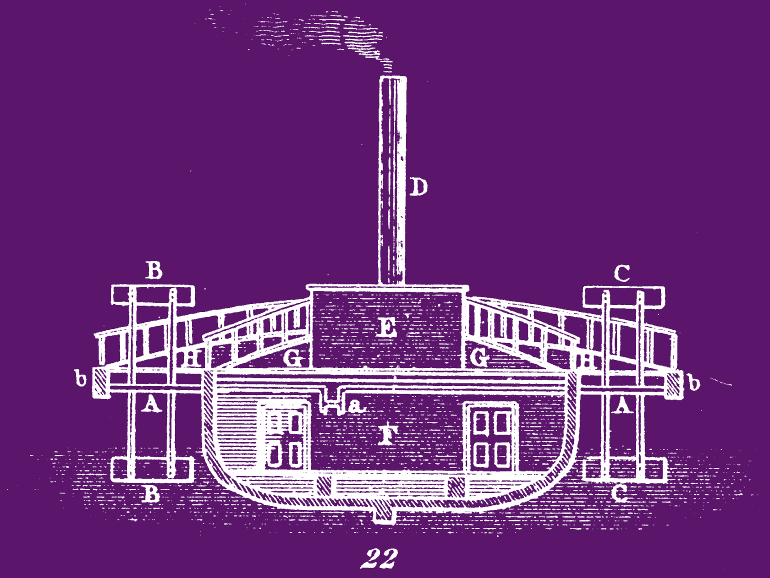
JUNE 2009 VOLUME 35 NUMBER 3

;login:

THE USENIX MAGAZINE



USENIX

The Advanced Computing Systems Association

18th USENIX SECURITY SYMPOSIUM

Montreal, Canada August 10–14, 2009

Join us for a 5-day tutorial and refereed technical program for security professionals, system and network administrators, and researchers.

2 Days of In-Depth Tutorials Taught by Industry Leaders, Including:

- Frank Adelstein & Golden G. Richard III on Learning Reverse Engineering: A Highly Immersive Approach (2 Day Class)
- Patrick McDaniel & William Enck on Building Secure Android Applications
- Phil Cox on Securing Citrix XenServer and VMware ESX Server

Keynote Address
Rich Cannings and David Bort of Google on the Android Open Source Project

Technical Program
26 refereed papers presenting the best new research in a variety of subject areas, including malware detection and protection, securing Web apps, and applied crypto

Invited Talks by Experts, Including:

- Jeremiah Grossman, WhiteHat Security, on "Web Security"
- Alex Sotirov on "Modern Exploitation and Memory Protection Bypasses"
- David Dagon, Georgia Institute of Technology, on bots

Co-Located Workshops:

EVT/WOTE '09
2009 Electronic Voting Technology Workshop/ Workshop on Trustworthy Elections
August 10–11, 2009

CSET '09
2nd Workshop on Cyber Security Experimentation and Test
August 10, 2009

WOOT '09
3rd USENIX Workshop on Offensive Technologies
August 10, 2009

HotSec '09
4th USENIX Workshop on Hot Topics in Security
August 11, 2009

MetriCon 4.0
Fourth Workshop on Security Metrics
August 11, 2009

Register by July 20, 2009, and save! **www.usenix.org/sec09/lg**

contents

;login:

VOL. 34, #3, JUNE 2009

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
Steve Gilmartin
proofshop@usenix.org

PRODUCTION
Casey Henderson
Jane-Ellen Long
Jennifer Peterson

TYPESETTER
Star Type
startype@comcast.net

USENIX ASSOCIATION
2560 Ninth Street,
Suite 215, Berkeley,
California 94710
Phone: (510) 528-8649
FAX: (510) 548-5738

http://www.usenix.org
http://www.sage.org

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) is published bi-monthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

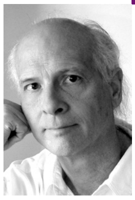
$90 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $125 per year.

Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

RIK FARROW

musings



Rik is the Editor of *;login:*.

rik@usenix.org

AS I HIKED DOWN PANORAMIC HILL, shortly after the HotPar workshop had completed in Berkeley, I marveled at the mounds of dirt pushed up by gophers. Were these the same species of gophers that made similar mounds in my yard? At home, the gophers come and go, apparently limited by effective predation. There is a natural balance between prey and predators, just as there is a balance that occurs in the acceptance and use of related types of technologies.

HotPar had been preceded by FAST, and you will find the FAST '09 summaries in this issue of *;login:*. Both HotPar and FAST sessions explored the edges of research in these two fields. And both left me with the impression that there are real trade-offs in computer technology, just like the natural balancing acts I can see when I am hiking.

Storage

I arrived too late to catch any of the morning FAST tutorials, but I did get to listen in on the Storage Class Memory (SCM) Technology tutorial [1] given by a group of IBM Almaden researchers. Flash-based Solid State Disks (SSDs) immediately come to mind when considering new types of storage. But these IBM researchers point out that flash does not rise to the level of what they consider SCM.

They expect that SCM will become a routine part of server-class systems in the future—that is, by 2013. SCM will sit between DRAM and disk, but mainly as a DRAM replacement, instead of replacing or supplanting the familiar hard drives of today. I considered this strange, as SSDs have replaced disks in laptops already, and Sun has products in the works for placing SSDs in storage systems. Besides, the current disadvantages found in flash seem to preclude flash from ever replacing random access memory. But the IBM guys are experts in their field, and this was their prediction.

Flash really shines at random read operations. But 7200 rpm consumer hard drives have the best cost/performance ratio for every other class of storage operations, something that surprised me (but I trust the Carnegie Mellon researchers who produced this result [2]). Flash has a big problem with writes. Flash, like any hard drive, always writes data in blocks. But flash works differently from disk, in that a block has to be erased before it can be written. Typical flash blocks are 128KB, so writing a single byte of data means finding a previously erased block, reading the current data block into an on-device buffer, modifying the block, and writing it out. On top of this, flash technology is limited to 100,000 writes per block, implying that long-term endurance of a flash device could be measured in seconds.

Flash devices handle endurance with the Flash Translation Layer (FTL), which uses write-leveling to distribute erasures and writes across the entire media. Some flash uses, particularly those typical of desktops and laptops, are predominantly random read, so flash can function well here. The higher cost of SSD implies that flash will appear first in higher-cost devices, and today you only see SSD being sold with laptops and netbooks—a natural fit.

During an SSD BoF at FAST, Milo Polte of Carnegie Mellon pointed out that we still don't know where SSDs best fit into system architecture. SSDs are currently used as disk replacements, but he and others present at the BoF questioned if this is the right place for them. SSD vendors make flash appear to be block-oriented, but future uses may be more like what the IBM researchers envision, with SCM acting as a main memory replacement and not as block-oriented storage devices.

Like hard drive manufacturers, SSD vendors are hiding the internal details and functioning of their products [3]. Intel's pricy X25E 32GB SSD, according to Polte, starts off with great write performance, but this plummets because the device hides the overhead of block erasure during its initial operation. Once erased blocks become scarce, the thread controlling block erasure, a slow process (2ms), cuts heavily into write performance.

We are watching the evolution of a new class of hardware, the SCM. Just like the gophers and their predators, this new class will eventually find its way into its proper niche. Unlike nature, human researchers and designers have control over the design of flash and SCM, and that will have a lot to do with where SCM will fit into the architecture of future systems.

The Pack Against the Three-Headed Hound

During the late 1980s, I had the privilege of getting to review new workstation hardware for *UnixWorld* magazine. I didn't even own a computer, as I always had at least one brand-new UNIX-based workstation at home. I studied the systems architecture, as well as the user manuals for CPUs (MIPS, SPARC, PowerPC, Alpha), trying to understand why certain workstations did better at different benchmarks. Floating-point performance depended largely on processor support, but general performance relied on good bus design.

My workstation focus left me largely unaware of the revolution in multiprocessing that was happening at the same time. There were good reasons for this—for example, I couldn't have my own Connection Machine [4] for review. Not only were delivery, setup, power, and cooling serious issues, but I really had no idea how to review such massive and specialized systems.

Other people within the design industry were paying close attention to various Symmetric MultiProcessors (SMPs), because they saw these as potential game changers in the high-end server market. Gregory Pfister of IBM in Austin had collected a lot of research which eventually became a book comparing various forms of SMP to clusters [5]. Ric Wheeler of Red Hat recommended that I read this book when I mentioned my interest in earlier SMP technologies.

I actually had owned this book, but gave it away without reading it. The cover depicts a three-headed dog fighting off a pack of dogs. I recall puzzling over this weird cover before packing the book up to send to a university library, something I did with many of the books publishers sent me out of the blue. Now I needed to buy a copy of the book I had given away many years ago.

Pfister is an entertaining writer who is also a master of his topic. The cover that puzzled me symbolizes SMPs as multi-headed beasts and a cluster as a pack of dogs. He explains that while an SMP has multiple "heads," the processors, it has to share the rest of the "body." The pack of dogs represents the complete computer systems in a cluster all working together to attack a problem. The SMP not only must share system resources, such as memory and I/O, but also has less resiliency, as there is only one system image running. The cluster can lose one or more of its members and continue running, as each member is a compete system.

The big trade-off between SMP designs and clusters has to do with memory latency. SMPs share memory, so they can communicate quickly when running parallel programs. Cluster members each have their own memory and rely on message-passing to synchronize while running similar programs. Message-passing latency, usually over commodity networks, is many times slower than interprocess communication in SMP systems.

NUMA, Non-Uniform Memory Access systems, or Cache Coherent NUMA as Pfister calls them, sit somewhere in between when it comes to communications latency. These systems have high-speed interconnects, like the AMD Barcelona, so shared memory can be close (controlled by the on-chip memory manager) or far (accessed via another CPU's memory manager). Cache coherency means that changes to values in one processor's memory will be visible to other connected processors, but only after a significant delay.

You may be wondering what SMPs of the past have to do with the systems of today and the near future. The connection is obvious once you consider that multicore chips are like SMPs; as the number of cores grows larger over time, CPU designers are facing the same types of issues faced by the much earlier SMP system designers, with access to memory being the most significant one.

Cluster systems have become very popular when it comes to applications that can have lots of data and can be partitioned easily. Google's search application should come to mind, as it involves storing immense amounts of data using a cluster file system (GFS) and manipulating that data with MapReduce operations. MapReduce allows data to be partitioned, with each map or reduce operating on data that is local. Hadoop, an open source project sponsored by Yahoo!, works in a similar fashion and has been adopted by companies working outside of the Internet search business, but with lots of data and tasks that are easy to partition.

Clusters certainly have their place today and in the future of large systems. Multicore systems will dominate both the server and the desktop, including each head/node in any cluster. This is why I wanted to read Pfister's book, which despite being over 10 years old is still very relevant (and still in print, which also says a lot).

The Line-up

Nothing like thinking about multiple racks of high-powered servers used by clusters to get one thinking about energy. Alva Couch, professor at Tufts University and a USENIX Board member deeply interested in Green IT, writes about the Greening of IT. I really like how Alva describes the two types of Green that IT cares about and how this impacts buying behavior.

Next up, Hasan et al. have written a nice article, based on their FAST '09 paper, about provenance. Margo Seltzer drew my attention to their paper as a potential *;login:* article, and I agreed it would work well. Hasan explains what provenance is and why it is important and becoming even more important, before describing their own work on turning provenance on shared data into something that can be trusted.

Andrew Leung, working with NetApp engineers and Ethan Miller, writes about Spyglass. Based on a related FAST '09 paper, Leung points out that metadata searches for all the files owned by a particular user greater than 100KB and modified during the past week, for example, can take too long to accomplish when storage systems become truly large. The UNIX `find` command doesn't scale well when used on petabyte systems (unless you plan on reading a nice book like Pfister's while you wait for a response). Leung et al. came up with a solution that uses a separate store of metadata to speed up searches by a couple of orders of magnitude.

Weihang Jiang et al. examined the role of system logs when troubleshooting storage system problems. Like past FAST research sponsored by NetApp, this research also relies on the vast storehouse of data collected by NetApp as customers use their storage devices. In this article, Jiang explains the relationship between various classes of storage failures, from the failure of a disk to that of software, and how logs can help in analyzing these failures. I've spent a fair amount of time looking at logs in my life, as I suspect is true of most USENIX members. The results of Jiang's research will, I expect, match your own experience.

Rudi van Drunen continues his series on hardware by explaining signals. Unlike power and voltage, signals need to deliver data reliability, and Rudi explains how factors such as distance and the design of cables affect data delivery.

David Blank-Edelman continues on his theme of Web-page scraping by taking on the challenge of extracting data from a messy page. Peter Galvin describes advances in installing the LAMP/SAMP stacks when using Solaris and OpenSolaris. Dave Josephsen tells us about what happened when the company he works for suffered from the TV version of slashdotting, an interesting tale. Robert Ferrell then regales us with his thoughts on social networking.

Elizabeth Zwicky and Sam Stover have written book reviews for this issue. Jason Dusek has vowed to return in the next issue with more reviews about programming books.

We finish up with FAST '09 and TaPP '09 conference reports.

Wrap-up

I still am astounded by the diversity I see in nature. Once I learned how to pay attention to my surroundings, I started to notice how the vegetation changed between the different faces of a hill, related to both sun and water, and how different critters exist in different environments.

As I walked down Panoramic Hill, I not only noticed the gopher holes but wondered if there were any pesky pack rats in the Bay Area. Pack rats build messy nests of gathered materials, and will even do so on car engines using your wiring harness as part of their nest. You really don't want pack rats in your area, believe me. Their range appears limited to desert areas, including most of Arizona, and some of the southern California deserts. Pack rats have evolved to fit a particular niche, and our technology just happened to get involved in very recent times.

I expect that new technologies such as SSDs, SCM, and multicore processors will also fit into their own niches. Computer scientists and technology companies invent devices, but these devices will fit naturally into whatever environment that matches them the best.

REFERENCES

[1] Wilcke, "Flash and Storage Class Memories" (similar to a portion of the FAST '09 tutorial): institute.lanl.gov/hec-fsio/workshops/2008/presentations/day3/Wilcke-PanelTalkFlashSCM_fD.pdf.

[2] Simsa, Polte, and Gibson, "Comparing Performance of Solid State Devices and Mechanical Disks," Parallel Data Laboratory, Carnegie Mellon University, 2008: http://www.pdsi-scidac.org/events/PDSW08/resources/slides/simsa_PDSW.pdf.

[3] Farrow, "Musings," June 2007: http://www.usenix.org/publications/login/2007-06/openpdfs/musings.pdf.

[4] Thinking Machines: http://en.wikipedia.org/wiki/Thinking_Machines.

[5] Pfister, *In Search of Clusters,* 2nd ed. (Prentice Hall, 1998), ISBN 0-13-899709-8.

PACKRATS OFTEN CONSTRUCT NESTS OF STICKS AND DEBRIS AROUND LARGE PRICKLY PEAR CACTUSES. THIS NEST IS SIX FEET ACROSS.

PACKRATS WILL USE HUMAN-PROVIDED STRUCTURES, SUCH AS WOODPILES OR CARS, WHEN BUILDING NESTS. WHEN USING A CAR, SUCH AS MY SUBURU, SHOWN HERE, ONLY THE INTERIOR LAYER, SHREDDED JUNIPER BARK FOR BEDDING, IS REQUIRED.

ALVA L. COUCH

Is it easy being green?

Alva L. Couch is an associate professor of Computer Science at Tufts University, where he and his students study the theory and practice of network and system administration. He served as program chair of LISA '02 and was a recipient of the 2003 SAGE Professional Service Award for contributions to the theory of system administration. He currently serves as Secretary of the USENIX Board of Directors.

couch@cs.tufts.edu

THERE REMAINS SOME CONFUSION about whether the term "Green Information Technology" refers to "being Green" or "saving Green ($)," and seeking convergence between the two often conflicting meanings is critical to developing a sustainable IT infrastructure.

Is it easy being Green? Unlike Kermit the frog, IT managers have a choice, and face difficult decisions between Green and not-so-Green approaches to IT. The dividing line between Green and not-so-Green is less clearly defined than it might seem at first glance. Considering the whole IT life cycle—or even the whole business life cycle that IT supports—can uncover subtle misconceptions that change what "being Green" entails.

What Is Green IT?

As with most terms forged mostly by marketing efforts, it is difficult to precisely define what is meant by "Green Information Technology" (Green IT). Considering the common threads from all vendor-supplied definitions, it seems to mean "considering the environment and environmental impact" in designing, building, managing, and decommissioning IT systems. This includes considering the impact of:

1. Power utilization, including the impact of power generation
2. Heat management, including heat release to the environment, as well as power requirements for moving heat around
3. Processes for manufacturing and disposal of computing hardware

Oddly enough, although there are plenty of available commercial approaches for "Greening" (1) and (2), there are few alternatives for coping with (3), although it remains part of the popular definition of "Green IT."

The root of this quandary is that there are two kinds of Green: the Green that refers to protecting the environment, and the Green that represents money kept in one's wallet. In addressing (1) and (2) above, one can often pursue the two kinds of Green at the same time, while addressing (3) requires that one kind of Green take precedence over the other. The former is easily sold to businesses, while the latter is an extremely hard sell.

For example, saving power in running an IT infrastructure is Green in two ways, because one both saves environmental impact (from power generation) and saves money (because power is a tangible cost of IT). Seemingly extreme changes such as putting up one's own solar panels or windmills can have a relatively short payback period and are undoubtedly Green in both ways (but there are some subtleties of life cycle analysis that most might easily miss, mentioned below).

On the Ground or in the Clouds?

The popular concept of Green IT is part of the motivation for many current research and development efforts.

Power awareness is one of the main practical outgrowths of autonomic computing, in the sense that many practical algorithms exist for minimizing power and cooling requirements while still responding to predicted loads. Some researchers think power awareness will be the main selling point for autonomic systems in the future, because it is one of the key business goals that a human administrator cannot feasibly attain.

While one can achieve power awareness even at small scales, it is one of the few aspects of computing that becomes more feasible and practical as scale increases. Power awareness can be achieved particularly well in cloud computing, in which one farm of servers serves multiple applications. One important side benefit of cloud computing is that one can cluster applications inside clouds and "optimize the ensemble" to save power [1]. Because a cloud serves multiple applications, it can reduce power needs by maintaining a capacity pool for all clients, not for each client separately.

Conflict Between the Greens

However, some aspects of Green IT—that are only Green in a single way—are much harder sells. Consider, e.g., how machine rooms are cooled. It is much cheaper (the monetary kind of Green) to cool the rooms with running groundwater, e.g., rivers, rather than using power and air-conditioning to release heat into the air, but this is considered to be very bad for the other kind of Green, because it affects and can radically transform the habitat for aquatic life. The practice of using natural cooling of this kind remains a highly controversial and emotional issue, perhaps because we know something of the potential impact from our use of groundwater to cool nuclear power plants [2].

The conflict between the two kinds of Green is perhaps most extreme when one considers equipment life cycles. An environmentally aware IT management strategy must also consider not only the impact of utilizing hardware, but also of manufacturing and disposing of hardware. Computer manufacturing continues to generate many hazardous wastes, with no easy solutions in sight. Disposal and recycling of computing equipment are also a growing problem, with the average computer obsoleted and replaced every three years or less.

Part of what makes managing equipment life cycles difficult is that the money to be made by planned obsolescence (even including upgrading to more power-efficient hardware) trumps the environmental impact of disposing of the so-called "obsolete" equipment. Here it seems that a viable Green alternative would have to reduce disposal impact while at the same time encouraging innovation and growth. Convergence between the two kinds of Green in this case would likely involve implementing a completely new and innovative business model for upgrading computing equipment, much like the maintenance agreements we currently buy for software. We would hang on to our computers longer if we could safely upgrade their capabilities, but one can be sure that for financial viability, such upgradable units would come with a relatively high price tag compared to that of today's "disposable" hardware.

Life Cycle Analysis

Considering equipment recycling in Green IT is an example of "life cycle analysis," which entails taking a broad view of the whole process of providing a service and taking into account the impact of every step from service inception to service decommissioning. The practice of life cycle analysis has disciplinary roots in chemical and process engineering, but can just as easily be applied to IT.

Taking the broad view of power has already led to some counterintuitive surprises with analogues in IT. In calculating the emissions of coal-burning power plants, accounting for emissions arising from moving the coal via truck or train—as well as the emissions from the power plant—demonstrates that burning locally available coal that burns less cleanly leads to lower total emissions than trucking in cleaner-burning coal from remote locations [3]. In the same way, the location of a data center affects the total power consumption of the enterprise in distributing data and serving computational needs.

Even taking seemingly positive steps can have hidden impacts. Mandating use of fluorescent bulbs to save energy leads to a secondary disposal problem for the mercury in the bulbs, in the same way that replacing all computer hardware with lower-power alternatives leads to a recycling problem for the higher-power obsolete hardware. And the apparently "Green" strategy of putting up solar panels may not look as Green when one considers the life cycle of the panels, their mean time to failure, and their manufacturing and recycling impacts. Solar cells are not created equal, so the lowest-impact solar power requires some careful planning and choices. Still, solar is much cleaner overall than burning fossil fuels, and most of the environmental impact from solar cells is from burning fossil fuels during the manufacturing process [4]. In considering the broad view, solar-powered boilers whose steam output is converted to electricity in the usual way are considered to be "less efficient" in producing power than solar cells, but may have even less environmental impact than solar cells when their impact is averaged over the life cycle of the equipment.

Thus, one might ask, if taking the broad view has such profound impact upon one's decisions, why is "Green IT" focusing only upon the data center? What different decisions would we make if we considered instead the life cycle of the entire business process? This question is at the root of some criticisms of the current concept of "Green IT."

"Green" Versus "Sustainable"

Lack of progress in Green directions—and seemingly limited opportunities for financially sensible Green—have given the term "Green IT" a bad name in some circles. It has become synonymous with talking about considering the environment while unapologetically continuing business practices that actively harm it. "Green IT" seems to be about compartmentalizing the environmental impact problem in the data center, "fixing the data center," and leaving the rest of the business process intact, wasteful, and blind to environmental concerns. The compartmentalization of "Green" extends even into the IT research community, which is for the most part studying "power awareness" (and strategies that save money) instead of a broader concept of "environmental awareness" (and strategies that cost money).

Some would prefer that we instead pursue a more aggressive goal of "Sustainable IT." While "Green IT" has come to mean "best effort" environmental awareness, "Sustainable IT" refers to running the data center with as little environmental impact as possible.

Even with "Sustainable IT," sustainability is still compartmentalized in the datacenter and cannot touch parts of business process that are consumers of IT. One is limited to making relatively small improvements in what is considered the purview of IT departments.

The lessons of life cycle analysis suggest that "Sustainable IT" is too narrow and that one should instead consider the sum total of all environmental impacts in all business decisions and target for a "Sustainable Enterprise," aiming for zero total impact for the whole enterprise.

For example, one major improvement in the business process would be to replace travel with telecommuting and virtual meetings. This is a business process change that is "Green" in both ways: it reduces corporate environmental impact and saves money by reducing consumption of fossil fuels during travel to and from meetings.

Green Versus Inertia

One often finds that a fantastic idea that would transform the way the world does business cannot be implemented, because the world is much too satisfied with the way it currently does business. The stark reality is that changes in business process can be incredibly expensive, involving retraining staff for new policies, as well as developing and managing new concepts of customer expectation. The true opponents of sustainability are inertia and the cost of changing how businesses are run and how business decisions are made. Truly sustainable infrastructure may well require a transformation of society, not just a transformation of technology, in very much the same way that our dependence upon oil can only be removed by a rather large shift in the way we live.

So true convergence between the two kinds of Green may be impossible, but that does not mean we can't redefine the two kinds in subtle ways to make convergence more likely. This is the true challenge and promise of sustainable computing.

The USENIX Board of Directors asked me, in my role as the Secretary of the Board, to look into the prospects for Green IT and explore what USENIX can contribute to the effort. I am open to input from everyone on this issue; please feel free to contact me at my email address, alva@usenix.org.

REFERENCES

[1] Niraj Tolia, Zhikui Wang, Manish Marwah, Cullen Bash, Parthasarathy Ranganathan, and Xiaoyun Zhu, "Delivering Energy Proportionality with Non Energy-Proportional Systems—Optimizing the Ensemble," *Proc. HotPower '08*, USENIX Association.

[2] U.S. Geological Survey Circular 1268, "Estimated Use of Water in the United States in 2000" (released March 2004, revised April 2004, May 2004, February 2005).

[3] D. Labbe, “NOx, SOx and CO2 Mitigation of Blended Coals Through Optimization,” to appear in *Proc. 19th Annual ISA POWID/EPRI Controls & Instrumentation Conference*, May 12-14, 2009, Chicago, IL.

[4] David Biello, “Dark Side of Solar Cells Brightens,” *Scientific American*, February 21, 2008: http://www.sciam.com/article.cfm?id=solar-cells-prove-cleaner-way-to-produce-power.

RAGIB HASAN, RADU SION, AND MARIANNE WINSLETT

secure provenance: protecting the genealogy of bits

Ragib Hasan is a PhD candidate in the Department of Computer Science at the University of Illinois at Urbana-Champaign. His research interests include secure provenance, regulatory compliant storage systems/databases, and other aspects of storage security.

rhasan@cs.uiuc.edu

Radu Sion is an assistant professor in the Department of Computer Science at Stony Brook University. His research interests span information assurance, practical cryptography, and large data and compute-intensive systems.

sion@cs.stonybrook.edu

Marianne Winslett is a research professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. Her research interests lie in information security and in the management of scientific data. She is an ACM Fellow, a former vice-chair of ACM SIGMOD, and recipient of the Presidential Young Investigator Award from NSF.

winslett@cs.uiuc.edu

THE ABILITY TO TRACK THE ORIGIN OF information is essential in science, medicine, commerce, and government. Applications such as digital rights protection, DNA testing, drug trials, corporate financial accounting, and national intelligence need to guarantee the integrity and authenticity of information as it flows between people and tasks. In this article, we describe what digital provenance means and how to provide strong integrity and confidentiality assurances for data provenance information. We present our provenance-aware system prototype that implements provenance tracking of data writes at the application layer, which makes it extremely easy to deploy. The prototype is efficient: for typical real-life workloads, its runtime overhead does not exceed 13% and is below 3% for most workloads.

Provenance

In 2006, the Picasso painting *Dora Maar au Chat* (Dora Maar with Cat) was auctioned at Sotheby's for US $95 million, becoming one of the most expensive paintings in the world. At the same time, someone listed several paintings for sale on eBay, supposedly complete with Picasso's signature. Although eBay quickly removed those listings, many purported Picassos are still on the market.

How do art buyers authenticate paintings? Many factors play a role, but one key element is provenance records that list the ownership history of an item and the actions performed on it. Provenance is widely used in the arts, archaeology, science, genealogy, and data archives, where it has been called the *fundamental principle of archiving*.

Provenance has traditionally been used to authenticate physical objects, but life today has become increasingly dependent on digital information that originated elsewhere, was processed by other people, and was stored in potentially untrustworthy storage. In such situations, it is increasingly important to know where the information comes from and how it has been processed and handled. In other words, to be able to trust a piece of information, we need to know its provenance.

Provenance is a highly overloaded term, but all definitions share the same core concepts: data provenance is a description of the origins, lineage, derivation, and transmission history of a digital object. Until now, scientists have been the primary users of data provenance systems, and provenance research has mainly focused on the tasks of modeling, representation, collection, annotation, and querying.

However, as provenance steps into mainstream computing, new challenges arise. With its increased use in financial, medical, and other non-scientific application areas, provenance information faces a host of security threats, including active attacks from adversaries. In high-stakes business and medical applications, insiders may have significant incentives to alter data records' history. For example, in big finance, regulatory and legal considerations mandate provenance assurances, and the US Sarbanes-Oxley Act sets prison terms for officers of companies that issue incorrect financial statements. As a result, officers have become very interested in tracking and securing the path that a financial report takes during its development, including both input data origins and authors. The US Gramm-Leach-Bliley Act, Securities and Exchange Commission rule 17a, and HIPAA also require documentation and audit trails for financial or medical records.

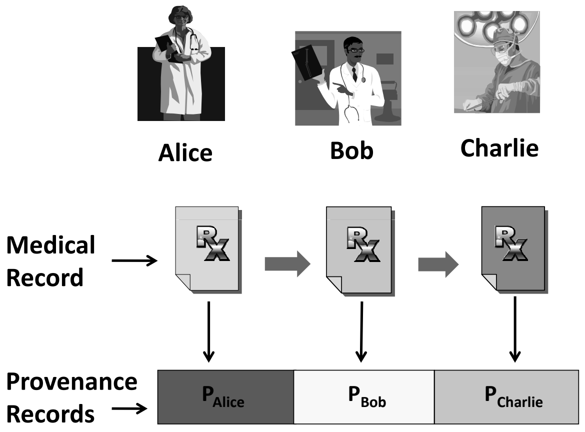


FIGURE 1: EXAMPLE SCENARIO. DANA GOES TO DR. ALICE, WHO REFERS HER TO DR. BOB FOR A TEST, AND THE TEST RESULTS ARE THEN PROCESSED BY DR. CHARLIE. BECAUSE OF A MISDIAGNOSIS BY DR. BOB, DANA SUFFERS HEALTH PROBLEMS AND SUES DR. CHARLIE, WHO WANTS TO USE THE PROVENANCE ($P_{ALICE}|P_{BOB}|P_{CHARLIE}$) OF DANA'S MEDICAL RECORDS TO PROVE HIS INNOCENCE. DR. BOB WANTS TO HIDE HIS ACTIONS BY RETROACTIVELY ALTERING HIS ENTRIES IN DANA'S MEDICAL RECORDS AND TAMPERING WITH THE PROVENANCE RECORD OF HIS DIAGNOSIS (P_{BOB}) TO MATCH.

When information crosses application and organizational boundaries and passes through untrusted environments, its associated provenance information is vulnerable to illicit alteration. For example, in Figure 1, Dr. Bob, wanting to hide evidence of his misdiagnosis, retroactively changes the diagnosis in his patient's record and tampers with the associated provenance record. Access control is insufficient to prevent this tampering, as Dr. Bob may have physical control over a machine where the information resides. Thus, the trustworthiness of the provenance records themselves is in question: we need *provenance of provenance*, i.e., a model for secure provenance.

Making provenance records trustworthy is challenging. Ideally, we need to guarantee *completeness*—all relevant actions pertaining to a piece of information are captured; *integrity*—adversaries cannot forge or alter provenance records; *availability*—auditors can verify the integrity of provenance information; *confidentiality*—only authorized parties can read provenance records; and *efficiency*—provenance mechanisms should have low overheads.

Here, we take the first step towards preventing forgery of history as stored in provenance records. We present a scheme for providing integrity and confidentiality assurances for provenance records, and we describe a proof-of-concept implementation for file systems that imposes only 1% to 13% overhead for typical real-life workloads.

A Model for Provenance

In what follows we use the term *document* to refer to the data item for which provenance information is collected, such as a file, database tuple, or network packet. At the IT layer, the *provenance* of a document is the record of actions taken on that document over its lifetime. Each access to a document D may generate a *provenance record* P. The types of access that should generate a provenance record depend on the domain, as do the exact contents of the record, but in general P may include the identity of the accessing principal; a log of the access actions (e.g., read, write) and their associated data (e.g., the bytes of the document or its metadata that were read/written); a description of the environment at the time of the action, such as the time of day and the software environment; and confidentiality- and integrity-related components, such as cryptographic signatures, checksums, and keying material. A *provenance chain* for document D is a non-empty time-ordered sequence of provenance records $P_1 \mid \cdots \mid P_n$.

In a given security domain (organization), *users* are principals who read and write documents and their metadata. Each organization has one or more *auditors*, who are principals authorized to access and verify the integrity of provenance records associated with documents. Documents move from one user to another, as email attachments, FTP transfers, or by other means. Provenance chains move with the documents. When a user modifies a document, a new provenance record describing the modifications is appended to the provenance chain and the user permits some subset of the auditors to read the new record. *Adversaries* are principals from inside or outside an organization who have access to a document and its provenance chain and who want to alter them inappropriately, as discussed below.

We cannot track provenance perfectly, because a provenance tracking system implemented at a particular level of the system is oblivious to attacks that take place outside the view of that level. For example, suppose that we implement provenance tracking in the OS kernel. If the kernel is not running on hardware that offers special security guarantees, an intruder can take over the machine, subvert the kernel, and circumvent the provenance system. Even with a trusted pervasive hardware infrastructure and provenance tracking at every level of the system, a malicious user who can read a document can always memorize and replicate portions of it later, minus the appropriate provenance information. Since we cannot fully monitor the information flow channels available to attackers, our power to track the origin of data by monitoring read operations is limited. Given a guarantee that users could never circumvent the provenance mechanisms, we could reliably track all information flows by recording all information that each user reads, in addition to what they write. However, promulgation of provenance for read operations can result in a combinatorial explosion in overhead that can make the system unusable [16]. In this work, we target applications that do not require tracking of reads.

Consider the version history that would result if a document were created and subsequently edited and transferred from user to user, with provenance information correctly and indelibly recorded all along the way. We call this a *plausible history* for the resulting document and its chain. We target applications whose provenance integrity needs are met by the following guarantee: *if a provenance chain does not give a plausible history for its associated document, we will detect this*. Such applications are common. For example, a retail pharmacy will not accept a shipment of drugs unless it can be shown that the drugs have passed through the hands of certain middlemen. If a criminal wants to sell drugs manufactured by an unlicensed company, he will want to forge a provenance chain that gives the drugs a more respectable history, so that he can move them into the supply chain. Our approach detects that the new chain is forged. The criminal will not want to take drugs manufactured and distributed through legitimate channels, strip off their distribution provenance records, and replace them by a record showing that he distributed the drugs himself, as this new plausible history for the drugs makes them worthless. Similarly, there is little danger that someone will remove the provenance chain associated with a box of Prada accessories and try to pass them off as another brand. Instead, the incentive is to pass off non-Prada accessories as Prada, and we detect this attack.

More precisely, suppose that we have a provenance chain ([*A*], [*B*], [*C*], [*D*], [*E*], [*F*]), in which, for simplicity, each record is denoted by the identity of its corresponding principal *A*, . . . ,*F*. We provide the following integrity and confidentiality assurances with respect to forgery of document history.

I1: An adversary acting alone cannot selectively remove other principals' records from the beginning or middle of a provenance chain without being detected at the next audit.
I2: An adversary acting alone cannot add records in the beginning or the middle of the chain without being detected at the next audit.
I3: Two colluding adversaries who have contributed records to a provenance chain cannot add records of other non-colluding users between theirs without being detected by the next audit.

For example, colluding users *B* and *D* cannot undetectably add records between their own, corresponding to fabricated actions by a non-colluding party *E*.

I4: Once the chain contains subsequent records by non-colluding parties, two colluding adversaries who have contributed records to a provenance chain cannot remove the record of any non-colluding user between theirs without being detected by the next audit.

For example, colluding users *B* and *D* cannot remove records made by non-colluding user *C*.

A user Gertrude who can read the last record in a chain can recreate the previous version of the document by removing that record and undoing its writes to the document. Thus Gertrude and her colleagues can roll back history, then edit the old version as desired, adding new provenance records that match their actions and thereby constructing a new plausible history. However, constraint I4 prevents Gertrude from attributing any of the new writes to those who are not collaborating with her. For example, suppose Gertrude undoes records *F* and *E*, where *F* is a stamp of approval from the Food and Drug Administration. Her collaborator *E* then alters the old version of the document, generating a new provenance record *E'*. If they then try to reaffix the old FDA stamp of approval *F* without the FDA's help and cooperation, that forgery will be detected by the next audit, as the resulting provenance chain does not correspond to any plausible history.

I5: Users cannot repudiate chain records.
I6: An adversary cannot claim that a valid provenance chain for one document belongs to a document with different contents, without detection by the next audit.
I7: If an adversary alters a document without appending the appropriate provenance record to its chain, this will be detected by the next audit.
C1: Any auditor can verify the integrity of the chain without requiring access to any of its confidential components. Unauthorized access to confidential provenance record fields is prevented.
C2: The set of parties originally authorized to read the contents of a particular provenance record for D can be further restricted by subsequent writers of D.

A Secure Provenance Scheme

We proposed a solution composed of several layered components: encryption for sensitive provenance record fields, a checksum-based approach to ensure provenance record integrity, and an incremental chained signature mechanism for securing the integrity of the chain as a whole. For confidentiality, we deployed a special keying scheme based on broadcast encryption key management to selectively regulate the access for different auditors. To provide fine-grained confidentiality, we used a cryptographic commitment-based construction.

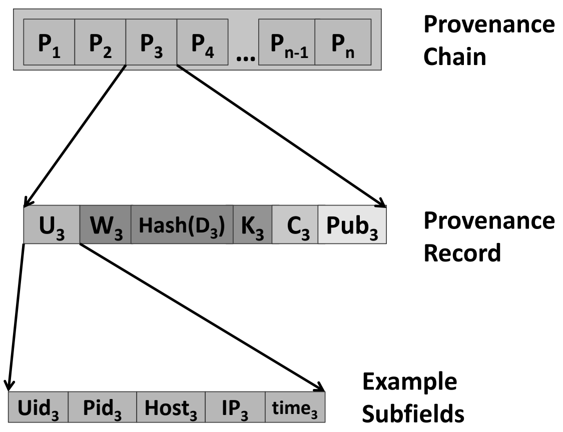


FIGURE 2: STRUCTURE OF A PROVENANCE CHAIN, SHOWING A PROVENANCE RECORD AND THE FIELDS OF ITS USER IDENTITY COMPONENT.

More precisely, each provenance record P_i summarizes a sequence of one or more actions taken by a user:

$$P_i = <U_i, W_i, hash(D), K_i, C_i, [public_i]>$$

where

- U_i is a plaintext identifier for the user;
- W_i is an encrypted or plaintext representation of the sequence of actions (the *modification log*) performed by the user;
- *hash(D)* is a one-way hash of the current content of the document;
- K_i contains keying material that auditors can use to decrypt the encrypted fields, as explained below;
- C_i contains an integrity checksum (defined below) for this provenance record, signed by user U_i;
- $public_i$ is an optional encrypted or plaintext public key certificate for user U_i.

As a practical matter, at the start of an editing session the provenance system should verify that the current contents of *D* match its hash value stored in the most recent provenance record. We discuss each of the record's fields below.

CONFIDENTIALITY

Certain fields or subfields of a provenance record may be sensitive, such as the identity or individual steps of a proprietary process used to clean experimental data. If all users trusted all auditors, then providing confidentiality for these sensitive fields would be straightforward—we could just encrypt all of them with a single public key, and give the private key to the auditors. If a user trusted only certain auditors, we could make several copies of the sensitive fields, encrypt each copy with the public key of a different trusted auditor, and include all of them in the new provenance record. While secure, this wastes space. Instead, as shown in Figure 3(a), we encrypt the sensitive fields of a record once with a new secret key, make multiple copies of the secret key, and encrypt each copy with the public key of a different trusted auditor. In this scheme, the new provenance record contains the encrypted sensitive fields plus several versions of the encrypted secret key, stored in the K_i field of the record. A trusted auditor can subsequently read the record, decrypt a copy of the secret key using the auditor's private key, and use the secret key to decrypt the sensitive fields. If there are many auditors, the record can be kept small by using a broadcast encryption tree to reduce the number of encrypted copies of the secret key. Other concerns such as separation of duty, or requiring a minimum number of auditors to cooperate when decrypting a secret key, can be addressed by using secret sharing and threshold encryption.

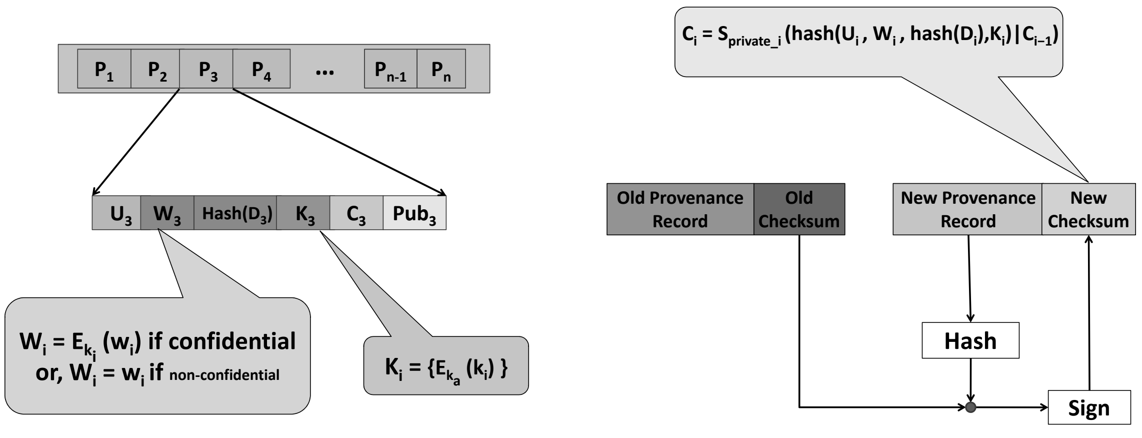
INTEGRITY

An audit must detect whether adversaries have removed or inserted elements from the chain and whether a chain has been switched from one document to another. To achieve this, the checksum C_i of a provenance record P_i is computed as shown in Figure 3(b). First we apply a cryptographic hash function to the tuple containing the user identity U_i, the hash of the document contents *hash(D)*, the modification log W_i, the key-related information *K*, and (if included in the record) the user's public key $public_i$. Then we concatenate the resulting hash with the checksum C_{i-1} of the previous provenance record P_{i-1}, sign the result with the user's private key, and store the signed result in the provenance record. More formally, the integrity checksum field C_i is:

$$C_i = S_{private_i}\ (hash(U_i,\ W_i,\ hash(D),\ K_i,\ [public_i])|C_{i-1})$$

where $S_{private_i}$ means that user U_i signs the hash with his or her private key.

To verify chain integrity, the auditor starts from the first record of the chain. The auditor extracts the user identity U_1 from the record and obtains $public_1$ from an external trusted source, or obtains $public_1$ from the record itself and uses an external trusted source to verify that $public_1$ is the public key of user U_1. The auditor uses the W_1 field (whether encrypted or plaintext), plus the U_1, K_1, *hash(D)*, and optional $public_1$ fields to generate a checksum *C* for the record. The auditor then uses $public_1$ to check that the signed checksum C_1 is in fact what would be produced by U_1 signing *C*. The auditor then moves on to the next record, remembering to include the signed checksum for the previous record in the computation of the checksum for the current record. Once the integrity of the chain is established, the auditor hashes the document *D* and verifies that the resulting hash value was stored in the last provenance record.



P_1 P_2 P_3 P_4 ... P_{n-1} P_n
U_3 W_3 Hash(D_3) K_3 C_3 Pub_3
$W_i = E_{k_i}(w_i)$ if confidential or, $W_i = w_i$ if non-confidential
$K_i = \{E_{k_a}(k_i)\}$
$C_i = S_{private\_i}(hash(U_i, W_i, hash(D_i), K_i) | C_{i-1})$
Old Provenance Record
Old Checksum
New Provenance Record
New Checksum
Hash
Sign

(A) **(B)**

FIGURE 3: (A) CONFIDENTIALITY. W_I IS THE MODIFICATION LOG, K_I IS A SECRET KEY THAT AUTHORIZED AUDITORS CAN RETRIEVE FROM THE FIELD K_I, K_A IS THE KEY OF A TRUSTED AUDITOR. (B) INTEGRITY. NEW CHECKSUM C_I IS A FUNCTION OF THE CURRENT DOCUMENT, NEW PROVENANCE RECORD, AND THE PREVIOUS CHECKSUM C_{I-1}.

Fine-Grained Control Over Confidentiality

Some portions of a provenance record may be quite sensitive. For example, suppose that because of a Freedom of Information Act request, a document containing sensitive information has to be released to the public. Usually this is done by redacting the sensitive content of the document. However, the provenance chain for the redacted document will also contain bits and pieces of the sensitive information here and there. We cannot simply remove every record containing sensitive information, as that will break provenance integrity checks. Encrypting the entire modification log just to hide a small piece of sensitive information is excessive.

To allow selective disclosure in cases like this, we replace each sensitive field (or sub-field) in the record by a *cryptographic commitment* for it, e.g., by appending a random number to the contents of the field and then hashing the result. We encrypt those random numbers with a secret key and leave them in the record for trusted auditors to use. When we compute the checksum, we use the commitments in place of the sensitive fields, and include the encrypted random numbers. The official provenance record includes the sensitive *and* non-sensitive fields, the commitments for the sensitive fields, the encrypted random numbers, and the checksum. When we release the provenance chain we can remove the sensitive fields, and subsequent integrity checks for the chain and document will still work correctly.

SUMMARIZING CHAINS

As the document is modified over time, the provenance chain can eventually become much larger than the document. System administrators may want to summarize a provenance chain by omitting all but the records corresponding to "important" actions. The original chain construction scheme does not allow summarization through removal of records. However, we can augment the chain by including additional independently computed checksums in each record. Each checksum is computed by taking the checksum of a previous record, combining it with the hash of the current record, and then signing it. For example, one checksum may connect the current record to the previous record, while another may connect it with the record two hops away in the chain. The additional checksums allow us to remove records and still be able to prove the integrity and chronological ordering of the remaining records.

ACTIONS RECORDED IN CHAINS

In our work, we recorded writes of document data and metadata in provenance records, as well as movements of a document across organizational boundaries. We do not record reads. When the document is deleted, the provenance chain may be retained for an application-appropriate period.

Implementation and Empirical Evaluation

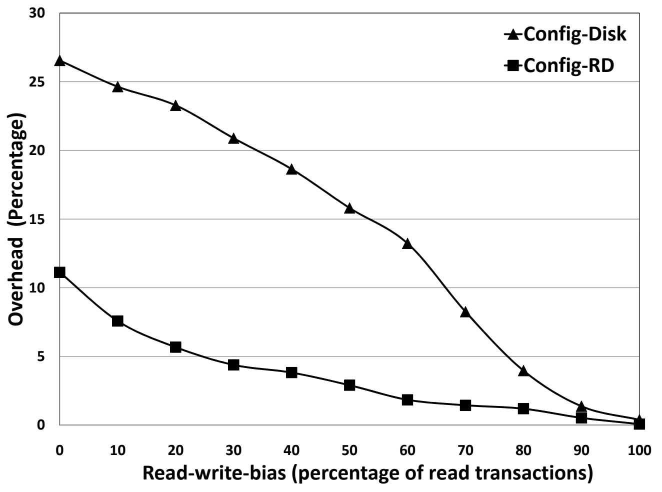


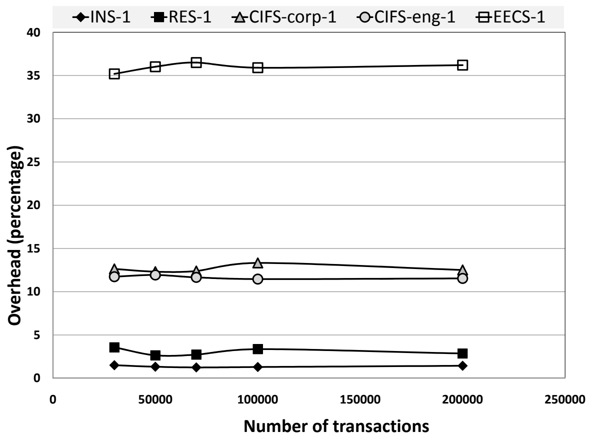
FIGURE 4: OVERHEAD OF SECURE PROVENANCE WITH POSTMARK. THE OVERHEADS ARE SHOWN FROM 0% READ BIAS (100% WRITE TRANSACTIONS) TO 100% READ BIAS (NO WRITE TRANSACTIONS), WITH OR WITHOUT A RAM DISK.

We designed and built a proof-of-concept prototype that provides secure provenance for files. The key design decision was where to place the provenance functionality. A kernel-layer or file-system-layer implementation makes provenance collection transparent to applications, but requires every user platform to run a modified kernel or file system, which hampers portability. A user-level implementation increases portability while still allowing a high degree of transparency. Thus we designed and implemented support for secure provenance through an application-layer C library called SPROV, consisting of wrapper functions for the standard file I/O library *stdio.h*.

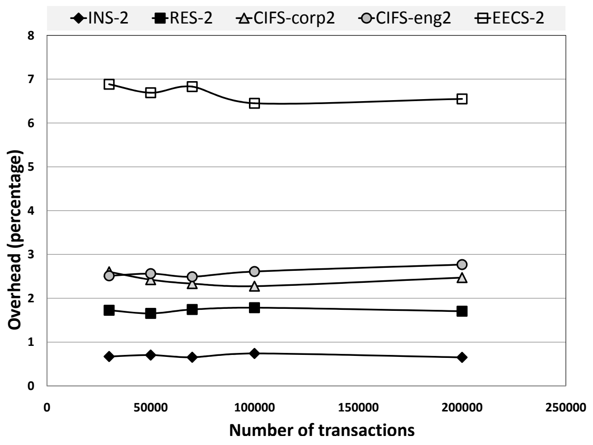
Our experimental testbed for SPROV was a workstation with an Intel Pentium 4 CPU at 3.4GHz, 2GB RAM, running Linux (Suse) at kernel version 2.6.11. In this configuration, each 1024-bit DSA signature took 1.5ms to compute. We used two modes for storing provenance chains—in the *Config-Disk* mode, the chains were stored on the disk, while in the *Config-RD* mode, we buffered the chains in a RAM disk and periodically flushed them to disk using a daemon.

We evaluated the overhead of SPROV using the standard Postmark benchmark for small files and workloads modeled on real-world traces. We ran

Postmark with secure provenance and without any provenance, and measured the runtime overhead. Modifying Postmark to use secure provenance required changing only 8 lines of code. A data set containing 20,000 Postmark-generated binary files with sizes from 8KB to 64KB was subjected to Postmark workloads of 20,000 transactions. Each transaction opened a file, issued a read or a write of size between 8KB and 64KB, then closed the file. We measured the performance overhead under different write loads by varying the percentage of write transactions from 0% to 100%, in 10% increments. As shown in Figure 4, the overheads start at 0.5% for both *Config-Disk* and *Config-RD*, ranging up to 11% for *Config-RD*.



(A)



(B)

FIGURE 5: EFFECT OF DIFFERENT WORKLOADS. (A) THE CONFIG-DISK SETTING. (B) THE CONFIG-RD SETTING.

Next we considered a more realistic scenario involving practical, documented workloads and file system layouts. We constructed a layout in the manner of Douceur and Bolosky [7], who showed that file sizes can be modeled using a log-normal distribution. We used the parameters μ^e=8.46, σ^e=2.4 to generate a distribution of 20,000 files, with a median file size of 4KB and a mean file size of 80KB. To that we added a small number of files with sizes exceeding 1GB, to account for large data objects [7].

We set the percentage of write and read transactions to match five studies of real-world file system workloads [8, 12, 18], where the write percentage ranged from 1.1% to 82.3%. These workloads came from an instructional (INS) and research (RES) setting [18], a campus home directory (EECS) [8], and CIFS corporate and engineering workloads (CIFS-corp, CIFS-eng) [12]. The RES and INS workloads are read-intensive, with the percentage of write transactions less than 10%. The CIFS workloads have twice as many reads as writes. The EECS workload has the highest write load, with more than 80% write transactions.

As shown in Figures 5(a) and 5(b), the read-intensive workloads have almost no provenance overhead, with less than 5% overhead for both RES and INS on ordinary disk and less than 2% with a RAM disk. Write-intensive workloads on ordinary disk incur higher overheads, but still less than 14% for CIFS and less than 36% for EECS. With a RAM disk, the overheads are less than 3% for CIFS and around 6.5% for EECS.

Related Work

Numerous research efforts have tackled the issues of collecting, storing, representing, annotating, and querying provenance data, but little has been done to secure that information [4, 10]. Researchers have categorized provenance systems for science [20] and explored the question of how to capture provenance information, typically relying on workflow instrumentation [2, 15]. Provenance management systems used by scientists include Chimera for physics and astronomy, myGrid for biology, CMCS for chemistry, and ESSW for earth science [20]. Other efforts propose to collect provenance information within databases [5, 22] social networks [9], and operating and file systems [16]; the latter offers the notable advantage of being hard to circumvent.

Researchers have proposed the use of *entanglement* to preserve distributed systems' history in a non-repudiable, tamper-evident manner [14]. Provenance-related information is supported by source code management systems such as SVN [6], GIT [13], CVS [3], and Monotone [1]; versioning file systems [17]; and *secure audit forensic logs* [19, 21], to provide integrity assurances for subsets of system and data state in a logically centralized authority model. Our work targets provenance information that is highly mobile and may traverse multiple untrusted domains, with no logically centralized repository or authority. Work on audit logs typically secures logs as a whole, but does not allow authentication of individual modifications. Because provenance information is associated with a digital object such as a file, this introduces attacks that are not applicable to secure audit logs. Finally, secure audit log schemes typically assume that at most a handful of parties will process the data and compute checksums, whereas multiple principals' access is required throughout the lifetime of a provenance chain.

Conclusion

Data provenance is growing in importance as more information is shared across organizational boundaries. In this article we introduced a cross-platform, low-overhead architecture for securing provenance information. We implemented our approach for tracking the provenance of *data writes*, in the form of a library that can be linked with any application. Experimental results show that our approach imposes overheads of only 1–13% on typical real-life workloads. Further details are available in our FAST '09 paper [11], and on our Web site (http://tinyurl.com/secprov).

ACKNOWLEDGMENTS

We thank Bill Bolosky and John Douceur for tips about file system distribution. Hasan and Winslett were supported by NSF awards CNS-0716532 and CNS-0803280. Sion was supported by Xerox, IBM, and the NSF through awards CNS-0627554, CNS-0716608, CNS-0708025, and IIS-0803197.

REFERENCES

[1] Monotone Distributed Version Control: http://www.monotone.ca/, accessed on December 22, 2008.

[2] R. Aldeco-Perez and L. Moreau, "Provenance-Based Auditing of Private Data Use," *Proceedings of the BCS International Academic Research Conference, Visions of Computer Science*, 2008.

[3] B. Berliner, "CVS II: Parallelizing Software Development," *Proceedings of the Winter 1990 USENIX Conference,* USENIX Association, 1990, pp. 341–352.

[4] U. Braun, A. Shinnar, and M. Seltzer, "Securing Provenance," *Proceedings of the 3rd USENIX Workshop on Hot Topics in Security (HotSec '08)*, USENIX Association, 2008.

[5] P. Buneman, A. Chapman, and J. Cheney, "Provenance Management in Curated Databases," *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, ACM Press, 2006, pp. 539–550.

[6] B. Collins-Sussman, "The Subversion Project: Building a Better CVS," *Linux Journal* 2002(94): 3.

[7] J.R. Douceur and W.J. Bolosky, "A Large-Scale Study of File-System Contents," *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS),* ACM Press, 1999, pp. 59–70.

[8] D. Ellard, J. Ledlie, P. Malkani, and M. Seltzer, "Passive NFS Tracing of Email and Research Workloads," *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, USENIX Association, 2003, pp. 203–216.

[9] J. Golbeck, "Combining Provenance with Trust in Social Networks for Semantic Web Content Filtering," *International Provenance and Annotation Workshop*, in volume 4145 of *Lecture Notes in Computer Science*, L. Moreau and I.T. Foster, eds., Springer, 2006, pp. 101–108.

[10] R. Hasan, R. Sion, and M. Winslett, "Introducing Secure Provenance: Problems and Challenges," *Proceedings of the ACM Workshop on Storage Security and Survivability (StorageSS)*, ACM Press, 2007, pp. 13–18.

[11] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance," *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST '09)*, USENIX Association, 2009.

[12] A.W. Leung, S. Pasupathy, G. Goodson, and E.L. Miller, "Measurement and Analysis of Large-Scale Network File System Workloads," *Proceedings of the 2008 USENIX Annual Technical Conference*, USENIX Association, 2008, pp. 213–226.

[13] J. Loeliger, "Collaborating Using GIT," *Linux Magazine*, June 2006.

[14] P. Maniatis and M. Baker, "Secure History Preservation through Timeline Entanglement," *Proceedings of the 11th USENIX Security Symposium*, USENIX Association, 2002, pp. 297–312.

[15] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga, "The Provenance of Electronic Data," *Communications of the ACM*, 51(4): 52–58 (2008).

[16] K.-K. Muniswamy-Reddy, D.A. Holland, U. Braun, and M.I. Seltzer. "Provenance-Aware Storage Systems," *Proceedings of the 2006 USENIX Annual Technical Conference*, USENIX Association, 2006, pp. 43–56.

[17] Z.N.J. Peterson, R. Burns, G. Ateniese, and S. Bono, "Design and Implementation of Verifiable Audit Trails for a Versioning File System," *Proceedings of the 5th USENIX conference on File and Storage Technologies (FAST '07)*, USENIX Association, 2007, pp. 93–106.

[18] D. Roselli, J.R. Lorch, and T.E. Anderson, "A Comparison of File System Workloads," *Proceedings of the 2000 USENIX Annual Technical Conference*, USENIX Association, 2000.

[19] B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics," *ACM Transactions on Information and System Security*, 2(2): 159–176 (1999).

[20] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in E-Science," *ACM SIGMOD Record*, 34(3): 31–36 (Sept. 2005).

[21] R. Snodgrass, S. Yao, and C. Collberg, "Tamper Detection in Audit Logs," *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB),* VLDB Endowment, 2004, pp. 504–515.

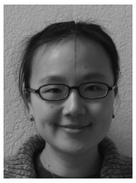
[22] J. Widom, "Trio: A System for Integrated Management of Data, Accuracy, and Lineage," *Proceedings of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR)*, January 2005.

ANDREW W. LEUNG, MINGLONG SHAO, TIMOTHY BISSON, SHANKAR PASUPATHY, AND ETHAN L. MILLER

Spyglass: metadata search for large-scale storage systems

Andrew W. Leung is a PhD student at the University of California, Santa Cruz. His advisor is Ethan L. Miller. He is currently researching new techniques for large-scale data management, with a particular emphasis on storage system search. He received his BS from the University of California, Santa Barbara.

aleung@cs.ucsc.edu



Minglong Shao is a member of the Advanced Technology Group at NetApp. She received a Ph.D. in Computer Science from Carnegie Mellon University in 2007. Her research interests are in the area of database and storage systems, with an emphasis on database system behavior on modern storage devices.

minglong@netapp.com

Tim Bisson is a software engineer at NetApp in Sunnyvale, California. He received his Ph.D. in computer science from the University of California, Santa Cruz, in 2007.

tbisson@netapp.com

Shankar Pasupathy is a member of NetApp's advanced technology group, where he leads the indexing project. He is involved in research related to indexing petabyte-scale storage, as well as building large content repositories.

Shankar.Pasupathy@netapp.org

Ethan Miller is a professor of computer science at the University of California, Santa Cruz, where he is the Associate Director of the Storage Systems Research Center. His research interests include archival storage systems, scalable metadata management, file system reliability and security, scalable distributed storage systems, and file systems for non-volatile memories.

elm@cs.ucsc.edu

ARE OUR GROWING STORAGE SYSTEMS a blessing or a curse? As storage systems expand to billions of files they become increasingly difficult to manage. Effective management requires quickly searching the metadata of the files being stored. Unfortunately, tools such as find and grep and off-the-shelf databases cannot easily scale to billions of files. Spyglass is a new approach to metadata search, which leverages file and query properties to improve performance and functionality and to allow people to focus on using rather than merely managing their data.

The Growing Data Problem

Our ever-growing storage systems have become an escalating problem for storage users and administrators. Increasing amounts of digital data coupled with decreasing storage costs have yielded systems that store petabytes of data and billions of files. Managing this rising sea of data is difficult because users and administrators need to efficiently answer a variety of questions about the files being stored. For example, a user may need files that he/she has forgotten the location for (e.g., "Where are my recently modified presentation files?"), or an administrator may need to reduce storage capacity (e.g., "Which files can be migrated to second-tier storage?"). These questions are extremely difficult to answer when one must sift through billions of files. Moreover, answering data management questions requires complex query functionality. Consider a scenario where a buggy script accidentally deletes a number of users' files. To fix the problem, an administrator would like to answer the question, "Which files should be restored from backup?" Restoring the entire backup takes enormous amounts of time and destroys any changes made since then. Ideally, an administrator would like to be able to search both the current and previous file versions, search for just those files affected by the script, and determine which critical files should be restored first.

Metadata search can greatly aid users and administrators in answering these questions. Metadata search allows ad hoc queries over file properties. These properties are structured *attribute,value* pairs and consist of metadata such as inode fields (e.g., size, owner, timestamps) and extended attributes (e.g., document title, retention policy, backup dates). Metadata search helps users and administrators understand the kinds of files being stored, where they are located, how they got there (provenance), and where they belong, and is a key step towards improving how we manage our data. Table 1 shows examples of some popular metadata search queries from a survey we conducted, which we detail later in the article.

File Management Question	Metadata Search Query
Which files can I migrate to tape?	size > 50 GB, atime > 6 months
How many duplicates of this file are in my home directory?	owner = john, datahash = 0xE431, path = /home/john
Where are my recently modified presentations?	owner = john, type = (ppt | keynote), mtime < 2 days
Which legal compliance files can be expired?	retention time = expired, mtime > 7 years
Which of my files grew the most in the past week?	Top 100 where size(today) > size (1 week ago), owner = john.
How much storage do these users and applications consume?	Sum size where owner = john, type = database

TABLE 1: USE-CASE EXAMPLES. METADATA SEARCH USE-CASES COLLECTED FROM OUR USER SURVEY. THE HIGH-LEVEL QUESTIONS BEING ADDRESSED ARE ON THE LEFT. ON THE RIGHT ARE THE METADATA ATTRIBUTES BEING SEARCHED AND EXAMPLE VALUES.

Metadata search is becoming increasingly popular and is common on desktop and small-scale enterprise storage systems. Most desktop file systems ship with a search tool that supports metadata search [1, 11]. Enterprise search appliances, such as FAST [4] and Google Enterprise [5], provide metadata search for relatively small-scale storage systems (e.g., up to tens of millions of files). These tools are becoming more prevalent; recent studies show that 37% of enterprise businesses already use such tools and that 40% plan to use them in the near future [6].

Unfortunately, it appears that we are still not ready to address metadata search at large scales. Searching storage systems with billions of files gives rise to a number of challenges that existing solutions do not address. First, cost and resources must be efficiently utilized. Existing enterprise-scale solutions address performance through dedicated CPU, memory, and disk hardware, which quickly becomes prohibitively expensive at large scales. It can cost tens of thousands of dollars just to search tens of millions of files [8]. An effective solution should be able to reside directly within the storage system, sharing resources without degrading search or native storage performance.

Second, there must be a way to quickly gather metadata changes. Large systems can produce millions of metadata changes per minute. However, existing solutions often gather metadata changes with a brute-force crawl of the storage system, which is not only prohibitively slow but also extremely resource-intensive. We have observed commercial systems take 22 hours to crawl 500GB and 10 days to crawl 10TB. Other approaches, such as intercepting file system requests, are often not possible because enterprise appliances are not integrated with the storage system.

Third, search and update performance must be highly scalable. It is very difficult to search billions of files in a timely manner. Poor search and update performance directly impacts overall effectiveness and usability. Existing solutions rely on basic off-the-shelf solutions—in most cases, general-purpose relational databases (DBMSes). While privy to decades of performance research, DBMSes are not designed or optimized for storage system search. As a result, they make few metadata search optimizations and have extraneous functionality that adds overhead. Our results show that a simple DBMS-based solution often requires many minutes and sometimes hours to search tens to hundreds of millions of files. The concept that a general-purpose DBMS should not serve as a "one size fits all" solution is not new and is actually touted by the DBMS community itself [12].

Re-Thinking Metadata Search

Addressing metadata search requires rethinking our approach to the problem, since we cannot rely on basic off-the-shelf solutions. As with any good system design, the key to improving performance and scalability is to understand and leverage metadata search properties. This requires understanding the properties of the files being searched and the queries being run. To do this we surveyed over 30 storage users and administrators from NetApp and the University of California, Santa Cruz, and analyzed metadata from three storage systems at NetApp. Armed with recent observations, we developed a new metadata search system, called Spyglass, which was presented at the 7th USENIX Conference on File and Storage Technologies (FAST '09) [9]. Spyglass introduces new approaches to index design, index updating, and metadata collection that leverage metadata search properties to improve performance and scalability.

We asked our survey participants, "What kinds of metadata queries would you like to perform?" Our survey yielded interesting insights, with some of the popular searches featured in Table 1. Queries almost always contain multiple metadata attributes, because querying a single attribute returns too many results to be useful. Also, queries can often be localized to only a part of the namespace, such as a home or project directory, by using a pathname in the query. Often participants had some idea where their data was located, and localizing queries helps to narrow the results. Finally, queries often involve "back-in-time" search of previous metadata versions to answer questions about how or when files have changed. Spyglass leverages this information by using a query execution method that utilizes all of the attributes in the query, embedding namespace information directly into the index, and versioning index updates.

The first storage system we analyzed served Web server data and stored about 15 million files. The second hosted engineering build space (source, object, and support files) and stored about 60 million files. The third stored employee home directories and contained about 300 million files. All three systems exhibit two important metadata characteristics: metadata values are highly clustered in the namespace, and the distribution of metadata values greatly changes when looking at a single metadata value versus a combination of values. Metadata clustering means that any particular metadata value, such as *owner,Andrew*, occurs only in a tiny fraction of directories (e.g., *Andrew*'s home directory) as opposed to being scattered across the namespace. In fact, most of the values we studied occurred in less than one-tenth of 1% of the directories! This is not a surprising result, really, since people use the namespace to group related files. Spyglass leverages this information by identifying just the few namespace locations that must be searched for in a query, thereby greatly reducing the scope of the search. Existing solutions do not utilize namespace information and must consider files from the entire storage system.

Our other observation is that the distributions of single metadata values differ greatly from those of multiple metadata values. Consider an example from our engineering storage system. As one might expect, there are many millions of *.c* source files, which comprise a large fraction of all file types on the system (we found that these distributions closely followed the power-law distribution). However, when we consider multiple attribute values at once, such as *file type,.c* and *owner,Andrew*, there are far fewer files with *both* values (we often found orders-of-magnitude fewer files). Spyglass leverages this knowledge by using *all* values when executing a query, which greatly reduces the number of files that must be considered. Existing DBMS-based solutions often rely on only a single attribute value when executing queries, and performance often suffers when distributions are skewed [10].

The Spyglass Design

Spyglass is designed to reside directly within the storage system and consists of two main components, shown in Figure 1: the Spyglass index, which stores metadata and serves queries, and a crawler that extracts metadata from the storage system.

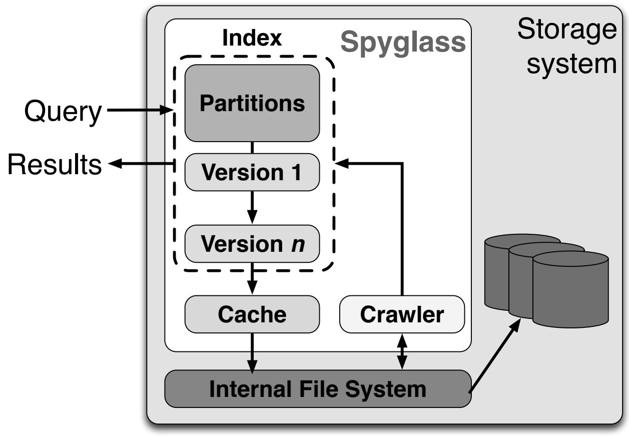


FIGURE 1: SPYGLASS OVERVIEW. SPYGLASS RESIDES WITHIN THE STORAGE SYSTEM. THE CRAWLER EXTRACTS FILE METADATA, WHICH GETS STORED IN THE INDEX. THE INDEX CONSISTS OF A NUMBER OF PARTITIONS AND VERSIONS, ALL OF WHICH ARE MANAGED BY A CACHING SYSTEM.

Spyglass introduces several new metadata search designs. First, *hierarchical partitioning* partitions the index based on the namespace, allowing the index to exploit metadata clustering and allowing fine-grained index control. Second, *signature files* [3] are used to automatically identify the partitions that are relevant to a query. Third, *partition versioning* versions index updates, which improves update performance and allows “back-in-time” metadata search. Finally, Spyglass utilizes snapshots in WAFL [7], the file system on which it was built, to collect metadata changes by crawling only files whose metadata has changed. In this article we briefly describe the design of the Spyglass index and how it uses hierarchical partitioning and signature files. A complete description of the design can be found in our FAST ’09 paper [9].

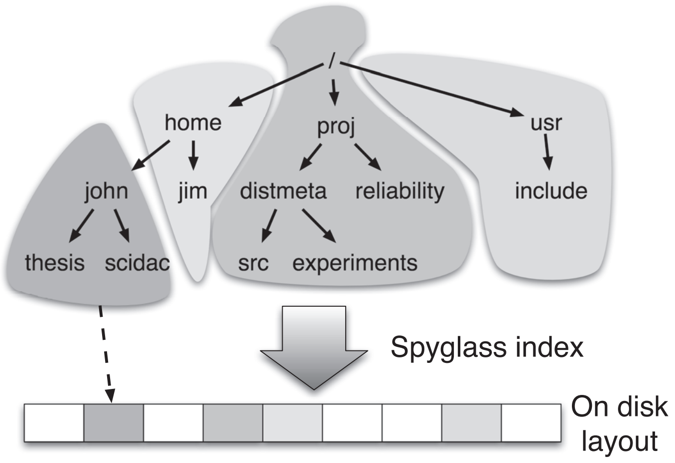


FIGURE 2: HIERARCHICAL PARTITIONING EXAMPLE. SUB-TREE PARTITIONS, SHOWN IN DIFFERENT COLORS, INDEX DIFFERENT STORAGE SYSTEM SUB-TREES. EACH PARTITION IS STORED SEQUENTIALLY ON DISK. THE SPYGLASS INDEX IS A TREE OF SUB-TREE PARTITIONS.

Hierarchical partitioning indexes files from separate parts of the namespace into separate partitions, providing flexible, fine-grained index control at the granularity of sub-trees. Figure 2 shows an example where files from separate sub-trees are indexed in separate partitions. The key idea is that we can now search only the parts of the storage system that are relevant to our query, without concerning ourselves with files from other parts of the system. Our finding that metadata values are highly clustered in the namespace indicates that in most cases we can search only a small fraction of the partitions. Spyglass keeps partitions relatively small (on the order of 100,000 files) and stores them sequentially on disk. This layout ensures very fast access to any one partition. Each partition indexes its metadata in a K-D tree [2].

Spyglass search performance is a function of the number of partitions that must be searched. Thus, when executing a query the question becomes, "Which partitions must be searched?" Spyglass can identify partitions in two ways. First, users can localize their queries (which we found to be common in our survey) and thereby control the number of partitions searched. Spyglass also uses signature files [3] to determine which partitions *may* have files relevant to a query. A signature file is a bit array with an associated hashing function that is a compact representation of a partition's contents. Each partition has a signature file for each attribute that it indexes, which are kept small so that they may fit in memory. Metadata values that are indexed in the partition have associated signature file bits set to one. If and only if *all* metadata values in a query map to one bits in the signature files is a partition read from disk and searched. Spyglass leverages the fact that there are far fewer files matching all query values than if only a single value is used. Since signature files are probabilistic, false positives can occur.

An evaluation of our Spyglass prototype shows significant performance improvements compared to a simple DBMS-based solution. Our evaluation was done using the real-world metadata we collected and sets of queries derived from our survey. We evaluated the performance of two popular DBMSes, PostgreSQL and MySQL, to serve as relative comparison points to DBMS-based solutions used in other metadata search systems. We found that Spyglass satisfies most queries in less than a second even as the system scales from 15 to 300 million files. Few queries took less than a second with our reference DBMSes. In fact, many queries require well over five minutes on the largest data set. A key reason for the performance difference is that Spyglass leverages hierarchical partitioning to greatly reduce the overall search space and often only requires a few small, sequential disk reads to answer a query. We also found that localizing a query to only a part of the namespace enhances performance to the point where Spyglass is up to four orders of magnitude faster than our reference DBMSes in some cases.

The Role of Search in the Storage System

Spyglass takes a first step towards improving how we manage our growing storage systems through new metadata search designs that leverage storage system properties. However, easily and effectively managing billions of files remains a daunting task. A key reason for this is that we must rethink the relationship between search and the file system. While becoming more common, file search (Spyglass included) remains an application that is often completely distinct from the file system. Yet search and file systems share a common goal: organizing and retrieving file data. Keeping search completely separate from the file system leads to duplicate functionality (e.g., each stores and maintains separate file index structures) and contention for resources (e.g.,a file requires memory and disk resources in both the file system index and search index). It's not hard to imagine the problems that will arise in the future when trying to index hundreds of billions of files in *both* the file system and a search index. Additionally, users are forced to interact with multiple interfaces (e.g.,the POSIX file system interface and the search interface), which needlessly complicates interactions with the storage system.

We believe improving the relationship between file search and the file system is key to improving how we manage our growing storage systems. We plan to explore this issue in our on-going Spyglass work. Our main goal is to explore how search can be integrated within the file system as first-class functionality. From an abstract level, Spyglass closely resembles a file system in which directory and file metadata are embedded together in partitions. Any file's metadata can be accessed in Spyglass by hierarchically traversing the partitions until the partition containing the file is reached, similar to a file system. The key differences from a file system are that in Spyglass files are grouped on disk by partitions rather than directories and that Spyglass versions each partition's file modifications (discussed in our FAST '09 paper) rather than perform in-place updates. Our on-going work looks at how an architecture like Spyglass may be used as the *main* metadata store for a storage system. Using an approach like Spyglass for all metadata storage means that only a single data structure, the Spyglass index, needs to be maintained, updated, cached, and searched. Doing so has the potential to improve overall performance by more efficiently utilizing resources and can greatly improve the functionality offered by the file system. The key research challenges will be achieving good performance for more general metadata workloads and achieving efficient real-time index updates.

ACKNOWLEDGMENTS

We would like to thank our colleagues in the Storage Systems Research Center and NetApp's Advanced Technology Group for their input and guidance. We also owe thanks to Remzi Arpaci-Dusseau, Stavros Harizopoulos, Jiri Schindler, and our FAST '09 shepherd, Sameer Ajmani, and the FAST '09 reviewers whose comments significantly improved this work.

This work was supported in part by the Department of Energy's Petascale Data Storage Institute under award DE-FC02-06ER25768 and by the National Science Foundation under award CCF-0621463. We thank the industrial affiliates of the SSRC for their support.

REFERENCES

[1] Apple, "Spotlight Server: Stop Searching, Start Finding": http://www.apple.com/server/macosx/features/spotlight.html, 2008.

[2] J.L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM,* 18(9): 509–517 (1975).

[3] C. Faloutsos and S. Christodoulakis, "Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation," *ACM Transactions on Information Systems,* 2(4): 267–288 (1984).

[4] FAST—Enterprise Search: http://www.fastsearch.com/, 2008.

[5] Google Enterprise: http://www.google.com/enterprise/, 2008.

[6] E.S. Groups, "ESG Research Report: Storage Resource Management on the Launch Pad," 2007.

[7] D. Hitz, J. Lau, and M. Malcom, "File System Design for an NFS File Server Appliance," *Proceedings of the Winter 1994 USENIX Technical Conference*, USENIX Association, 1994, pp. 235–246.

[8] Goebel Group, Inc., "Compare Search Appliance Tools": http://www.goebelgroup.com/sam.htm, 2008.

[9] A. Leung, M. Shao, T. Bisson, S. Pasupathy, and E.L. Miller, "Spyglass: Fast, Scalable Metadata Search for Large-Scale Storage Systems," *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST '09)*, USENIX Association, 2009, pp. 153–166.

[10] C.A. Lynch, "Selectivity Estimation and Query Optimization in Large Databases with Highly Skewed Distribution of Column Values," *Proceedings of the 14th Conference on Very Large Databases (VLDB)*, 1988, pp. 240–251.

[11] Microsoft Windows Search 4.0: http://www.desktop.google.com/, 2007.

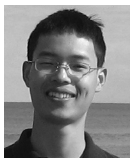
[12] M. Stonebraker and U. Cetintemel, "'One Size Fits All': An Idea Whose Time Has Come and Gone," *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, 2005, pp. 2–11.

WEIHANG JIANG, CHONGFENG HU, SHANKAR PASUPATHY, ARKADY KANEVSKY, ZHENMIN LI, AND YUANYUAN ZHOU

storage system problem troubleshooting and system logs

Weihang Jiang is a senior research engineer at Pattern Insight. He received a Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign.

Weihang.jiang@gmail.com



Chongfeng Hu did his graduate study at the University of Illinois at Urbana-Champaign. He is interested in operating systems and software reliability. He is now a software engineer at Microsoft.

chu7@cs.uiuc.edu

Shankar is a member of NetApp's advanced technology group, where he leads the indexing project. He is involved in research related to indexing petabyte-scale storage, as well as building large content repositories.

Shankar.Pasupathy@netapp.org

Arkady Kanevsky is a senior research engineer at NetApp Advanced Technology Group. Arkady has done extensive research on RDMA technology, storage resiliency, scalable storage systems, and parallel and distributed computing. He received a Ph.D. in computer science from the University of Illinois in 1987. He was a faculty member at Dartmouth College and Texas A&M University prior to joining the industry world. Arkady has written or co-authored over 60 publications and is a chair of DAT Collaborative and MPI-RT standards.

arkady@netapp.com

Yuanyuan Zhou is an associate professor at the University of Illinois, Urbana-Champaign. Her research spans the areas of operating system, storage systems, software reliability, and power management.

yyzhou@uiuc.edu

CUSTOMER PROBLEM TROUBLEshooting has been a critically important issue for both customers and system providers. A recent study indicates that problem-diagnosis-related activity is 36–43% of TCO (total cost of ownership) in terms of support costs [4]. Additionally, downtime can cost a customer 18–35% of TCO [4]. The system vendor pays a price as well. A survey showed that vendors devote more than 8% of total revenue and 15% of total employee costs on technical support for customers [10]. In this article, we explain how our FAST '09 paper made two major contributions to better understanding how logs pertain to solving problems.

We provided a characteristic study of customer problem troubleshooting using a large set (636,108) of real-world customer cases reported from 100,000 commercially deployed storage systems in the past two years. We studied the characteristics of customer problem troubleshooting from various dimensions, as well as correlation among them. Our results show that while some failures are either benign or resolved automatically, many others can take hours or days of manual diagnosis to fix. For modern storage systems, hardware failures and misconfigurations dominate customer cases, but software failures take a longer time to resolve. Interestingly, a relatively significant percentage of cases occur because customers lack sufficient knowledge about the system. We observed that customer problem reports with attached system logs are invariably resolved much faster than those without logs.

We also evaluated the potential of using storage system logs to resolve these problems. Our analysis shows that a failure message alone is a poor indicator of root cause, and that combining failure messages with multiple log events can improve low-level root cause prediction by a factor of three. We then discuss the challenges in log analysis and possible solutions.

Data Selection

We used two primary databases in selecting customer case data for analysis in our research: a *Customer Support Database*, which contains details on every customer case that was human-generated or auto-generated, and an *Engineering Case Database* for problems that cannot be resolved by customer support staff, which are escalated to engineering teams.

We analyzed 636,108 NetApp customer cases from the Customer Support Database over the period 1/1/2006 to 1/1/2008. Of these, 329,484 were human-generated and 306,624 were auto-generated. Overall, these represent about 100,000 storage systems.

For each of these 636,108 customer cases, *problem category* and *resolution time* were retrieved from the Customer Support Database. For each of the 306,624 auto-generated customer cases, we also retrieved the critical event that led to the creation of the case. However, the human-generated cases do not include such information.

The goal for resolving any customer case is to determine the problem root case as soon as possible. Since such information in the Customer Support Database is unstructured, it was difficult to identify problem root cause for solved cases. However, the Engineering Case Database includes the problem root cause at a fine level. We used 4,769 such cases that were present in both the Customer Support and Engineering Case databases to analyze problem root cause and its correlation with critical events.

To study the correlation between problem root cause and storage system logs, we retrieved the AutoSupport logs from the NetApp AutoSupport Database. Since not all customer systems send AutoSupport logs to NetApp, 4,535 out of 4,769 customer cases have corresponding AutoSupport log information.

Problem Resolution

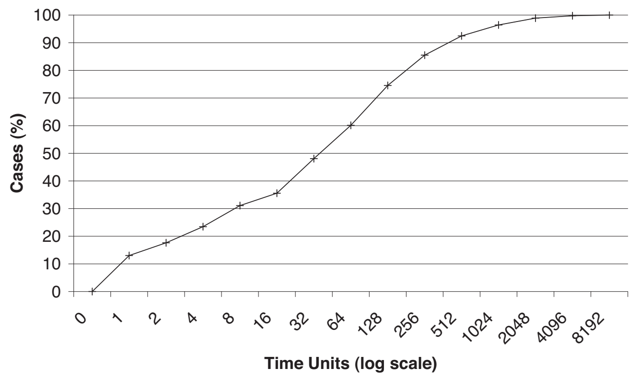


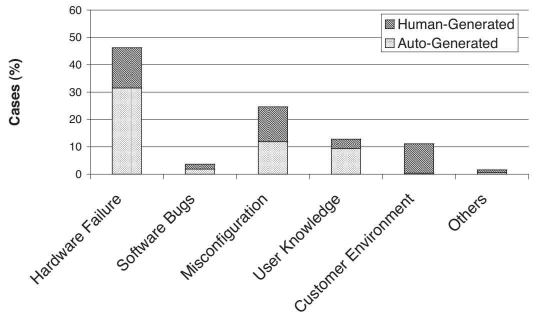
FIGURE 1: CUMULATIVE DISTRIBUTION FUNCTION (CDF) OF RESOLUTION TIME FOR ALL CUSTOMER CASES. WE ANONYMIZE RESULTS TO PRESERVE CONFIDENTIALITY.

One of the most important metrics of customer support is problem resolution time, the time between when a case is opened and when the resolution or a workaround is available for a customer. The distribution of problem resolution times is the key to understanding the complexity of a specific problem or problem class, since it mostly reflects the amount of time spent on troubleshooting problems. This time should not be directly used to calculate MTTR (Mean Time To Recovery), since it does not capture the amount of time to completely solve the problems (e.g., for hardware-related problems, it does not include hardware replacement.)

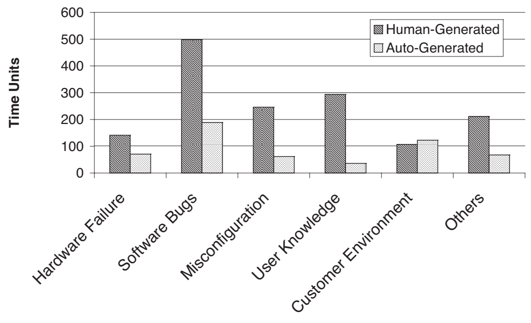
Figure 1 shows the Cumulative Distribution Function (CDF) of resolution time for all customer cases selected from the Customer Support Database. Troubleshooting can take many hours. For a small fraction of cases, resolution time can be even longer. Since the x-axis of the figure is logarithmic, the graph shows that doubling the amount of time spent on problem resolution does not double the number of cases resolved. While the AutoSupport logging system is an important step in helping troubleshoot problems, this figure makes the case that better tools and techniques are needed to reduce problem resolution time.

PROBLEM ROOT CAUSE CATEGORIES

Analyzing the distribution of problem root causes is useful in understanding where one should spend effort when troubleshooting customer cases or designing more robust systems. Although a problem root cause is precise, such as a SCSI bus failure, in this section we lump root causes into categories such as hardware, software, misconfiguration, etc. For all the customer cases, we study resolution time for each category, relative frequency of cases in each category, and the cost, which is the average resolution time multiplied by the number of cases for that category.



(A) CATEGORIZATION OF PROBLEM ROOT CAUSES



(B) AVERAGE RESOLUTION TIME PER PROBLEM ROOT CAUSE CATEGORY

FIGURE 2: PROBLEM ROOT CAUSE CATEGORIES AND TIME TO RESOLUTION.

In Figure 2, *Hardware Failure* is related to problems with hardware components, such as disk drives or cables. *Software Bug* is related to storage system software, and *Misconfiguration* to system problems caused by errors in configuration. *User Knowledge* concerns technical questions, e.g., explaining why customers were seeing certain system behaviors. *Customer Environment* involves problems not caused by the storage system itself. Figure 2(a) shows that hardware failure and misconfiguration are the two most frequent problem root cause categories, contributing, respectively, 47% and 25% to all customer cases. Software bugs account for a small fraction (3%) of cases. We speculate that the reason software bugs are relatively uncommon is that software undergoes rigorous testing before being shipped to customers. Besides tests, there are many techniques [2, 6, 7, 8] that can be applied to find bugs in software. Figure 2(b) shows that software bugs take a longer time to resolve on average, but since their number is so small, their overall impact on total time spent on all problem resolutions is not very high, as Figure 3 demonstrates.

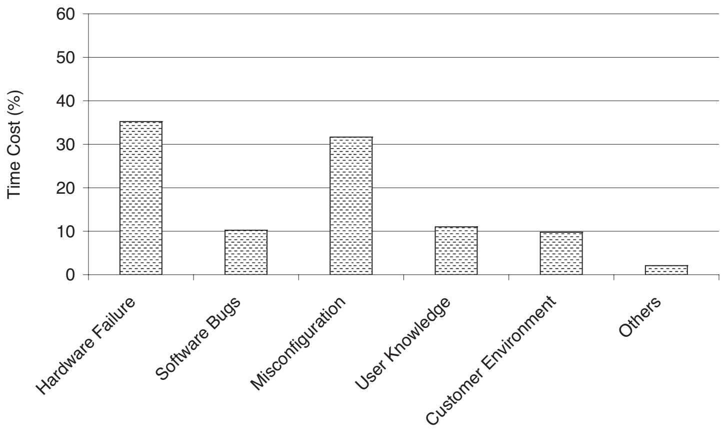


FIGURE 3: RESOLUTION TIME SPENT ON PROBLEM ROOT CAUSE BY CATEGORY. ALTHOUGH SOFTWARE PROBLEMS TAKE LONGER TO RESOLVE ON AVERAGE, HARDWARE FAILURE AND MISCONFIGURATION-RELATED PROBLEMS HAVE A GREATER IMPACT ON CUSTOMER EXPERIENCE.

It is interesting to observe that a relatively significant percentage of customer problems are because customers lack sufficient knowledge about the system (11%) or customers' own execution environments are incorrect (9%) (e.g., a backup failure caused by a Domain Name System error). These problems can potentially be reduced by providing more system-training programs or better configuration checkers.

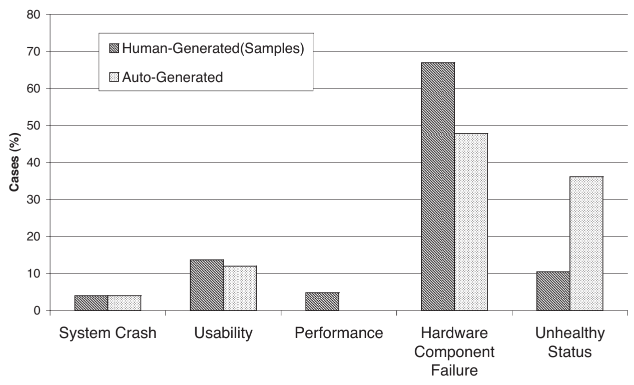
Figure 2(b) is our first indication that logs are indeed useful in reducing problem resolution time. Auto-generated customer cases, i.e., those with an attached system log and problem symptom in the form of a critical event message, take less time to resolve than human-generated cases. The latter are often poorly defined over the phone or by email. The only instance where this is not true is when the problem relates to the customer's environment, which is difficult to record via an automated system.

PROBLEM IMPACT

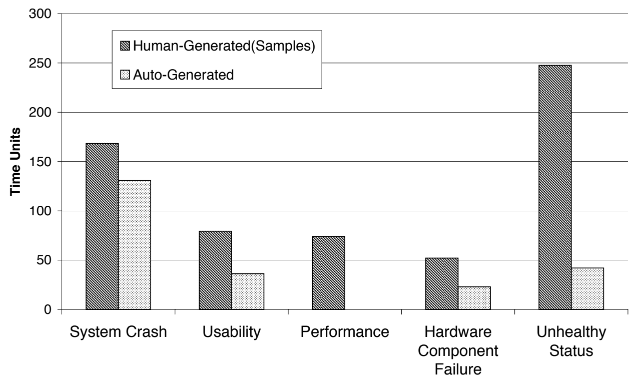
In the previous subsections, we have treated all problems as equal in their impact on customers. We now consider customer impact for each problem category. To do this, we divide customer cases into six categories based on impacts ranging from *system crash*, which is the most serious, to the low-impact *unhealthy* status. "System crash" here means crash of a single system, which might not lead to service downtime with a cluster configuration. The other categories, from higher to lower impact, are *usability* (e.g., inability to access a volume), *performance*, *hardware component failure*, and *unhealthy status* (e.g., instability of the interconnects, low spare disk count). Hardware failures typically have low impact, since the storage systems are designed to tolerate multiple disk failures [3], power-supply failures, filer-head failures, etc. However, until the failed component is replaced, the system operates in degraded mode where the potential for complete system failure exists, should its redundant component fail.

Since human-generated customer cases do not have all impact information in structured format, we randomly sampled 200 human-generated cases and manually analyzed them. For auto-generated problems, we include all the cases and leverage the information in Customer Support Database.

For both human-generated and auto-generated cases, the classification is exclusive: each problem case is classified to one and only one category. The classification is based on how a problem impacts customers' experience. For example, a disk failure that led to a system panic will be classified as an instance of *system crash*. If it did not lead to system crash (i.e., RAID handled it), it is classified as an instance of *hardware component failure*. It is important to notice that, in our study, the *performance* problems are problem cases that lead to unexpected performance slowdown. Therefore disk failures leading to expected slowdown with RAID reconstruction processes are classified as *hardware component failures* instead of *performance* problems.



(A) DISTRIBUTION OF PROBLEMS WITH DIFFERENT IMPACT



(B) AVERAGE RESOLUTION TIME OF PROBLEM WITH DIFFERENT IMPACT

FIGURE 4: PROBLEM IMPACT.

Figure 4(a) shows the distribution of problems by impact. One obvious observation is that there are far fewer high-impact problems than low-impact ones. More specifically, *system crash* only contributes about 3%, and *usability* problems contribute about 10%. Low-impact problems such as *hardware component failure* and *unhealthy status* contribute about 57% and 23%, respectively.

While high-impact problems are much fewer, as Figure 4(b) shows, they are more time-consuming to troubleshoot. This is due to the complex interaction between system modules.

Finally, as we observed in the previous section, auto-generated cases take less time to resolve than human-generated ones.

Feasibility of Using Logs for Automating Troubleshooting

We investigated the feasibility of using additional information from system logs and answered the following two questions: Does problem root cause determination improve by considering log events beyond critical events? What kinds of log events are key to identifying the problem root cause?

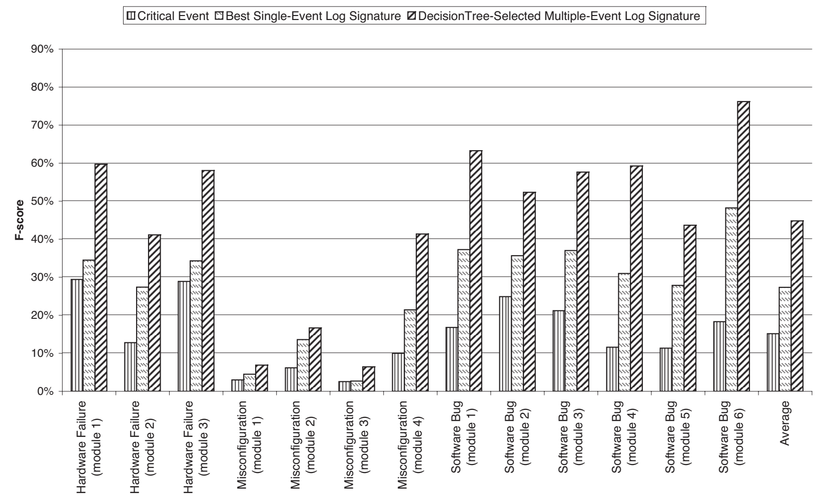


FIGURE 5: COMPARISON AMONG THREE METHODS OF USING LOG EVENTS. F-SCORE INDICATES HOW ACCURATE A PREDICTION CAN BE MADE ON A MODULE-LEVEL PROBLEM ROOT CAUSE USING LOG INFORMATION.

ARE ADDITIONAL LOG EVENTS USEFUL?

To study whether additional log events are useful, we considered three methods of using log event information and compared how well they can be used as a module-level problem root cause signature. We defined a signature as a set of relevant log events that uniquely identify a problem root cause. Such a signature can be used to identify recurring problems and to distinguish one problem from another unrelated one, thereby helping with customer troubleshooting. It is important to note that we are not designing algorithms to find log signatures; instead, we are manually computing log signatures to study how they improve problem root cause determination.

As a baseline, our first method is to use only the problem's critical event as its signature. The second method is similar to method one, but instead of just looking at critical events to deduce a root cause signature, we search all log events looking for the one log message that best indicated the module-level root cause. The third method is to use a decision tree [1] to find the best mapping between multiple log events and the problem root cause. The resulting multiple log events can be used as the root cause signature. For each module-level problem root cause, *F-score* is used to measure how well the signature can predict the problem root cause [9]. For more details about the methodology, refer to our conference paper [5].

As Figure 5 shows, for all customer cases, using only critical events as the problem signature is a very poor predictor of root cause. On average, it only achieves an *F-score* of about 15%. Using the best-matched log event, instead of just critical events, can achieve an *F-score* of 27%. By comparison, the average *F-score* achieved by the decision tree method for computing problem signatures is 45%, 3x better than using critical events. Based on these results, we conclude that accurate problem root cause determination requires combining multiple log events rather than examining a single log event or critical event. This observation matters, since customer support personnel usually focus on the critical event, which can be misleading. Furthermore, as we show in the next section, there is often a lot of noise between key log events, making it hard to manually detect problem signatures.

Although we use the decision tree to construct log signatures that are composed of multiple log events, we do not advocate this technique as the solution for utilizing log information. First of all, the accuracy (F-score) is still not satisfactory, due to log noise, which we discuss later. Moreover, the effectiveness of the decision tree relies on training data. For problem root causes that do not have a large number of diagnosed instances, a decision tree will not provide much help.

CHALLENGES OF USING LOG INFORMATION

To understand the challenges of using log information and identifying key log events to compute a problem signature, we manually analyzed 35 customer cases sampled from the Engineering Case Database. These customer cases were categorized into 10 groups, such that cases in each group had the same problem root cause.

For these customer cases, we noticed that engineers used several key log events to diagnose the root cause. Table 1 summarizes these cases and characteristics of their key log events.

Symptom	Cause	# of Key Log Events	Distance (secs)	Distance (# events)	Fuzziness?
Battery low	Software bug	2	5.8	1.6	No
Shelf fault	Shelf intraconnect defect	3	49.4	3.8	Yes
System panic	Broken scsi bus bridge	4	509.2	34.4	No
Performance degradation	Fc loop defect	2	3652	69.4	No
Power warning	Incorrect threshold in code	2	5	2.4	Yes
Raid volume failure	Software bug	3	196	66.5	No
Raid volume failure	Non-zeroed disk insertion	3	80	35	Yes
Raid volume failure	Interconnect failure	3	290.5	126	Yes
Shelf fault	Shelf module firmware bug	4	18285.5	21.5	No
Shelf fault	Power supply failure	3	31.5	3.5	No

TABLE 1: CHARACTERISTICS OF LOG SIGNATURES. WE MANUALLY STUDIED 35 CUSTOMER CASES AND PLACED THEM INTO 10 GROUPS, WHERE THE CASES IN EACH GROUP HAD THE SAME PROBLEM ROOT CAUSE. BASED ON DIAGNOSIS NOTES FROM ENGINEERS, WE WERE ABLE TO IDENTIFY THE KEY LOG EVENTS, WHICH CAN DIFFERENTIATE CASES IN ONE GROUP FROM CASES IN ANOTHER. "# OF KEY LOG EVENTS" IS THE TOTAL NUMBER OF IMPORTANT LOG EVENTS (INCLUDING CRITICAL EVENTS) NEEDED TO IDENTIFY THE PROBLEM. "DISTANCE" IS CALCULATED AS THE LONGEST DISTANCE FROM A KEY LOG EVENT TO A CRITICAL EVENT FOR EACH CUSTOMER CASE, AVERAGED ACROSS ALL CASES.

Based on these 10 groups, we made the following major observations:

(1) Logs are noisy.

Figure 6 shows the Cumulative Distribution Function (CDF) of the number of log events in AutoSupport logs corresponding to customer cases. As can be seen in the figure, for a majority of the customer cases (75%), there are more than 100 log events recorded within an hour before the critical event occurred, and for the top 20% of customer cases, more than 1000 log events were recorded.

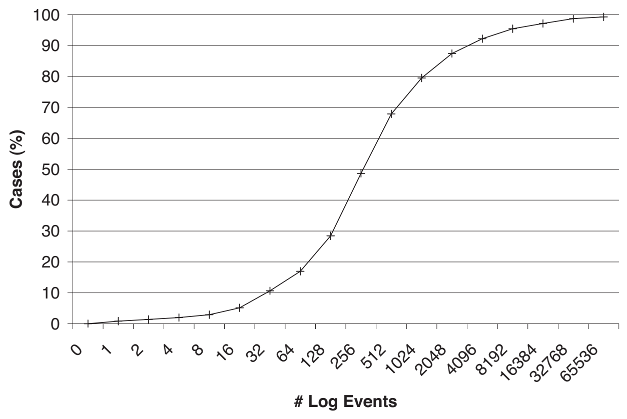


FIGURE 6: CUMULATIVE DISTRIBUTION FUNCTION (CDF) OF NUMBER OF LOG EVENTS WITHIN ONE HOUR OF THE CRITICAL EVENT. FOR THIS FIGURE, WE USE THE SAME DATA SET AS IN FIGURE 5. WE ONLY COUNT THE LOG EVENTS GENERATED AND RECORDED BY THE AUTOSUPPORT SYSTEM WITHIN ONE HOUR BEFORE THE CRITICAL EVENT, SINCE ENGINEERS OFTEN ONLY EXAMINE RECENT LOG EVENTS FOR PROBLEM DIAGNOSIS.

In comparison, as Table 1 shows, there are usually only 2–4 key log events for a given problem, implying that most log events are just noise for the problem.

(2) Important log events are not easy to locate.

Table 1 shows the distance between key log events and critical events, both in terms of time and of the number of log events. For 5 out of 10 problems, at least one key log event is more than 30 log events away from the critical event, which captures the failure point. For all problems, there are always some irrelevant log events between the key log events and the critical event. In terms of time, the key log events can be minutes or even hours before the critical event.

(3) The pattern of key log events can be fuzzy.

Sometimes it is not necessary to have an exact set of key log events to identify a particular problem. For example, in Table 1's problem 7, it is not necessary to see the "raidDiskInsert" log event, depending on how the system administrator added the disk drive. In problem 2 the same shelf intraconnect error can be detected by different modules, and different log messages can be generated for it depending on which module reports the issue.

Conclusion

In this article we present a study of the characteristics of customer problem troubleshooting from logs, using a large set of customer support cases from NetApp. Our results show that customer problem troubleshooting is a very time-consuming and challenging task that can benefit from automation to speed up resolution time. We observed that customer problems with attached logs were invariably resolved sooner than those without logs. We show that while a single or critical log event is a poor predictor of problem root cause, combining multiple key log events leads to a threefold improvement in root-cause determination. Our results also show that logs are challenging to analyze manually, because they are noisy, and that key log events are often separated by hundreds of unrelated log messages. Please refer to our conference paper [5] for our ideas for an automatic log analysis tool that can speed up problem resolution time.

As with similar studies, it was impossible to study a handful of different data sets, especially for customer support problems, due to the unavailability of such data sets. Even though our data set (which is already very large with 636,108 cases from 100,000 systems) is limited to NetApp, we believe that this study is an important first step in quantifying both the usefulness of and challenge in using logs for customer problem troubleshooting. We hope that our study can motivate characteristic studies about other kinds of systems as well and inspire the creation of new tools for automated log analysis for customer problem troubleshooting.

REFERENCES

[1] Leo Breiman, J.H. Friedman, R.A. Olshen, and C. Stone, *Classification and Regression Trees*, Statistics/Probability Series (Belmont, California: Wadsworth Publishing Company, 1984).

[2] Ben Chelf and Andy Chou, “The Next Generation of Static Analysis: Boolean Satisfiability and Path Simulation—A Perfect Match,” *Coverity White Paper*, 2007.

[3] Peter Corbett, Bob English, Atul Goel, Tomislav Grcanac, Steven Kleiman, James Leong, and Sunitha Sankar, “Row-Diagonal Parity for Double Disk Failure Correction,” *Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST ’04)*, USENIX Association, 2004, pp. 1–14.

[4] Crimson Consulting Group, “The Solaris 10 Advantage: Understanding the Real Cost of Ownership of Read Hat Enterprise Linux,” *Crimson Consulting Group Business White Paper*, 2007.

[5] Weihang Jiang, Chongfeng Hu, Shankar Pasupathy, Arkady Kanevsky, Zhenmin Li, and Yuanyuan Zhou, “Understanding Customer Problem Troubleshooting From Storage System Logs,” *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST ’09)*, USENIX Association, 2009.

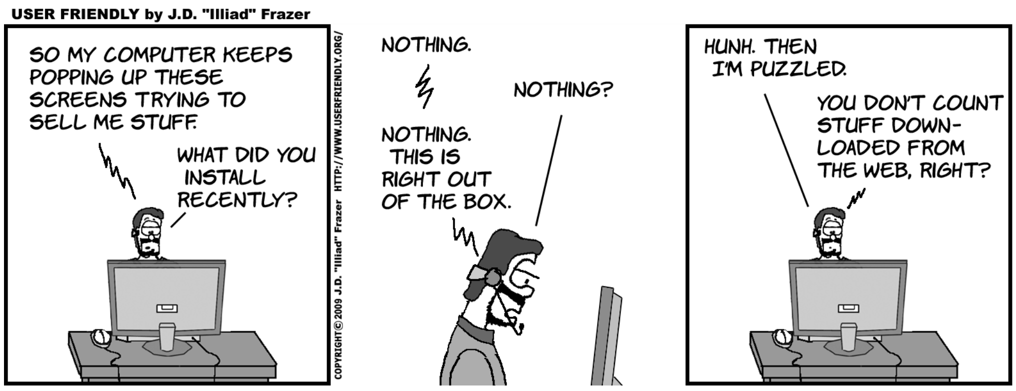
[6] S. Johnson, “Lint, a C Program Checker,” Computer Science Technical Report 65, Bell Laboratories, December 1977.

[7] Zhenmin Li, Shan Lu, Suvda Myagmar, and Yuanyuan Zhou, “Cp-Miner: A Tool for Finding Copy-Paste and Related Bugs in Operating System Code,” *Proceedings of the 6th USENIX Conference on Symposium on Operating Systems Design and Implementation (OSDI ’04)*, USENIX Association, 2004.

[8] Zhenmin Li and Yuanyuan Zhou, “Pr-Miner: Automatically Extracting Implicit Programming Rules and Detecting Violations in Large Software Code,” *ESEC/FSE-13: Proceedings of the 10th European Software Engineering Conference Held Jointly with the 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2005, pp. 306-315.

[9] C.J. Van Rijsbergen, *Information Retrieval* (Newton, MA: Butterworth-Heinemann), 1979.

[10] Association of Support Professionals, "Technical Support Cost Ratios," 2000.


USER FRIENDLY by J.D. "Illiad" Frazer
SO MY COMPUTER KEEPS POPPING UP THESE SCREENS TRYING TO SELL ME STUFF.
WHAT DID YOU INSTALL RECENTLY?
COPYRIGHT © 2009 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
NOTHING.
NOTHING?
NOTHING. THIS IS RIGHT OUT OF THE BOX.
HUNH. THEN I'M PUZZLED.
YOU DON'T COUNT STUFF DOWN-LOADED FROM THE WEB, RIGHT?

RUDI VAN DRUNEN

signals

Rudi van Drunen is a senior UNIX systems consultant with Competa IT B.V. in The Netherlands. He also has his own consulting company, Xlexit Technology, doing low-level hardware-oriented jobs.

rudi-usenix@xlexit.com

WHEN TRANSMITTING INFORMATION, we do not need to transmit power (covered in my previous article) but information and data expressed with signals. In this article I explain how data gets encoded, sources of noise, and various solutions. We will primarily examine Ethernet as an example.

Encoding Information

In the digital world we distinguish two discrete values a signal can have, 0 or 1, and these values correspond to certain voltages. The mapping between the voltage on a point and the digital value it represents is determined by the electronics family the circuit is built in and the supply voltage [1]. This chart gives the HIGH (often "1") and LOW (often "0") values and thresholds for most logic families. We can learn from this that the lower the supply voltage is, the lower the voltage margin between a HIGH and a LOW signal is.

The larger the (voltage) margin between the "1" and "0" signal, the less susceptible the system is to external noise, but the more power is needed in the driver electronics to drive the circuit. There is a trend to lower the voltages in systems to be more energy (and heat) efficient, but this will make the system less immune to outside factors.

For external interfaces the levels can be completely different. For example, the serial RS 232 interface is originally specified with a "0" value corresponding to 12 V and a "1" value corresponding to –12 V. Nowadays most systems are able to receive the 12 .. –12 V levels, but send only 5 and 0 V levels.

Information can also be encoded in the current, as used in RS 422 or MIDI interfaces. Here, when a current is flowing a "1" is represented, whereas no current represents a "0" value.

Analog Versus Digital

With simple analog signals the information is encoded in the value (mostly the voltage) at a given time. More complex signals (often modulated signals) can have different encodings. For example, with the FM radio the information is encoded in the frequency of the analog signal. With analog TV the color and intensity are complexly encoded in phase differences. Another encoding example can be found in process control systems, where a current loop interface is commonly used. This interface uses current (4 to 20 mA) to encode the (analog) sensor value. An advantage of current loop interfaces is that they can operate over moderately long distances (>10 miles).

Here, note the difference between the current loop interfaces in the analog and digital worlds. The analog interface uses the amount of current to show the value; the digital interface uses current or no current to show the two values.

Conversion

Conversion from analog to digital is done using an analog to digital converter (ADC). These components or circuits have an analog voltage input and some way to output the signal digitally. We find these components, for example, in your sound input circuitry or the environmental sensor system.

The other way around, converting digital to analog, is done by a digital to analog converter (DAC), as found in the sound output or your (analog) video (VGA) output.

Transmission

As we transmit an analog value using a conductor that (continuously) carries a certain voltage to ground, to transmit the very same amount of digital information we might need *more than one* conductor, each representing a bit. The number of bits we need is determined by the accuracy and resolution we need to express the value. These bits can be transmitted either serially or in parallel, where you need as many signal lines as bits you want to transmit.

If you need to transport temperature information, for example, 70 def F (with a resolution of 1 deg F), from a sensor to a computer system, you can encode this in analog by sending a voltage of 70 mV through an analog channel (using a ground and a signal wire) or digitally by using 8 bits, which gives you a range of values from 0 to 255 (= $2^8 - 1$). (You can do with less; the number of bits you need is determined by the range of the information you want to send, the resolution, and the encoding.) The value of the 8-bit word determines the temperature value.

To do this in parallel you need eight signal lines carrying 0100 0110 (the actual voltage on the lines is not relevant) or just one line where you send a bit, say, each second, by putting the appropriate voltage on the signal line (see also Figure 1).

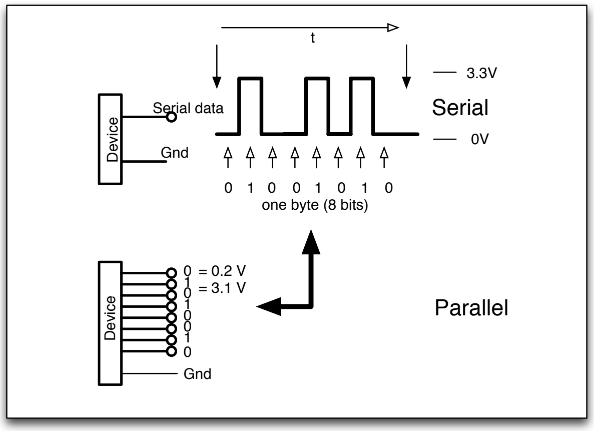


FIGURE 1: SERIAL VS PARALLEL

In general, analog signals are more susceptible to external interference, such as noise or hum, as the voltage directly relates to the value. With digital signals we have defined a noise margin. Even though the absolute value of the voltage is not relevant, it should be within boundaries defined by the logic family or standard for that interface. If you transmit serially, the time factor comes into play; if in the transmission the timing of the signal changes, the read value might change.

In the previous example this might be the case if, due to the nature of the transmission channel, a “0” takes longer to transmit than a “1.”

Outside Influence

All digital signals with a serial nature or changing parallel values generate noise. As the value of the signal changes rapidly (from a logic 1 to a logic 0, or the other way around), the cable carrying this signal generates noise in the electric and magnetic spectrum.

A source of interference is called Radio Frequency (RF) noise. We experience this if we put a radio receiver next to a piece of computing equipment. You then can “tune in” to a number of different sounds that are generated from the computing equipment. In the computer we find all kinds of signals that have a square wave nature.

The Fourier series tells us that we can see a square wave as the sum of a number of sine waves (harmonics) all of different frequencies. For example, a square wave with a frequency of 1 MHz can be described by adding up sine waves of 1, 3, 5, 7, 9 . . . MHz with different amplitudes. So, the square wave voltage over time can be described as:

$$U(t) = \frac{4}{\varphi}(\sin(2\varphi ft) + \frac{1}{3}\sin(6\varphi ft) + \frac{1}{5}\sin(10\varphi ft) + ...)$$

As we use radials, the factor 2π is used in this expression.

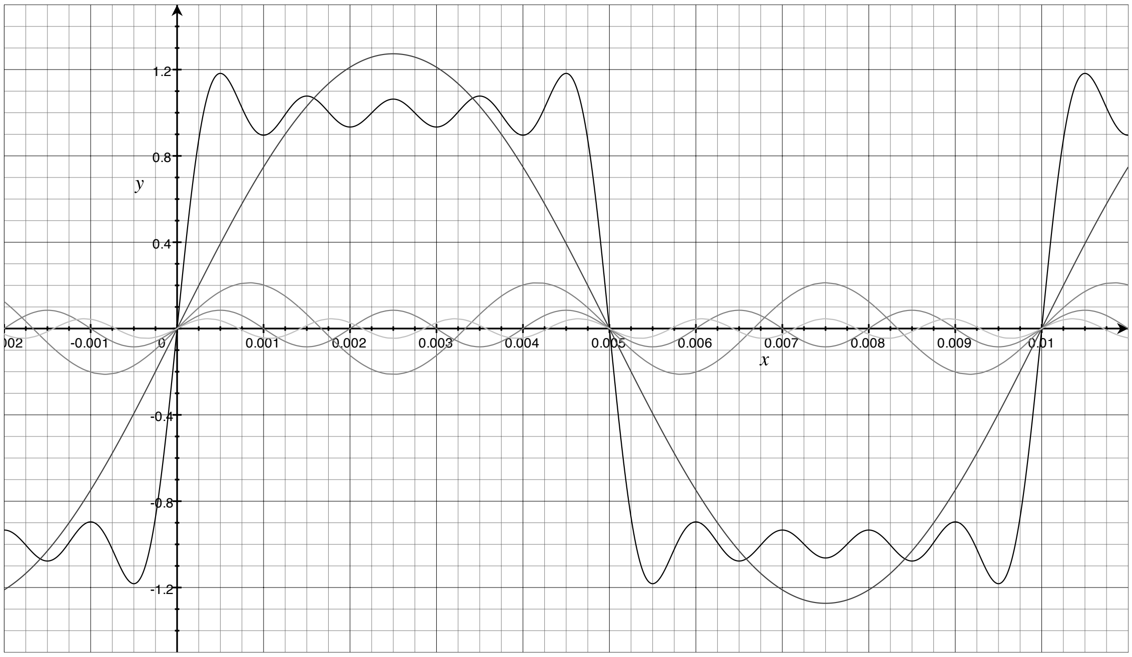


FIGURE 2: BUILDING A SQUARE WAVE FROM SINE WAVES

Figure 2 (preceding page) shows a square wave built from different sine waves. In our example, with a square wave of 1 MHz you would hear signals on your radio receiver at 1, 3, 5 . . . MHz when tuned in.

The higher-frequency harmonics are decreasing in amplitude. Compared to the ground frequency (the frequency of the square wave), the third harmonic is 1/3 of the ground wave amplitude, the 5th harmonic is 1/5 of the amplitude.

The FCC (or in Europe, the ETSI), which regulates the radio spectrum, tests all equipment in order to be sure that the (RF) radiation that originates from this equipment is within the boundaries of the legislation that prevents interference.

An example of this interference is Gigabit Ethernet, which is notorious for polluting the RF spectrum. We can lessen the RF noise generated by Gigabit interfaces and cabling (1000 BASE-T) by using shielded patch cables. Another way to reduce RF pollution by Gigabit Ethernet is to use fiber connections (1000 BASE-X) where possible.

The RF noise signal will, for example, prevent your GPS (or long wave) antenna for your time server from working and distort your cordless phone or your paging system when these are close to a non-shielded network cabinet.

Not only does equipment itself generate RF radiation, but the cabling used generates or can pick up a considerable amount of RF noise, and also works as an antenna amplifying this noise. To minimize this effect, new (signal) cables often have a ferrite coil integrated into them. You can see that as a big lump in the cable. The ferrite coil works as a choke to prevent part of the RF noise from radiating out of the cable by virtually creating a barrier for high-frequency signals.

Preventing the noise from getting out of the system or cable is one thing, but there is still so much ambient noise getting into (or out of) our systems and cabling that we need measures to prevent it from disturbing electronics and communication channels. There are a number of ways to overcome the influence of external factors in transmission channels. These techniques are applied in various interface standards in our digital world as well. We will discuss these here.

Noise

When you need to transport a digital signal over a long distance in a harsh environment, you want to prevent the signal from becoming too noisy. Therefore you want to minimize the influence of external factors, such as fluctuating magnetic or electric fields, which cause voltage noise to be inducted in the conductors of the cable. If, for example, you run a signal cable parallel to a power cable, you will see when there is a current flowing through the power cable for there will be a hum caused by an induction on the signal cable, which may distort the signal. In the analog world, the signal might be coming from a sensor (maybe your environmental monitoring sensors); in the digital world, it can be a systems interconnect of some sort.

Mobile phones are another example of noise generated by external (electric/magnetic) fields. If the transmitter of a mobile phone is communicating with a base station, for example, the electric field will generate heavy noise in all conductors in the vicinity. You will notice this in small signal analog systems, but it might disturb digital systems as well when the (voltage) margin between a 0 and a 1 is small.

To prevent the build-up of the hum voltage, you can start by not running signal cables parallel to power cables and by putting a metal shield around the signal-carrying conductors, actually putting the signal carrying conductor or conductors in a "Faraday cage" [2]. When you add the conductive shielding, you need to be aware of ground loops (described in the first article of this series). For example, you might use FTP (foil-shielded twisted pair) cables instead of UTP cabling.

There is also a technique where you use two conductors to transmit the signal and drive them in an inverse way with the signal to transmit. Both conductors will pick up about the same stray noise in transit. At the end, both signals electronically contain the same output as the input signal, but the stray noise will be cancelled out (see Figure 3).

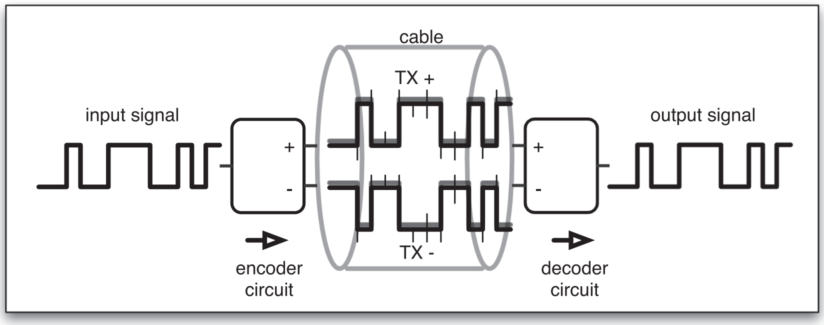


FIGURE 3: SYMMETRIC SIGNALS

This is the way Ethernet transmits the signals. To drive and receive the signals, small transformers are used. These transformers make sure that the inverse driving and subtraction of signals take place, and, by not making an electrical circuit between both systems, they also prevent ground loops from occurring (see Figure 4).

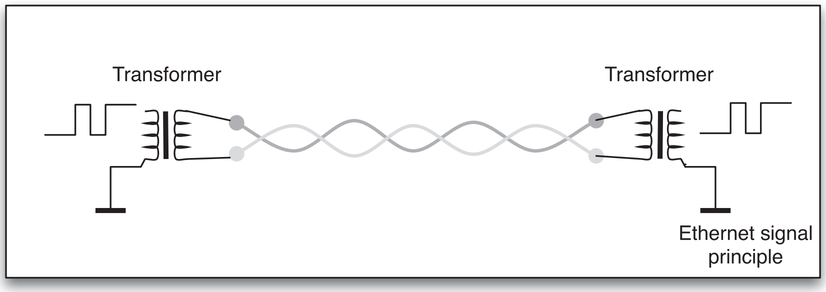


FIGURE 4: ETHERNET CABLING

The twisting of the conductors in an Ethernet cable will amplify the fact that both conductors pick up the same noise.

Differential SCSI, like Ethernet, transmits symmetrically. SCSI also protects against external influences by separating the data conductors with a ground conductor, which acts as a shield against noise or crosstalk (signals in the same cable influencing each other).

Timing Issues

Now that serial data transmission (e.g., USB or SATA/SAS) is replacing older parallel interfaces such as the Centronics printer interface (IEEE 1248 [3]), ATA, or SCSI, another factor comes into play. In order to preserve the exact timing relationship between the encoded bits in the cable it is important both to encode the bits in a way that protects the signal from outside factors and to use good cabling.

You can encode the bit (1 or 0) into the value of the voltage or into the change of voltage. Serial connections always need some kind of synchronization between the sender and receiver in order to sample the voltage values at the same relative moment in time in both sender and receiver. Having your data encoded in a steady voltage necessitates more overhead in time, and thus channel capacity to add synchronization bits, than does encoding your data in the changes of voltage. With the latter you can intelligently extract synchronization from the data signal.

I will describe how cabling can affect timing of a signal. Figure 5 shows the equivalent circuit of a piece of signal cable. The cable can be thought of as an infinite number of series resistors and parallel capacitors. The better quality the cable has, the smaller the capacitance and inductance effects are.

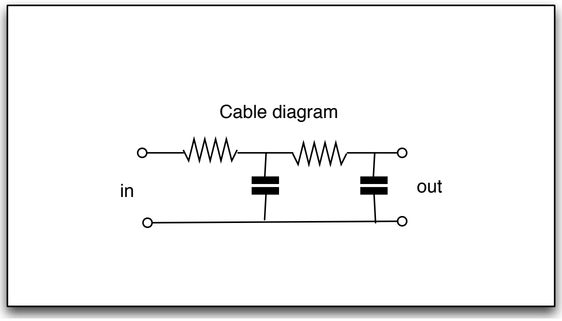


FIGURE 5: EQUIVALENT CIRCUITRY OF A CABLE

In Figure 6 we show the input signal of a cable and the output signal, in which, due to the nature of the cable, the input and output signals are quite different. There are delays (phase shift, skew) that are caused by the capacitors to be charged, and fields that have to be built up in the inductors. You also see overshoot and undershoot due to these effects. The bad thing here is that the effect is not symmetrical between the “0” and the “1” value, thus creating a different timing in the output as perceived by the electronics on the receiving end.

It is important to understand that there is often not only a delay/skew in the pulse on the output of the cable, but also an asymmetric effect changing the pulse length. The two combined can totally mess up the timing relationship in a system.

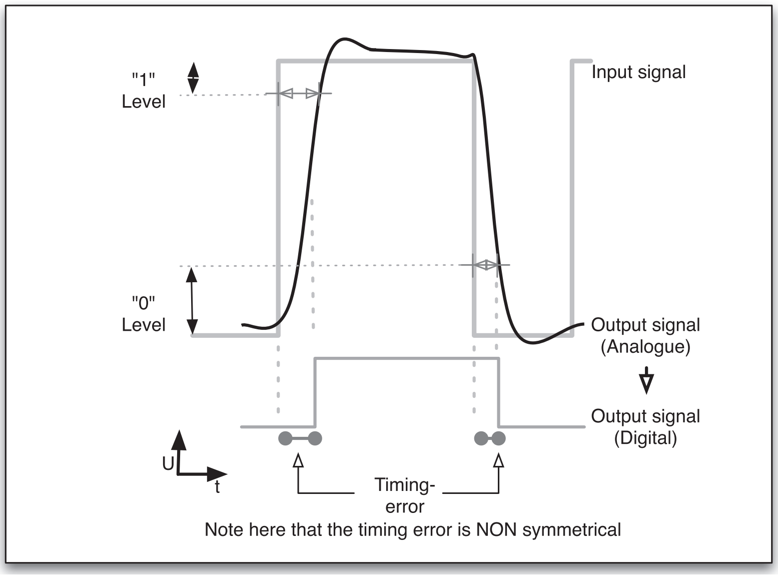


FIGURE 6: INTRODUCING TIMING ERRORS DUE TO BAD CABLING

If your data (information encoded in your signal) relies on exact timing, a short, good-quality cable is needed. A typical example can be found in the USB bus. If you make the USB connection longer than 5 m (18 ft.), data transfer will often not be reliable anymore, or the electronics will restrict the maximum transfer rate. In order to minimize these effects, the right length of high-quality, uncoiled, twisted-pair cable should be used. If you need longer transmission channels, select another technique or use regeneration boxes that receive and retransmit or decode and re-encode your information on the cable.

Note that this is also an issue present in parallel interfaces if we quickly transmit multiple words in sequence after each other.

In Figure 6 we also see that a bad cable can change the edges of the signal. This is particularly important with video interfaces. If the edges of the signal become less straight, you will see ghost images (shadow) on your screen. In comparison to analog video, this effect is much less present in digital video, since we digitize the signal first.

On this subject, we might explain the term *slew rate*. Slew rate is defined as the maximum rate of change of a signal, which in turn is defined by the environment. A cable can change the slew rate completely: when the cable is not able to let higher frequencies pass, the edges of the resulting signal on the output are not as steep.

$$SR = \frac{d(U\,out)}{d(t)}$$

The slew rate (SR) is defined in terms of units of voltage (volts, or V) per unit of time (milliseconds, or ms). On modern video cards we see slew rates of >200 V/ms.

Reflection

Another effect you might see in cabling is reflection. Each type of cable has a characteristic impedance (resistance to an AC signal). The material used and the way the cable is manufactured determine the characteristic impedance. If the impedance of the cable does not match the circuitry on both ends, you might end up with reflection. Reflection means that part of the signal is not being completely "absorbed" by the receiving end but is partly reflected by it back into the cable, thus distorting the signals.

For example, the characteristic impedance of a piece of (UTP) Ethernet cable is 100 Ω, so the receiving and transmitting circuitry need to be adapted to that in order not to create distortions in the signal. Abrupt changes in the nature of the cable—unevenly distributed twists or bends, for example (more important in other types of cable, mostly RF)—also create reflections and thus distortions of the signal.

In the early days this was extremely important in systems using coax-based, thin (10base-2) or thick (10base-5) Ethernet [4]. Coax-based Ethernet uses a cable with a characteristic impedance of 50 Ω connecting all devices in a long line with T-shaped connectors. On both ends of the line one needed a 50 Ω terminator plug to prevent signal reflections against an open end. If one of the terminators was not present the complete network failed, as the signals on the cable got too distorted to be used by the interfaces of the connected equipment.

In 10base-5 Ethernet (the thick yellow cable you had to use clamps, or "vampire taps," with), you were only allowed to put an adapter to connect a machine to the cable every 2.5 meters (8.5 ft). This was required in order to avoid signal distortion in the cable resulting from reflections caused by the tap.

SCSI is another interface requiring termination for proper operation. To prevent signal distortions due to reflections, one needs to terminate the SCSI bus. This can be done by a physical terminator on the cable end or by turning termination to "on" on the last device on the bus. The first device (the host) on the bus often has termination built in. SCSI terminators are just resistors connected to each data or handshake line and are commonly joined to signal ground.

By using reflection we can measure the length of a cable. This is done using a TDR (Time Domain Reflectometer). This device measures the time it takes a pulse to be reflected on the end of the cable and to arrive back at the sending end. The TDR can calculate the characteristics of the cable and its length. Often this function is integrated in a network test set.

General

All these factors get worse if the signal frequency (data rate) gets higher. At current data rates and frequencies used in systems that drop the supply voltage (thus lowering the noise margin between the 1 and 0 signals), external and internal (crosstalk) influence on signals becomes more important. Designers of printed circuit boards need to be aware of this, but the system designer connecting pieces of equipment also needs to be aware of the types and limitations of the interconnects.

Conclusion

With today's faster systems, cable quality is crucial in keeping the timing relations of the signals within spec. Cable with low resistance and low capacitance will introduce less distortion in your signal. It is also important to align cables properly and not to use cables that are too long. Finally, make sure your transmission channel (the cable) is of the correct type and is terminated on both sides, to prevent signal distortion due to reflection.

REFERENCE

[1] Low Voltage Logic Threshold Levels: http://www.interfacebus.com/voltage_LV_threshold.html.

[2] http://en.wikipedia.org/wiki/Faraday_cage.

[3] http://en.wikipedia.org/wiki/IEEE_1284.

[4] http://en.wikipedia.org/wiki/Thin_Ethernet.

DAVID N. BLANK-EDELMAN

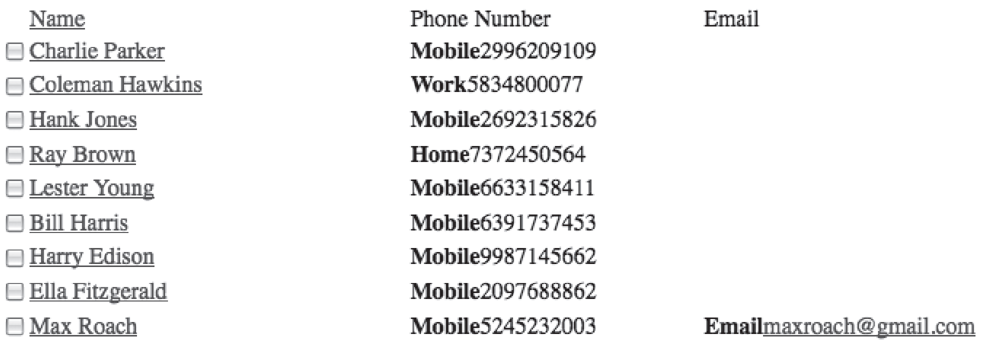
practical Perl tools: en tableau

David N. Blank-Edelman is the director of technology at the Northeastern University College of Computer and Information Science and the author of the O'Reilly book *Automating System Administration with Perl* (the second edition of the Otter book), newly available at purveyors of fine dead trees everywhere. He has spent the past 24+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and one of the LISA '06 Invited Talks co-chairs.

dnb@ccs.neu.edu

A FRIEND CAME TO ME A WHILE BACK with a problem. He had just purchased an iPhone and needed a way to get his address book from his old phone into the new one. His old phone had software that would let it sync the address book information to a service provided by the carrier. That carrier, let's call them "rhymes-with-horizon" to avoid naming names, hadn't engineered their service to make it easy to take your data with you. There was no "download your address book" (or "export as CSV") feature. At best, they offered a Web interface where you could view and edit the data to a certain extent.

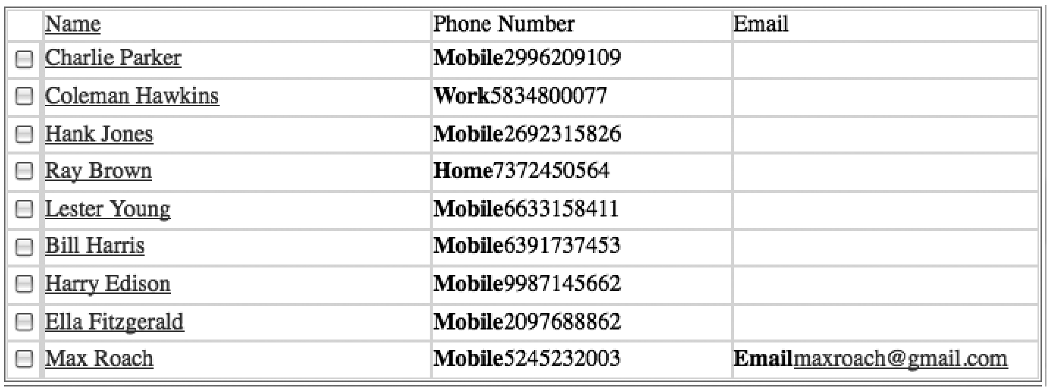
But a Web interface is better than nothing, because if we can see the data in a Web page, we can probably scrape it and return it to its rightful owner. The tricky thing here is the Web page they provided is kind of yucky. The data is embedded in a huge table and there's lots of other markup goop and JavaScript throughout. A simple cut-and-paste won't work for my friend. To get some idea of what I mean, Figure 1 shows a portion of what the table looked like in the browser (with the names and phone numbers changed).



Name	Phone Number	Email
☐ Charlie Parker	**Mobile**2996209109	
☐ Coleman Hawkins	**Work**5834800077	
☐ Hank Jones	**Mobile**2692315826	
☐ Ray Brown	**Home**7372450564	
☐ Lester Young	**Mobile**6633158411	
☐ Bill Harris	**Mobile**6391737453	
☐ Harry Edison	**Mobile**9987145662	
☐ Ella Fitzgerald	**Mobile**2097688862	
☐ Max Roach	**Mobile**5245232003	**Email**maxroach@gmail.com

FIGURE 1: DATA AS RENDERED IN THE BROWSER

To make it a little more legible, Figure 2 (next page) is what it looks like if I outline the table cells using the Firefox Web Developer add-on:



	Name	Phone Number	Email
☐	Charlie Parker	**Mobile**2996209109	
☐	Coleman Hawkins	**Work**5834800077	
☐	Hank Jones	**Mobile**2692315826	
☐	Ray Brown	**Home**7372450564	
☐	Lester Young	**Mobile**6633158411	
☐	Bill Harris	**Mobile**6391737453	
☐	Harry Edison	**Mobile**9987145662	
☐	Ella Fitzgerald	**Mobile**2097688862	
☐	Max Roach	**Mobile**5245232003	**Email**maxroach@gmail.com

FIGURE 2: OUTLINING TABLE CELLS

And that's where we'll pick up the story for this edition's column. In this column we're going to look at an approach for extracting data from even ugly HTML tables. Given how much information is now presented to us in HTML tabular form, it is generally useful to know how to grab the data and work with it on your own terms. In a previous column we looked at the WWW::Mechanize module for navigating Web sites and retrieving certain content. In this column, we're going to assume you've already retrieved the HTML document containing the table of interest (perhaps using WWW::Mechanize) and you now need to process its contents.

There are a number of ways we could approach this problem. We could shred the document using a set of complex regular expressions, but that's no fun at all. It would be a better idea to treat the HTML table like any other HTML and use some of the general-purpose HTML parsing modules like HTML::Parse and HTML::TreeBuilder. Those modules make it much easier to find the <tr> and <td> elements in the document and proceed from there. But probably the best tack we could take would be to use one of the specialized table parsing modules to do the heavy lifting, so that's what we'll do here.

Using HTML::TableExtract for Basic Data Extraction

Regular readers of this column (you know, the ones that have bought all of my albums and have the set of well-worn Practical Perl Tools tour t-shirts) might recall that I'm a big fan of HTML::TableExtract. We'll start with that module and then head into some more advanced territory.

The first step after loading HTML::TableExtract is to specify which table in the document should be considered for extraction. HTML::TableExtract offers several ways to specify the table: the two most commonly used ones are by table headers and by depth/count. With the first method you initialize an HTML::TableExtract object with the names of the column headers you care about from the table in question:

```
use HTML::TableExtract;
my $te = HTML::TableExtract->new(
        headers => [ 'Name', 'Phone Number', 'Email' ] );
```

When we ask the module to parse the data, it will attempt to find all of the tables with those headers and retrieve the data in those columns for every row in those tables.

This usually works quite well, but sometimes you encounter tables that don't play nice with a header specification: for example, tables without any labeled headers. In those cases HTML::TableExtract lets you specify a depth and count to identify the table in question. "Depth" refers to the level of embedding for a table. If the table is not embedded in any other table, it is at depth level 0. If the table you care about is in another table, that would be depth level 1. Once you establish depth, you then provide an instance number to point at the specific table (both depth and count start at 0). For example, the second table on a page would be depth => 0 and count => 1. The first embedded table in the first table in the document would have depth => 1 and count => 0. These numbers are set in a similar fashion to the headers:

```
my $te = HTML::TableExtract->new( depth => 1, count => 1 );
```

Our sample document has identifiable headers, so our program will start off like the first sample above. We can then perform the actual parse of the HTML file like so:

```
$te->parse_file('contacts.html') or die "Can't parse contacts.html: $!\n";
```

Now our object (if the parse succeeded) will let us query the tables matched and retrieve all of the rows in those tables:

```
foreach my $table ( $te->tables ) {
   foreach my $row ( $table->rows ) {
      print '|' . join( '|', @$row ) . '|' . "\n";
   }
}
```

Usually at this point we're home free, because the information in the table is sufficiently simple that the extraction yields the data we need. But, alas, with our sample document we get stuff that looks like this (I've removed a bunch of whitespace to save magazine trees, but you get the idea):

```
|
          Charlie Parker
             |
                Mobile2996209109
             |<A0>
             |
```

Yucko.

More Advanced Data Extraction with the HTML::Tree Family

Basically, each table cell in our example has a bunch of whitespace and who-knows-what in it, making for a very messy extraction. Here's a snippet of the HTML found in a table row with the whitespace stripped and the elements indented for readability:

```
<tr>
   <th>
      <input type="checkbox" name="contact5"
         value="931dc428-0 11b-1000-86fb-bfd83474aa25"
         onclick="tc(this, '5');">
   </th>
   <td style="padding-top:13px;"></td>
   <td>
      <span class="name less" style="max-width: 182px" id="x5">
         <a href="javascript:toggleContact('5')"
            title="Toggle contact">Max Roach</a>
      </span>
   </td>
```

```
      <td>
         <span class="mobile"><strong>Mobile</strong>5245232003</span>
      </td>
      <td class="end">
         <span class="email"><strong>Email</strong>
            <a href="mailto:maxroach@gmail.com">maxroach@gmail.com</a>
         </span>
      </td>
   </tr>
```

Cleaning up the HTML in this fashion was made much easier by first passing it through the great HTML Tidy program at http://tidy.sourceforge.net/.

There are at least two things we can learn about the data when we peer at it closely:

1. There's a lot of gunk (JavaScript, useless table columns, attributes, markup, etc.) we're going to want to ignore.
2. The information we do care about is found in three places:
   a. An anchor tag (<a>) holds the contact's name.
   b. A <span> holds the phone number. That span has a class attribute (class="mobile") that will let us know the kind of phone it is.
   c. A span with a class of email holds the email address if there is one.

We're not entirely stuck at this point, because HTML::TableExtract has at least one more trick up its sleeve. If you load it like this:

```
use HTML::TableExtract qw(tree);
```

it will bring in a method from the HTML::TreeBuilder module (part of the HTML::Tree package which contains HTML::TreeBuilder, HTML::Element, and HTML::ElementTable). The tree() method from HTML::TreeBuilder can turn an extracted table into an HTML::ElementTable structure (composed of HTML::Element objects):

```
foreach my $table ( $te->tables ) {
        my $tree = $table->tree;

   # ... do stuff with HTML::Element/HTML::ElementTable objects
}
```

This gives us a tree-like data structure composed of the HTML elements in the table. Here's an example dump of the tree created for the previous HTML row snippet to give you an idea of the tree that is created:

```
   DB<1> print $row->dump
<tr> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26
   <th> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.0
      <input name="contact11" onclick="tc(this, &#39;11&#39;);"
      type="checkbox" value="931dc428-011b-1000-86ff-bfd83474aa25" />
      @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.0.0
   <td style="padding-top:13px;"> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.1
   <td> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.2
      <span class="name less" id="x11" style="max-width: 182px">
      @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.2.0
         <a href="javascript:toggleContact(&#39;11&#39;)" title="Toggle
         contact"> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.2.0.0
            "Max Roach"
   <td> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.3
      <span class="mobile"> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.3.0
```

```
            <strong> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.3.0.0
              "Mobile"
            "5245232003"
        <td class="end"> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.4
          <span class="email"> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.4.0
            <strong> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.4.0.0
              "Email"
            <a href="mailto:maxroach@gmail.com">
            @0.1.7.0.2.0.1.0.2.5.2.0.1.0.26.4.0.1
            "maxroach@gmail.com"
```

This output shows the element (indented to show its level in the tree), a unique identifier, and any textual contents of the element. With that structure we should be able to tease apart the structured (albeit yucky) HTML contents of the table cells in question.

OK, so now it's clobberin' time. Our main tool for taking all of this apart is the HTML::Element method look_down(). We tell it which elements we want in the tree and it will return either the first element that matches that specification (if called in a scalar context) or all of the elements that match (if called in a list context). Our first use of it is to get all of the table rows:

```
my @table_rows = $tree->look_down( '_tag', 'tr',
        sub { $_[0]->look_down( 'class', 'name less' ) } );
```

This line of code requests elements that fit the two-part specification of

1. Find all of the <tr> tags . . .
2. . . . that contain an element with a class attribute of "name less".

look_down() then returns the list of matching HTML::Element objects that fit this bill. To get the actual data, we'll iterate over the objects returned and extract what we need:

```
foreach my $row (@table_rows) {

    my $name   = $row->look_down( 'class', 'name less' );
    my $work   = $row->look_down( 'class', 'work' );
    my $home   = $row->look_down( 'class', 'home' );
    my $mobile = $row->look_down( 'class', 'mobile' );
    my $email  = $row->look_down( 'class', 'email' );

    push @contactlist,
      [
        $name->as_trimmed_text(),
        ($work     ? ( $work->content_list )[1                          : '',
        ($home)    ? ( $home->content_list )[1]                         : '',
        ($mobile)  ? ( $mobile->content_list )[1]                       : '',
        ($email)   ? ( $email->content_list )[1]->as_trimmed_text()     : '',
      ];
}
```

The extraction starts with a gaggle of look_down() method calls, each seeking a class attribute with a specific value. Some of the method calls will return an HTML::Element; the rest will not succeed in their search and will return `undef` instead. Our next step will be to store the information found by the successful searches.

To understand what is going on in the push() statement you may need to flip back to the HTML example code we showed earlier. For the name field we can scoop up any text found in the sub-tree (as_trimmed_text()), because the only piece of text in an element with the class attribute of "name less" is the actual name. Retrieving the other data is a little bit trickier because it has a pesky label next to the actual number: for example, <strong>Mobile<strong>.

Our look_down() calls have found <span> elements that look like this:

```
DB<1> print $mobile->dump
<span class="mobile"> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.2.3.0
  <strong> @0.1.7.0.2.0.1.0.2.5.2.0.1.0.2.3.0.0
    "Mobile"
  "2996209109"
```

The <span> element has two things in it: a <strong> sub-element and the actual text value we want (the phone number). We really only care about the text value, so we just reference the second element returned by content_list() as in ($mobile->content_list)[1]. The email address <span> needs an extra as_trimmed_text() because the address is stored in an <a> sub-element instead of plain text like the phone numbers.

At the end of this rigmarole, we've got a bunch of lists in @contactlist, each list containing one contact record. We could easily spit it out as a comma-separated value file, like this:

```
foreach my $record (@contactlist) {
  print join( ',', map { '"' . $_ . '"' } @$record ), "\n";
}
```

with the resulting output looking like this:

```
"Charlie Parker","","","2996209109",""
"Coleman Hawkins","5834800077","","",""
"Hank Jones","","","2692315826",""
"Ray Brown","","7372450564","",""
"Lester Young","","","6633158411",""
"Bill Harris","","","6391737453",""
"Harry Edison","","","9987145662",""
"Ella Fitzgerald","","","2097688862",""
"Max Roach","","","5245232003","maxroach@gmail.com"
```

The Mac OS X address book is happy to import a CSV file of this format, so job done.

Eagle-eyed readers (i.e., those not falling asleep on the keyboard) may have noticed that our use of HTML::TableExtract in the last section didn't buy us very much. We still had to grovel around in a parsed tree of HTML elements to get anything done. We could have ditched HTML::TableExtract and gone right to something like HTML::TreeBuilder.

That's a perfectly valid criticism. In most cases, HTML::TableExtract hands you back the data elements you want; in this case, it just helped us find the right table in the document. There is at least one other excellent module for table parsing, called HTML::TableParser, we could have used, but my preliminary experiments with it in this context showed that the ugly HTML in the document gave it a tummy-ache as well. We'll have to save it for another task.

Hopefully, this column has given you an idea of how to extract data from both simple and complex HTML tables. Take care, and I'll see you next time.

PETER BAER GALVIN

Pete's all things Sun: AMPing up your Web environment

Peter Baer Galvin is the chief technologist for Corporate Technologies, a premier systems integrator and VAR (www.cptech.com). Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is co-author of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide. Peter blogs at http://www.galvin.info and twitters as "PeterGalvin."

pbg@cptech.com

ONE OF THE FRUSTRATIONS FACED BY system administrators (and believe me, there are many frustrations) is the frequent repetition of tasks already performed by other admins. Consider how many times Emacs has been installed in the history of computing, for example. It's frightening how many man-hours are wasted in performing the same computing tasks. And that's ignoring the whole Web-surfing thing.

Over time, many of these repetitive tasks have gotten optimized. In the Emacs example, I used to have to download the source code, compile it, and install the results. I would have to repeat this task (except the download) for any given operating system or computer architecture, and for any given Emacs release that I wanted to use or provide to my users. Now, depending on the tools provided by a given operating system, one `apt-get` or `pkg` command can check if Emacs is currently installed, determine its version number, check Internet repositories for later versions, and, if found, download and install the binaries of the newer version.

While that functionality is astoundingly useful for a given application, we still face other repetitive administration tasks involving application installation. Configuration of applications has no uniformity beyond the fact that there are usually configuration files and command-line options to inspect, modify, test, and implement. Now consider the more daunting and time-consuming task of installing and configuring a set of tools that must work together. Clearly there are "right" and "wrong" configurations, but beyond that there are optimum configurations. Getting the components to work well together, with maximum performance and reliability, is still an exercise in reading manuals, checking forums, using Web search engines, testing, and repeating until relatively happy. Usually these results are actually not optimum, but good enough for a given environment given the amount of time and effort already expended and the diminishing returns of expending more of the same. As DTrace has shown many of us, there are still many performance problems at many, many sites caused by configuration issues (as well as by poor coding).

And let's not forget the frustration of knowing that for a given common set of tools, this task has been performed before and will be performed again by legions of admins.

{L,S}AMP

A common set of tools deployed on Web servers and for Web application development includes Apache, MySQL, and the PHP/Perl/Python programming languages. These open source tools have great features and utility, but can each present installation and production readiness challenges. Implementing AMP, as those three tool sets are collectively known, is done frequently and it frequently requires a lot of effort, regardless of the operating system. Note that even operating systems that ship with some or all of those components provide administration challenges as versions of various components change and as other applications require certain versions of each component.

The AMP stack, as built for Linux (LAMP) and Solaris (SAMP), tries to reduce the admin's work from hours to minutes. It provides a pre-integrated, pre-optimized, and pre-tested set of tools that can turn a standard server into a Web server or a Web development server. Previously, this Sun pre-built, integrated, and tested stack of applications used to be called "coolstack." Sun has renamed it the "Web Stack" because it is no longer coolthreads-specific. (Coolthreads is one of the names given to the lower-power many-thread CPUs Sun ships. They are also known as Niagara or T CPUs.) The current Web Stack also includes GlassFish, Sun's open source application server, and Tomcat. These are optional, and a more limited version of the Web Stack is available without them. The Web Stack home page [1] includes links to videos and download options.

The Web Stack is available in binary form in RPM and SVR4 formats for installation on Solaris 10 SPARC, Solaris 10 x86, OpenSolaris, and Red Hat Enterprise Linux 5 U2. As of this writing, the following components were included in the Solaris x86 version of version 1.4 of the Web Stack from Sun:

Apache Server 2.2.9
lighttpd 1.4.19
Squid 2.6.STABLE17
MySQL 5.0.67
PHP 5.2.6
Ruby 1.8.6
Python 2.5.2
memcached 1.2.5
Apache Tomcat 5.5.27
GlassFish 9.1.02

AMPed Up

The Web Stack journey starts with a download (after registration) of the appropriate build. For the purposes of this column, I tested the Solaris 10 x86 and OpenSolaris versions. A check of the release notes published in the documentation wiki tells the current version information and any important bug or installation information. Oddly enough, the release notes do not include mention of GlassFish. It's a fairly separate package, in spite of its inclusion. For example, downloading the compressed tarball and untar-ing it reveals an `install` script to install all of the components, except for GlassFish. GlassFish is installed via its own `sjsas` script. Clearly this is sub-optimal and, hopefully, it will be better integrated in the future.

Support on the Web Stack is available, if desired, as part of Sun's GlassFish Portfolio offering [2]. The Sun documentation claims that Web Stack will follow a regular release schedule where it is updated, integrated, tested, and released as a bundle for convenient deployment. Certainly the current Web Stack is a good start on this, and a continuation of this effort by Sun would be a boon for all administrators running Web servers and wanting to use this collection of tools. Short of paying for support (or as well as), there is an active Web Stack community with discussion forums [3]. Also, the online documentation wiki [4] looks to be quite extensive. It includes "getting started" guides as well as an FAQ and links to the communities associated with each of the packages. Sun of course also sends interested folks to the individual open source project pages of the various packages. There is no magic here in terms of Sun creating a new branch of any of these projects. Rather, Sun has just built, integrated, and tested existing tools.

To install the full set of tools, I first executed `# ./install amp`, and then `# ./sjsas-9_1_02-solaris-i586.bin`. Note that the install tool can also install any one of the packages, if a subset of the tools is desired rather than the full AMP stack.

OpenSolaris is a different kettle of fish altogether. Web Stack releases are being timed to coincide with the distributions of OpenSolaris. For example, a Web Stack version was released at the time of OpenSolaris 2008.5, and another was released concurrently with OpenSolaris 2008.11. This practice is expected to continue.

Rather than the download-unpack-install cycle that was needed for Solaris 10, OpenSolaris installation is done via the new `pkg` mechanism. One command is all it takes: `$ pfexec pkg install amp-dev`. This same command will refresh the Web Stack when a newer version comes out. The `pfexec` command enables privileges associated with the current user. If you are logged in as a user who has the root role, `pfexec` will enable that role and execute `pkg` with root privileges. Though frowned upon, an alternative is to simply become root and execute `# pkg install amp-dev`. There is also a package management GUI that could likewise be used to install the amp-dev package. To just install the runtime components of all the AMP packages (say, to run a Web server but not to develop Web software) `$ pfexec pkg install amp` would do the trick.

Actually, a bit more than the `pkg` command is needed for a full, ready–to-use environment. For example, in the OpenSolaris GUI, each user would need to execute `Applications > Developer Tools > Web Stack Initialize` to initialize the development environment. There are other extra steps needed to start Apache and MySQL. These tasks are all well documented in the Getting Started Guide.

Once again, GlassFish is not tightly integrated into the Web Stack. In fact, a separate sequence is needed to install and configure it on OpenSolaris. The full sequence is documented in a blog [5]. In essence it involves pointing the system at a pkg repository that contains GlassFish, doing a `pkg install glassfishv2`, creating an application server domain, and starting GlassFish in that domain.

One added bonus of using the Web Stack on Solaris is the ability to use DTrace to debug problems and optimize performance of your code. The Web Stack tools have DTrace support enabled where possible and appropriate. Once again the Getting Started Guide is handy, showing how to DTrace various aspects of the Web Stack. The Guide also includes the steps needed to transfer a configuration and its contents between systems—for example, when moving a development configuration into production.

In the following example, my version of OpenSolaris was not within the allowed range of the current Web Stack package, so I upgraded my OpenSolaris from release 109 (snv_109 in the /etc/release file) to 110, then installed the amp-dev package, bringing in the entire suite, minus GlassFish. Finally, I installed the GlassFish package.

```
$ pfexec pkg install amp-dev
Creating Plan \
pkg: pkg: the following package(s) violated constraints:
   Package pkg:/SUNWj6rt@0.5.11,5.11-0.110 conflicts with constraint in in-
stalled pkg:/entire:
                      Pkg SUNWj6rt: Optional min_version: 0.5.11,5.11-0.109 max
version: 0.5.11,5.11-0.109 defined by: pkg:/entire

$ more /etc/release
          OpenSolaris 2009.06 snv_109 X86
      Copyright 2009 Sun Microsystems, Inc. All Rights Reserved.
          Use is subject to license terms.
                                              Assembled 05 March 2009

$ pfexec pkg image-update
DOWNLOAD    PKGS    FILES    XFER (MB)
Completed                               631/631              13962/13962
196.45/196.45

PHASE          ACTIONS
Removal Phase    4957/4957
Install Phase     6370/6370
Update Phase      18110/18110
PHASE          ITEMS
Reading Existing Index  9/9
Indexing Packages                       631/631

A clone of opensolaris-4 exists and has been updated and activated.
On the next boot the Boot Environment opensolaris-5 will be mounted on '/'.
Reboot when ready to switch to this updated BE.

---------------------------------------------------------------------------
NOTE: Please review release notes posted at:
 http://opensolaris.org/os/project/indiana/resources/relnotes/200811/x86/
---------------------------------------------------------------------------

$ pfexec init 6

$ more /etc/release
          OpenSolaris 2009.06 snv_110 X86
      Copyright 2009 Sun Microsystems, Inc. All Rights Reserved.
          Use is subject to license terms.
                                              Assembled 19 March 2009

$ pfexec pkg install amp-dev
PHASE          ITEMS
Indexing Packages      631/631
DOWNLOAD    PKGS    FILES    XFER (MB)
Completed                               52/52                19332/19332
335.22/335.22

PHASE          ACTIONS
Install Phase     22102/22102
Reading Existing Index  7/7
```

```
Indexing Packages     52/52
Optimizing Index...
PHASE          ITEMS
Indexing Packages     683/683
$ ls /usr/apache2/2.2
bin      build  include  lib   libexec  man   manual
$ pkg install glassfishv2
DOWNLOAD   PKGS    FILES    XFER (MB)
Completed                        11/11              6197/6197
110.19/110.19

PHASE          ACTIONS
Install Phase     7193/7193
PHASE          ITEMS
Reading Existing Index  7/7
Indexing Packages  11/11
$ pfexec pkg contents -m amp-dev
set name=fmri value=pkg:/amp-dev@0.5.11,5.11-0.101:20081210T004605Z
set name=description value="AMP Development cluster"
set name=info.classification
value="org.opensolaris.category.2008:Development/Integrated Development
Environments"
set name=publisher value=dev
depend fmri=SUNWsvn@1.4.3-0.101 type=require
depend fmri=SUNWmysql5@5.0.67-0.101 type=require
depend fmri=netbeans type=require
        ...
```

For more information on all things SAMP, LAMP, and Web Stack, a good starting place is the BigAdmin page [6].

Conclusion

Installing a full set of the latest Web server and Web service development components can require hours of work, depending on the operating system and its starting status. Getting that set of tools configured and running can increase that time immensely. Getting all of those tools running optimally and keeping them updated with the latest versions can be a never-ending system administration task. Sun's Web Stack is a great help, easing many tasks of such a project. In spite of a few rough edges it is a time and sanity saver.

Random Tidbits

While on a development theme, for those interested in using DTrace for debugging applications under development, I recommend reading the new DLight tutorial [7]. DLight is part of Sun Studio and is a visual DTrace, in essence.

Sun has announced flash storage (SSD) disks available for some of its servers. They have a central Web page [8] for information on the offerings, including an analyzer tool to run on a server to determine if flash would improve performance. I expect flash to be a major driving factor in improving server I/O performance in the next few years.

REFERENCES

[1] http://www.sun.com/software/webstack/.

[2] http://www.sun.com/software/products/glassfish_portfolio/support.jsp.

[3] http://opensolaris.org/os/project/Webstack/.

[4] http://wikis.sun.com/display/WebStack/Sun+Web+Stack+Documentation.

[5] http://blogs.sun.com/hyau/entry/opensolaris_is_now_ready_onamazon.

[6] http://wikis.sun.com/display/BigAdmin/LAMP-SAMP.

[7] http://sun.systemnews.com/articles/133/4/Solaris/21537.

[8] http://www.sun.com/storage/flash/resources.jsp.

DAVE JOSEPHSEN

iVoyeur: the dangers of marketing

Dave Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and senior systems engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper award for his co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

I, LIKE SO MANY OF MY PEERS, HAVE occasionally observed that if not for the presence of users, my systems and networks would perform perfectly. I suspect, however, that if we were ever to achieve our collective long-fancied dream of . . . "removing" the users, that running systems would be rather less fun, not to mention less profitable (but still worth it, I dare say). Though they are often worthy of my disdain, users are, I must admit, beyond necessary in my current line of work, where we live or die by the number of folks voluntarily using the Web sites we host; they are in fact . . . desirable.

That I should actually encourage people to use the services I'm providing smacks of poetic justice. At least it's not something my career to this point has prepared me for. Having been brought up mostly in large corporate environments, I've been of the mind-set that work gets done in spite of the users. Not even users, in fact, but lusers [1], pebcaks [2], and suits [3]. Users went from annoying to desirable in my estimation gradually. Truthfully, I still haven't entirely made the transition. It's been like slowly coming to the realization that carbs might not be as bad as everyone says. Intellectually I get it, but blueberry muffins still inspire pangs of guilt.

That users could be something to fear, however, was an entirely new and unconsidered possibility. Worse, I had no time to prepare mentally. I went into a meeting one day and came out a changed man upon being given the news that not only would a site we were hosting be given the front-page ad on msn.com, but that there would be several prime-time TV commercials during, for example, "ER," "Lost," and something called "The NCAA Playoffs."

What kind of traffic increase can you expect to get from a TV ad? I did a quick mental inventory and discovered that I didn't know anybody who could easily answer that question. I've heard more than my share of slashdottting horror stories, but these anecdotes don't help, because nobody who tells them was capable of serving the traffic, much less measuring it. Alas, Google wasn't much help either. So in the hope that some terrified sysadmin finds it helpful, this month I offer to you my first experience with a real marketing behemoth.

I should back up a bit and give you some context. I left my quite cushy corporate job three years ago to work at a tiny little tech company. Well, that isn't exactly what happened. I left under some duress, and wasn't getting along with my cushy corporate overlords. I had a distressing tendency to get things done, you see. Things that everyone wanted but nobody would dare request. So maddening was my penchant for implementing systems that the company needed yet management hadn't asked for that I was flung from manager to manager like a radioactive potato. I had 16 or so managers while I was there, and yet despite the re-orging, the time tracking, and even the Microsoft project, up would pop a monitoring system, a code promotion system, a documentation system. You get the idea. They were all free and they worked well enough that no one could stomach having them dismantled, but they didn't endear me to management. When the managers finally prevented me from getting anything done for nine months or so, I started sending out the resumes. When I gave notice two weeks later they were quite surprised, having been happy that I was finally settling into the corporate culture.

I was the first dedicated sysadmin-type in my current company. We have not many employees—fewer, in fact, than the number of Web pages we host. We have a great niche hosting "loyalty management" (points/miles etc.) programs for much larger companies. Because of the size of our customers compared to ourselves, the job works out to be what a lot of sysadmins might consider a dream job. Suffice it to say, no pigeonholing goes on here; servers, firewalls, routers, load balancers—the sysadmins (both of us) are armpits deep in all of it, and it's interesting work. The problems are large and varied, at least for me. On any given day I might, for instance, be designing BGP-based failover for our geographically dispersed co-located data centers, extending our LDAP schema, configuring LUNs on a SAN, creating VLANs, or writing bash scripts to transfer and load Oracle data.

The best part about this job is having the complete freedom to choose the right tools to do it. As a result we have quite an eclectic mix of tools, including OpenBSD routers on rather nifty Axiomtek-embedded hardware [4], Fujitsu SANs, various distributions of Linux including (shameless plug) SourceMage [5], running on everything from gray boxes to Sun x4550s, and, of course, the usual suspects that you'd expect from Cisco. Most of the best stuff we have is home-grown. I am especially proud of our load-balancer tier; it does things no commercial balancer can, and as much as I'd like to co-opt this article into fully describing it, I'll restrain myself and leave it to my cohort to publish, it being his brainchild. How's that LISA paper coming, Jer?

The problem with freedom is that pesky responsibility, so with the freedom of choosing and building your own tools comes the responsibility of building a solution bullet-proof enough to someday survive national TV and radio ads. By now I hope you can begin to relate to the sinking feeling I experienced in my stomach when given the details of our customer's marketing plans—plans that included not only a litany of TV and Web advertising, but 25% market saturation on a slew of national radio stations. Sure, we'd load-tested the environment, but at the time that thought provided little comfort. It's this feeling, or something like it, that I imagine made Microsoft so popular in the '90s.

I'll cut to the chase here and tell you that we've survived the onslaught (so far), and that the marketing was able to increase our traffic by more than an order of magnitude. Figure 1 is a graph of the traffic during the initial marketing ramp-up. The y-axis has been altered to protect our customer, but otherwise the data is unaltered.

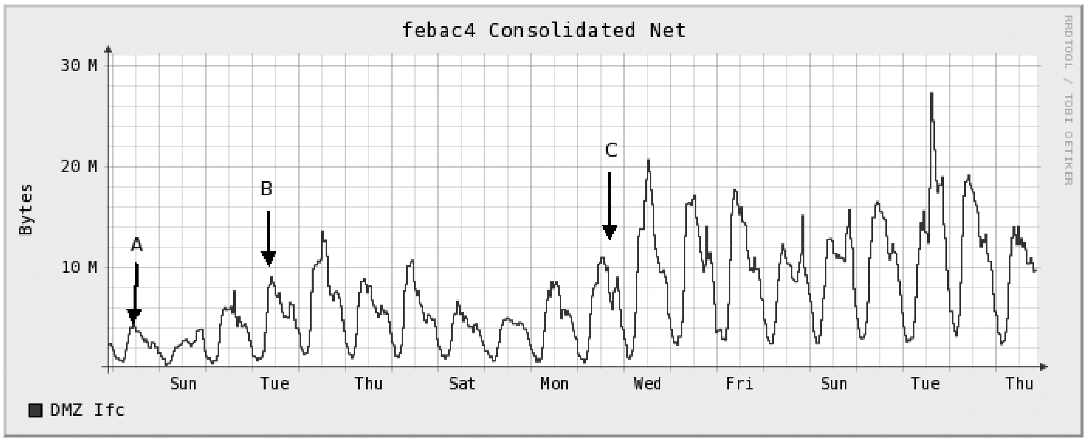


FIGURE 1

The point marked A shows our normal pre-marketing traffic. Point B marks the beginning of the Web marketing campaign, which included the afore-mentioned msn.com front-page ad, and point C marks the beginning of the radio and TV ads, the first of which appeared during a popular prime-time ABC show.

We used the week or so of lead-time we were graciously given to scale up our app tier. This was done easily enough, because the app tier servers are Xen virtual machines. We quickly doubled their number, and distributed the app across them. We also built up some additional balancers in case we needed to scale up the balancer cloud, but as of this writing we haven't had to use them. Quite to our relief, the traffic increase was pretty much ideal from our perspective, *lots* of extra traffic but nothing the systems couldn't handle.

What about security? Does being on TV make you a bigger target? To that question I can give an emphatic yes. And while I can't give a whole lot of detail here, I can say that application-layer attacks (injections, URL traversal, cross-site whatevertheheck, etc.) have increased nearly 600%. The lower-level stuff—scans, brute-force attacks, etc.—have increased in kind. The cycles we spend mitigating attacks are not trivial. Were I advising someone building an infrastructure from scratch, my advice would be to make attack mitigation, logging, etc., a design consideration. But that sort of goes without saying nowadays.

For now at least, it seems I've reached a happy medium with my users, though I suspect I remain more frightened of them than they of me. That particular role reversal is troubling, to be sure, but worth it, I dare say.

Take it easy.

REFERENCES

[1] http://catb.org/jargon/html/L/luser.html.

[2] http://catb.org/jargon/html/P/PEBKAC.html.

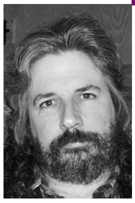
[3] http://catb.org/jargon/html/S/suit.html.

[4] http://www.axiomtek.com.tw/Products/ViewProduct.asp?view=676.

[5] http://www.sourcemage.org/.

ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is an information security geek biding his time until that genius grant finally comes through.

rgferrell@gmail.com

WHEN I WAS A KID IN NORTH TEXAS in the sixties—I realize young people, or at least those who can still comprehend entire sentences that contain no abbreviations, tend to tune out anything that follows the introductory phrase "When I was a kid," but I doubt there are many of them reading this column in the first place—"networking" pretty much meant meeting other people face-to-face at church picnics, the Rotary Club fundraiser bake sale, or that annual carnival in the Piggly-Wiggly parking lot. The term "social networking" was redundant and therefore unnecessary. This was, of course, in the days when lack of legitimate linguistic function was still an etymological disqualifier.

Along came computer networks, ARPANET, and, eventually, the public Internet (thanks, Al Gore), and the landscape of social interaction changed almost overnight. Impersonal communication has more or less always been possible via messenger and then postal mail, of course, but the delay between statement and response was so great that it couldn't really be compared with actual conversing. You had days, even weeks or months, to consider and compose your contribution to the dialogue. The advent of the telegraph shortened this latency considerably, and then widespread adoption of telephony brought remote conversations to real-time status. There were no visual clues inherent in this verbal intercourse, though, so miscommunication was rampant.

Now we have (lucky us) "social networking," which term is really nothing more than an insidious oxymoron. Yes, it involves interactions among lots of people, but those people almost never come into physical proximity in their daily genuflections to Face(less)book. Social networking really is a throwback to the old "party line" days of the early telephone system, where many people talk over one another or eavesdrop silently in an anonymous amorphous mass. Humans are primed genetically to respond to even the subtlest visual clues about the mood and intentions of another human via body language. Emoticons just don't convey the same level of information, no matter how cleverly employed. Maybe if we keep communicating this way for another hundred thousand years or so we can develop some kind of "verbal-only perception" sense, but I doubt it. We haven't, as a species, made a lot of progress towards developing sense of any sort.

It isn't that I think the whole concept of social networking is outright bogus or stupid. There are things about it I really admire, most of which revolve around the fact that it keeps kids who might otherwise be doing drugs, creating hip-hop, or planning careers as investment bankers somewhat distracted. I personally get a certain amount of entertainment envisioning people raiding Naxxramas while sending email, texting, tweeting, and talking on the cell phone simultaneously, in much the same way that I used to enjoy watching the Surgery Channel whenever flesh-eating bacteria played a prominent role.

At the risk of exposing myself as the rapidly aging fuddy-duddy I quite definitely am, the single most amusing (disturbing) facet of the social networking phenomenon (plague) from my perspective is the willingness of MySpies users to expose themselves, both anatomically and psychologically, at the drop of a virtual hat. Never before in history has intelligence gathering regarding the innermost demons of prospective students/employees/spouses been so simple and convenient. It takes a lot of the tedium out of being a sexual predator when potential victims post their complete hopes, dreams, aspirations, and itineraries online for your casual perusal, lemme tell you. It's as though you have a 21-inch LCD window into every teenager's private diary. With a soundtrack and video. You don't even need the little brass key.

The business-related heads of the social networking hydra I don't so much mind. LinkedIn, for example, seems nominally useful if for no other reason than I like to look at the interesting societal cladistics generated by the interrelationships therein represented. You are in a maze of twisty passages, all alike. My life has been indelibly enriched by the realization that there are only four degrees of LinkedIn separation between myself and an individual who claims to be a confidant of Valiant Thor, an avowed native of the planet Venus. I suppose if I were in the import/export business that might present some potentially lucrative commercial possibilities, but shipping costs are somewhat problematic. Then there is also the whole "virtually everything is quickly reduced to a molten mess on Venus" inconvenience. On the plus side, getting through Customs should be a piece of cake.

Blogging and its ADHD-riddled bastard offspring, tweeting, are the latest and arguably most pathogenic mutations of the always-connected neurophage. Blogging at least has the potential for relating moderately enlightening information, mired deeply though such gems are in a planet-sized morass of the mundane, the inane, and the profane. Tweeting, on the other grossly deformed hand, is nothing more than a low-budget horror flick where thousands upon thousands of zombies stumble around mumbling incoherently to themselves while the incoherently mumbling zombies who pass nearest them pretend briefly to be listening.

It is good practice for membership in state or federal legislative bodies, I guess.

book reviews

ELIZABETH ZWICKY, WITH SAM F. STOVER

97 THINGS EVERY SOFTWARE ARCHITECT SHOULD KNOW: COLLECTIVE WISDOM FROM THE EXPERTS

Richard Monson-Haefel, editor

O'Reilly, 2009. 195 pages.
ISBN 978-0-596-52269-8

This is a sweet collection of advice. As you might expect from a collection, some advice overlaps and some conflicts. That's OK. Actually, if I had to pick my top advice for software architects, there are some important apparent conflicts that appear very fast (for instance, always plan for the future, but not too far in the future, and don't expect it to actually work—all of which is better discussed in the book). If you can't handle balancing opposing and important concerns, don't try to architect anything.

This isn't going to teach you how to be a software architect. It's more a tool for software architects who want to improve their practice. The advice is mostly in the "simple but not easy" category, so it's the sort of book you want to read a little at a time, think on, come back to when you need a pick-me-up, argue with your colleagues about.

Plus, I'm glad it's 97 items. I hate it when people force their lists into round numbers. (Although I do have my suspicions that ending up with a prime number of items was not entirely accidental. It is possible that I hang out with too many people who notice this sort of thing.)

NETWORK KNOW-HOW: AN ESSENTIAL GUIDE FOR THE ACCIDENTAL ADMIN

John Ross

No Starch Press, 2009. 251 pages.
ISBN 978-1-59327-191-6

Networking is complex, and yet most networks are not big enough to need a full-time network administrator. My home network is full-featured to an unusual extent (there aren't very many nodes, for a household of computer people, but a mix of expediency and curiosity leads to some baroque complexity). Even though we're running exotic feature-sets on routers purchased on eBay, most of the time the thing just works. This is even more true for most households and even most small businesses.

So what happens when new features are needed or, worse yet, it stops working? Well, my household is in good shape, but most networks aren't, so a book like this could be extremely useful.

And this book is OK. Instead of to troubleshooting, it's mostly devoted to setting things up, which I find is usually the easy part, although a bit of help understanding what's going on and how the pieces fit together is definitely useful. Some of its advice is purely mystifying: no, really, I asked around, and people don't usually spell out IP addresses number-by-number ("one nine two dot three five . . . " instead of "one ninety-two dot thirty-five . . . "). The author may say "why-fie" for Wi-Fi but I guess he says "why" differently than I do. No home network I know of changes WPA encryption keys once or twice a month. (Frankly, most people change them when they move. Yes, it would be safer to change them all the time, but then all my friends would be typing in new passwords every time they come over.) "Modem" is not a geeky term for "modulator-demodulator," but, rather, the other way around. Not that there's a non-geeky term. And honestly, I know that these things are very confusing and some skipping details is necessary and experts disagree on fine nuances, but a bridge is not a device that sits between two different networks, and it is not fair to say that NAT is a primary characteristic of a router.

All of this made me very cranky. Possibly unreasonably cranky; it's like listening to somebody singing slightly off-tune. The fact is, there's a lot of useful information here. The presentation is relatively accessible, suitable for people who are a bit technical but not network-literate, and there's practical advice for small networks with little or no support staff, which is hard to find elsewhere. The information on troubleshooting, while sparse, is practical and accurate. It's not the book I was hoping for, but it's a lot better than nothing.

PHOTOSHOP CS4 PHOTOGRAPHER'S HANDBOOK

Stephen Laskevitch

Rocky Nook, 2009. 258 pages.
ISBN 978-1-933952-42-0

If you're new to Photoshop and are interested in doing normal photograph things with it, this is a good starting point. It describes a good working process, firmly based in current Photoshop best practices (every pixel is sacred! never destroy data!). It does a quick but reasonable job of introducing you to the basics of pixel-based photographs, assuming very little about your knowledge of digital photography. It's not an advanced Photoshop technique manual, but it does cover the techniques you're likely to need to get the best out of your photos, plus the most popular fun tricks.

Oddly, the thing I liked least about this book was the layout. I found that navigation was sometimes tricky. The book actually covers three or four applications, depending on how you count—Photoshop, Bridge (which ships with Photoshop), Lightroom (which can be bundled with Photoshop but is a separate, extra-cost application), and Adobe Camera Raw (which is a plugin, but with all the features of a separate application). These applications overlap a lot, so almost every task can be undertaken in at least two of them. This means a lot of back and forth. There are handy little color blocks to tell you what application is being discussed, but it still changes every few paragraphs in some places. Couple this with the need to put in lots of screen shots and illustrations, and I found it hard to follow from time to time.

THE MANGA GUIDE TO DATABASES

Mana Takahashi, Shoko Asuma, and Trend-Pro Co., Ltd.

No Starch Press, 2009. 208 pages.
ISBN 978-1-59327-190-9

This is a perfectly reasonable introduction to databases, including normal forms, and basic SQL queries. It's not particularly deep; you wouldn't want it to be your DBA's main text or anything, but a person who pays attention and does the exercises will be able to, for instance, understand what's so horrible about most of the SQL examples you see on www.thedailywtf.com, or what a DBA is talking about. It's enough to do some basic database work, if you're a reasonably technically oriented person to begin with.

I would recommend judging this book by the cover. If you look at the big-eyed fairy and think "Bleargh," really, it's not going to get any better. There's a princess, and a love interest. If you look at it and think, "Cute. That could make SQL bearable," then you're in the right place. (At least this time the love interest is not creepy.)

It's fatally easy to skim, so the unmotivated reader can easily come away with a feeling of virtue and not much actual knowledge. In some ways, it's like one of those girly cocktails; it's pink and fluffy, but it packs a concealed punch. Unfortunately, in this case you won't take it in without noticing. I found that I was periodically going back to re-read.

UNIX AND LINUX FORENSIC ANALYSIS DVD TOOLKIT

Chris Pogue, Cory Altheide, and Todd Haverkos

Syngress, 2009. 230 pages.
ISBN: 978-1-59749-26-0

REVIEWED BY SAM STOVER
(SAM.STOVER@GMAIL.COM)

This little book was a pleasant surprise: well written, upfront about the targeted audience, and full of interesting information. When I first started reading, I immediately formed the "another Windows user who is amazed by basic *nix capabilities" opinion. While there is a little of this, it's not overwhelming, and the basics covered are solid. Since the forensic process touches just about everything hardware and software, this is a great book for someone who doesn't know much *nix but wants to learn, and that was what the author intended.

Weighing in at a lean 230 pages, the book contains eight chapters and an appendix. The first chapter, "Introduction," is very short and covers what is covered, what is not covered, and who the target audience is. I think this is a pretty important chapter for *nix geeks, because, unlike some other books, this one does a great job of laying out everything so the reader isn't taken by surprise as they read the book. If you find yourself considering this book for purchase, definitely read the Intro, which does a great job of telling you whether you'll benefit from it.

"Understanding Unix," the second chapter, it covers the expected *nix basics: differences between UNIX and Linux, some basic file system stuff, and an introduction to shells. The third chapter, "Live Response: Data Collection," starts to delve into the forensic process a bit and how this differs from Windows to *nix. Someone with experience using EnCase, FTK, and other Windows forensic tools will find some familiar material here. Chapter 4, "Initial Triage and Live Response: Data Analysis," hits on numerous *nix commands that replace or augment the typical Windows forensic toolkit. I've said it before and I'll say it again, the majority of whiz-bang features included in most Windows forensic toolkits are simply commands that *NIX geeks have been using for years, and that becomes pretty clear in this chapter. I sincerely doubt that my target audience needs a refresher on more, less, and tail, so unless you want to see how they fit in the forensic process, you might be bored by Chapter 4.

Chapter 5 lists the "Hacking Top 10" tools, which, again, should be familiar to any self-respecting geek: netcat, nmap, nessus, nikto, wireshark, etc. Good intro chapter for the Windows user, but old hat to *nix folks.

Chapters 6 and 7 deal with "The /proc File System" and "File Analysis" respectively, and they do a really great job. While I wouldn't expect you to buy this book for two chapters alone, if you need a refresher on /proc, Chapter 6 is a good place to start. Since a lot of forensic analysis is actually file analysis, understanding the file system is pretty important, and these two chapters provide what you need. Chapter 8, entitled "Malware," actually deals more with anti-virus solutions (Panda and Clam) than actual malware—my only real gripe with the book. There is a pretty interesting discussion of malware on the *NIX platform, and it's not just the ubiquitous "Linux is more secure than Windows because of X, Y, and Z" but some well-thought-out points and expectations for the future.

This book might be a little too basic for the *nix maestro who wants to learn forensics, but I'd still recommend considering it. Also, while I don't think this was the intent of the author, I think this is a great introduction for any budding *nix enthusiast, because it deals with a lot of core and basic concepts inherent to *nix that anyone, not just a Windows forensic analyst, can learn from. A solid intro book.

USENIX notes

USENIX MEMBER BENEFITS

Members of the USENIX Association receive the following benefits:

FREE SUBSCRIPTION to *;login:*, the Association's magazine, published six times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on such topics as security, Perl, networks, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

ACCESS TO ;LOGIN: online from October 1997 to this month: www.usenix.org/publications/login/.

DISCOUNTS on registration fees for all USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals: www.usenix.org/membership/specialdisc.html.

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, and election of its directors and officers.

FOR MORE INFORMATION regarding membership or benefits, please see www.usenix.org/membership/ or contact office@usenix.org. Phone: 510-528-8649

USENIX BOARD OF DIRECTORS

USENIX GOES GREENER

Jane-Ellen Long,
Director of IS & Production

USENIX continually seeks additional ways to enhance its environmental responsibility. Below are some of the steps we've taken. We welcome additional suggestions from you, our membership.

These days, we publish event proceedings online for attendees before the event and make the files world-accessible from the opening day of the event. We are working to reduce our offerings of pre-printed proceedings; those we do print use recycled paper.

At our events, we offer both proceedings and training materials on reusable flash drives instead of throw-away CD-ROMs or paper, and we give our attendees reusable cloth bags made from recycled materials instead of the usual plastic. None of our conference materials come in plastic packaging. Our signs are mounted on biodegradable foamcore with 15% recycled content. We choose hotels that have a "green policy," and we work with them to further minimize negative impact on the environment. Our attendees are given an assortment of ways to help us in these efforts, including returning their badge holders to us for reuse.

For our informational contacts with you, we focus on Web and email to minimize our printed and mailed materials. Those materials we do mail use recycled paper. Instead of mailing membership materials, we leave it up to members whether to print their membership cards from the online file.

In the office, we've virtualized both servers and workstations, automated shutdown of our desktops, and chosen new equipment with an eye to reducing power consumption and heat generation. Computer equipment has a replacement schedule longer than five years and is donated to the Alameda Computer Recycling Center at the end of its life. We're also moving services from our office machine room to efficient datacenters and service providers.

Finally, the USENIX Reserve Fund invests globally in the bluest of blue chip companies, and these U.S. companies cannot derive revenue from energy, alcohol, or tobacco products.

NOTICE OF ANNUAL MEETING

The USENIX Association's Annual Meeting with the membership and the Board of Directors will be held during the 18th USENIX Security Symposium, on August 10–14, 2009, Montreal, Canada. The time and place of the meeting will be announced onsite and on the conference Web site, www.usenix.org/sec09.

conference reports

THANKS TO OUR SUMMARIZERS

FAST '09: 7th USENIX Conference on File and Storage Technologies

San Francisco, CA
February 24–27, 2009

OPENING REMARKS AND AWARDS

Summarized by Dutch Meyer (dmeyer@cs.ubc.ca)

Program Chairs Margo Seltzer and Ric Wheeler opened FAST '09 by thanking the contributors and stressing the value of interaction between students and attendees. Best Paper awards were presented to "CA-NFS: A Congestion-Aware Network File System" by Batsakis, Burns, Kanevsky, Lentini, and Talpey of NetApp and Johns Hopkins, and to "Generating Realistic Impressions for File System Benchmarking" by Agrawal, Arpaci-Dusseau, and Arpaci-Dusseau of the University of Wisconsin, Madison.

Next, Garth Gibson took the floor to present the 2009 IEEE Reynold B. Johnson Information Storage Systems Award to Marshall Kirk McKusick. McKusick was recognized "for fundamental contributions in file system design, mentoring file system designers, and disseminating file system research." In his acceptance speech, McKusick stressed two themes: first, the collaboration between hardware and software experts, and second, the lessons drawn from his work on the Berkeley Fast File System (FFS).

In thanking the awards committee, McKusick praised their equal consideration of hardware and software nominees, a trend he hoped would continue. He stressed that it is necessary to incorporate both sides of the hardware and software interface with respect to storage. As an example, he pointed to what he termed the "don't lie to me" bit, which tells mechanical disk drives to confirm that data is written only when it actually reaches the persistent medium. He also pointed to the FAST conference itself as a unique forum in that it draws heavily from a diverse community of hardware, software, academic, and industry experts. This characteristic, he argued, was key to the conference's success.

Reflecting on his own work, McKusick began by noting that FFS was initially created in a "target rich environment." Because comparable systems were so slow, it was relatively easy to demonstrate significant improvements quickly. However, to remain relevant has required constant effort. In its initial version, FFS weighed in at a mere 1,200 lines of code. The current version, which remains a canonical file system after 30 years, has grown to 55,000 lines of code. This increase is the result of steady improvement in the attempt to remain competitive with new file system offerings and ideas. As he finished, McKusick quipped that in comparison, ZFS's 120,000 lines of code had been written in just a few years.

KEYNOTE ADDRESS

Summarized by Dutch Meyer (dmeyer@cs.ubc.ca)

- ***Cloud Storage FUD (Failure, Uncertainty, and Durability)***
  Alyssa Henry, General Manager of Amazon Simple Storage Service (S3)

Alyssa Henry presented Simple Storage Service's (S3) goals and her team's experiences with developing an Internet-scalable and accessible storage system over the past four years. The presentation mixed colorful anecdotes with a description of the project's motivation and design.

Henry's major theme was that the S3's broad audience had many different requirements and workloads. To meet such a wide range of users, S3 needed to be designed to tolerate a large degree of uncertainty. It is possible to identify trends when a service is in active use, but these trends are dynamic in practice and cannot be relied upon. Underlying the design of the service is a low-cost "pay as you go" model, which is supported by leveraging Amazon's software expertise against commodity hardware and balancing the system's architecture against the need to keep costs low.

Failures are intrinsic to systems at this large scale, and Henry pointed out that even low probability errors begin to happen regularly. She classified service disruptions across two axes: duration, with errors ranging from temporary to permanent, and scope, ranging from few clients to many. Failures with sufficiently small scope and low duration are essentially harmless, while persistent errors with large scope are catastrophic.

The overarching strategy employed by S3 is to broaden the class of harmless failures into the region that would otherwise be considered catastrophic. Since perfection is not attainable, the goal is to balance the odds of service disruption against financial and complexity costs. The specific methods used toward this end are a mix of traditional approaches such as redundancy, rate limiting, and hardware diversity, with some newer ideas. Amazon's focus on the eventual consistency model was cited as one way in which they break from tradition. Amazon Web Services also routinely force failures to occur—for example, by yanking power plugs when a system is to receive downtime and by turning off whole datacenters for management. By stressing error paths, often one can work with more assurance that the countermeasures employed continue to be effective.

In closing, Henry commented that storage services represent lasting relationships that require trust. She also noted that reliability at low cost remains a difficult problem. She directed parties interested in more information to visit Werner Vogels' blog (http://www.allthingsdistributed.com/) and the Amazon Web Services blog (http://aws.typepad.com/aws/).

Sameer Ajmani from Google followed up on the hardware diversity comment to ask if software diversity was also a viable strategy. To this, the presenter noted that all aspects of the system represent design tradeoffs. For S3's needs, software diversity would be going too far, at too great expense. David Rosenthal from Stanford University asked why the system doesn't publish a numerical goal for reliability, as there is for availability. He pointed out that 100% reliability is not realistic and that the provided EULA has no penalty for data loss. Henry reiterated that the team's reliability goal was 100%, while in practice the service level agreement specified 99.9%, and this seemed to satisfy customers. S3 performs internal measurements of the error rate, but the resulting data is not disclosed. Stephen Spackman from Quantum asked how Amazon balances the trade-off between centralizing data in one country and offshoring it. S3 pushes this issue to the end user, who can choose between any combination of US and EU S3 offerings.

AUGMENTING FILE SYSTEM FUNCTIONALITY

Summarized by Phillipa Gill (phillipa@cs.toronto.edu)

- ***The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance***
  Ragib Hasan, University of Illinois at Urbana-Champaign; Radu Sion, Stony Brook University; Marianne Winslett, University of Illinois at Urbana-Champaign

Ragib Hasan presented a secure provenance scheme implemented at the application level. Hasan highlighted that most previously developed provenance schemes were applied in the domain of scientific applications. The provenance system developed in this paper is designed to be more secure so that it may be applied in finance or business applications. The authors designed a system that aims to prevent undetectable history rewriting. That is, any change or deletion of provenance information must be detectable.

Hasan described their method for providing secure provenance. The method uses the concept of a provenance chain which is made up of provenance entries (records of users' modifications and context). Adversaries in the system include users who may add or delete provenance entries or collude with each other to modify entries (more detail about adversaries in Hasan et al., *ACM Storage SS 2007*). Auditors are trusted entities who may verify the accuracy of the provenance chain by using checksums to prevent undetectable changes to the provenance entries. This is done by computing the checksum of an entry as a function of the previous entry's checksum and the new entry. Selective confidentiality is provided by encrypting the modification details and distributing the encryption key in such a way that only the auditors trusted by a user can see modification details. The scheme also allows selective disclosure to third parties by redaction of sensitive attributes without invalidating the integrity checksums. Hasan also talked about experimental results: for most typical real-life workloads, this secure provenance scheme incurs only 1%–13% runtime overhead.

The audience raised the issue of how the provenance system could handle a document that may be composed of multiple elements (e.g., an HTML page). Hasan stated that the current scheme is applied to a single file but that a provenance chain may also be constructed for a whole document. The issue of security guarantees when multiple users collude and create multiple entries was also raised. Hasan stated that the colluding users can only modify their own entries and would not be able to tamper with provenance entries from benign users.

Hasan provided a URL for more information: http://tinyurl.com/secprov. [See p. 12 for an article about this paper.]

- ***Causality-Based Versioning***
  Kiran-Kumar Muniswamy-Reddy and David A. Holland, Harvard University

Kiran-Kumar Muniswamy-Reddy began by explaining a motivating example for causality-based versioning. He described the situation in which a piece of software is installed, but when it is uninstalled some changed files remain. A versioning system would enable a user to roll back the system but would not provide information about which files were modified. In another example, users continue their database work even after a malicious entity begins to tamper with the database. In these cases it is important not only to roll back the system, but also to know which pieces of data were modified (that is, which bits to keep).

Muniswamy-Reddy contrasted two different versioning schemes, open-close and version-on-write. Open-close is coarse-grained and versions when files are opened or closed, while version-on-write is fine-grained, creating a version for each write. Open-close has much lower overhead than version-on-write, but also provides less fine-grained versioning.

Muniswamy-Reddy then described the Cycle-Avoidance and Graph-Finesse algorithms proposed by his paper. Cycle-Avoidance preserves causality, but only uses local information when deciding to create a new version. As a result, it creates more versions than necessary. However, by only using local information, it has lower overhead. The Graph-Finesse algorithm uses global knowledge and as a result creates fewer versions. However, it has higher overhead than the Cycle-Avoidance algorithm. Implementation results were shown and the authors concluded that adding causality to versioning-based systems only increases overhead by 7%.

The audience asked how the system would compare to the open-close method on a single process. Muniswamy-Reddy emphasized that the causality becomes necessary only when a second process is present. The audience also asked how the algorithm would perform in a system with a large number of files. The author stated that for Cycle-Avoidance it would depend on the size of the local data, but that Cycle-Avoidance would perform much better than Graph-Finesse, which uses global data.

- ***Enabling Transactional File Access via Lightweight Kernel Extensions***
  Richard P. Spillane, Sachin Gaikwad, Manjunath Chinni, and Erez Zadok, Stony Brook University; Charles P. Wright, IBM T.J. Watson Research Center

Richard Spillane presented the audience with the results of his work extending the kernel to support transactional file access. Spillane described Valor, a file interface that requires only a small amount of modification to the page writeback mechanism and some additional module code. Valor adds seven new system calls to the kernel that allow processes to utilize atomic, consistent, isolated, and optionally durable transactions. In serial overwrite benchmarking tests, Spillane noted that Valor is 2.75 times slower than ext3, but it has lower overhead than Stasis, which runs 4.8 times slower than ext3. Spillane also showed that Valor outperforms Berkeley DB by a factor of 8.22.

An audience member asked if Valor performed dependency resolution between transactions. Spillane referred the questioner to the paper for details of Valor's isolation semantics. The audience also asked for more detail on the kernel and user-space implementations. Spillane explained that moving transactional support to the user level is difficult because performance will be impacted. Also, kernel support gives transactional support true transparency. The impacts of the transactional support on non-transactional I/O were also discussed. Spillane stated that non-transactional writes to a page that was being written to by a transaction would have to wait for both page writeback and any isolation locks on that page to be released. Margo Seltzer commented that the benchmarking workload used was not one that Berkeley DB (BDB) was made for and that the configuration would have also impacted the performance of BDB. She also noted that the page size chosen for BDB was sub-optimal. Spillane pointed out that Stasis provides an upper bound on the performance of BDB since it also utilizes a user-space page cache implementation, but Stasis is not restricted to writing into a B-Tree.

DIAGNOSIS

Summarized by Ragib Hasan (rhasan@uiuc.edu)

- ***Understanding Customer Problem Troubleshooting from Storage System Logs***
  Weihang Jiang and Chongfeng Hu, University of Illinois at Urbana-Champaign; Shankar Pasupathy and Arkady Kanevsky, NetApp, Inc.; Zhenmin Li, Pattern Insight, Inc.; Yuanyuan Zhou, University of Illinois at Urbana-Champaign

Weihang Jiang presented a tool for analyzing storage system logs to assist in customer troubleshooting. Today's complex storage systems need to deal with constant failures, which cause costly service downtime. Manual troubleshooting is also very costly for vendors. Problems may happen at different layers. Customer problem issues are reported to vendor support centers in two ways: human-generated reports (e.g., phone call, email) and automated built-in monitoring tools (e.g., storage system logs). Jiang said that manual processing of customer service requests often has a long turnaround time. He argued that by analyzing logs in a systematic way, it is possible to troubleshoot many problems automatically.

The authors analyzed a large problem database containing 600,000 problem cases and 300,000 logs. They found that hardware fault and misconfiguration are the main causes of problems. Software bugs caused only 3% of the errors, but they required a larger amount of troubleshooting time. Most of the customer problems are low-impact, and only 3% caused system crashes. Jiang described three techniques for analyzing logs: using critical events only; using single events; or combining multiple events. A score is computed based on how well the event signature can uniquely identify the cause. Jiang commented that they found logs to be noisy and verbose. Important log events are not easy to locate or link together. However, it is possible to identify and link patterns in the logs with specific types of problems. By applying clustering techniques, the tool described in the paper can help identify the causes of problems and help in troubleshooting.

An audience member asked what the authors do when an error starts as a hardware error but later causes software errors. Jiang replied that they only consider the initial cause when classifying errors. When asked why software errors are so expensive, Jiang explained that hardware errors are easy to solve simply by replacing the malfunctioning hardware component. But replacing software is not so easy. The audience raised the question of whether proactive action can be taken on the fly when a SCSI bus problem is detected. Jiang argued that not all SCSI errors lead to a problem, and finding the correlation between the SCSI error and a problem/crash is challenging. Finally, a question was raised about the nature of the study and whether the underlying system's properties changed during the study, e.g., after a software version update. Jiang replied that instead of a single type of system, they had studied a large number of logs from different storage systems, and most of the errors were caused by failing hardware rather than software. [See the related article on p. 31.]

- ***DIADS: Addressing the "My-Problem-or-Yours" Syndrome with Integrated SAN and Database Diagnosis***
  Shivnath Babu and Nedyalko Borisov, Duke University; Sandeep Uttamchandani, Ramani Routray, and Aameek Singh, IBM Almaden Research Center

Nedyalko Borisov presented DIADS, a tool that provides a holistic view of query execution and assists SAN and DBMS administrators during troubleshooting. The authors applied machine learning techniques and expert knowledge, and also implemented a data abstraction called Annotated Plan Graph (APG) that carefully integrates the DBMS and SAN monitoring data.

Borisov discussed various challenges in building the APG and how to solve them. He introduced a running example of SAN misconfiguration that causes performance degradation of a business intelligence query. In the DIADS workflow, the administrator first specifies the queries that had satisfactory and those that had unsatisfactory performance. The set of operators correlated with the query's performance are then identified, and an anomaly score is computed using the kernel density function. Later, these operators are used to look into related SAN components and calculate the anomaly score for them. Next, a symptom database is used to identify the root cause(s). Borisov discussed several challenges, such as expressiveness of the symptoms and missing symptoms. Once a root cause is identified, DIADS calculates its impact on the query performance. This reduces false positives and negatives and allows identification of high-impact root causes. Borisov concluded by presenting DIADS' evaluation scenarios, which consisted of incremental increase in complexity of the problems and investigation of the tool's ability to diagnose them.

An audience member asked whether DIADS can diagnose problems when the real SAN is replaced with a virtualized one. Borisov explained that monitoring data needs to be collected from the virtualization level of the SAN, and that DIADS will then be able to provide diagnosis. Another questioner asked whether the authors have considered creation of a performance metrics library to provide more useful monitoring data. Borisov argued that it is not known in advance when a problem will happen; thus, having intrusive data collection enabled all the time would cause a significant performance hit on the production system.

WORK-IN-PROGRESS REPORTS (WIPS) PART ONE

Summarized by Michelle Mazurek (mmazurek@andrew.cmu.edu)

- ***Progress on FileBench***
  Andrew Wilson, Sun Microsystems

Wilson discussed recent additions to FileBench, a model-based approach to improving file system benchmarking. Existing macro benchmarks are too time-consuming, while existing micro benchmarks are not comprehensive enough. In addition, the wide variety and lack of standardization in benchmarking can be frustrating. FileBench solves this problem by allowing the tester to model workloads in a high-level language, quickly run the test, and collect results. Recently added features include random variables, multi-client support, extension to support file sets as well as individual files, and composite operations that resemble inline subroutines. An NFSv3 plugin is nearing completion.

- ***When "More and More" Does Not Help: Sensible Partitioning of Cache***
  Hamza Bin Sohail, Purdue University

Bin Sohail presented a new approach to partitioning the buffer cache among applications. Current algorithms that do not partition the cache can result in cache hogging by certain processes, to their detriment. Bin Sohail proposes partitioning the cache and allocating larger portions to "good" processes whose instantaneous hit ratio increases as cache allocation increases. Simulation results indicate that statically allocating more cache to a historically "good" process and less cache to a historically "bad" process results in increased system performance. The next step is to partition the cache dynamically by monitoring processes as they run; it remains to be seen how effective this method will be and how much overhead it will require.

- ***Solving TCP Incast in Cluster Storage Systems***
  Vijay Vasudevan, Hiral Shah, Amar Phanishayee, Elie Krevat, David Andersen, Greg Ganger, and Garth Gibson, Carnegie Mellon University

Vasudevan discussed the TCP incast problem, which occurs when synchronized reads in a cluster environment cause TCP timeouts, resulting in a throughput collapse. The default 200 ms delay between TCP retransmissions is too long and wastes resources; can reducing or eliminating this lower bound provide a safe, effective, and practical solution? The standard TCP implementation relies on timers with millisecond granularity; a 5 ms timeout improves performance on systems with small stripe widths but fails for larger stripe widths. Using the Linux kernel's high-resolution timer with microsecond granularity, the incast problem can be avoided for at least 47 concurrent senders. As datacenters move toward increased bandwidth and more servers, response time at the latency of the network will be increasingly important. For more information, see tinyurl.com/incast.

- ***Improving I/O Performance by Co-scheduling of I/O and Computation on Commodity-Based Clusters***
  Saba Sehrish, Grant Mackey, and Jun Wang, University of Central Florida

Sehrish presented a framework for more efficient scheduling of Map-Reduce tasks in a fault-tolerant system like Hadoop. The scheduling algorithm improves efficiency by intelligently assigning multiple DFS (Distributed File System) blocks per map task, based on data locality. The algorithm considers three cases: dependent DFS blocks combined statically as an application requirement, independent DFS blocks combined statically to improve performance, and independent DFS blocks combined dynamically to improve performance. In the first case, the node with the most participating DFS blocks is chosen as the host node for the task, and each remaining block is retrieved from the node with minimal latency. The second case resembles the first, but the latency of transferring data to the host node is compared with the overhead of creating a separate task for the remote node to determine the optimal configuration. In the third case, one task is created for each set of co-located participating blocks; the number of tasks and the number of blocks per task are determined dynamically.

- ***Predictable and Guaranteeable Performance with Throughput, Latency, and Firmness Controls in Buffer-Cache***
  Roberto Pineiro and Scott Brandt, University of California, Santa Cruz

Tolerance of I/O performance degradation ranges from services that require hard realtime guarantees to those that require soft guarantees and even those that tolerate best-effort. Pineiro proposed a system that supports a mix of such services by providing different levels of predictable and guaranteeable performance in the buffer cache. To enforce hard guarantees, Pineiro focused on coordination of components in addition to conservative assumptions. The system uses device time utilization rather than softer metrics like bandwidth to manage devices, and it enforces hard isolation of components. I/O rate and deadline requirements are enforced both into and out of the buffer cache, which is partitioned according to I/O properties and performance requirements. Test results comparing this system to Linux using CFQ (Completely Fair Queuing) yield more stream isolation, more stable performance relative to the guarantee type, and a slight improvement in overall throughput.

- ***NFSv4 Proxy in User Space on a Massive Cluster Architecture: Issues and Perspectives***
  Philippe Deniel, Commissariat à l'Énergie Atomique, France

Deniel noted that server architectures are evolving from individual clusters to aggregations of clusters. This presents a problem for using NFS, as exponential growth in clients will overwhelm the servers. Deniel proposes using proxy servers based on the existing NFS-GANESHA tool and running in user space to solve this problem. Serving an aggregation of clusters with one main proxy server would be perfect for read-only workloads; in the real world, write, create, and delete operations lead to cache incoherency. To solve this, Deniel is developing a protocol for communication among proxy servers. The first implementation is expected by the end of the year.

- ***Comparing the Performance of Different Parallel File System Placement Strategies***
  Esteban Molina-Estolano, Carlos Maltzahn, and Scott Brandt, University of California, Santa Cruz; John Bent, Los Alamos National Laboratory

Molina-Estolano presented a trace-driven simulation approach to comparing file-placement strategies used by different parallel file systems. The goal is to compare only the file placement strategies rather than the file systems themselves. Simulated clients using various placement strategies are driven by traces from different workloads, including scientific computing and Web server workloads. The effects of normalizing chunk size and turning off redundancy were also considered. Preliminary results, which measured balance across the cluster, found the PanFS and Ceph strategies to be comparably balanced. Turning off redundancy in Ceph has limited effect, but turning off redundancy in PanFS increases balance. In addition, reducing chunk size in Ceph increases balance. Future simulations will measure performance in addition to balance. Molina-Estolano also discussed the need for more workload traces, particularly those related to data-mining and enterprise workloads.

- ***Overlapped HPC Checkpointing with Hardware Assist***
  Christopher Mitchell and Jun Wang, University of Central Florida; James Nunez and Andrew Nelson, Los Alamos National Laboratory

Mitchell noted that as high-performance computing systems get larger, they spend more and more time on failure mitigation processes such as checkpointing and error recovery, reducing system utilization. To solve this problem, either checkpoints must write less data or they must happen faster. Mitchell proposed adding a fast, non-volatile checkpoint buffer between the application and the file system. The prototype system will have three main components: a fleet of servers with connected buffers, a daemon to migrate checkpoint data from the buffer to the file system, and an API allowing application developers to access this checkpoint method. Currently, the servers and daemon are operational and the API is nearing completion. Preliminary testing shows significant improvements over existing methods.

- ***Moderated Collaboration to Modify Shared Files Among Wireless Users***
  Surendar Chandra and Nathan Regola, University of Notre Dame

Chandra presented a system for wireless file collaboration among multiple authors. Analysis indicates that contemporary users mainly use wireless devices such as laptops and are typically available for only short sessions with relatively long duration between them. Traditional approaches such as mandatory locking and epidemic propagation fail when multiple users are online at the same time. Chandra proposed creating one writable version of each collaboration file per user; users can also hoard read-only copies of other users' versions, distributed via epidemic propagation. Each user manually reconciles his changes with those of the other authors until convergence is achieved. Logs that track causal provenance allow users to determine whether their changes have been incorporated into the latest version. A prototype of this system, developed using FUSE, achieves acceptable performance in terms of memory used as well as file transfer time.

- ***Probabilistic Reputation for Personal Trust Networks***
  Avani Wildani and Ethan Miller, University of California, Santa Cruz

Wildani discussed the issue of trust verification in peer-to-peer storage. Out-of-band trust verification such as OpenPGP's web of trust is difficult and expensive. Wildani proposed an alternative approach where trust is calculated dynamically by individual nodes and used to make locally optimal decisions. Individual nodes sort peers into trust clusters, where the innermost cluster is most trusted. When a node successfully reads a file from a peer, the reputation of that peer is updated. When writing, a node sends out a number of replicas proportional to the perceived trustworthiness of the recipients. This system limits the severity of an attack: if a node is compromised, the attacker only gains information about that node's trust relationships. Because there is no central repository of reputation, there is no single attack point for poisoning trust relationships. Development of a simulated system using PlanetLab is in progress.

WORK-IN-PROGRESS REPORTS (WIPS) PART TWO

Summarized by Rik Farrow

- ***Can Clustered File Systems Support Data Intensive Applications?***
  Rajagopal Ananthanarayanan, Karan Gupta, Prashant Pandey, Himabindu Pucha, Prasenjit Sarkar, Mansi Shah, and Renu Tewari, IBM Research

Mansi Shah said that extreme data applications, such as Web page indexing and genome searches, demand a storage layer that is scalable and cost-effective, as well as fault-tolerant. Special file systems like Hadoop Distributed File System (HDFS) and the Google File System can also ship computation to the nodes that contain the data to be searched. Shah argued that cluster file systems, such as Lustre and IBM's GPFS, can do that as well. They experimented with GPFS, first increasing the block size, which worked poorly. But by changing the block allocation scheme to mimic a large block size, and through exposing block location via an ioctl() call, they were able to match the performance of HDFS while still maintaining performance for legacy tasks in GPFS.

- ***Data Destruction: How Can You Destroy Data and Prove It Is Destroyed?***
  Dan Pollack, AOL LLC

AOL leases systems and thus has a strong interest in data destruction techniques that do not involve destroying hardware. When they return equipment, they must be able to prove to auditors that they have destroyed any data in storage. At the same time, storage systems are getting larger: overwriting a disk at 1GB/sec translates into 3.6TB/hour, which is slow if you have petabytes to destroy. And the increased workload on drives during the overwriting process can result in failed drives before the overwrite can be completed. They saw a five-fold increase in device failures during the most recent attempt at destroying data. Pollack asked the community for help in coming up with effective ways of destroying data.

- ***SmartStore: A New Metadata Organization Paradigm with Semantic-Awareness***
  Yu Hua, Huazhong University of Science and Technology; Hong Jiang, University of Nebraska—Lincoln; Yifeng Zhu, University of Maine; Dan Feng, Huazhong University of Science and Technology; Lei Tian, Huazhong University of Science and Technology and University of Nebraska—Lincoln

Lei Tian posed the problem of finding files in very large systems where there may be millions of files and nearly an exabyte of storage. Tian said their system, SmartStore, is different from Spyglass (FAST '09) in that it groups and stores files according to their metadata semantic correlations. They use Latent Semantic Indexing to measure semantic correlations and construct multiple logical R-trees that improve search. Tian used graphs to demonstrate the dramatically improved performance of range searches (e.g., all files that took less than 30 minutes to generate and are less than 2.6GB) and top-k queries (e.g., find the top ten matching files) over a conventional file system and a DBMS with stored metadata. Their prototype emulates I/O behaviors of a large storage system by scaling up I/O traces both spatially and temporally for testing purposes.

- ***Making the Most of Your SSD: A Case for Differentiated Storage Services***
  Michael Mesnier and Scott Hahn, Intel Corporation; Brian McKean, LSI Corporation

Michael Mesnier introduced the notion of Differentiated Storage Services (DSS) as a method for making the most out of SSDs when mated with disks. They modified ext3 by adding policies that assign quality of service (QoS) levels for different classes of writes, with metadata, journal, directory, and small files being given priority. The main idea is to separate policy from hardware implementation so that file systems can assign a QoS to a write request that the hardware can optionally respond to. Mesnier argued that simply using an SSD coupled with a hard drive as a cache wastes potential performance gains. He also said that this work applies not only to SSD but to other storage hierarchies.

- ***On the Consistability of Storage Systems***
  Amitanand Aiyer, Eric Anderson, Xiaozhou Li, Mehul Shah, and Jay J. Wylie, HP Laboratories

Amitanand Aiyer defined consistability as an attempt to describe the different levels of consistency found in a storage system at any point in time. In a perfectly performing storage system, the system may provide atomic consistency. But in a system that experiences some fault 20% of the time, the system provides atomic consistency 80% of the time and regular consistency 100% of the time. An example of a fault would be network partitioning, where the ability to get that last value put into the system degrades into the ability to get one of K most recent values.

- ***Speedy and Scalable File-System Benchmarking with Compressions***
  Nitin Agrawal, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

Nitin Agrawal explained how their tool, Compressions, can be used to simulate very large storage systems. As the amount of disk storage has grown, it has become more difficult to create realistic simulations for benchmarking, and synthetic benchmarks are also hard to create. Compressions allows an evaluator to run a large benchmark using a much smaller disk—for example, 100GB to simulate 1 terabyte. They do this by doing away with all data blocks and laying out metadata blocks (inodes, directories, indirect blocks, etc.) more efficiently on disk. All data writes are discarded, and reads are supplied with mock data that may resemble what is expected. Delays are added to disk operations to simulate full-scale operation while using just 10% as much disk space.

- ***Out-of-Place Journaling***
  Ping Ge, Saba Sehrish, and Jun Wang, University of Central Florida

Ping Ge presented work on improving the performance of journaling file systems. Journaling file systems maintain system integrity through atomic writes by writing to the log (journal) and then by committing the log entries. The second step proceeds by copy-on-write (COW) or by updating pointers to the data written by the log. Ping Ge presented a third mechanism, which they have implemented and tested in ext3: they use a mapping layer between the file system and the device driver to map logical blocks. After data gets written to the log, the mapping layer commits this data by redirecting requests for the data to the blocks in the log. If the system crashes, description records in the log can be used to rebuild the mapping layer.

POSTER SESSION

Summarized by Madalin Mihailescu (madalin@cs.toronto.edu)

- ***On the Consistability of Storage Systems***
  Amitanand Aiyer, Eric Anderson, Xiaozhou Li, Mehul Shah, and Jay J. Wylie, HP Laboratories

Amitanand Aiyer proposed quantifying the consistability modes of a storage system under various operating conditions. The intuition is that systems that offer different consistency levels can be compared by estimating the percentage breakdown of the levels each system achieves in the presence of various failure scenarios. Current work is being conducted to better understand design/implementation trade-offs for a key-value store.

- ***Can Clustered File Systems Support Data Intensive Applications?***
  Rajagopal Ananthanarayanan, Karan Gupta, Prashant Pandey, Himabindu Pucha, Prasenjit Sarkar, Mansi Shah, and Renu Tewari, IBM Research

Mansi Shah argues that cluster file systems such as Lustre and PVFS can be tweaked to support data-intensive applications, thus eliminating the need for specialized file systems, e.g., GFS and Hadoop DFS. One advantage with this approach comes from deploying a single file system that supports both data-intensive and legacy applications. The authors changed IBM's GPFS to accommodate the Map-Reduce framework. Preliminary evaluation shows comparable performance with Hadoop DFS.

- ***Adaptive Context Switch for Very Fast Block Device***
  Jongmin Gim, Kwangho Lee, and Youjip Won, Hanyang University, Korea

Jongmin Gim notices that context-switching processes when performing I/O could be inefficient when using current non-volatile memory drives, such as SSDs. This is based on the observation that context-switch overhead can be as high as 300μs, while I/O access time on SSDs is an order of magnitude lower. To address this problem, the authors built an adaptive context-switch algorithm. The algorithm tries, by analyzing I/O response times, to determine whether context switch is beneficial. Preliminary results show performance improvements of up to 16%.

- ***SmartStore: A New Metadata Organization Paradigm with Semantic-Awareness***
  Yu Hua, Huazhong University of Science and Technology; Hong Jiang, University of Nebraska—Lincoln; Yifeng Zhu, University of Maine; Dan Feng, Huazhong University of Science and Technology; Lei Tian, Huazhong University of Science and Technology and University of Nebraska—Lincoln

SmartStore targets the problem of efficient metadata retrieval in large-scale storage systems. In particular, it focuses on range queries and top-k queries. It uses Latent Semantic Indexing to group semantically correlated files, based on their metadata. Furthermore, an R-tree is used to store the metadata, based on the grouping obtained from LSI. Compared against an R-tree scheme without semantic knowledge and a DBMS, SmartStore has very low query latency numbers.

- ***Comparing the Performance of Different Parallel File System Placement Strategies***
  Esteban Molina-Estolano, Carlos Maltzahn, and Scott Brandt, University of California, Santa Cruz; John Bent, Los Alamos National Laboratory

Esteban Molina-Estolano proposed a comparison among the various placement strategies implemented in current parallel file systems. He implemented a basic simulator for the placement strategies used in Ceph, PanFS, and PVFS. Preliminary evaluation using real and synthetic I/O traces showed the three file systems having comparable placement techniques in terms of balance. The chunk size and redundancy strategy used by each file system have an impact on the balance. Future work includes improving the simulator to allow for performance comparison.

- ***Supporting Data-Intensive Applications on Accelerator-Based Distributed Systems***
  M. Mustafa Rafique, Ali R. Butt, and Dimitrios S. Nikolopoulos, Virginia Tech

Mustafa Rafique argues that current large-scale clusters that leverage computational accelerators, such as GPUs, for high-performance computing implement either ad hoc or specific solutions. Thus, there is a need for understanding alternative designs in this space, depending on the capabilities of various accelerators, in the context of data-intensive applications. Accelerators are classified based on their compute power and, mainly, the extent to which they can manage external resources, e.g., I/O devices. This led to four configurations, which were evaluated against a number of Map-Reduce applications. The experimental setup was built using Sony PS3s and a multicore cluster. Future work will try to make the framework more generic.

- ***Exploiting the Overlap Between Temporal Redundancy and Spatial Redundancy in Storage System***
  Pengju Shang, Saba Sehrish, and Jun Wang, University of Central Florida

Pengju Shang proposed bridging the gap between application-level temporal redundancy techniques (e.g., a database log record) and storage-level spatial ones (e.g., RAID). The authors built a transactional RAID, or TRAID, for database systems, which aims to take advantage of the redundancy overlap. TRAID lowers the log wait time and log size while ensuring the database ACID semantics. Results show that TRAID can improve RAID by 40–50%, depending on the RAID version. Current work is being done to adapt this technique to versioning file systems, e.g., ext3cow.

- ***Using Realistic Simulation to Identify I/O Bottlenecks in MapReduce Setups***
  Guanying Wang and Ali R. Butt, Virginia Tech; Prashant Pandey and Karan Gupta, IBM Almaden Research

Dumbo is a simulator for the MapReduce framework. It can be used to analyze application performance by understanding the impact of various configuration parameters for typical MapReduce deployments, e.g., storage, compute capacity, network topology, or data layout. The analysis is done with minimal resources. A prototype implementation managed to uncover a network-related performance inefficiency in Hadoop, an open source version of MapReduce.

[Editor's Note: Many more posters were not summarized or have been summarized as WiPs. See http://www.usenix.org/events/fast09/poster.html for the full list of posters and their abstracts.]

SCHEDULING

Summarized by Brandon Salmon (bsalmon@ece.cmu.edu)

- ***Dynamic Resource Allocation for Database Servers Running on Virtual Storage***
  Gokul Soundararajan, Daniel Lupei, Saeed Ghanbari, Adrian Daniel Popescu, Jin Chen, and Cristiana Amza, University of Toronto

The paper includes two parts: building a latency model, and building a resource partitioner which operates on this latency model. They consider three resources in their models and controllers: file system cache, database buffer pool, and disk bandwidth.

To build the cache models, they observe that if the caches are LRU, then the larger cache dominates the smaller, so they simulate both the file system and database caches as a single cache of the size of the larger cache. To model the disk, they model the latency based on the latency the application would have with the disk to itself, and they assume a large sharing quanta. To handle cases where the models are inaccurate, they use cross-validation, and in regions where the models are inaccurate, they sample and interpolate.

Given these models, they can provide multi-level resource allocations, which give up to 3x performance improvement over a conventional single-level resource allocator.

- ***PARDA: Proportional Allocation of Resources for Distributed Storage Access***
  Ajay Gulati, Irfan Ahmad, and Carl A. Waldspurger, VMware Inc.

This paper focused on providing proportional resource allocation, based on tickets given to each VM to specify its relative importance, to virtual machines running on several different physical hosts without requiring coordination between the physical machines.

Host-level schedulers are not sufficient, since each host may have multiple VMs. To address the problem they keep a queue for each host based on the proportion of the tickets given to that host. To avoid problems with choosing metrics, each VM is given some number of slots which it is allowed to keep full at any given time. However, PARDA needs to be sure that the queue is appropriately deep to avoid increasing latency above a threshold.

To do so, each VM tracks the latency to a common shared file and then adjusts the queue length appropriately. Evaluations show that this allows PARDA to provide proportional sharing with minimal impact on performance.

- ***CA-NFS: A Congestion-Aware Network File System***
  Alexandros Batsakis, NetApp and Johns Hopkins University; Randal Burns, Johns Hopkins University; Arkady Kanevsky, James Lentini, and Thomas Talpey, NetApp

  Awarded Best Paper!

Conventional NFS systems are prone to problems with congestion on the network when too much traffic is going to the servers. However, in conventional systems the clients do not know how to trade off the various resources consumed by operations. CA-NFS approaches this problem by assigning a price to each resource in the system based on the current scarcity of that resource, and then combining to provide a utilization metric. CA-NFS considers the resources: server CPU, client and server network, server disk, client and server memory, and client read-ahead effectiveness.

This allows clients to make decisions about whether to send asynchronous writes or reads back to the server, for example, or whether to hold them in local memory instead. It does so by comparing the price it would be willing to pay to free its resources with those consumed by the server. Evaluations show that these methods provide a 20% performance improvement over standard NFS for several workloads.

TOOLS YOU WISH YOU HAD

Summarized by James Hendricks (James.Hendricks@cs.cmu.edu)

- ***Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality***
  Mark Lillibridge and Kave Eshghi, HP Labs; Deepavali Bhagwat, University of California, Santa Cruz; Vinay Deolalikar, HP Labs; Greg Trezise and Peter Camble, HP Storage Works Division

Mark Lillibridge presented his results on data deduplication for disk-to-disk backup. As disks get bigger and cheaper, backing up to disk rather than tape makes sense and provides many benefits. Unlike tape, the random access available with disks allows for deduplication of data. Data deduplication replaces duplicate data with pointer(s) to the original data. One approach to data deduplication is chunk-based deduplication, in which data is broken into chunks and the chunks are hashed. Under the standard of implementation, the hash values are then looked up in a table kept in RAM. If the hash value for a particular chunk is already present in the table, that chunk has already been stored, so only a pointer to that chunk is stored. If no hash value is found, that entire chunk is stored and its hash value is added to the table. The problem with this approach is that 100 terabytes of physical disk requires over 1 terabyte of hash values, which exhausts available RAM. One option is to store hashes on disk, but then each chunk lookup requires a slow disk lookup.

Storing hashes on disk but caching recently used hash values does not work, because backup streams exhibit little temporal locality. For example, a file will be read today, then terabytes of other data will be read, and then the file will be read tomorrow. Instead of temporal locality, backup workloads exhibit chunk locality, which means that if a chunk reoccurs, it tends to occur near other chunks that were nearby when it was last seen. Rather than tracking all hashes in RAM, this paper uses a technique called sparse indexing. Consecutive chunks of data are grouped into segments, and only a few hashes are sampled per segment. The sampled hashes form a sparse index that fits in RAM. To back up a segment, its samples are looked up in the sparse index to find previously backed up segments that contain a lot of chunks in common with the new segment. The new segment is then deduplicated against a few of the found segments by loading in from disk lists of the chunks contained in those segments. By chunk locality, over 99% of the duplicate data can be removed this way even though only a small number of segments rather than the entire store are deduplicated against. Thus, sparse indexing allows deduplication of large-scale backup data.

Hakim Weatherspoon of Cornell University asked how much of the gain was due to a common chunk such as the chunk of all zeroes. Mark said that removing the top 100 most common chunks often reduced the data size by 1%, but sometimes by up to 10%. Hugo Patterson, CTO of Data Domain, asked about read performance compared to the approach in last year's paper from Data Domain. Mark said that chunks are stored in the same way as last year's paper proposed, so read performance should be similar. Bill Bolosky of Microsoft Research asked why chunk size makes much difference. Mark said that files often aren't quite the same. Bill also questioned whether compressing data before storing would do as well as suggested in the paper, because a lot of data is already compressed (music, archives, etc.). Mark replied that it varies widely.

- ***Generating Realistic* Impressions *for File System Benchmarking***
  Nitin Agrawal, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

  Awarded Best Paper!

"For better or for worse, benchmarks shape a field" (Dave Patterson). Nitin Agrawal presented a tool, Impressions, that creates representative file system images for benchmarking. The community already knows properties of metadata and disk fragmentation, and Nitin argued that there is a need for an easy-to-use tool to create representative, controllable, and reproducible file system images. Impressions provides such functionality by taking file system distributions as input along with user-specified parameters such as total file system size. For example, Impressions can use the metadata distribution from the presenter's FAST '07 paper. Impressions has an advanced mode where several knobs can be turned and a basic mode that provides reasonable defaults.

Impressions uses a generative probabilistic model to create files and directories. Each directory is created, then a parent is chosen according to a probability model. Each file is created and its size, extension, and parent are created by a similar model. Traditionally, file sizes were assumed to be distributed lognormal. More recent studies have shown a bimodal distribution. Impressions' model of file system size is hybrid, using a lognormal body but a Pareto tail. Files are generated according to size model and then are attached to directories.

The tool will be available soon at http://www.cs.wisc.edu/adsl/Software/Impressions.

Ajay Gulati of VMWare asked if the tool could be augmented to run in reverse, such that one could run it on a machine to extract statistical properties and then generate a reasonable file system. Nitin said he had something similar for in-house testing but a full version for release is not available. Drew Wilson of Sun said FileBench already does this. Nitin replied that his contribution was to allow one to contribute newer designs and data sets and to make it easier to plug in distributions. Mike McThrow of Cal Poly San Luis Obispo asked if image creation is reproducible. Nitin said the random seeds can be set for reproducibility. Geoff Kuenning of Harvey Mudd University asked how long it takes to build a big image. Nitin said the tool currently writes images at10MB/sec, but the tool could be optimized to go much faster. The last questioner asked if filename length was modeled. Nitin said filename lengths in the data sets were anonymized.

- ***Capture, Conversion, and Analysis of an Intense NFS Workload***
  Eric Anderson, HP Labs

This paper describes new industrial-strength NFS tracing techniques needed to capture workloads at scale. The goal was to collect a customer's NFS traces, but standard techniques failed due to the huge volume of data. Many improvements were incremental, but all were needed to achieve the goal. Eric wanted to highlight two big takeaway points. First, if you take traces, read the paper and apply the techniques. For example, future traces should not drop packets—lindump, driverdump, or endacedump can capture traces without dropping packets. The improved data analysis techniques allow for handling these gigantic traces on modest systems. Second, if you need workloads, look at the ones he describes here. The workload is much different and significantly larger than prior workloads. All of the data and tools are open-sourced under a BSD license, so anyone can reproduce the results of the paper or conduct further analysis. Eric argued that there is an acute need for new traces. There are many different workloads but few traces over the past decade, so trace-based studies do not reflect most modern workloads. Eric encouraged the community to publish more traces and more trace analysis.

The customer was a feature animation (movie) company. The applications read models, textures, and animation curves and wrote intermediates and pictures. There were thousands of clients, tens of NFS servers, twenties of NFS caches, many rack switches, and a few core routers. The workload was different from many that have been previously studied. For example, most files have a single read, so any prefetch mechanism must work across files. The workload was very intense, which reduces the need to arbitrarily speed up a replay of the trace to evaluate a system. The workload also had many small files, which means that replaying the trace would really stress a system's performance.

The tools are available from http://tesla.hpl.hp.com/opensource/, and the traces are available from http://apotheca.hpl.hp.com/pub/datasets/animation-bear/.

Brent Welch of Panasas asked if Eric could figure out the working set for the textures and models. Eric replied that he didn't do much analysis on the trace, but he hopes other researchers will explore the trace in more depth. Brent Callaghan of Apple asked if packet reassembly was done to process readdir. Eric replied that a flaw discussed in the paper prevented this.

METADATA AND OPTIMIZATION

Summarized by Avani Wildani (agadani@gmail.com)

- ***Spyglass: Fast, Scalable Metadata Search for Large-Scale Storage Systems***
  Andrew W. Leung, University of California, Santa Cruz; Minglong Shao, Timothy Bisson, and Shankar Pasupathy, NetApp; Ethan L. Miller, University of California, Santa Cruz

Searching through petascale storage systems has become more and more difficult to manage. Current techniques include crawling metadata or building a DBMS that mirrors the system's metadata. Leung described the approach used in Spyglass as creating a versioned metadata index that gets stored as part of the storage server.

The authors surveyed users to get a list of requirements and analyzed storage systems used at NetApp and UCSC. They found that most searches involved multiple search parameters and had strong locality. For example, a user searching for lost files only needs to search within her own partition of the storage system. Spyglass takes this further, with hierarchical partitioning and signature files that use a bit map to provide hints for the classes of metadata found within each partition. Spyglass also takes advantage of a feature of NetApp's WAFL file system, snapshots, so that updating the index only involves looking at new versions of files. They compared the performance of Spyglass to systems based on PostgreSQL and MySQL and found that Spyglass could answer most queries in less than a second, something that the DBMS versions could only occasionally accomplish.

An audience member asked how they deal with directory renaming. Andrew answered that since the index itself is versioned, the next version reflects the rename. Versions are merged over time to recover space, and the partitioning strategy isn't strict, so even if the directory is moved, the worst that happens is that two partitions get searched.

- ***Perspective: Semantic Data Management for the Home***
  Brandon Salmon, Carnegie Mellon University; Steven W. Schlosser, Intel Research Pittsburgh; Lorrie Faith Cranor and Gregory R. Ganger, Carnegie Mellon University

Brandon Salmon made a polished presentation clearly demonstrating the need for semantic file naming and transparent file migration for nontechnical users. He began by describing case studies of users to learn how file names are used. As an example, Brandon described how a student might want to find a song by a particular artist and copy it to another device she owns. The problem is that iTunes, for example, organizes files semantically, by artist, album, and song, whereas the Mac Finder uses hierarchical names that do not map clearly to the iTunes view.

Perspective addresses the core issues in several ways. It captures the decentralized nature of devices by being P2P. It allows semantic management of data, allows for rule-based data management, and provides a way for automation tools to do things for the user while giving the user readable feedback. Perspective provides a global namespace across devices. Files are accessible through FUSE, and any replica of a file can be modified at any time. Devices aren't forced into a topology, and conflicts are handled similarly to previous systems. Views are used in file management: if a user wants all of her files on a given device—on a cell phone or a desktop, for example—she can specify that in a view. An automated system may ask to move files across from the phone to the desktop if the phone fills, and it will modify the views as it does. The views are human-readable, to make it easy to customize any automated decisions.

The first question was what happens if devices run out of space when copying a file for redundancy. Brandon answered that Perspective will never drop a file. The next questioner asked about results showing 60% accuracy in user testing of Perspective and wondered what was difficult. Brandon said that the interface they designed is overwhelming at first, and that the notion of hierarchy, as displayed with click and expand, is difficult for some people. After the Q&A completed, Brandon continued to be plied with questions.

- ***BORG: Block-reORGanization for Self-optimizing Storage Systems***
  Medha Bhadkamkar, Jorge Guerra, and Luis Useche, Florida International University; Sam Burnett, Carnegie Mellon University; Jason Liptak, Syracuse University; Raju Rangaswami and Vagelis Hristidis, Florida International University

Medha Bhadkamkar explained that BORG uses a special partition, called BOPT, as a write cache and for storing frequently accessed blocks. They created heat maps for different workloads (office, developer, subversion server, and Web server) and discovered that unique reads are a small portion of disk data but are spread out over the entire volume. Also, non-sequential block accesses repeat on certain workloads, and there is substantial overlap in the working sets across days. Thus, past I/O information can be used to reorganize data and improve performance. BORG identifies block access patterns in the workloads and copies them sequentially to the BOPT partition. The size of the partition is controlled by an administrator and includes room for a write buffer, reducing seek latency.

BORG operates in the background, is independent of the file system, and can be dynamically inserted and removed. It maintains consistency with a page-level consistency map. The architecture consists of user space components, the analyzer and planner, and kernel space components, the profiler, indirector, and the BOPT-space reconfigurator. The analyzer operates when needed and creates a weighted, directed graph representing the frequency of the accesses between nodes. These graphs become a master graph for the planner component to create a new layout. The indirector directs all writes to the BOPT partition as well as reading blocks stored in this partition. This process is iterative and continuous. If the BORG module is removed, the dirty blocks are copied back into the file system. Disk-busy times were reduced by up to 80% with optimal parameters.

Someone wondered that as the workload on the Web server was 1% busy time, what the effect was on response time. There was an improvement of up to 46%. Another person wondered what happens if the BOPT partition is full and a write occurs. The write buffer was full during the sensitivity analysis but not for the rest of the experiments. The final questioner asked Medha to compare this work to the performance of a logging file system (LFS). Medha responded that this work is based on LFS in that it tries to make writes sequential, but it also has good read performance, unlike LFS.

DISTRIBUTED STORAGE

Summarized by Dutch Meyer (dmeyer@cs.ubc.ca)

- ***HYDRAstor: A Scalable Secondary Storage***
  Cezary Dubnicki, Leszek Gryz, Lukasz Heldt, Michal Kaczmarczyk, Wojciech Kilian, Przemyslaw Strzelczak, and Jerzy Szczepkowski, 9LivesData, LLC; Cristian Ungureanu, NEC Laboratories America; Michal Welnicki, 9LivesData, LLC

Michal Welnicki described HYDRAstor as a scalable deduplication system for the enterprise market. Deduplication systems are faced with huge volumes of data, requiring scalability at a low cost per terabyte. They also must perform deduplication globally, be able to differentiate between high and low value data, and provide failure-tolerant restore performance. HYDRAstor is a system initially from NEC research but now successfully commercialized as a grid storage platform at a variety of capacities. Its architecture consists of front-end nodes that export an NFS or CIFS interface and can be scaled for performance, and back-end nodes that can be scaled for capacity.

The basic unit of storage in HYDRAstor is a synchron, a collection of subsequently written blocks stored linearly on disk. Metadata for the system is stored separately for performance reasons. Failure tolerance is provided through erasure coding with a standard N of M loss model. Data in HYDRAstor can be located at any of the back-end nodes, and request routing is performed with a DHT. Scalability, load balancing, and data relocation provide dynamic performance stability, which was said to be challenging to implement in concert with deduplication. In his evaluation, Welnicki showed an instance of HYDRAstor operating at a full throughput of 600MB/sec. Then a node failure is introduced and the system begins reconstruction. This has the effect of dropping the user's throughput to 400MB/sec.

Niraj Tolia of HP Labs asked Welnicki if hashes were evaluated when data is written out to avoid hash collision. Welnicki clarified that they weren't, but they agreed that they themselves didn't believe that such precautions were necessary, given the unlikelihood of such an event.

- ***Smoke and Mirrors: Reflecting Files at a Geographically Remote Location Without Loss of Performance***
  Hakim Weatherspoon, Lakshmi Ganesh, and Tudor Marian, Cornell University; Mahesh Balakrishnan, Microsoft Research, Silicon Valley; Ken Birman, Cornell University

Hakim Weatherspoon presented Smoke and Mirrors, which attempts to provide a higher-performance solution to safely mirroring data on geographically remote servers, specifically targeting industrial applications. Data stored in a single cluster is vulnerable to loss if the cluster experiences a catastrophic error. Since such a data-loss event could be disastrous, geographically remote shadowing of data is a very attractive option. However, designers of such a system are faced with a key decision with respect to performance and consistency: when can data be reported as safely reaching persistent storage? The performant option is to send a confirmation when the data reaches nonvolatile RAM on the local cluster; however, this is not the most reliable option. A safer but slower approach is to wait for data to be confirmed at the remote location, a process that is fundamentally limited by the speed of light.

Smoke considers a middle-ground position where confirmations can be sent once the border router transmits data towards the remote site. Since datacenters operate over low error-rate fiber, error rates nearing those of disks can be achieved by sending multiple copies of each packet to the network. Forward error correction is used to send the redundant packets and works as an efficient way to mitigate network error conditions. The result is a system that is safer than the local synchronization case and faster than the remote synchronization case. Weatherspoon was very clear in stating that the position was a compromise in which data could still be lost.

The system was evaluated with Emulab over 1GB links. Congestion and latency were introduced to simulate a geographically remote fiber connection. Weatherspoon showed how in this scenario Smoke was able to operate without data loss, at the cost of 3 extra network packets per 8 packets of goodput, and with very low overhead. Weatherspoon pointed to moving the experiments onto large private ring networks as the next step towards wider adoption.

The audience's response included concern from two attendees about the assumption that each packet would experience error in a probabilistically independent manner. Weatherspoon pointed to the forward error correction as handling some of the errors associated with, for example, transient network congestion that could result in a batch of dropped packets. David Rosenthal of Stanford admonished everyone present to monitor the loss rate on their networks (as was done in Smoke), and Stephen Spackman at Quantum praised Weatherspoon for "standing up to the tyranny of TCP."

- ***Cumulus: File System Backup to the Cloud***
  Michael Vrable, Stefan Savage, and Geoffrey M. Voelker, University of California, San Diego

Michael Vrable presented his work on Cumulus, which is a backup system designed for use with cloud-based storage systems. Cloud storage services are an emerging area of interest. The very high reliability that can be offered promises to greatly simplify backup, which continues to be a difficult problem. However, there is no clear solution to integrating with such a system.

Vrable began by classifying cloud storage systems along a spectrum from thick to thin. At the thick end, every aspect of storage is integrated, providing better efficiency and easier use. At the thin end, the user is provided with basic building blocks, but must develop their storage application independently; Amazon's S3 is an example.

Cumulus addresses the question of how effectively one can develop a backup service with the simple interface provided by thin cloud services. It operates over a simple get/put object API, with other operations being developed to run on the client. The goal of the system is to minimize resources and costs. Internally, a backup is structured as a directed acyclic graph that mirrors the file system hierarchy. Over time, new backups are created by duplicating the previous root node and using copy-on-write. The backup model is therefore incremental after the initial backup. Complicating this design is the fact that some storage providers may have per-file costs. To optimize for this, files can be collected into larger segments, but at the cost of potential segment fragmentation. Such a segment would increase costs by adding storage overhead. Cumulus implements a defragmentation utility to mitigate this problem.

In evaluating the system, Vrable showed the result of a seven-month trace of user data consisting of small data updates on the order of 10MB/day of new content and 30MB/day of modified content. Cumulus was compared in simulation against an optimal backup strategy. The assessment concluded that cleaning is necessary, as it maintains 95% segment utilization versus 50% without cleaning. It also established optimal values for segment size and cleaning threshold given a particular pricing structure. Two existing tools for S3 were found to operate at storage costs 19%–200% higher than Cumulus. Cumulus was also said to be competitive with Mozy, a thick cloud service that provides unlimited storage at $5 per month for noncommercial users.

Irfan Ahmad from VMware asked if Cumulus could provide deduplication or data scrubbing services. Vrable answered that Cumulus relies on the provider to provide reliability, although there is related work on auditing a service provider. As a client-oriented service, deduplication could be provided on the client side but could not capture duplication across clients.

DATA INTEGRITY

Summarized by Brandon Salmon (bsalmon@ece.cmu.edu)

- ***WorkOut: I/O Workload Outsourcing for Boosting RAID Reconstruction Performance***
  Suzhen Wu, Huazhong University of Science and Technology; Hong Jiang, University of NebraskaóLincoln; Dan Feng, Huazhong University of Science and Technology; Lei Tian, Huazhong University of Science and Technology and University of Nebraska—Lincoln; Bo Mao, Huazhong University of Science and Technology

As systems increase in scale, online RAID reconstruction is likely to become a common mode of operation. To address this problem, WorkOut uses a "surrogate array" made of spare resources, such as spare RAID disks, to provide improved performance for arrays currently performing reconstruction.

Writes to new data, or writes that hit in cache sent to the reconstructing array, will be redirected to the surrogate array, and subsequent reads to this data can also be sent to the surrogate array. To route requests, the array keeps a mapping table in NVRAM. Once the array has finished reconstructing, the data on the surrogate array is recopied onto the original array, although overwrites may help speed this process as well.

Evaluation shows that the use of a surrogate array can decrease reconstruction time by up to 5x and also improves the latency of the foreground workload.

- ***A Performance Evaluation and Examination of Open-Source Erasure Coding Libraries for Storage***
  James S. Plank, University of Tennessee; Jianqiang Luo, Wayne State University; Catherine D. Schuman, University of Tennessee; Lihao Xu, Wayne State University; Zooko Wilcox-O'Hearn, AllMyData, Inc.

James Plank summarized and evaluated a variety of erasure coding schemes and libraries in order to give system designers without deep erasure coding expertise the ability to evaluate libraries for use in their systems.

The evaluations had several key results. First, the open-source erasure coding libraries can keep up with disks, even on slow processors. Within the codes, special-purpose RAID-6 codes were more efficient than general-purpose Reed-Solomon codes. Cauchy Reed-Solomon codes were also more efficient than Reed-Solomon codes. For Cauchy Reed-Solomon codes, the matrix choice was important to performance.

On the Mac machine they tested, the number of XORs was a good measure of performance, but caching behavior was also important on the Dell machine tested.

- ***Tiered Fault Tolerance for Long-Term Integrity***
  Byung-Gon Chun and Petros Maniatis, Intel Research Berkeley; Scott Shenker and John Kubiatowicz, University of California, Berkeley

This paper describes Bonafide, a key/value pair store designed to maintain fault tolerance over a long period in the face of Byzantine faults. To attack the problem, Bonafide divides the service into two tiers: a trusted tier, which must not fail, and an untrusted tier, which can fail frequently without compromising the service.

The trusted tier is able to make changes to state, while the untrusted tier is able to respond to read requests but not change the actual state of the system. This division allows Bonafide to leverage a large number of untrusted devices to prolong the life of the service while still maintaining appropriate fault-tolerance.

CONTROLLERS AND CACHING

Summarized by Chris Frost (frost@cs.ucla.edu)

- ***A Systematic Approach to System State Restoration during Storage Controller Micro-Recovery***
  Sangeetha Seshadri, Georgia Institute of Technology; Lawrence Chiu, IBM Almaden Research Center; Ling Liu, Georgia Institute of Technology

Disk controller firmware contains many interacting components (e.g., RAID, I/O routing, and error detection) and completes many asynchronous (concurrent) and short-running tasks per second. Current firmware recovers from transient errors by rebooting, halting all drive progress for around four seconds. Seshadri et al. want to increase the availability of drives without rewriting the large software base of existing controllers while supporting high performance and dealing with dynamic dependencies and complex recovery semantics. Their approach is to enable per-task recovery when possible, falling back to a system restart only very rarely.

When an error occurs within a task, the task may continue (ignore/correct for error) or retry (rollback), but only if no other tasks have been affected by state changes made by the failed task. The basis of Log(Lock) follows from all global state modifications being protected by locks (or similar primitives). These locks guide recovery. Log(Lock) tracks recovery points, starting points for recovery upon error, and whether each recovery point is safe to use given the current system state. Log housekeeping is split between the system and code the developer has added. The developer adds start-and-stop tracking calls associated with locks. With Log(Lock), a typical error causes a 35% throughput decrease for six seconds. A reboot would take four seconds and have zero throughput. In their experiments, task-level recovery is applicable for 99% of errors, and runtime overhead is less than 10%.

A student asked how interdependent the threads of context are or how often dirty reads exist. Seshadri answered that although dirty reads are present most of the time in many types of systems, they have seen that in disk controllers they rarely exist, because the time scales are so small. For example, a typical task completes in five milliseconds.

- ***CLIC: CLient-Informed Caching for Storage Servers***
  Xin Liu, Ashraf Aboulnaga, Kenneth Salem, and Xuhui Li, University of Waterloo

Kenneth Salem presented CLIC, an approach to implementing a hinted two-tier caching system. CLIC learns how to respond to hints rather than using prebuilt, ad hoc rules. Multiple levels of caching introduce two issues: caching an item at multiple levels can waste cache space, and caches farther from the client do not see all requests and so their temporal locality is poor. Cache hints—for example, a client writing a page with the intent to replace another page—allow caches to behave appropriately. However, existing hint implementations use manual rules, which do not support new hints without changes, may have poor responses to hints, and can be difficult to implement when multiple clients use the cache. CLIC is a hint-aware cache replacement policy for second-tier caches that learns appropriate hint responses rather than relying on manually specified rules. CLIC separates the generation of hints (generated by clients) from the interpretation of hints (the storage server).

In CLIC, each page in the cache is associated with the hint set with which it was most recently read or written. CLIC orders the hint sets by their learned priority and evicts pages associated with the lowest-priority hint set. Priorities are determined using the results of previous requests for the given hint set. CLIC takes steps to reduce the space needed to track these statistics; in their measurements, CLIC needed less than 1% of the cache space for this tracking.

Kenneth compared the TPC-C and TPC-H hit ratios achieved by CLIC, an ad hoc hint policy (TQ), two policies that do not use hints (LRU and ARC), and the optimal policy (knowledge of the future) for varying cache sizes. CLIC and TQ usually dominated LRU and ARC, CLIC often dominated TQ, and OPT typically dominated all.

One audience member asked how CLIC performs with a large number of clients and with a varying dynamic workload. Kenneth answered that they have not tested with a large number of clients or time-varying workloads. However, at present, CLIC occasionally throws out all logs, using exponential decay. Another person asked how CLIC responds to clients that use hints to try to hurt other clients. Kenneth responded that CLIC helps the clients that benefit the most from the cache. Finally, someone asked whether the authors had thought of CLIC giving feedback to the client about which hints are the most useful. Kenneth said they had not and that this might be interesting.

- ***Minuet: Rethinking Concurrency Control in Storage Area Networks***
  Andrey Ermolinskiy and Daekyeong Moon, University of California, Berkeley; Byung-Gon Chun, Intel Research, Berkeley; Scott Shenker, University of California, Berkeley, and ICSI

Andrey Ermolinskiy discussed two limitations of using distributed locking to coordinate reads and writes among clients of a Storage Area Network (SAN), and he presented a new coordination approach that addresses these two limitations through optimistic, instead of strict, concurrency by adding logic to storage system nodes.

Using distributed locking to coordinate access to shared state has two issues: (1) it does not guarantee correct serialization of requests, and (2) it requires a majority of the locking server nodes to be available. Minuet guarantees the correct serialization of disk requests and removes the need to contact any locking server. Instead, the storage nodes themselves, via a guard, mediate requests and can reject incorrectly ordered requests. In Minuet, requests are augmented with session annotations that are used to order the requests. Distributed transactions can be constructed atop Minuet using logging and recovery.

Remaining challenges to Minuet-like systems include the adoption of guard logic into storage arrays, storage and bandwidth overheads of session metadata, and the programming model change of request rejection and forced lock revocations.

One audience member asked why an equivalent system cannot be built on top of SCSI-3 reservations. Andrey responded that he had not considered this. Another person asked about the efficacy of caching version numbers for concurrency control. Andrey answered that their implementation does scale; version numbers are stored in NVRAM on storage devices and perhaps could be stored on disk with some also in RAM.

First Workshop on the Theory and Practice of Provenance (TaPP '09)

February 23, 2009
San Francisco, CA

INVITED TALK

- ***Causality, Responsibility, and Blame: A Structural-Model Approach***
  Joe Halpern, Cornell University

  Summarized by Kiran-Kumar Muniswamy-Reddy (kiran@eecs.harvard.edu) with assistance from Peter Macko

Halpern's talk introduced the theoretical aspects of causality, responsibility, and blame and their implications for the provenance community.

The world can be modeled using structural equations that model the outcome of events using one or more random variables. The values of exogenous variables come from outside the model, while endogenous random variables depend on other variables in the system. There are, however, two choices of uncertainty over which reasonable people can disagree: first, we do not know whether our model is correct or sufficiently detailed, and, second, we may not be certain about the values of some variables. Consequently, we should introduce probabilities into the model that reflect our confidence in it. A model with no probabilities is completely deterministic.

Causality can be informally defined as follows. A set of events A caused event B, assuming that both A and B actually happened. Changing the outcome of A and possibly also the outcomes of some other events X would cause B not to happen. But if A happened, so would B, regardless of the outcomes of X. For example, this definition can gracefully handle the following scenario: it is sufficient to drop one match in order to burn a forest. If two people drop matches and the forest burns down, the definition correctly identifies both of them as the causes.

This definition of causality has one very important implication: it is not transitive. This challenges the interpretation of provenance as a form of causality, since provenance is believed to be transitive. The intransitivity of causality can be illustrated using the following example: a patient has an illness that can be cured by one dose of medicine, but two doses would kill him. On Monday, doctor A gives the patient a dose of medicine. On Tuesday, doctor B does not give him the medicine, because he already received it the day before. On Wednesday, the patient is alive and well. Clearly, doctor A caused doctor B not to administer the medicine, and by doing so, B caused the patient to be alive. If causality were transitive, A would also be a cause that the patient is alive, but this is not true: if doctor A did not give the medicine, the patient would still be alive on Wednesday, regardless of the subsequent action of doctor B.

While causality is binary, the degrees of responsibility and blame can be expressed as numbers between 0 and 1. Your responsibility is 0 if you are not a cause, or 1/(k+1), if k–1 is the minimum number of variables that must be changed in order to change the outcome of the event. For example, in a 6–5 voting scenario the responsibility of the six voters is 1, while in an 11–0 scenario the individual degrees of responsibility would be smaller. The degree of blame is the expected degree of responsibility given all possible situations considered by an agent. For example, consider the following situation: ten marksmen are ordered to shoot a prisoner. Nine of them receive fake bullets and only one gets a live bullet, but no one knows which one. If none of them misses the target, only the marksman with the live bullet is responsible with degree 1, while each of them has the same degree of blame 1/10.

Moving on to applications: causality can be used for model checking. If there is an error in the specification of a program, causality can be used to perform coverage estimation. In particular, one can ask the question, "Which parts of the spec cause the program to be satisfied?" If 90% is irrelevant, then the spec is flawed. Causal models can also be used when one is uncertain about provenance.

One member of the audience asked if the model the speaker presented considered the Trio definition of provenance. The speaker was unfamiliar with Trio, but it seems as though the two models address the same issues. For example, Trio tries to answer questions such as "Why does a tuple appear in a result?" which is basically causality.

MORNING SESSIONS

Summarized by Kiran-Kumar Muniswamy-Reddy (kiran@eecs.harvard.edu)

- ***A Formal Model of Provenance in Distributed Systems***
  Issam Souilah, University of Southampton, UK; Adrian Francalanza, University of Malta, Malta; Vladimiro Sassone, University of Southampton, UK

In this talk, the speaker presented a provenance-based calculus to study trust in distributed systems, in particular, a formalism that is an extension of Pi-calculus. The basic idea is to annotate all data with their provenance. Users can use this provenance to make decisions; for example, a product made in China may be more desirable than a product made in Zimbabwe. They considered a few more approaches before deciding on this approach: static analysis does not scale. A dynamic analysis cannot do a full-blown verification or use proof-carrying code as decisions. Decision criteria need to be computationally lightweight. With their method, provenance tracking is automated and is orthogonal to programming. Their approach also ensures provenance annotation standardization and provides circular reasoning with respect to trust. One more contribution of the work is that they provide a definition for provenance correctness and prove the correctness for the provenance-tracking semantics that they have proposed.

One member of the audience asked what happens if the same value is sent to two different processes in two different channels. The speaker said that the messages sent on individual channels are considered to be different copies, so their approach holds. Next, James Cheney asked what happens if one of the entities lies and sends the wrong provenance. The speaker replied that they are assuming that the system is running in a trusted environment, i.e., the principals tell the truth about themselves so others can make decisions based on that.

- ***Towards Semantics for Provenance Security***
  Stephen Chong, Harvard University

  Summarized by Kiran-Kumar Muniswamy-Reddy (kiran@eecs.harvard.edu) with assistance from Peter Macko

Stephen Chong presented formal definitions for provenance security that ensure that we do not reveal sensitive data and the provenance does not reveal sensitive information. The work assumes a simple language-based model, where the program has input locations and produces single output. The provenance trace T describes the execution of a program. The partial provenance of T allows parts of T to be elided (hide some values or even entire statements from T). At this point, an audience member asked who was deciding what provenance to elide. Stephen replied that the system assumes the existence of some access control scheme that basically tells whether some provenance should be elided. Each input location has a security policy for data and provenance (high or low security). A user knows the values of low-security inputs and is given output and partial provenance trace.

Basically, users are allowed to know the existence of objects but may not be allowed to know that they were involved in generating an output. In this system, T is secure if it accurately describes the execution and does not contain any high provenance locations, but that does not prevent reasoning using correlations. T satisfies provenance security if it is accurate and there exists another execution where the high provenance point is not even involved.

One member of the audience asked if there was a unique minimal reduction. The speaker replied that this was still an open question.

- ***Scalable Access Controls for Lineage***
  Arnon Rosenthal, Len Seligman, Adriane Chapman, and Barbara Blaustein, The MITRE Corporation

Adriane Chapman described securing the sensitive information that lineage contains. Role-based Access Control (RBAC), used in prior lineage security work, does not scale. RBAC might lead to situations where a new role has to be created for each user, leading to role explosion. Instead, Chapman proposed a model based on Attribute Based Access Control (ABAC). In classic ABAC, attributes are used to determine access. If the attributes, such as age or taste, satisfy some properties, then access is allowed. However, ABAC rules are hard to change once specified. Instead, they allow stakeholders to specify the access allowed, and when the stakeholders have conflicting opinions, they reconcile them. In their system, the basic ownership defaults are that the process node is owned by the process creator, the data node is owned by the data creator, and the edge between nodes is the union of stakeholders. They have further extended ABAC to return a fake node if a user is denied access to a node.

Margo Seltzer pointed out that not just the nodes of the provenance graph but even the edges connecting the graph have provenance. Chapman replied that taking the union of the permissions at the edges is sufficient. Next, Erez Zadok asked if they had considered the MLS model, to which Chapman replied that they had not thought about it. Another member of the audience asked if giving away a bogus node isn't giving away some information. Chapman replied that they are still working on this aspect. Another member asked if the fact that some entities have the ability to create nodes does not leak information. Chapman replied that they are looking into it.

- ***On Explicit Provenance Management in RDF/S Graphs***
  P. Pediaditis, G. Flouris, I. Fundulaki, and V. Christophides, ICS-FORTH

The context of the work is provenance management in RDF/S (a collection of data and schema triples). RDF/S (Resource Description Framework Schema language) is used to add semantics to RDF triples by imposing inference rules. This entails new implicit triples (facts) that are not explicitly asserted. Deletion/update of some triples will ensure that some of the implicit information will be lost, even though some of that information is still valid. To solve this, they introduce RDF/S graphs that help these issues by performing queries and updates.

RDF/S graphs are a set of RDF named graphs that are associated with a URI and have a set of triples whose ownership is shared by the named graphs that constitute the graph set. The authors further extended RQL to handle provenance queries and to support updates to graph sets through an extended version of RUL.

An audience member said that if things change, inference might change, so why should we care about retaining it? The speaker replied that this is because they are using coherent values and not operational semantics. Another member of the audience asked if they had studied whether querying semantics could be simulated with foundational or operational semantics. The speaker replied that there has been a lot of study in the literature, but there is no clear marking between the two.

- ***Application of Named Graphs Towards Custom Provenance Views***
  Tara Gibson, Karen Schuchardt, and Eric Stephan, Pacific Northwest National Laboratory

Tara Gibson described making provenance more accessible to users. Workflow provenance is very detailed and presents a machine view of things, which is probably much more detailed than a human can understand. In this talk, Tara presented a filtering technique that extends SPARQL (SPARQL Protocol and RDF Query Language) to help avoid information overload. Furthermore, since this is a generic extension to the query language, users do not have to constantly rewrite code every time they want to customize their views.

In particular, their approach leverages extensions to RDF Named Graphs (NG). They extend SPARQL to include the new keyword APPLY, which tells the query interface what views should be applied to the query result. Based on the views specified, each NG is grouped into a user-defined node. The new node is then created based on the properties associated with the NG definition. They then restore links between the new aggregate nodes in the result. Currently, they are working on implementation of the system.

An audience member asked if they know of any properties that must hold for their extension to transform the nodes and whether their method destroys some of these properties. The speaker replied that since they still maintain the original graph at the lower level, users can get back to it if they want to. Another member from the audience asked if they allow users to abstract things at runtime. The speaker replied that in theory it is possible.

- ***Authenticity and Provenance in Long Term Digital Preservation: Modeling and Implementation in Preservation Aware Storage***
  Michael Factor, Ealan Henis, Dalit Naor, Simona Rabinovici-Cohen, Petra Reshef, and Shahar Ronen, IBM Research Lab in Haifa, Israel; Giovanni Michetti and Maria Guercio, University of Urbino, Italy

Michael Factor discussed his team's work on long-term digital preservation, which involves processes, strategies, and tools to allow future usability of digital assets. Their work leverages the Open Archival Information System (OAIS) standard for digital preservation. It consists of content information (data and representation information) and preservation descriptive information (reference, provenance, context, fixity, and representation information).

Architecting a preservation store that supports authenticity and provenance is a challenge. To this end, their work (and the talk) presented a novel model for managing authenticity in a preservation environment and an implementation that integrates the concept of provenance.

Someone asked if the OAIS standard makes it easy to preserve information. The speaker replied that their work provides a concrete implementation of the model. Another audience member asked if there are scenarios where a bit stream can change without semantics also changing. The speaker replied that an example of this is when you change the format of a document from MS Office 97 to Office 2000. The challenge is to ensure that data has not been corrupted during the migration from one format to another. A final questioner pointed out that many archival experts do not consider format conversion an appropriate mechanism and prefer the use of virtual machines. The speaker replied that some data is application-independent, and that does need to be updated as systems are updated.

- ***Steps Toward Managing Lineage Metadata in Grid Clusters***
  Ashish Gehani and Minyoung Kim, SRI International; Jian Zhang, Louisiana State University

Ashish Gehani explained that their work is set in the context of the grid, i.e., distributed systems with non-interactive workloads that involve a large number of files. The goal is to provide low-latency lineage queries that enable a number of applications: for example, dynamic toolchain selection, safety/reliability (check tool dependencies), etc. The speaker then presented a set of discarded approaches which included use of auxiliary files, a local database, in-band encoding, and headers and footers to store provenance, as well as making the file server provenance-aware.

They are currently experimenting with a hybrid approach which uses an overloaded namespace for storing provenance. In this approach, the system sends provenance when the end of a file is reached. This approach transparently makes protocols such as FTP and SCP provenance-aware. They also built a scheme where lineage is replicated at the nodes that actually consume it, thus ensuring that lineage is more readily available at those nodes. Finally, they store all the lineage in a HyperTable distributed database, ensuring that all clients have access to the lineage.

An audience member asked why they don't store all the provenance in a central location and then just query it. The speaker replied that as long as the users do not care about synchronous queries, the central-location approach should not be a problem.

INVITED TALK

- ***The State of Provenance in 2019***
  Margo Seltzer, Harvard University

Summarized by Peter Macko (pmacko@fas.harvard.edu)

It is the year 2019, and we won! Provenance is everywhere. It is secure, reliable, mandatory, and globally searchable. All kinds of storage systems collect provenance, including local file systems, network-attached storage, and storage in the cloud. All major programming languages are provenance-aware; consequently, all programs implemented in them collect provenance. Web browsers are provenance-aware, and so is the entire Web.

Provenance is secure and verifiable, and users can easily restrict access to the provenance of their objects in order to prevent release of confidential information. For example, if we had this technology back in 2009, the public would never learn the value of the Facebook settlement. Provenance collection cannot be turned off, which makes forging digital signatures or plagiarism easily detectable! All viruses can be tracked immediately back to the Web sites they were downloaded from, which helped to solve the virus problem.

How did we get there? Back in 2004, efficient provenance collection was not even thought to be possible. Soon, the first few provenance-aware (PA) systems appeared: domain-specific solutions, workflows, and storage systems. The next big step was the development of PA programming languages, which introduced more detail into provenance collection and essentially eliminated false provenance. Then the first few PA applications were developed, which was soon followed by the emergence of PA protocols. Most remarkably, email provenance solved the spam problem. And, starting in 2010, there was some fundamental work on graph databases and formalism, which greatly improved querying provenance graphs.

Fortunately, space and computational overhead of provenance collection simply became irrelevant during the past 10 years. The cost of storage went down, so that a terabyte is essentially free. Processors have hundreds of cores, and we still do not know how to exploit all this parallelism. You can devote a hundred cores solely to provenance collection and no one would ever notice any performance impact.

Formalism was also necessary for the success of provenance. It was discovered that causality is not transitive, which forced us to rethink what provenance really is. The community then approached provenance from the perspective of an information flow, and soon provenance semantics and calculus were developed. Related security work solved issues of the reliability and verifiability of provenance, which helped provenance become accepted as the method for system auditing. Another important aspect of security is protecting provenance itself in order to prevent leakage of confidential information.

The success of provenance would not be possible without proper standardization (in order to allow interoperability) and policies. Provenance was standardized through a grassroots effort, creating a need for provenance before standardizing it, and instead of inventing the standard, standardizing the practice. The need for provenance was created by the financial crisis of 2009: In 2010, the government passed an act that required companies to prove additional regulatory compliance, and provenance was perfectly suited for this purpose. The provenance community jumped on this bandwagon and was soon joined by industry.

This talk spawned a lot of interest and started a long discussion. One member of the audience reminded the speaker of a thriving black market of legacy systems without provenance, while another talked about "provenance-free Finland." When asked about the people who do not care about provenance, the speaker explained that they would not be affected in any way—provenance collection is transparent, essentially free, and comes with reasonable security defaults.

Other members of the audience were concerned with the Big Brother aspect of provenance and the (in)ability to post anonymously on the Internet. The speaker agreed that provenance has the potential for misuse, but this risk can be made minimal if the security settings are correctly configured. However, there would always be new ways to find exploits. Other issues raised by the audience included topics such as pre-existing unannotated data, false positives, and implicit information flows.

FIRST AFTERNOON SESSION

Summarized by Peter Macko (pmacko@fas.harvard.edu)

- ***A Framework for Fine-grained Data Integration and Curation, with Provenance, in a Dataspace***
  David W. Archer, Lois M.L. Delcambre, and David Maier, Portland State University

Some tasks require fine-grained data integration from a large variety of sources, which is done manually by experts in the particular field. This process usually involves copying and pasting data from database tables, emails, documents, and Web pages, followed by manual editing and cleaning. The talk presented the research in capturing the provenance of such process. The collected information can be used to answer questions such as: where does the data in this column come from? if there are multiple conflicting data sources, which of them was preferred by the user?

The research group developed a provenance-aware editor of tabular data, which logs all user actions. When data is copied and pasted from an external application, the editor annotates it with information about where the data came from. This information is currently provided by Microsoft applications via an Office add-in, or it can be entered manually. The editor explicitly supports entity resolution, in which the user merges two rows that correspond to the same entity (event) and resolves any conflicting values. Similarly, the user can resolve two columns that correspond to the same attribute.

The system can parse the edit log and produce a provenance graph for any given data value. The graph shows where the value came from, when it was entered, and, if the user performed any form of resolution, what the conflicting values were. The log can also be queried to learn a wide variety of information about both individual values and sets of values, such as the names of all data sources, the age of the oldest source, or the total number of resolutions.

- ***The Case for Browser Provenance***
  Daniel W. Margo and Margo Seltzer, Harvard University

Web browsers keep track of a large amount of information, which causes "a little big data management problem." The amount of data is tractable for a computer but not for users. The traditional solution is bookmarks. More advanced solutions include auto-complete, history search, and the smart location bar. These features are based on history and usage statistics, which is, in fact, provenance. In this talk, Daniel Margo explored how this provenance, which is already collected by the browser, can be used to provide additional sophisticated features.

One of the possible use cases is determining the lineage of downloaded files, because in many cases the URL alone is not sufficient. For example, learning that a picture was downloaded from ImageShack is usually not good enough. Instead, browsing history can be used to determine the exact sequence of user actions to obtain the given downloaded file. Provenance can also be used to improve history search and personalize Web search. For example, the browser can use it to learn that when a user searches for "rosebud," she is interested in gardening, but not in *Citizen Kane*. The browser can use this extra information to refine Web search results or clarify the search query by adding the word "flower" to the query.

This talk was followed by a long discussion—almost as long as the talk itself. Some members of the audience were concerned about security: what if someone steals the collected provenance? what if someone modifies it? The speaker pointed out that browsers already collect all the data (they are just not using it), so if this is the concern, we should already have it now. He also explained that we do not need to worry about users faking their provenance, because, in the end, it would only hurt them. When asked about how much history is necessary for these features, the speaker admitted that he did not study this issue. He mentioned that his personal browsing history was sufficient to get reasonable results.

Other audience members were concerned about the irrelevant parts of the history, such as clicking on an uninteresting link or the possibility of including irrelevant steps in a lineage of a downloaded file. The speaker explained that uninteresting links are being handled gracefully by the applied graph algorithms. When tracing back through the download lineage, most users can identify parts that they recognize before reaching the earlier irrelevant actions.

- ***Provenance as Data Mining: Combining File System Metadata with Content Analysis***
  Vinay Deolalikar and Hernan Laffitte, Hewlett Packard Labs

A large amount of information is stored in an unstructured setting as (text) documents without provenance. This talk presented a method of discovering the provenance of a document by combining file system metadata with data mining techniques. The advantage of this approach is that it does not assume any modification of the file system; consequently, it can also operate on legacy documents.

The algorithm is based on the following two observations: two documents that are one link apart in a provenance chain tend to be similar, and the direction of the information flow can be inferred from the file creation and modification times. The algorithm starts by coarsely clustering documents by their feature vectors, such as TF-IDF (term frequency-inverse document frequency). It then identifies the cluster that contains the document of interest, reclusters it finely, and adds other related documents (such as those in the same directory) to the working set. Then, starting at the document of interest, the algorithm works backward by finding files with similar feature vectors until some stopping criterion is met.

The experimental results presented by the speaker showed that this approach is indeed effective and produces few false positives, but many members of the audience were skeptical. The speaker clarified that this algorithm works even if you do not keep old versions of your documents, although many users do indeed keep some. Other concerns raised by the audience included scalability, reproducibility of results, and application of this technique to other domains, such as images or media.

FINAL SESSION

Summarized by Richard P. Spillane (necro351@gmail.com)

- ***Story Book: An Efficient Extensible Provenance Framework***
  R. Spillane, Stony Brook University; R. Sears, University of California, Berkeley; C. Yalamanchili, S. Gaikwad, M. Chinni, and E. Zadok, Stony Brook University

Richard Spillane argued that provenance-aware systems should make it easier to support application-specific provenance tracking and should at the same time utilize a simpler design. Simply logging provenance is a straightforward design that leads to a more stable and reliable implementation, but it consumes too much disk space. Spillane introduced Story Book, a system that utilizes a write-optimized database and a FUSE-based file system to log provenance records and general compression algorithms to reduce disk consumption. Rather than maintain a provenance graph in memory while capturing provenance events, these events are instead logged and compressed. Later, during queries, a provenance graph is constructed from the indexed log records. To support application-specific modifications, Story Book allows developers to record different provenance for different file types upon file system access. Story Book also has an API to allow applications to manually insert application-specific provenance.

To evaluate the performance of Story Book, Spillane compared Story Book to Waldo, the log indexing tool implemented in PASS (Muniswamy-Reddy et al., *2006 USENIX Annual Technical Conference*), and also implemented an alternative version of Story Book that stores provenance log records in a traditional read-optimized database (Berkeley DB). Spillane noted that Story Book performs as expected: its write-optimized version is 2.8 times faster than its read-optimized version when processing provenance log records. In comparison, Waldo performs 3.5 times faster than Story Book's write-optimized version. Waldo, however, was inserting pre-compressed data while Story Book was inserting uncompressed data; Waldo is a subset of PASS, which performs its compression before log indexing. Spillane noted that read performance for write-optimized Story Book is 3 times slower than for read-optimized Story Book, but argues that this is an unimportant workload for provenance tracking, which he asserts is generally a logging workload.

One audience member asked whether the authors had considered providing multiple alternative provenance graphs to users on queries. Spillane replied that they had not but that Story Book's design did not preclude such a feature. Another member asked whether Story Book provided provenance graphs on queries. Spillane asserted that a full and complete provenance graph is provided on queries.

- ***Making a Cloud Provenance-Aware***
  Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo Seltzer, Harvard University

Kiran-Kumar Muniswamy-Reddy outlined the motivation for a provenance-aware storage system (PASS) which used a remote database (Web service) as a backing store. He argued that storage sizes for large scientific data sets are becoming unwieldy and that using a shared storage back end is a practical way to share storage costs among multiple labs. If, however, some lab still wants to track the provenance of files accessed by all labs, a PASS built on top of the Web service is required. Muniswamy-Reddy clarified, however, that creating a PASS on top of a Web service is complicated by the differing semantics of different Web service providers. He argued that a practical PASS should satisfy: (1) a Read Correctness constraint, which dictates that reads on data should be accompanied with up-to-date provenance; (2) Causal Ordering, which dictates that the provenance of a file-update or process action is always complete as of the time that operation occurred; and (3) Efficient Query, which specifies that provenance queries should be feasible. Muniswamy-Reddy outlined the evolution of a PASS built on top of a Web service that eventually satisfies all these constraints.

Muniswamy-Reddy described three systems: (1) S3, an object store that allows the insertion and removal of key and value pairs; (2) SimpleDB, which supports at least insertion into and query on a basic table with 256 attributes; and (3) SQS, which acts as a basic queuing service for clients to pass messages through. He first described a system that uses only S3 to store both data and the provenance of this data but is limited in query performance, violating the Efficient Query constraint. He improves query performance by storing provenance data in SimpleDB and data in S3; however, this violates the Read Correctness constraint. Muniswamy-Reddy's final design utilizes the work of Branthner et al. and uses SQS as write-ahead logging order to ensure Read Correctness. The final design incurs an estimated 32.2% overhead on top of simply copying the data (without storing provenance) to a remote Web service. Muniswamy-Reddy estimates that roughly 71,000 operations on the remote Web service would be required to recover the provenance of a file in his benchmark data set, which he asserts is reasonable.

One audience member asked what kinds of modifications Muniswamy-Reddy would make to one of the Web services he utilized in order to better support provenance capture. He replied that he would modify the service to capture and store the provenance internally, without involving the client.

- ***Transparently Gathering Provenance with Provenance Aware Condor***
  Christine F. Reilly and Jeffrey F. Naughton, University of Wisconsin, Madison

Christine F. Reilly described the modifications she made to Condor, a distributed job executing environment, to track the provenance of files modified by jobs. Condor by itself does not do this, but Reilly utilizes two new extensions to extend Condor to track provenance: Quill, a tool to scan and insert Condor logs into an SQL database, and FileTrace, a tool to track interactions between jobs and a shared file system. This extended system is called Provenance-Aware Condor, or PAC. The primary advantage of PAC is that it requires no modification to the source code of any job that could run in Condor.

To evaluate PAC, Reilly extrapolated the size of a provenance table for jobs that ran for a year from shorter-running jobs. She estimated that the largest provenance table would be 1.15TiB (tebibyte) after a year of running, and the second largest table would be 0.9TiB. Reilly evaluated four synthetic applications in PAC and in an unmodified Condor system (no Quill or FileTrace extensions). The paper shows the overheads for these synthetic workloads as negligible. Finally, Reilly looked at the application of PAC to tracking the provenance of a Web site called DBLife, which is an online community designed to manage information about the database research community. Limitations of PAC were illustrated in this section of the talk—namely, the lack of application-specific knowledge needed for answering DBLife-specific provenance queries.

One questioner asked about the overhead of the FileTrace tool itself on top of an untracked process. Reilly had not yet investigated this. Some audience members commented that despite the small amount of file data modified by a job, the estimated size of the provenance table after a year was indeed large, contrary to Reilly's claim. When asked about how PAC tracks processes that are being waited upon by parents or are already being tracked, Reilly responded that such applications weren't typically run on a Condor system.

USENIX Upcoming Conferences

2009 USENIX Annual Technical Conference

JUNE 14–19, 2009, SAN DIEGO, CA, USA
http://www.usenix.org/usenix09

Workshop on Hot Topics in Cloud Computing (HotCloud '09)

Co-located with USENIX '09

JUNE 15, 2009, SAN DIEGO, CA, USA
http://www.usenix.org/hotcloud09

18th USENIX Security Symposium

AUGUST 10–14, 2009, MONTREAL, CANADA
http://www.usenix.org/sec09

2nd Workshop on Cyber Security Experimentation and Test (CSET '09)

Co-located with USENIX Security '09

AUGUST 10, 2009, MONTREAL, CANADA
http://www.usenix.org/cset09

3rd USENIX Workshop on Offensive Technologies (WOOT '09)

Co-located with USENIX Security '09

AUGUST 10, 2009, MONTREAL, CANADA
http://www.usenix.org/woot09

2009 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '09)

Co-located with USENIX Security '09

AUGUST 10–11, 2009, MONTREAL, CANADA
http://www.usenix.org/evtwote09

4th USENIX Workshop on Hot Topics in Security (HotSec '09)

Co-located with USENIX Security '09

AUGUST 11, 2009, MONTREAL, CANADA
http://www.usenix.org/hotsec09

Fourth Workshop on Security Metrics (MetriCon 4.0)

Co-located with USENIX Security '09

AUGUST 11, 2009, MONTREAL, CANADA
http://www.securitymetrics.org/content/Wiki.jsp?page=Metricon4.0

22nd ACM Symposium on Operating Systems Principles (SOSP '09)

Sponsored by ACM SIGOPS in cooperation with USENIX

OCTOBER 11–14, 2009, BIG SKY, MT, USA
http://www.sigops.org/sosp/sosp09/

23rd Large Installation System Administration Conference (LISA '09)

Sponsored by USENIX and SAGE in cooperation with LOPSA

NOVEMBER 1–6, 2009, BALTIMORE, MD, USA
http://www.usenix.org/lisa09

ACM/IFIP/USENIX 10th International Middleware Conference

NOV. 30–DEC. 4, 2009, URBANA CHAMPAIGN, IL
http://middleware2009.cs.uiuc.edu/

8th USENIX Conference on File and Storage Technologies (FAST '10)

Sponsored by USENIX in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee (MSSTC), and IEEE TCOS

FEBRUARY 23–26, 2010, SAN JOSE, CA , USA
http://www.usenix.org/fast10
Submissions due: September 10, 2009

7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 28–30, 2010, SAN JOSE, CA , USA
http://www.usenix.org/nsdi10
Submissions due: October 2, 2009

USENIX Conference on Web Application Development (WebApps '10)

Co-located with USENIX '10

JUNE 20–25, 2010, BOSTON, MA , USA
http://www.usenix.org/webapps10
Submissions due: January 11, 2010

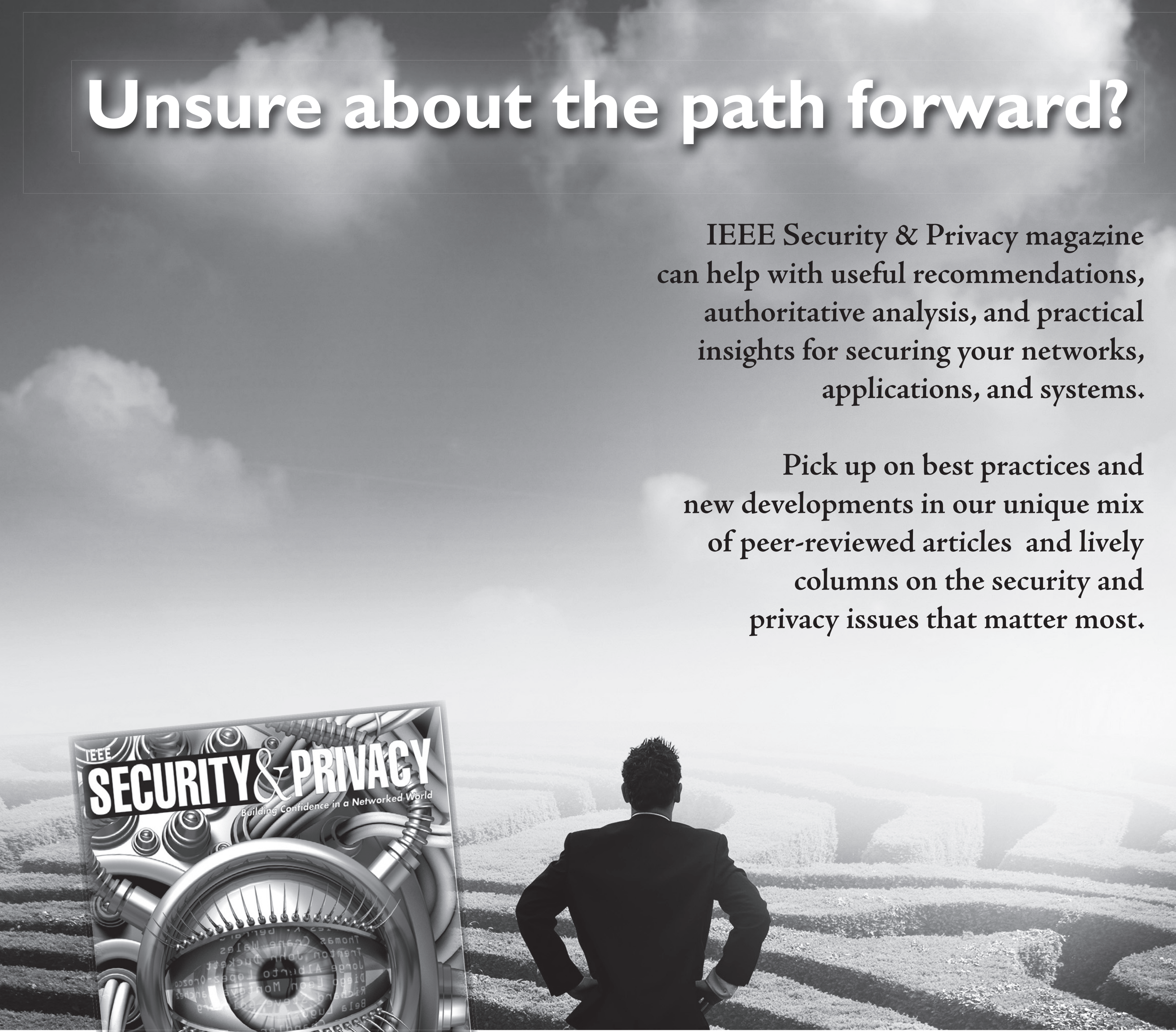
USENIX: The Advanced Computing Systems Association

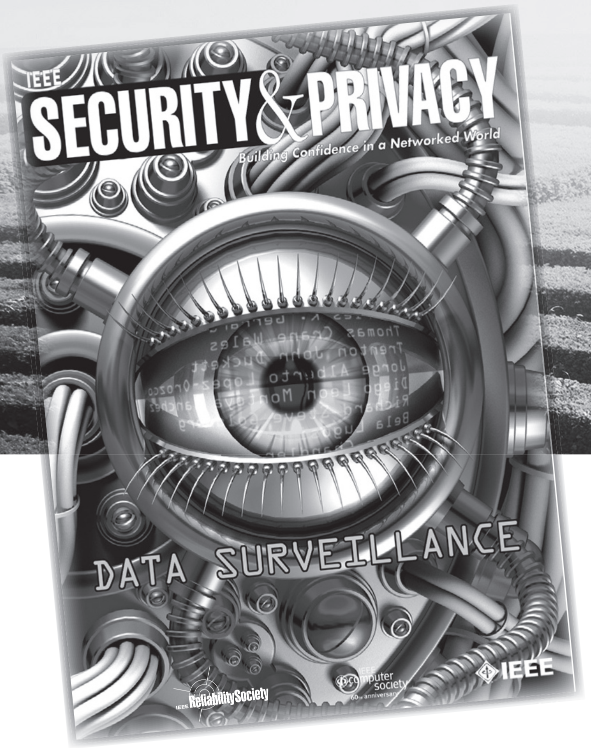
TECHNICAL SESSIONS AND TRAINING PROGRAM INFORMATION AND HOW TO REGISTER ARE AVAILABLE ONLINE AND FROM THE USENIX OFFICE:
http://www.usenix.org/events | Email: conference@usenix.org | Tel: +1.510.528.8649 | Fax: +1.510.548.5738

HELP NET SECURITY

10 years of information security news
UPDATED EVERY DAY.

net-security.org




IEEE
SECURITY & PRIVACY
Building Confidence in a Networked World
DATA SURVEILLANCE
IEEE ReliabilitySociety
computer society
IEEE

NETWORK WORLD LIVE PRESENTS

IT Roadmap
CONFERENCE & EXPO

2009 SCHEDULE

Boston
May 6

Atlanta
June 24

Philadelphia
July 14

Dallas
September 2

Seattle
September 17

Wash, D.C.
October 27

San Fran
December 10

The 7 most-important days in IT'09

Save the date for IT Roadmap when it comes to a city near you. The annual, one-day conference and expo that's close to home. That brings out IT's best and brightest. And surrounds you with the tools you need to succeed in 10 key areas of IT.

Visit **NetworkWorld.com/calendar2009**—qualify to attend free—and you'll get the latest news and insights from Network World editors. Forecasts from IT's most-followed analysts. Case histories from successful end users. Answers from key vendors. A 360° day of strategies and solutions designed to help you do more with less in today's tough economic times.

Mark your calendar now and register online for the city nearest you.

10 TRACKS OF IT ALL IN 1 DAY

1. Application Delivery
2. Building a Resilient & Dynamic Data Center
3. Green IT
4. Network Management, Automation & Control
5. SaaS, Cloud Computing & Managed Services
6. Securing the Core
7. Security & Compliance
8. Unified Communications, VoIP & Collaboration
9. Virtualization
10. WAN, Wireless & Mobility Strategies

REGISTER NOW AND QUALIFY TO ATTEND FREE
www.networkworld.com/calendar2009 | 1-800-643-4668

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

Need TECH TRAINING that puts you ahead in the game?

Join us for 6 days of practical training on topics including:

- Virtualization
- Security
- Solaris
- And more!

Save the Date!

LISA'09

SPONSORED BY USENIX & [sage] IN COOPERATION WITH LOPSA

23RD LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE

November 1–6, 2009, Baltimore, MD

http://www.usenix.org/lisa09/lg