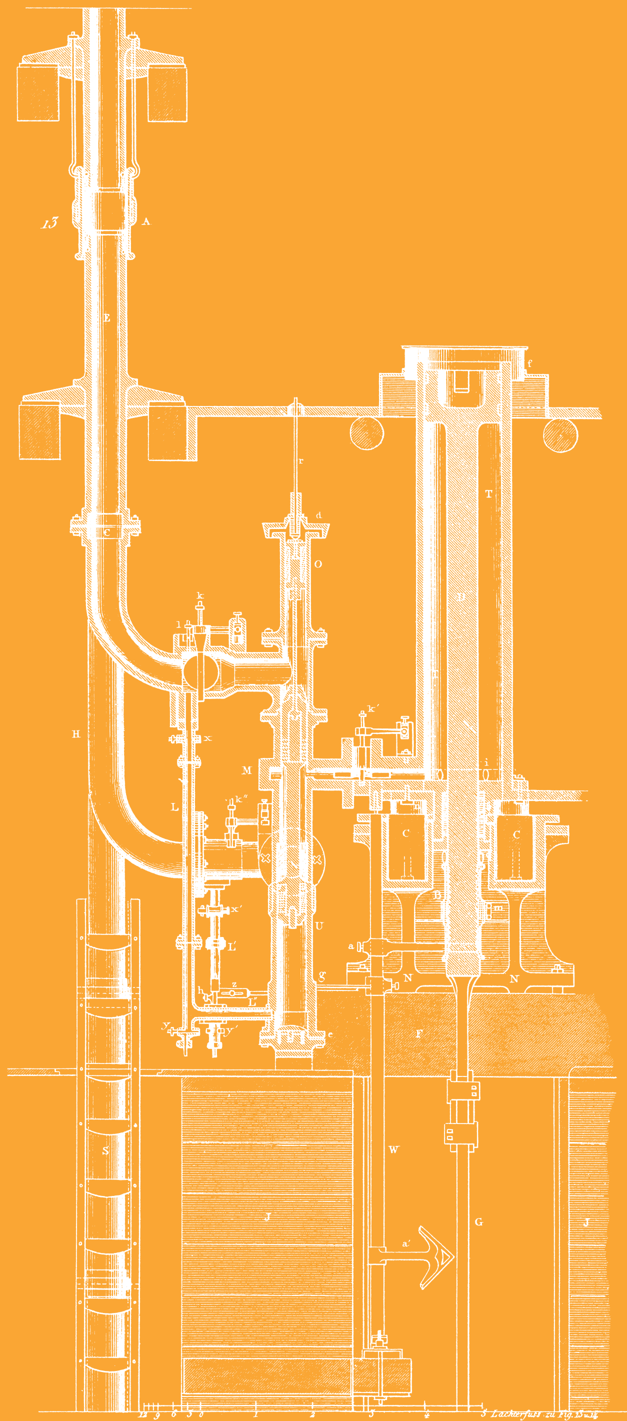
October 2010 VOLUME 35 NUMBER 5

;login:

THE USENIX MAGAZINE



OPINION

Musings 2
RIK FARROW

SYSADMIN

Teaching System Administration in the Cloud 6
JAN SCHAUMANN

A System Administration Parable: The Waitress and the Water Glass 12
THOMAS A. LIMONCELLI

Migrating from Hosted Exchange Service to In-House Exchange Solution 18
TROY MCKEE

NETWORKING

IPv6 Transit: What You Need to Know 23
MATT RYANCZAK

SECURITY

Secure Email for Mobile Devices 29
BRIAN KIROUAC

COLUMNS

Practical Perl Tools: Perhaps Size Really Does Matter 36
DAVID N. BLANK-EDELMAN

Pete's All Things Sun: The "Problem" with NAS 42
PETER BAER GALVIN

iVoyeur: Pockets-o-Packets, Part 3 48
DAVE JOSEPHSEN

/dev/random: Airport Security and Other Myths 53
ROBERT G. FERRELL

BOOK REVIEWS

Book Reviews 56
ELIZABETH ZWICKY, WITH BRANDON CHING AND SAM STOVER

USENIX NOTES

USA Wins World High School Programming Championships—Snaps Extended China Winning Streak 60
ROB KOLSTAD

CONFERENCES

2010 USENIX Federated Conferences Week:

- 2010 USENIX Annual Technical Conference Reports 62
- USENIX Conference on Web Application Development (WebApps '10) Reports 77
- 3rd Workshop on Online Social Networks (WOSN 2010) Reports 84
- 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10) Reports 91
- 2nd Workshop on Hot Topics in Storage and File Systems (HotStorage '10) Reports 100
- Configuration Management Summit Reports 104

2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '10) Reports 106

USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events

24th Large Installation System Administration Conference (LISA '10)

Sponsored by USENIX in cooperation with LOPSA and SNIA

NOVEMBER 7–12, 2010, SAN JOSE, CA, USA
http://www.usenix.org/lisa10

ACM/IFIP/USENIX 11th International Middleware Conference (Middleware 2010)

NOV. 29–DEC. 3, 2010, BANGALORE, INDIA
http://2010.middleware-conference.org/

9th USENIX Conference on File and Storage Technologies (FAST '11)

Sponsored by USENIX in cooperation with ACM SIGOPS

FEBRUARY 15–18, 2011, SAN JOSE, CA, USA
http://www.usenix.org/fast11

8th USENIX Symposium on Networked Systems Design and Implementation (NSDI '11)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

MARCH 30–APRIL 1, 2011, BOSTON, MA, USA
http://www.usenix.org/nsdi11

European Conference on Computer Systems (EuroSys 2011)

Sponsored by ACM SIGOPS in cooperation with USENIX

APRIL 10–13, 2011, SALZBURG, AUSTRIA
http://eurosys2011.cs.uni-salzburg.at

13th Workshop on Hot Topics in Operating Systems (HotOS XIII)

Sponsored by USENIX in cooperation with the IEEE Technical Committee on Operating Systems (TCOS)

MAY 8–10, 2011, NAPA, CA, USA
http://www.usenix.org/hotos11
Submissions due: January 15, 2011

2011 USENIX Federated Conferences Week including USENIX ATC '11, WebApps '11, and HotCloud '11

JUNE 12–17, 2011, PORTLAND, OR, USA

20th USENIX Security Symposium (USENIX Security '11)

AUGUST 10–12, 2011, SAN FRANCISCO, CA, USA

For a complete list of all USENIX & USENIX co-sponsored events, see http://www.usenix.org/events.

contents

;login:

VOL. 35, #5, OCTOBER 2010

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.

©2010 USENIX Association

USENIX is a registered trademark of the USENIX Association. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. USENIX acknowledges all trademarks herein. Where those designations appear in this publication and USENIX is aware of a trademark claim, the designations have been printed in caps or initial caps.

OPINION

Musings — 2
RIK FARROW

SYSADMIN

Teaching System Administration in the Cloud — 6
JAN SCHAUMANN

A System Administration Parable: The Waitress and the Water Glass — 12
THOMAS A. LIMONCELLI

Migrating from Hosted Exchange Service to In-House Exchange Solution — 18
TROY MCKEE

NETWORKING

IPv6 Transit: What You Need to Know — 23
MATT RYANCZAK

SECURITY

Secure Email for Mobile Devices — 29
BRIAN KIROUAC

COLUMNS

Practical Perl Tools: Perhaps Size Really Does Matter — 36
DAVID N. BLANK-EDELMAN

Pete's All Things Sun: The "Problem" with NAS — 42
PETER BAER GALVIN

iVoyeur: Pockets-o-Packets, Part 3 — 48
DAVE JOSEPHSEN

/dev/random: Airport Security and Other Myths — 53
ROBERT G. FERRELL

BOOK REVIEWS

Book Reviews — 56
ELIZABETH ZWICKY, WITH BRANDON CHING AND SAM STOVER

USENIX NOTES

USA Wins World High School Programming Championships—Snaps Extended China Winning Streak — 60
ROB KOLSTAD

CONFERENCES

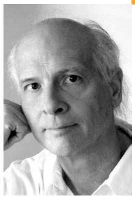
2010 USENIX Federated Conferences Week:

- 2010 USENIX Annual Technical Conference Reports — 62
- USENIX Conference on Web Application Development (WebApps '10) Reports — 77
- 3rd Workshop on Online Social Networks (WOSN 2010) Reports — 84
- 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10) Reports — 91
- 2nd Workshop on Hot Topics in Storage and File Systems (HotStorage '10) Reports — 100
- Configuration Management Summit Reports — 104

2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '10) Reports — 106

RIK FARROW

musings



Rik is the Editor of *;login:*.

rik@usenix.org

AS I WAS GOING OVER THE LINEUP OF articles for this issue of *;login:*, I found myself thinking about just how cool it is that we have not just virtualization, but services that make it easy to spin up new systems on a moment's notice. We can use the new "system" for testing, and as soon as we are done with it, it is gone. Poof.

At the same time, I found myself pondering this brave new world, facing the quandaries it creates: spinning up VMs is really easy, but how do we manage all these systems?

Greybeard

I learned system administration the way many people have—through trial and error. I had no mentors, as the people who could manage UNIX systems were still few and far between. I was fortunate in one way, though: I was being paid to learn how to manage UNIX systems and to write about it. And the people who were paying me provided systems to play with.

I fondly recall sitting in Becca Thomas's backyard in San Francisco, playing with a Xenix system while drinking a beer. My goal was to understand how dump and restor worked, with a particular focus on dump levels. The man page writer had suggested using a Tower of Hanoi sequence of dump levels, but I wanted, really needed, to know why. I couldn't just follow some unknown person's suggestions, as I knew nothing about this person's reasoning or reputation.

I wrote several system manuals for manufacturers of microcomputer-based multi-user UNIX systems, and each time I understood more. Then I ghost-wrote chapters for several books before I started my own.

Now that it was my book, I had to use my own computer. I kept it locked inside a special desk that I had designed (lots of ventilation), and added deadbolts with keys for both inside and out to all house doors, to make it more difficult for someone to steal the computer. The physical security seems ridiculous to me today, as the computer really only had value to me—a thief would be hard-pressed to use a UNIX system running System V Release 3 with one megabyte of RAM and 34 megabytes of hard disk. But that was my experimental system, as well as where Thomas and I wrote. Primitive by today's standards, but a big deal back then.

A Wider World

I was missing out on a much wider world, but didn't know it. I also did consulting as a sysadmin, in those few places that needed a small multi-user system. I thought I was working in the real world of UNIX, but individual systems were quickly becoming a thing of the past.

When I proposed a title for my book, *The Handbook of System Administration*, I was stunned to discover that my publisher had contracted with another group to write a similar book, and they had already chosen that name. I later learned that this was Evi Nemeth and friends, whose fourth edition [1] of their book has just arrived, but too late for a review to be included in this issue.

Nemeth et al. were taking a very different approach to sysadmin. Their environment was the University of Colorado in Boulder, and they had access to lots of systems. From my perspective, telling people how to attach vampire taps [2] to thick Ethernet cables seemed far afield from sysadmin, yet this was an important topic in their first edition. And this pointed to something very important that my co-author and I had completely missed.

Computers would soon be connected to networks, and only rarely would they be used alone. While our book had an excellent chapter on using UUCP over dialup, they included basic IP networking. Neither book dealt with methods of managing groups of computers (beyond the files handled by Sun's NIS [3] or rdist), and for the next several years, this would remain the case. Managing multiple systems would rely on tools that could copy files from a central server to "managed" systems that were essentially all clones.

Back to the Future

Long gone are the days of having one system to manage. Before I was finished writing my first book, I was managing a development network of different vendors' workstations. My bosses did not allow me to use NIS, so just adding a user meant doing this at the console on each system.

Today, sysadmins manage tens to hundreds of systems. They obviously do not walk around to each one, login or su to root, and type commands—at least I hope not. Instead, they will use one of the many configuration management tools to handle the work for them (see the Configuration Management Summit summary in this issue, p. 104).

Jan Schaumann's article about using Amazon's EC2 as a sysadmin teaching resource is what inspired this column. Schaumann explains how important hands-on experience is in learning system administration, and his article in this issue includes links to his syllabus. Having taught sysadmin myself, I read his syllabus eagerly and liked what I saw. Schaumann makes good use of the virtual resources provided (and donated) by Amazon.

What Schaumann leaves out, for the most part, is how to manage multiple systems, perhaps with different OSes, simultaneously. I can imagine that doing so would be a topic for a more advanced class in sysadmin, as you must understand the basics before you can use tools that will duplicate your commands on possibly hundreds of systems. Actually, just thinking about setting a novice sysadmin loose with a configuration management tool is enough to make me shudder.

But learning how to manage multiple systems seems like a perfect fit for working in virtualized environments. Instead of the novice screwing up key systems, he can screw up, uh, configure, several OS instances, then learn how to clean up his mistakes. Or he can just start over, as killing off an instance and spinning up a new one erases past errors.

The Lineup

I've mentioned Jan Schaumann's article already, so let's move on to the next. Tom Limoncelli teaches us by example about satisfying customers. You might wonder how keeping water glasses topped up fits really well with different styles of handling system administration customers, but it does. Just take a few minutes and read his article.

Troy McKee takes us on an adventure where, instead of moving to the Cloud, he migrates from a hosted service. McKee covers the ins and outs of getting mailboxes and other configurations for Exchange moved from a hosted service to an internal one, with some hard-won knowledge learned along the way.

Matt Ryanczak shares some tips on finding IPv6 transit providers. Ryanczak points out that getting good IPv6 connectivity today is not unlike finding good IPv4 connectivity in 1994, as IPv6 is really a different protocol and only slowly gaining the first-class support found with IPv4. Ryanczak doesn't try to convince you to try out IPv6—he just explains some of the important steps you will need to take some day soon.

Brian Kirouac takes us down a different path, one that has become more important with the broader acceptance of smartphones. Kirouac describes how to create and use self-signed certificates to support authenticated and encrypted email for iPhones and Android-based mobiles. Chris Paget demonstrated interception of GSM voice during DefCon this summer, using homebrew equipment, leaving one to wonder if data interception can be far behind [4].

David Blank-Edelman explores places where size really does matter—those times when you need a Perl module and don't have much memory available. He takes us on a fantastic voyage with ::Tiny.

Peter Galvin suggests that we take another look at NAS. Starting with some history, Galvin contrasts NAS and SAN and provides excellent insights that may help you with your network storage decisions.

Dave Josephsen completes his series about monitoring using Argus. Josephsen demonstrates a couple of tools for extracting and sorting events or records of interest from the vast amount of flow information collected from networks.

Robert Ferrell compares airport security to network security and finds many parallels. As Ferrell writes, air travel is definitely UDP.

Elizabeth Zwicky has reviewed four books this time, starting with *Hackers*. I encouraged her to read the revised edition, and it was enlightening to read her opinions of a book I once found inspiring. Sam Stover reviews *Network Flow Analysis* and waxes enthusiastic. Brandon Ching reviewed *High Performance JavaScript*, appearing almost as excited about this book as Stover was about *Flow*.

This issue includes seven sets of reports, starting with USENIX Annual Tech and WebApps—the two main conferences from the 2010 USENIX Federated Conferences Week—followed by most of the workshops of that Week, and ending with the 2nd USENIX Workshop on Hot Topics in Parallelism.

My wife has been trying to get me to clean up my office for years now. With the advent of virtualization and services like EC2, I really don't need either of my old SPARCstations any more (plus, they are *really* slow). And the various PCs, plus extra hard drives, for running different Linuxes and BSDs seem sort of superfluous.

It is really hard to dump my SPARCstation IPC, a 25 MHz system with a 10 megabyte hard drive that cost me $6000 (with a developer's discount!) back in 1990. Perhaps I can just donate the still working RGB monitor to someone.

REFERENCES

[1] Evi Nemeth, Garth Snyder, Trent R. Hein, and Ben Whaley, *UNIX and Linux System Administration Handbook*, 4th ed. (Pearson, August 2010), 1300 pp.: http://www.amazon.com/UNIX-Linux-System-Administration-Handbook/dp/0131480057/ref=sr_1_1?s=books&ie=UTF8&qid=1280184162&sr=1-1.

[2] Vampire taps: https://secure.wikimedia.org/wikipedia/en/wiki/Vampire_tap.

[3] Naming and Directory Services (DNS, NIS, and LDAP): http://docs.sun.com/app/docs/doc/817-4843/6mkbebda4?a=view, https://secure.wikimedia.org/wikipedia/en/wiki/Network_Information_Service.

[4] "Hacker Spoofs Cell Phone Tower to Intercept Calls": http://www.wired.com/threatlevel/tag/gsm/.

JAN SCHAUMANN

teaching system administration in the cloud

Jan Schaumann is a Systems Architect at Yahoo!, a nice place to stay on the Internet. He is also a part-time instructor at Stevens Institute of Technology, where he teaches classes in system administration and in UNIX programming. Like most people, he has his own ideas as to what exactly "cloud computing" is supposed to mean.

jschauma@netmeister.org

LEARNING SYSTEM ADMINISTRATION IN an academic environment requires students to have full superuser access to multiple operating systems (OSes), but many schools' resources cannot provide this kind of access. I have successfully used Amazon's Elastic Compute Cloud (EC2) as an alternative to traditional computer labs or Live-CDs. Using EC2, free for students in most cases, allowed me to offer more flexible and valuable assignments and learning possibilities beyond the boundaries of traditional university resources. There are, however, some disadvantages as well.

System administration has long been a profession that is learned primarily by experience, where people grow into a position in order to fulfill the requirements of an organization rather than follow a career path well-defined by courses, degrees, and certifications. Up until fairly recently, formal classes in system administration as part of a Computer Science or Engineering curriculum were uncommon, but in the past few years more and more institutions have recognized the industry's need for academic courses that adequately prepare students for the multitude of responsibilities within this field.

To really have students understand and appreciate some of the most general aspects of system administration, they need to gain practical experience. They need to administer a system, to have superuser access, to have a chance to configure a system for a specific service, and to make the kind of spectacular mistakes that experienced system administrators value—if only in hindsight.

This normally conflicts with the requirements of the IT staff at the universities: students would require access to a number of different OSes when the school's system administrators strive for a certain level of homogeneity. In order to understand OS installation concepts, filesystem tuning, and other low-level principles, students need to perform these tasks themselves. Learning to debug network connectivity issues or being able to actually see the payload of captured network traffic requires access to raw sockets, which the staff responsible for the security of the campus network would certainly rather not provide.

System administrators are unusually creative and frequently able to provide solutions to help overcome any given limitation. Isolated labs can be made available to students, or various virtualization technologies may make it possible for students to set up and control OS instances outside the standard university environment. All of these incur a significant maintenance penalty and still, in the end, students normally remain confined to a sandbox of some kind, necessarily limited in their learning environment in one way or another.

When I started teaching system administration at Stevens Institute of Technology (not entirely coincidentally, at a time when I was working as a system administrator at Stevens Institute of Technology), I created the syllabus under the assumption that at the very best I could only convey parts of the many topics relating to system administration. “Aspects of System Administration” [1], therefore, covers very broad topics in each lecture, using practical assignments to let students understand and experience what was talked about.

Elastic Compute Cloud

After years of struggling to give all students equal access to the resources required for the diverse assignments and frequently having to change the assignments to be able to make use of available resources, I finally decided to do what many companies do. I outsourced the resource provision problem into “the Cloud”—specifically, to Amazon’s Elastic Compute Cloud (EC2).

Having recently experimented with different cloud platforms, I knew that this approach should make it possible for me to make available to each student an unlimited number of servers, different OS instances, not in a restricted network using theoretical or contrived examples but on the Internet, where they could have full superuser access. Students wouldn’t be restricted by lab hours or firewalls. And, as an instructor, I could create assignments that let students experience having full control over their systems.

As a result, at the beginning of the semester each student was required to create an account with Amazon’s Web Services (AWS)—account creation itself is free of charge, although a valid credit card needs to be provided—and familiarize themselves with the concepts of cloud computing in general and Amazon’s EC2 in particular. Furthermore, they needed to download and install the tools (provided by Amazon) to create, manipulate, and maintain EC2 instances from their preferred platform [2]. Thanks to Amazon’s educational program [3], students were given up to $100 in compute resource credits—more than enough for all required course work.

Since EC2 OS instances are billed based on their actual usage, it should be noted that I did have to repeatedly instruct students to remember to shut down their instances after their work was done, so as to avoid using up their allocated free credits. In the end, one student managed to leave two instances running for over a week, using up over two-thirds of his free credits. However, the current price for a standard or “small” instance (32bit, 1.7GB RAM, 1 virtual core, 160GB local instance storage) of $0.085 per hour made it possible to expect students to cover any required additional compute time themselves. At the end of the semester, almost all students still had several hours of compute credits left, allowing them to further experiment with EC2 and deepen their experience for free.

Assignments

The first assignment was for students to create an OS instance following the instructions provided in Amazon's "Getting Started with EC2 Guide" [4] and run a few simple commands. Anticipating the learning curve all students would experience, students had a full week for this assignment; once comfortable with the tools, spinning up a new instance and running the required commands would normally take minutes.

Amazon offers two main ways of interacting with your OS instances: via the Web browser and by using a set of command-line tools. Both ways utilize the well-defined and open API underneath, the programmatic use of which could well be expanded into a separate assignment or even a stand-alone class. Not very surprisingly, initially many students chose to use the GUI-oriented Web interface for this task. Subsequent assignments required the use of the CLI as a way to illustrate how flexible tools can be combined into a powerful program able to control a number of remote instances. These tools make it possible for students to create or destroy—on demand—an OS instance, create and apply firewall rules and access restrictions, and monitor the status of their instance(s). Once created, an OS instance comes up and can almost immediately be accessed via one of the ssh keys associated with the student's AWS account.

As an instructor, I was surprised at the ease with which students adapted to using the cloud. The turnaround time of just a few minutes from requesting a new OS instance to being up and running and ready to log in still seemed fast to me, while students quickly accepted this as a convenience to be relied on. Only a few students struggled initially with the concepts of different firewall rules and ssh keys that one can associate with one's OS instances. The overall quick adoption of the new tools allowed me to be much more flexible in creation of assignments, many of which were adapted, changed, or scaled during the course of the semester as the capabilities of the resources and the students' own experience became clearer.

One assignment, for example, required students to compare a number of different package management systems on a few different OS flavors (each system's tools native to the OS as well as a single cross-platform system [5]). The lessons learned included not only an appreciation of the differences (and similarities) between software package management systems and an understanding of the intricate complexities of software dependencies, but also nicely illustrated how different libraries provided by different OSes can influence the compilation process of the same set of sources in, shall we say, "interesting" ways. Without root access on four different UNIX flavors (for this exercise alone), these lessons would have been abstract and theoretical, if understood at all.

Another assignment's objective was for students to learn how to use tcpdump(1) to observe and analyze DNS-related traffic. Students had to set up one instance as a DNS resolver and another as a client pointing to the server. While running tcpdump(1) on both hosts, they could then observe how one query from the client caused the server to first contact one of the root servers, then one of the nameservers responsible for the TLD, for example, before returning the results; a subsequent query was observed to not leave the DNS server, as it had cached the results. The detailed analysis of the tcpdump(1) output—graphical analysis tools such as Wireshark were explicitly forbidden—caused the students to really understand how to read network packets, what kind of lookups are done, and when a server replies with what kind of information. Previous lectures on the topic of the DNS hierarchy proved to me that experience is necessary to illustrate clearly to students what happens behind the curtain for almost any and every network connection initiated on a host. It became clear that only the actual inspection of network packets on a DNS resolver as they're generated really reached the students.

From an instructor's perspective, it quickly became clear to me that the availability of a fresh, new OS instance to test each student's project as well as my own sample solutions was invaluable, but something more was lurking on the horizon. The fast turnaround time makes it feasible to spin up an instance to, for example, quickly test a program for cross-platform compatibility without having to run your own (energy-wasting and mostly idle) server farm for this purpose.

Cloud Limitations

From the above, it should be obvious that I'm quite enthusiastic about using Amazon's EC2 service in order to teach system administration adequately. Most of the practical assignments would not have been possible (to the same degree or at all) without the resources provided in this way; all of the assignments in the cloud have produced significantly better understanding on the students' part than previously assigned similar exercises. However, as much as our industry would make it seem, "the Cloud" is no panacea: there are limitations and downsides.

A slight disappointment was that at the time of this writing, Amazon did not offer IPv6 networking capabilities in EC2, something I had hoped to be able to expose students to in practice. Our school did not provide IPv6 connectivity itself, either. Using a tunnel broker would require the use of so-called "elastic IPs" for the students' instances and might thus be possible, although unfortunately we did not have the time to pursue this in the last semester.

In previous iterations of my class, I made students perform an OS installation from scratch, by hand (i.e., without the use of either a GUI or a menu-driven install program). This has always been a great opportunity for students to learn to understand the concepts of disk partitions, filesystem creation, boot loaders and boot order, OS set installation and initial configuration that previously had only been discussed in class and normally are hidden from the end user by the installer.

The virtualization of the OS and abstraction of the instance creation process made it impossible for students to execute these steps. However, a different opportunity awaits: if time and students' prerequisites permit, it ought to be possible to allow them to construct a custom Amazon Machine Image (AMI), an exercise that would convey valuable experience and understanding of rapid deployment systems and abstraction of system components in a virtualized compute world.

Finally, a big challenge lies in the fact that an OS instance has a limited life—when it is shut down, it ceases to exist, and any data residing on the instance is lost. Hence, for the most part, students configured their hosts but "lost" all their work when they shut it down or rebooted it. This can be overcome by making use of Amazon's Elastic Block Storage, which offers off-instance, persistent storage, but while not particularly difficult to use, this is another hurdle and part of the learning curve when using "the Cloud." More than once students reported that they had completed their assignment but had to redo it after accidentally shutting down a host. Frustrating as that was for the students, it was satisfying for me to hear that they also reported that the second time around was much easier, implying that they had internalized some of the lessons as intended.

Each of these problems and limitations does, it should be pointed out, offer an opportunity for students to expand their understanding of cloud computing, of how services, OSes, and host instances are seen in a virtualized world, and how the workarounds or solutions create other interesting possibilities.

As noted above, system administrators are creative. Hence, it comes as no surprise that we should be able to determine viable alternatives to using one company's commercial service. In the end, as so often, it boils down to being a trade-off: each alternative has a cost, just as the original solution does. Will the cost (explicit or implicit) be worth the (perceived or actual) gain?

At the beginning of this article, I mentioned some of the disadvantages of more traditional solutions: isolated labs create an environment that can only approximate real networks; restricting assignments to resources already available on campus necessarily removes some of the flexibility the instructor has in choosing them; and developing or providing more diverse or flexible laboratories or restricted services incurs the cost of many work-hours on the IT staff.

If a school is willing to invest in making available a virtualized environment to students by using Xen, VMware, or any of the other technologies, then it ought to be possible to offer many of the same benefits as Amazon's EC2 environment. However, the maintenance overhead of such facilities is non-negligible and is still likely subject to network access control restrictions (inbound and/or outbound). At the same time, ad hoc creation of OS instances by students themselves as needed requires a high degree of automated virtualization maintenance, which may not always be available.

Based on this, I do believe that outsourcing the infrastructure maintenance is an appealing solution. At the same time, I consider it important to expose students to emerging technologies and what already has become, to some degree anyway, an industry standard solution: teaching system administration in the cloud includes lessons on system administration *of* the cloud. The experience gained this way is not restricted to one vendor's product, either: not only are the underlying concepts illustrated by the practical experience useful when approaching other cloud-specific solutions, there exist AWS-compatible open source implementations such as Eucalyptus [6]; many companies even base their own cloud strategy on these solutions and are actively looking to hire new talent with experience in this area.

Conclusion

"Learning by experience," valuable and irreplaceable as it is, requires two things: time and opportunity. We cannot provide students with more time—learning to appreciate true scalability can only come with years of experience in a number of different environments. By sowing the seeds of a different, deeper understanding of the practical concepts of system administration, by letting students gain hands-on experience actually creating and maintaining Software as a Service (SAAS), by exposing them to scalable and elastic Infrastructure as a Service (IAAS) and letting them see for themselves how, underneath, such services are maintained or created, we can offer the opportunity to learn system administration in a new and very different way.

Teaching system administration in (and of) the cloud will be a core element of my own class in the future. Whether that will happen in Amazon's EC2 implementation or through other resources is largely irrelevant. As usual, it's the underlying principles that count, and in many ways throughout the last semester the teacher was as much a disciple as the students, enjoying the process along the way.

REFERENCES

[1] Course Web page: http://www.cs.stevens.edu/~jschauma/615A/.

[2] Sample command-line tool invocation: http://www.usenix.org/publications/login/2010-10/pdfs/cli.html.

[3] Amazon's AWS in Education program: http://aws.amazon.com/education/.

[4] Amazon's EC2 Getting Started Guide: http://docs.amazonWebservices.com/AWSEC2/2009-11-30/GettingStartedGuide/.

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

[6] Eucalyptus: http://www.eucalyptus.com/products/overview.

THOMAS A. LIMONCELLI

a system administration parable: the waitress and the water glass

Thomas A. Limoncelli has written or co-written four books, including *Time Management for System Administrators* (O'Reilly) and *The Practice of System and Network Administration* (Addison-Wesley). He is a system administrator at Google in NYC and blogs at http://EverythingSysadmin.com.

tal+usenix@everythingsysadmin.com

I PRESENT TO YOU, DEAR READER, THIS parable about how the different ways we organize our work result in different levels of customer satisfaction.

It was the summer of 2008. July. I ate dinner in three different restaurants on three consecutive nights. I noticed something about the way that different restaurants have different schemes for how they manage to refill my water glass.

Interrupt Driven

Tuesday night I had dinner at Friendly's. In case you are not familiar with Friendly's: it is a mid-priced restaurant chain that markets to families with children. There are three within easy driving distance of where I live in the suburbs of New Jersey. The ones near me are usually staffed by high school kids, probably their first job.

As we ate dinner, the waitress refilled our water glasses every time we asked. She would hear and acknowledge our request, take our glasses to the kitchen, and return with new glasses full of water. She felt she was giving us excellent service. We asked, she provided promptly.

While she felt she was prompt, we saw it differently. It took a long time to get her attention. If she was taking orders from a large table with many children, or indecisive adults, it would be a while before we would be able to talk to her. She felt she was being immediately responsive to our requests; we saw her as being difficult to get service from.

The result:

- Best case: She was available immediately and we got our water quickly.
- Worst case: It took 5–10 minutes to reach her, and we got our water quickly, though we were without our half-full glasses while she was refilling them in the kitchen.
- Average case: not so good.

Batching with Interrupts

Wednesday night I had dinner at a fancy restaurant in the Hell's Kitchen neighborhood of New York City. While the name Hell's Kitchen comes from the seedy history of this part of town, lately it has gentrified and is the home of many excellent restaurants. New York City's waitress and waiter subculture has very high standards. These people work hard, are paid little, and live in a place where the cost of living is huge. Tipping starts at 20 percent in this city for good reason. Since I started working at Google's NYC office I have gotten to know many fine restaurants here.

At this restaurant our waitress refilled our water glasses differently. Every now and then she would pause from taking and fulfilling orders and sweep through her section with a pitcher of water. Every glass that wasn't full would be topped off.

If someone requested water sooner, she would bring out the pitcher, refill his or her glass, and then sweep through the rest of her section.

In other words, she was performing periodic batch processes and sometimes the batch process was kicked off due to an interrupt being triggered.

The result:

- Best case: Refilled before you asked.
- Worst case: Similar to the waitress at Friendly's but we always had glasses at our table, and fewer glasses were being rewashed.
- Average case: Sometimes we wait, but usually we don't. On average, pretty good.

Delegation

Thursday night I had dinner at the Skylight Diner in the Clinton neighborhood of New York City. Skylight is a Greek diner. If you are unfamiliar with the concept of the Greek diners that are typical of the New York and New Jersey area, there are a few things you should know. First, they are not called "Greek" because the food is Greek. They are usually owned and run by people of Greek descent. NYC is, very much, a city of immigrants and it is one of the things that makes this city so wonderful. Second, their menus are huge. Page after page of items from burgers to pasta to seafood to sautés. Some even serve Greek dishes, too. Third, the food is usually excellent and the portions are amazingly huge. Finally, if you hear the term "diner" and think "truck stop," you are 180 degrees wrong. To redeem yourself, come visit and be hungry. Very hungry.

At the Skylight our waitress never refilled our glasses. That was the responsibility of the busboys. The busboys were continually roving, taking away our empty dishes and refilling our water glasses. If your water glass is empty and you ask your waitress for more water, they turn to the nearest busboy, point at you, and say, "Más agua aquí."

This is the power of delegation or, one might say, automation.

The result:

- Best case: Refilled before you asked.
- Worst case: Refilled when you ask.
- Average case: Nearly always the same as the best case.

If you have a large party at a diner, they will simply leave a pitcher of water at the table. The entire process becomes self-service.

The result:

- Best case: Water exactly when you need it.
- Worst case: Waiting for a pitcher to be refilled.
- Average case: Usually close to the best case.

Organizing Work

We system administrators organize our work as differently as these three waitresses manage water refills.

When we are new, we are interrupt-driven. Someone asks, we respond. We feel we are doing a great job, because we are being responsive. We are, however, ignorant of the time our customers wait to get our attention.

On a busy day we are unreachable. We pride ourselves on having an excellent best case, but our average case is terrible. Worst of all, we are running ourselves ragged; interrupts manage our time, and we have little control.

We improve our situation when we become batch driven. We improve the average case. Sometimes we fear we are reducing the probability that someone will receive best-case service, since people won't get service on demand. The reality is that this case is very rare and actually isn't accounting for the wait time before they can make the request. The average case is greatly improved. The average case is pretty important.

There are some ways to turn interrupts into batching.

When we are interrupted, rather than jumping to that task, we have a choice. Listen to the request. Pause. Take time to consider: should I record this, delegate it, or do it?

I can record this request in my to-do list or open a "ticket" in my "request tracking system."

I can delegate it if we have set up a division of labor, where each sysadmin specializes in various things.

If it is truly urgent or if it is my job to take requests of this stripe, I can do it. However, this is the last option. I would rather record it, so that I can schedule it after higher-priority items or batch up similar requests.

Ask Questions

In the past I assumed all requests were urgent. Now I always ask, "How soon do you need this?" It is surprising how many urgent-sounding requests aren't urgent when we take the time to ask. "Oh, I'm about to go to Boston. Can this be done before I get back?" or "The new employee starts on October 18. Anytime before that is fine."

It is funny how often we forget to ask that question.

Our ability to record a task is dependent on having a place to record it. Keeping a to-do list is one way. (See my recommendations in [1].) However, it is important to have a request tracking system that lets our customers make requests without interrupting us. These "helpdesk automation" products or "request tracking systems" come in many shapes and sizes. There are open source ones such as Request Tracker [2] or ORTS [3], and commercial products too many to list.

Keeping requests in such a system permits us to record all communication related to the request in one place, permits customers to check on the status without bothering us, permits us to work as a team better (hand off tasks to co-workers), and lets management both observe our work without being annoying and produce metrics to help them do their own jobs better.

A request tracking system also lets us abide by priorities better. When there is a backlog, we can find the high priority tasks easily and work on them first. We can sort the list of tickets by due date.

Emergencies

Emergency requests are the one thing requiring an interrupt-driven response. Sadly, we find some customers who claim everything is an emergency. We can fix this by having a written definition of what constitutes an emergency. This policy must be written and agreed to at the management level. At a newspaper an "emergency" might be something that would directly prevent tomorrow's edition from getting out on time, but not something that is one degree away. At a school, an emergency might be something that prevents a scheduled class activity from happening on the date listed in the teacher's lesson plan or syllabus (but only if the instructor gave prior notice).

Just as stoves, pots, and pans are tools that a chef has in a kitchen, a request tracking database and a written definition of "emergency" are tools needed by a system administrator.

Better Batching

We can batch our work in other ways.

We can do all related tickets as a batch. For example, sometimes doing a DNS update requires certain tools to be open: a control panel, a certain command line, or maybe a text editor open to a particular configuration file. Once we've done one DNS update, we can search for all the other requests related to DNS and do them too. Or we can also ignore all non-urgent DNS-related requests until 4 p.m. each day and do them then.

We can batch by location: gather all the tickets that require physically visiting Building 47 and do them in one trip.

We can batch up all the requests from a particular user. When we are feeling overloaded it can be very satisfying that, while there are dozens or hundreds of tickets sitting idle, at least we've made one user very happy.

If working on the requests requires communicating with the user, it can be faster to get the person on the phone and walk though each request, completing them in real time. Even better, do all the tickets that don't require talking with the user, get them on the phone, and work through the remaining. The user sees a flurry of emails related to status updates on those tickets and then suddenly receives a phone call. They feel like they've won the lottery.

It's more efficient, too. It takes a certain amount of time to open a communication channel with a person (tracking them down, setting up an appointment, walking to their office, or opening an Instant Message window). Once you've suffered that overhead, you might as well do all their open tickets or at least report on their status.

I encourage batching by doing certain requests on certain days. Requests for activating network jacks might be done on Tuesdays and Thursdays. Rather than changing backup tapes every day, use a tape jukebox large enough that they only need be changed every Monday and Friday; a larger jukebox permits them to be changed monthly, an even larger one can practically eliminate human interaction.

Delegation and Specialization

Sometimes we delegate or specialize like the Skylight Diner. We can delegate more easily when things are documented, but we don't need fancy documentation. A bullet list of steps on a wiki can be "just enough."

Specialization is only possible as our organization grows. Often an IT team begins as a single individual who carries the entire burden for the small company.

Then another person is hired and the two share the load. Fast-forward a few years and we find a 10-person IT team all struggling to do all tasks. At what point should they have specialized? Probably after the second or third person. It depends on each organization. Different organizations need different specializations. Typically people specialize where specific knowledge is needed. If the environment requires a particularly large and ever-growing storage system, some may specialize in storage. Even small teams have one person who knows networking better than others, and is responsible for the Internet gateways and firewalls.

With proper documentation everyone can do all the basic "everyday" tasks related to provisioning ("adds, changes, and moves"). Leave the uncommon tasks to the specialist (grow the service, optimize, add entirely new features).

In other words every system administrator on the team should be able to connect a new machine to the network, update DNS, and so on. The specialist might be the one who has the knowledge required to create a new subnet and DNS zone or to modify firewall rules.

When tasks are documented it is easier to optimize and automate them. We can strategically select specific parts to automate (which bullet items on the wiki page), or we can automate the entire process.

Automation

Giving a table of customers at a restaurant their own pitcher of water turns a burdensome request into a self-service feature. In system administration it is often easiest to create self-service versions of provisioning requests. Take, for example, providing VPN service to your users. Setting up the VPN server is something done once; there is no benefit to automating. However, adding new accounts should be a repeatable process, and therefore easy to automate.

At first one might automate the process so that a system administrator can enable or disable service for a single user very easily. That gives them a tool that frees up their time for other things. The next step would be to make this a self-service operation: a Web page that users can click on to make the request, the request is approved by a human, and the system does the right thing. Some people might be pre-approved; for example, users in the LDAP Group "engineers" might be pre-approved for VPN access. Or only people in the LDAP group "visitors" require approval. Now more than freeing up your time, you have a tool that empowers users to help themselves.

A restaurant doesn't have as many opportunities for automation as system administrators do. Yes, they could build robots to take our orders and deliver food. That would be awesome. However, what we love about restaurants is the human aspect. It is a luxury to be served. While restaurants lack opportunities to automate, they can improve workflow through batching and better organization.

As system administrators we have many choices about how we do our work: interrupt-driven, batching, delegating, automating, self-service.

What do you do?

Things to Think About

1. In the past week at work, were you interrupt-driven, batching, delegating, automating, or creating self-service systems?

2. What are three tasks you do at work that could be batched?

3. When someone makes a request, how do you know how soon he or she needs it?

4. What specializations are in your team? Are they recognized formally?

5. How would you reorganize how you do your own work? How would this make things better and worse for you? For your customers?

6. How would you reorganize your team or IT organization? How would this be better or worse, for you and your customers?

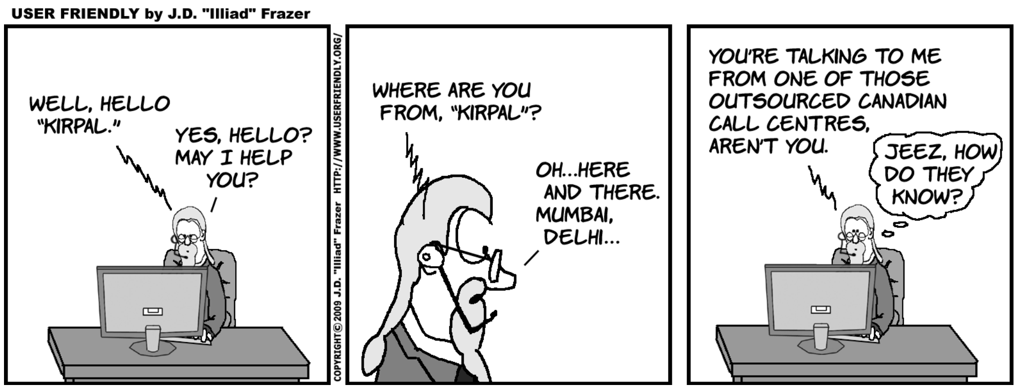
7. In answering question 5 and 6, many changes were proposed. Which specific changes would have the biggest impact? Which one change would have the most immediate benefit?

REFERENCES

[1] Thomas A. Limoncelli, *Time Management for System Administrators* (O'Reilly Media, 2005).

[2] RT: Request Tracker: http://www.bestpractical.com.

[3] ORTS: http://www.orts.org.


USER FRIENDLY by J.D. "Illiad" Frazer
WELL, HELLO "KIRPAL."
YES, HELLO? MAY I HELP YOU?
COPYRIGHT© 2009 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
WHERE ARE YOU FROM, "KIRPAL"?
OH...HERE AND THERE. MUMBAI, DELHI...
YOU'RE TALKING TO ME FROM ONE OF THOSE OUTSOURCED CANADIAN CALL CENTRES, AREN'T YOU.
JEEZ, HOW DO THEY KNOW?

TROY MCKEE

migrating from hosted Exchange service to in-house Exchange solution

Troy McKee is an IT manager with an eclectic range of experience, from geology and engineering to the games industry and software development. Having worked in just about every sized environment and on many platforms, he currently prefers to work wherever he finds a challenge. Troy has a master's degree in IT from American Intercontinental University. He is currently employed in the financial services industry, which is quite challenging.

tmckee@leadclick.com

IN THIS ARTICLE I REVIEW SOME OF the issues involved in migrating from a hosted Exchange service to an in-house Exchange solution. I will briefly discuss the choice to migrate, the pros and cons of an in-house Exchange solution versus a hosted Exchange service, and the real cost considerations. I describe the actual migration process, including the necessary planning and testing. And then there are the gotchas: there are several potential issues that need to be dealt with in such a migration, some of which are not obvious when testing the environment.

Hosted Exchange Services vs. In-House Exchange Solutions

Most businesses start small enough in the beginning that having an in-house Exchange server doesn't make sense. A hosted Exchange service from a reputable provider is a good choice as long as the numbers make sense. When a company gets large enough, the costs for a hosted Exchange service will become expensive enough that it will make sense to examine whether it is time to migrate to an in-house Exchange solution.

The first step in determining when to migrate is to look at the needs of the business and determine if the hosted service meets the growing company needs. One need may concern cost, but other needs may involve features, availability, SLA, security, and SOX compliance, among other business needs. If the need for features and affordability is met by the hosted provider, then there is no reason to change. Some cost considerations determine when to migrate: hardware and software costs are depreciable costs spread out over the life of the installation of the application and are not monthly or yearly costs. Each copy of Microsoft Office includes a CAL (Client Access License) for connecting to Exchange, so it is not necessary to buy Exchange CALs for every user. Exchange CALs are necessary for all concurrent OWA (Outlook Web Access) sessions, and that is a good gauge of the number of Exchange CALs needed. Prices for a hosted solution go up for each new mailbox and with each additional service and are paid every month. At the end of the host email contract, you don't own anything from a hosted service but your data.

There were several reasons that led the company to change to running our own Exchange server. As we grew, it became more apparent that we needed more control and the latest features that only Exchange 2007 would offer. Our hosted provider wasn't planning to upgrade to Exchange 2007 in the near future, and the cost for hosted service was growing far more rapidly than if we ran it ourselves, so we made the plan to migrate.

One of the other advantages of migrating to Exchange 2007 was support for our Mac users. We had about 10% of our users running OS X using Entourage 2008. The connectivity for Entourage 2008 and Exchange 2007 made everyone happy, even me as an Exchange admin.

While Exchange 2007 has many great features for the users, it has added a great deal of complexity. The Exchange 2007 server roles help to define functionality and reduce the attack surface by limiting the roles a server has to provide services for, but this mostly applies in a larger organization. For a rapidly growing company it can be a hindrance. Exchange 2007 resembles Exchange 5.5 in the level of administrative complexity, which is bad, but SP1 for Exchange 2007 helped with some of that. It took a while to work all that out and get the configuration that had both the features and the security we wanted. This delayed rollout of the project by more than a month.

The Migration

Once we had a configuration that worked, we had to develop a plan for migrating all the mail for our users who had their mail hosted on a server we didn't own. This presented several problems. First, we didn't have console access to the hosted mail servers, and we couldn't connect to them in any way that would allow us to transfer data from one server to another. Because of the poorly designed Web interface at the hosted mail service, I couldn't get a list of users, contacts, distribution lists, and members saved to a file. All that had to be recreated by hand. I was eventually able to capture usernames in a messy fashion, save them to a text file, remove all the HTML header info, and then import them into a file I could use for reading in a script. Still not the automated solution I was hoping for, but I don't think the hosted Exchange provider was that interested in providing tools for migrating away from their service.

To avoid downtime, we prepared alternate forwarding addresses for each user from a second email domain, though we could have used a subdomain. The alternate domain was necessary for testing and to avoid email disruption while it was being transferred from the hosting to the in-house service. It may take 24–72 hours for the MX record to be transferred and for that DNS info to completely propagate, so this is probably the best way to make sure email delivery continues until all external DNS information has propagated across the Internet.

Mail forwarding on the hosting service admin console had to be set for each user, but there was no way to do this in bulk at the console. It may be possible to create a script using JavaScript, Greasemonkey, or Perl to scrape the addresses and create the forwarding addresses, but that is likely to abrogate the Terms of Service with the hosted Exchange provider. Please make sure there are no legal or contractual issues before attempting to automate this process on the hosted Exchange provider's Web interface.

In addition, a contact address had to be created for each address that was forwarded to, but that could be automated. Powershell is now the shell scripting language of choice for Exchange; it is simple, it is object oriented, and it works very well in this instance. Something like the following should work to import contacts from a .csv file:

```
$csv = Import-Csv "C:\Contacts.csv"
foreach($line in $csv)
{
New-MailContact -Name $line.DisplayName -ExternalEmailAddress
$line.EmailAddress -OrganizationalUnit "users" -Alias $line.Alias
}
```

Once the contacts are created, the alternate addressing needed to be configured on the new Exchange server. The email address policy in Exchange 2007 made this very easy, as rules could be set to create alternate domain addresses and select a default domain address. Exchange needed to be configured to receive email for both domains. We made sure that a receive connector was in place to accept email from the hosting service and then forwarded a test address to the alternate address. We verified that it was received. Then we verified that the test account could send email and that it showed the correct sending address. We also had that test account send a reply and verified that it was received and showed the correct address. We set the forwarding up for the test account, sent to that account from an outside address, and made sure it was forwarded to the new Exchange server. Then we sent a reply and made sure it was received and showed the correct address.

Once we had recreated the users, conference rooms, distribution lists, contacts, and everything else, we needed to populate them with data. We decided it would be easier to download each user's mail to an Outlook client, saved as a PST file, and then upload to the new server from a new mail profile than to try to get something from the hosting provider, a belief that turned out to be accurate.

Exporting each user's data to a PST file and importing to the new server took a long weekend, but allowed us to migrate the users' email without any downtime for them. As each user was migrated, we sent a message to the user with instructions for accessing their email on the new server via Outlook and via OWA, then forwarded their email from the hosted mail to our new mail server. Then we exported their email and imported to the new server. When all the data was imported, we changed the MX records and let that propagate out to the rest of the world.

Monday morning, the users came in and checked their email and the last message in their mailbox on the hosted server was the message that they had been migrated. Some needed help creating new mail profiles to connect to the new Exchange server, but the process was well documented and tested, and most users had no problems.

Surprises

Now, here comes the bad part: Once we had everyone sending and receiving email from our mail server, we discovered a few problems. First, our conference rooms were not accepting new meeting requests and nobody could see anyone else's availability when creating new meetings. Secondly, and more importantly, replying to messages which were originally generated on the hosted mail solution created errors and the messages were not delivered. In addition, once users had done this, even new messages to the recipient would fail. This is basically the error the sender would see:

```
Delivery has failed to these recipients or distribution lists:
```

```
Username <mailto:%2FO%3DCOMPANY%2FOU%3DEXCHANGE
%20ADMINISTRATIVE%20GROUP%20(FYDIBOHF23SPDLT)%2FCN
%3DRECIPIENTS%2FCN%3Dusername>
```

```
You are not allowed to send this message because you are trying to send on
behalf of another sender without permission to do so. Please verify that you
are sending on behalf of the correct sender, or ask your system administrator
to help you get the required permission.
```

A call to Microsoft Support and 12 business hours later (we didn't wrap up the issues from the call until the next day, or perhaps the day after that) and the Offline Address Book issue and mail delivery problems were fixed. The OAB had to be retargeted, regenerated, and downloaded to most users. Also, the problem with the failed email delivery had to be fixed. The solution, not obvious but not as hard as I had feared it would be, was to create an X500 address for each user which mimicked the old hosted solution server info. That is done by creating a custom address and labeling it as an X500 address. The first part of the address had to mimic the server name of the old server from the hosted Exchange service. That information could be obtained from one of the corrupted email addresses in the OAB. It would look something like this:

```
/O=OEXCH010/OU=FIRST ADMINISTRATIVE GROUP/CN=RECIPIENTS/
CN=USERNAME
```

The first part of the X500 address is the server name of the hosted Exchange service. The second section is the administrative group or storage group the users account was on the hosted service. The third part is the identifier for an email recipient. The last section is the username of the email recipient. This information can be gathered from a test email from a migrated test account or maybe even from the header information of an email, if you know what you are looking for. This info can be gathered from the email header of any email sent from the hosted provider. It would be best to confirm this with the hosted email provider unless you have tested it.

With this and the OAB solution, users could now send and receive email without problems. The final issue was the conference rooms.

The auto-accept issue was simple. That just needed to be set from the Exchange Management Shell for each conference room. The following command will set that:

```
Set-MailboxCalendarSettings <Identity> -AutomateProcessing:AutoAccept
```

It was obvious that the real issue was a rights or permissions issue. I tried granting the distribution list, which included the entire staff, rights to the conference rooms, but that did not work. The correct solution was to create a security group, add the conference rooms to the security group, and grant the distribution list rights to the security group. This solved the problem with visibility of calendar info for the conference rooms as well as availability info for users when scheduling new meetings.

Conclusion

While there were significant issues to overcome, and some unexpected surprises, the migration was cost-effective and resulted in a stable email platform that provided all the features the company needed. I could have benefitted from more testing, particularly testing a migration of two or more mailboxes and a conference room and then testing send and receive between them and outside mailboxes, which would have found the major gotchas that the users discovered. I learned some important lessons about the strengths and weaknesses of hosted services and the types of things that can go wrong when I had only half the info to plan for the migration. I have discovered that for the migration process, it is important to think like a sysadmin, but for developing test cases, I needed to think like a user.

MATT RYANCZAK

IPv6 transit: what you need to know

Matt Ryanczak has worked as a system and network administrator, primarily on Linux and on Solaris, and other UNIX platforms, for the past 17 years. Throughout his career, Matt has worked in many different facets of information technology while employed at companies ranging from OEMs to Internet service providers and space and defense vendors. He is currently employed at the American Registry for Internet Numbers, where he serves as the network operations manager.

ryanczak@arin.net

PEOPLE HAVE BEEN PROCRASTINATING about implementing IPv6 for over 10 years, but the day is coming soon [1] when this will leave your network disconnected from the only part of the Internet that is growing and innovating, the IPv6 Internet. At ARIN, we have been using IPv6 since 2003. In this article I will make some suggestions about choosing an IPv6 transit provider and offer some tips for determining if a provider's network is tunneling IPv6 over IPv4 or otherwise offering sub-optimal IPv6 services.

The IPv6 Internet

While the names may sound similar, the protocols themselves are quite different. So different, in fact, that IPv4 and IPv6 are not layer 3 compatible. The good news is that they are layer 2 compatible in most cases, meaning that IPv4 and IPv6 can share the same wire, although even then an IPv6 host cannot connect directly to an IPv4 host; in much the same way that Appletalk and IPX are not compatible, IPv4 and IPv6 are distinct, incompatible protocols. Ultimately, this means that there are two Internets—one running IPv4 and the other IPv6. How do you connect to the IPv6 Internet? Does your current IPv4 transit provider offer IPv6 support? Is all IPv6 transit the same? Will your existing routers, firewalls, and other hardware and software support IPv6? What is different and what do you need to know to connect your network to the IPv6 Internet? I will attempt to answer these questions or provide the pointers you need to get the answers yourself.

In some ways the IPv6 Internet today is comparable to the IPv4 Internet of fifteen or twenty years ago. In 1993 there were only a handful of companies providing Internet connectivity; today there are only a few providers providing IPv6 connectivity. In 1993 there was a large disparity in the performance of transit provider networks, because many providers were not well connected or were under-provisioned. Today, many networks are not well connected to the IPv6 Internet, or their IPv6 network is not given the same level of support as their IPv4 network. In fact, only a handful of networks currently run IPv6 natively on their backbones, and many providers rely on tunnels, IPv6 carried inside of IPv4, to transit IPv6.

Backbones and Tunnels

One question you must ask potential transit providers is how they transmit IPv6 over their network. Is their IPv6 network overlaid on top of their IPv4 backbone, or does it use a separate physical infrastructure? If the networks are separate, are they completely separate or do they diverge at particular points? A provider may advertise its modern 10-gigabit IPv4 backbone only to run its IPv6 network on older or more poorly provisioned equipment. Does the provider offer all of its services, from Web hosting and co-location to email and conferencing, over IPv6? A provider with broad product support over IPv6 is likely to be committed to providing good service no matter which protocol is used. Do they have a completely native IPv6 backbone, or do they tunnel portions of their IPv6 network over IPv4 in order to bypass devices that do not yet support IPv6? Tunnels can cause problems, especially if they are not well documented.

One reason tunnels can be problematic is that sudden changes in maximum transmission unit (MTU) can cause packet loss and complete failure for some protocols, such as HTTP. While IPv6 supports path MTU (PMTU) discovery, filters on routers and firewalls can cause this mechanism for detecting MTU to fail. Understanding where tunnels exist and what their MTU is can help in troubleshooting problems. If the MTU is too small, you may not be able to consider the provider at all, because your packets will never make it across their network.

There is a very easy way to detect tunnels by using the ping command. Most UNIX-like operating systems use ping6, rather than ping, to send ICMP packets over IPv6. You can use ping6 with the -s option to specify the packet size and the -Mdo (Linux) or -m (*BSD) to disable fragmentation. The default value in most cases is 56 bytes which, when combined with the ICMP header, equals 64 bytes. You can take advantage of the ability to set the packet size with the ping6 command to find tunnels. The first thing you need to do is determine the path to test. This can be done using traceroute6 or the mtr command [2]. Both of these commands are available on most modern UNIX-like operating systems. I prefer mtr. Here's example output using mtr from my house to ARIN's Web server:

```
    matt@fry:~$ mtr -6 -n www.arin.net

Host                      Loss %   Snt    Last    Avg    Best     Wrst    StDev
1. 2001:470:e1ce:1::1     0.0      27      0.2     0.2    0.2      0.2    0.0
2. 2001:470:7:287::1      0.0      26     38.7    26.6   13.4     66.7    12.9
3. 2001:470:0:90::1       0.0      26     52.2    33.0   11.0     61.6    14.9
4. 2001:504:0:2:0:1:745:1 0.0      26     53.5    33.9   10.4    112.4    24.5
5. 2001:500:4:10::12      0.0      26     18.8    45.4   11.6    229.8    47.2
6. 2001:500:4:11::2       0.0      26     63.3    43.3   14.6    238.4    45.0
7. 2001:500:4:12::3       0.0      26     84.9    39.8   17.2    119.3    24.7
8. 2001:500:4:13::81      0.0      26     41.2    38.3   15.0    409.1    75.9
```

Using this output I can then use the ping6 command to ping each address listed in the path to determine if there is a tunnel along the way. Knowing that my local MTU is 1500, I'll issue pings to the address on the far side of my default gateway using ping6 –s 1480 –Mdo 2001:470:7:287::1. This produces the following output:

```
matt@fry:~$ ping6 –s 1480 –Mdo 2001:470:7:287::1

PING 2001:470:7:287::1(2001:470:7:287::1) 1480 data bytes
From 2001:470:e1ce:1:240:63ff:fef9:f6fb icmp_seq=1 Packet too big:
mtu=1480
```

I can then back down the packet size until packets get through:

```
matt@fry:~$ ping6 –s 1432 –Mdo 2001:470:7:287::1

PING 2001:470:7:287::1(2001:470:7:287::1) 1432 data bytes
1440 bytes from 2001:470:7:287::1: icmp_seq=1 ttl=63 time=12.8 ms
1440 bytes from 2001:470:7:287::1: icmp_seq=2 ttl=63 time=9.30 ms
```

I now know that the MTU for that leg of the packets' journey is just north of 1432 bytes. While this does not necessarily mean there is a tunnel, there is a very good chance that there is. At the very least there is a hop with a very odd MTU size and this is something that could cause you problems down the road.

Details, Details

Understanding what type of data and which protocols you will send across the network is just as important when selecting an IPv6 provider as it was when selecting an IPv4 provider. The usual questions apply. How will their network handle your data? What is their uptime guarantee? No network has a perfect track record of uptime, but they should be able to guarantee their product with an SLA that gives credit for unplanned outages. Any provider should be able to provide uptime, latency, packet loss, and other statistics. It is important to make sure the statistics they are citing are for IPv6. It is also worth asking them if the statistics they provide are for IPv6 only or for both IPv4 and IPv6.

It can be difficult to get providers to divulge non-marketing–driven information about their network, especially details such as tunnel configurations and whether tunnels exist at all. Asking for a native IPv6 connection typically refers to your link to the ISP and does not necessarily imply that you are asking for native IPv6 from your connection to the ISP, through their network, and on to their providers and/or peers. It is important to ask if they are tunneling upstream from your connection. You may learn that they are using a tunnel to connect to their upstream providers and peers.

At least two Tier 1 providers have a native IPv6 backbone, and several other smaller providers have native IPv6 backbones that reach across the globe. There are providers that can offer tunneled service over existing IPv4 service, even when the IPv4 service is coming from another provider altogether. There are quite a few ways to get IPv6 transit right now, and in some cases it is easier and even cheaper to obtain IPv6 transit than IPv4.

Types of Transit Providers

TUNNEL BROKERS

There are several organizations that will provide any network with free IPv6 transit using tunnels at no cost. These organizations, called tunnel brokers, are typically not-for-profit, but in some cases they are free services offered by for-profit companies. Some tunnel brokers will even allocate your network a /48 of IPv6 address space. That is 1,208,925,819,614,629,174,706,176 hosts, or 65,536 subnets, for free! In some cases BGP is allowed by providers and they will announce your IPv6 address block if you already have one, or just give you a feed of the IPv6 routing table. I mentioned above that tunnels can make troubleshooting more difficult, but the price of establishing an IPv6 tunnel cannot be beat, which makes this a good way to get your feet wet. You won't get an SLA for the price, but you will gain experience. To this day we still use tunnel broker accounts at ARIN for testing IPv6 outside of our own network. A good list of tunnel brokers is available at the IPv6Day Web site [3].

INTERNET EXCHANGE POINTS

Another way to participate in the IPv6 Internet is to join an exchange. At least one provider will provide free IPv6 transit if you peer with them at one of the common Internet exchange points (IXPs). In fact, if you have access to an IXP, you likely have access to several carriers willing to provide you IPv6 transit services. You will also find many smaller networks willing to peer with you over IPv6. Most IXPs now support both IPv4 and IPv6, which makes finding IPv6 peers and transit providers very easy. This approach to obtaining transit offers you great speeds, an SLA, and the opportunity to peer with other networks over IPv6, which can help reduce transit costs.

THE LAST MILE

So what if you want IPv6 transit with an SLA over your last-mile connection? How many providers are you likely to find that can provide you IPv6 over an OC3 or T1 to your data center or office? There are more than you might think. In 2002, when we started looking into IPv6 transit at ARIN, we had a very difficult time finding anyone that could offer us a T1 running native IPv6. We did eventually find a provider that gave us service on an experimental basis. Much of the upstream network was tunneled. The first few months involved a lot of troubleshooting, but the service worked and we did get an SLA. Today things have improved considerably. If you speak to one of the big carriers, you are likely to find IPv6 ignorance in the form of, "We don't do that," or "What is IPv6?" But if you are persistent and don't take no for an answer you may discover that IPv6 support is available. In some cases the service is tunneled over your existing IPv4 services, but you can find providers that support this configuration with an SLA.

ARIN is located in the Washington DC metro area, and something I have found over the years is that IPv6 support seems to be better among smaller ISPs. I know of several small ISPs in the DC area that offer native IPv6 transit over any last-mile connection they offer. I believe this is because smaller providers can be more agile, making it easier to implement IPv6 and have a leg up on their bigger competitors. This may be true for your area as well; you may find that the little guy is better connected to the IPv6 Internet than the big guy providing his transit! This is probably not a long-term thing, but it is worth looking at smaller providers in your area; you may be surprised at what you find.

INTERNET 2

There is at least one other way of getting IPv6 transit, and that is by joining the Internet2. The Internet2 is a nonprofit consortium of universities, corporations, government agencies, and laboratories. Not just anyone can join, but if your university, corporation, or agency is involved in research, you may qualify for a connection. Visit the Internet2 Web site [4] for more information about the project and to find out whether your organization is eligible to participate.

Selecting a Transit Provider

It is important to apply the same criteria you used to select an IPv4 provider when selecting an IPv6 provider. Because there are so many tunnel brokers in the mix, you will probably have several providers to choose from, even if you do not have a local ISP that supports IPv6 directly. You need to make sure that the IPv6 transit being offered meets your requirements. Earlier I mentioned that the IPv6 Internet is, in some ways, less mature than the IPv4 internet. Companies are still building out their IPv6 networks and, in quite a few cases, have not started at all. Keep this in mind when speaking to providers about services. There are several important questions to ask:

- How are they connected to the IPv6 Internet?
- Do they have multiple IPv6 providers?
- How much IPv6 bandwidth do they have?
- Is that bandwidth dedicated or is it shared with IPv4 or possibly other protocols?
- Do they peer with other networks over IPv6?
- If they do peer, how many peers do they have?
- In which exchanges do they have a presence and use IPv6?

It is also very important to ask a potential provider if they offer the same SLA for IPv6 as IPv4. The answers to these questions will impact your IPv6 service provider selection decision.

Finding transit is the hardest task in your quest to establish a connection to the IPv6 Internet. Once you have a connection, things get easier. In the next part of this article I'll go over what you need to know about your routers and other equipment in order to get them to work with IPv6.

Connecting Your Network

It may seem like enabling IPv6 support in your routers is going to be difficult, but generally this is the easy part. If you use Cisco or Juniper routers produced in the past six or seven years you can be fairly certain that your device supports IPv6. You may need to update the operating system on your router to a more current version, or in the case of Cisco, you may need to purchase the advanced IP services pack to enable IPv6 support. There is a good chance that you have gained IPv6 support in your routing hardware through regular software updates. If you use Linux, FreeBSD, or OpenBSD as a routing platform, you will have support out of the box, though depending on the operating system you will have some choices to make regarding firewall [5] and routing protocol daemons.

Assuming you have a router that is IPv6 capable, using it to connect to an IPv6 transit provider is no different from using it to connect to an IPv4 transit provider. In my experience, both Cisco and Juniper offer very good support for IPv6. Generally speaking, tasks like adding IPv6 addresses, ACLs, and static routes are very similar to doing the same tasks in IPv4. One thing that is critically important to understand is that IPv6 and IPv4 are treated separately in terms of access control lists. It is important that you recreate your IPv4 access control lists in IPv6. Your router will end up containing completely separate ACLs for IPv4 and IPv6. Keeping your access policies in sync between IPv4 and IPv6 can be painful, but it is very important. One thing that complicates this is the differences between the two protocols. For example, ICMP has a lot more functionality in IPv6; if you apply an IPv4 mindset to developing IPv6 ACLs, you are likely to break IPv6 features such as path MTU discovery. I would recommend that you read up on IPv6 and understand how it uses ICMP differently from IPv4. If you understand this key difference, you will have a good start in being able to construct ACLs that do not cause breakage.

Another important thing to remember about configuring IPv6 on your router is that, much like ACLs, routing is separate from IPv4. If you keep in mind the fact that IPv6 is a separate protocol, it makes sense that routing would be separate. You must maintain a separate IPv6 routing table. You will have to explicitly configure IPv6 static routes or routing protocols for your network. You may need to establish a BGP session over IPv6 with your transit provider or with peers. Any routing policies you have must be maintained in both IPv4 and IPv6, which can make consistency a challenge. Routing in IPv6 works very much the same as in IPv4. Once you are used to the fact that IPv6 addresses just look different, you are left with something that, from a router's perspective, works very much like IPv4.

IPv6 does offer many new features, including additional security, advanced routing, and multicast capabilities, but knowledge of those features is not necessary to get your network connected to the IPv6 Internet. It is very easy to use knowledge you have gained building and maintaining IPv4 networks to enable IPv6 on your network. Once you have established transit there will still be a lot of work needed to get all of your hosts and services IPv6 enabled, but you have overcome one of the most difficult parts of the process. Taking advantage of new features in IPv6, enabling IPv6 in software, and bringing IPv6 to the desktop, laptop, and server are all subjects deserving of their own articles.

I hope this has provided you with enough useful information to establish IPv6 transit on your network. I think that you will find that implementing IPv6 is not a back-breaking, budget-busting exercise. Rather, there is a good chance your equipment and even your existing providers support IPv6 in some form, leaving you with a task that is easily manageable across networks of all sizes and complexities.

REFERENCES

[1] Mark Kosters and Megan Kruse, "IPv6: It's Time to Make the Move," *;login:*, vol. 33, no. 2 (April 2008): http://www.usenix.org/publications/login/2008-04/openpdfs/kosters.pdf.

[2] MTR: My Trace Route/Matt's Trace Route: http://www.bitwizard.nl/mtr/.

[3] The IPv6 Day Tunnel Broker List: http://www.ipv6day.org/action.php?n=En.GetConnected-TB.

[4] The Internet 2 Project: http://www.internet2.edu/.

[5] David Piscitello, "Are Commercial Firewalls Ready for IP Version 6?" *;login:*, vol. 33, no. 2 (April 2008): http://www.usenix.org/publications/login/2008-04/pdfs/piscitello.pdf.

BRIAN KIROUAC

secure email for mobile devices

Brian Kirouac is the CTO and a principal security consultant for Security Horizon, Inc., and a faculty member of the University of Advancing Technology. He has an MS in both computer science and management, as well as a few certifications, including the CISSP, PCI-QSA, ISRM, and ISAM. He is a contributor to *The Security Journal* and a co-author of *IT Security Interviews Exposed: Secrets to Landing Your Next Information Security Job.*

bkirouac@securityhorizon.com

FOR SMALL BUSINESSES, SETTING UP mobile devices to send and receive email is fairly easy. Most smartphones can guide the user through the configuration process. The default configurations use plaintext (unencrypted) connections. Configuring these devices to connect in a secure manner takes more work. If you wish to do this on the cheap and use self-signed certificates, there are more steps required. This article attempts to guide you through the steps to configure a server to provide, and a mobile device to utilize, secure email services *cheaply*.

For many years I have been a staunch Linux user. Over the past few years I've slowly started to become an Apple fanboy. Started with an iPod, then an iTouch for my daughter. My first MacBook purchase was two years ago. My latest purchase was the iPad. I love the combination of toy and work device (I am writing this article on my iPad as I fly from COS to IAD). My wife loves the toy part.

Like most of us, I have both work and personal email accounts. For almost 20 years I have used ssh tunnels to secure my email connections, both for work and home. This has worked flawlessly but only for laptop or desktop use. For my Blackberry I had to open the IMAP port for connections from the Blackberry Internet Service (BIS) servers. I wasn't overly happy with this but it did serve a purpose. To get email on my iPad and my employees' iPhones something different had to be done.

BIS connections come from a limited-size, known set of IP addresses. iOs devices connect from all over the world with no set IP address. My goal was to enable access to send and receive email on iOs devices. Being a security company and not wanting to end up on "The Wall of Sheep" at conferences, the access had to be secure, meaning strongly encrypted.

Secure encrypted email access has been around for a long time: IMAPS, SMTPS, SMTP utilizing STARTTLS. All of these use SSL to encrypt the connection. I set up my first Sendmail server with an SSL certificate close to 15 years ago. This is nothing new.

Most SMTP servers do not care if you use a self-signed certificate. Setting up IMAPS for the BIS access also allowed the use of self-signed certificates. The iOs (and Android) devices, on the other hand, *do* care. By default these devices will reject self-signed certificates, but the use of self-signed certificates with these smart devices is doable.

For my home systems, I did not want to fork out the money for a commercial certificate. My company is a small business and did not want to spend money on a certificate that would only be used for internal employees. By pure coincidence, one of my customers switched from Blackberry to Droid and wanted the same type of access to their email. So now my problem was to create secure email communications for iOs and Android devices, allowing them to both send and receive email through company-controlled servers. The only resources available are man hours—no funds will be allocated. Get it done now!

The first thing I did was to use my GoogleFu to see who else has done this. I received very disappointing results. I could find parts of the process but not one that covered the complete process. Most of the guides I found referred to using a script that came with the individual applications; none of them was a comprehensive guide.

I am going to step through the process I used to configure a server to provide the required services. These steps must be completed prior to configuring your email accounts on the smartphones. The servers run Fedora Core or CentOS. The default install for these distributions uses Sendmail for SMTP and Dovecot for IMAP. For authentication, SASL will be used as provided by cyrus-sasl. For Webmail we use Squirrelmail on Apache HTTPD (although this article will limit its discussion of SSL configuration to a gloss on Apache's and nothing further).

To follow this how-to make sure you have the following packages installed:

- openssl
- httpd
- dovecot
- sendmail
- sendmail-cf
- cyrus-sasl
- iptables

The first hurdle to cross was generating a certificate that could be used across Sendmail, Dovecot, and Apache. This one certificate will also be for the global *.company.com domain. The reason for one certificate is to make it easier on the end users, since they will have to install the certificate on each of their devices. Some of our clients are not computer-centric and do not need nor want to understand the underpinnings of the system or jump through hoops for security. They want things to "just work."

The system administrator in me desires things to work after an unattended reboot. So I also have my certificate created without a passphrase. To extend the time between repeating these steps the certificate is good for 3650 days (almost 10 years).

The Linux distributions we use come with /etc/pki. In this directory is a set of folders designed to make things easier for creating and maintaining SSL certificates. For me this just makes things more complicated than they need to be. Instead of having folders for each type of service, I prefer just one folder that holds my certificates.

Making Certificates

The first step is to find your Certifying Authority certificate. Fedora and CentOS both come with one already in /etc/pki/tls/certs. This file belongs to the ca-certificates package.

In this same directory is a Makefile and a script called make-dummy-cert, which is part of the openssl package. Modifying the make-dummy-cert script is what I chose to create my certificates in a repeatable manner. Listing 1 contains the script after modification. Save this script as make-company-cert and run it.

```
./make-company-cert
```

This will generate three files: companyname-full.pem, companyname-key .pem, and companyname-cert.pem.

```
#!/bin/sh
umask 077
# The name of the file you wish to generate.
target="companyname"
# The answers to the questions openssl will ask.
answers() {
  # Country Name (2 letter code) [GB]:
  echo --
  # State or Province Name (full name) [Berkshire]:
  echo Colorado
  # Locality Name (e.g., city) [Newbury]:
  echo Colorado Springs
  # Organization Name (e.g., company) [My Company Ltd]:
  echo Company Name, Inc.
  # Organizational Unit Name (e.g., section) []:
  echo IT Department
  # Common Name (e.g., your name or your server's hostname) []:
  echo *.companyname.com
  # Email Address []:
  echo root@companyname.com
}

PEM1=`/bin/mktemp /tmp/openssl.XXXXXX`
PEM2=`/bin/mktemp /tmp/openssl.XXXXXX`
trap "rm -f $PEM1 $PEM2" SIGINT
answers | /usr/bin/openssl req -newkey rsa:1024 -keyout $PEM1 -nodes -x509
-days 3650 -out $PEM2 2> /dev/null
cat $PEM1 >> ${target}-key.pem
cat $PEM2 >> ${target}-cert.pem
cat $PEM1 >  ${target}-full.pem
echo ""   >> ${target}-full.pem
cat $PEM2 >> ${target}-full.pem
rm -f $PEM1 $PEM2
```

LISTING 1: MODIFIED VERSION OF MAKE_DUMMY_CERTS FOR CREATING SELF-SIGNED CERTIFICATES

Configuring Servers to Use SSL

Adding this certificate to Apache HTTPD was easy. By default the mod_ssl module is installed with the configuration file /etc/httpd/conf.d/ssl. Edit this file and change the following lines:

- Add the newly created certificate to the configuration file.

```
SSLCertificateFile /etc/pki/tls/certs/companyname-full.pem
```

- Comment out the other certificate lines.

```
SSLCertificateKeyFile
SSLCertificateChainFile
SSLCACertificateFile
```

The next step was to add the certificate to Dovecot. Edit the file /etc/dovecot.conf. For the protocols line choose:

```
protocols = imap imaps
```

Regular unencrypted IMAP is left for those still using the ssh tunnels, as I did not want to deal with changing everyone's configuration. We do not set up POP or POPS, so email stays on the server, where it can be backed up. If the user removes the email from the server to their laptop/desktop, we are no longer responsible for backing it up.

To configure Dovecot to use the SSL certificate, edit the following lines.

```
ssl_cert_file = /etc/pki/tls/certs/companyname-cert.pem
ssl_key_file = /etc/pki/tls/certs/companyname-key.pem
```

The next beast to slay is the line noise configuration of Sendmail. Adding the generated certificate to the configuration is easy. Go to the directory /etc/mail and edit the file sendmail.mc and add the following lines:

```
define('confCACERT_PATH', '/etc/pki/tls/certs')dnl
define('confCACERT', '/etc/pki/tls/certs/ca-bundle.crt')dnl
define('confSERVER_CERT', '/etc/pki/tls/certs/companyname-cert.pem')dnl
define('confSERVER_KEY', '/etc/pki/tls/certs/companyname-key.pem')dnl
```

To enable the different submission agents add the following lines:

```
DAEMON_OPTIONS('Port=smtp, Name=MTA')dnl
  DEFAULT Port 25
DAEMON_OPTIONS('Port=submission, Name=MSA, M=Ea')dnl
  DEFAULT Port 587
DAEMON_OPTIONS('Port=smtps, Name=TLSMTA, M=s')dnl
  DEFAULT Port 465
```

The next step is to configure Sendmail to use SASL for authenticating users before allowing them to relay email through your server. First you have to decide what type of authentication you wish to use. So that my users have just one password to forget, we use plain authentication which is tied to their host username and password. (In a possible future article, I'll discuss generating and using client certificates for authentication.) To the sendmail.mc file add the following lines:

```
define('confAUTH_OPTIONS', 'A p y')dnl
TRUST_AUTH_MECH('LOGIN PLAIN')dnl
define('confAUTH_MECHANISMS', 'LOGIN PLAIN')dnl
```

For testing purposes leave the p out of confAUTH_OPTIONS. When the p is included it will not accept plain text logins over unencrypted connections, which makes testing the new configuration a bit more complicated.

Generate your new sendmail.cf by running the following while in the /etc/mail directory:

```
make sendmail.cf
```

Now cyrus-sasl must be configured. By default sasl wants to use it's own database of usernames and passwords. This file is /etc/sasldb2. If this file does not exist sasl will fail and your users will not be able to authenticate. To create this file I just used touch:

```
touch /etc/sasldb2
```

Adjusting the Firewall

To ensure that users can access the newly configured services, iptables needs to be configured. Edit /etc/sysconfig/iptables and add the following lines (typically just after the rule to allow ssh (port 22) access):

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 25 -j
ACCEPT
-A RH-Firewall-1-INPUT -s 192.168.1.0/24 -m state --state NEW -m tcp -p tcp
--dport 143 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j
ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 465 -j
ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 587 -j
ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 993 -j
ACCEPT
```

Now run the following commands to restart the services:

```
service httpd restart
service saslauth restart
service sendmail restart
service dovecot restart
service iptables restart
```

Ensure that all of these services are configured to start automatically on reboot. (I missed this step for one of the services on the customer's machine. The customer requested we reboot his server a few days after getting everything working. I missed the saslauth autostart and the user could receive but not send email with his Droid.)

```
chkconfig httpd on
chkconfig saslauth on
chkconfig sendmail on
chkconfig dovecot on
chkconfig iptables on
```

Testing

Now to test and ensure that everything is working. Testing from an external source is suggested. Use your browser to test the HTTPS access. Your prompts and how you view the certificate for HTTPS access will vary from browser to browser.

To test Sendmail, connect to port 25. Issue an ESMTP hello with `ehlo me` and you should get a response back similar to Listing 2. The response contains the `250-STARTTLS` line that shows it allows TLS connections. The `250-AUTH LOGIN PLAIN` shows the allowed authorization mechanisms.

Note: If you included the `p` option to the `confAUTH_OPTIONS` option in your sendmail.mc you will not see the `250-AUTH`.

```
user@host# ncat 1.2.3.4 25
220 companyname.com ESMTP Sendmail 8.14.3/8.14.3; Sun, 18 Jul 2010
14:59:55 -0600
ehlo me
250-companyname.com Hello localhost [4.3.2.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-STARTTLS
250-DELIVERBY
250 HELP
221 2.0.0 companyname.com closing connection
user@host#
```

LISTING 2: AFTER CONNECTING TO YOUR SMTP PORT, YOU SHOULD SEE STARTTLS IN THE LIST OF ENHANCED STATUS CODES.

To test Dovecot, follow Listing 3. The inclusion of STARTTLS in the welcome message shows the certificate has been loaded and encrypted connections can be used.

```
user@host# ncat localhost 143
* OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID
ENABLE STARTTLS AUTH=PLAIN] Dovecot ready.
^C
user@host#
```

LISTING 3: TEST DOVECOT (YOUR IMAP SERVER) BY CONNECTING TO PORT 143 AND LOOKING FOR STARTTLS AGAIN.

Now reboot your server and test again. Once you are satisfied that the server is working properly you can start working on the clients.

Client Side

For the clients to accept the self-signed certificate as valid they must install the certificate on the device or application. Since they cannot get email, the best way is for them to download via HTTP(S). Copy the certificate to a location on the Web server accessible from the mobile device and your laptop/desktops. Depending on the device or application, the file extension will be either .crt or .pem. Listing 4 shows the commands I used to move the certificate files into place.

```
mkdir /var/www/html/certs
cp /etc/pki/certs/companyname-crt.pem /var/www/html/certs/cert.pem
ln /var/www/html/certs/cert.pem /var/www/html/certs/cert.crt
```

LISTING 4: PLACING CERTIFICATES WHERE THEY CAN BE DOWNLOADED

On Safari/Firefox as well as iOs/Droid devices, open the default Web browser and connect to the URL http://www.companyname.com/certs/cert.crt. You will be presented with a dialog box to install the profile on your device/or application. On the iOs device, this will create a new Profile in your configuration. This can be viewed under Settings->General->Profiles. Your profile will be labeled as *.companyname.com.

On your smartphone/mobile devices you can now proceed to create your email accounts. During the initial process choose the SSL or TLS options as offered.

For Thunderbird to accept the certificate, you must download the certificate to the local machine. You cannot use the ".crt" URL since your browser will think this is a certificate that needs to be installed. Thus you must use the URL http://www.companyname.com/certs/cert.pem. Save this file to a good location. Then within Thunderbird open the Preferences window. Choose Advanced->Certificates->View Certificates. Use the Import… button to import your new certificate.

You should now be able to check and send email securely from anywhere in the world where you have Internet connectivity. The fear of appearing on the Wall of Sheep should also be diminished (this fear should *never* go away).

Have a happy and secure computing day!

DAVID N. BLANK-EDELMAN

practical Perl tools: perhaps size really does matter

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), 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. David is honored to have been the recipient of the 2009 SAGE Outstanding Achievement Award and to serve on the USENIX Board of Directors beginning in June of 2010.

dnb@ccs.neu.edu

I REALIZE THIS IS A VERY SENSITIVE topic for some people, so I am going attempt to broach this subject with as much care and finesse as possible. Today we are going to be talking about being small—how to accept and embrace being small, how to use what you have, that sort of thing. In fact, we're going to make you feel good about using things that are downright tiny. Or perhaps I should say "::Tiny", because the subject of this column will be the modules whose names end in ::Tiny.Took

Many of the modules in this family were inspired by work done initially by Adam Kennedy and a series of modules he created. These modules were designed to take some of the more useful but heavyweight modules in the Perl ecosystem and partially reimplement them as lean and mean as possible. There's a somewhat tongue-in-cheek module called Acme::Tiny, not written by Kennedy, whose doc lists a set of "commandments" that represents the gestalt of the ::Tiny family:

1. The module should be implemented in "as little code as possible."
2. The module should implement a useful subset of functionality.
3. The module should use at least 1/10th the amount of memory overhead, ideally less than 100k.
4. The module MUST have no non-core dependencies.
5. The module MUST be only one single .pm file.
6. The module should be back-compatible to at least 5.004.
7. The module should omit functionality rather than implement it incorrectly.
8. If applicable, the module should be compatible with the larger module.

::Tiny Markup

So let's actually look at some of the modules from this family. The first set I'd like to explore is in the general area of markup language processing. Diving into the deep end with the most markup of markup languages, XML::Tiny is a module that could be used if you had some basic XML parsing you needed done but didn't want the memory or module size overhead of the more complete parsers like XML::LibXML. For example, if we used the

Yahoo! Geolocation service we discussed in the June 2006 column, it would hand us back a result that looked like this without the indentation:

```
<?xml version="1.0"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:yahoo:maps" xsi:schemaLocation="urn:yahoo:maps
           http://api.local.yahoo.com/MapsService/V1/GeocodeResponse.
xsd">
  <Result precision="address">
    <Latitude>37.859576</Latitude>
    <Longitude>-122.291659</Longitude>
    <Address>2560 9th St</Address>
    <City>Berkeley</City>
    <State>CA</State>
    <Zip>94710-2516</Zip>
    <Country>US</Country>
  </Result>
</ResultSet>
<!-- ws05.ydn.ac4.yahoo.com uncompressed/chunked Sun Jul 25 17:34:42
PDT 2010 -->
```

XML::Tiny will give you back a data structure that looks like this:

```
0 ARRAY(0x100b5baf8)
  0  HASH(0x100d0be48)
    'attrib' => HASH(0x100d0bd40)
      'xmlns' => 'urn:yahoo:maps'
      'xmlns:xsi' => 'http://www.w3.org/2001/XMLSchema-instance'
      'xsi:schemaLocation' => 'urn:yahoo:maps
      http://api.local.yahoo.com/MapsService/V1/GeocodeResponse.xsd'
    'content' => ARRAY(0x100d0be18)
      0 HASH(0x100d0bf38)
        'attrib' => HASH(0x100d0bff8)
          'precision' => 'address'
        'content' => ARRAY(0x100d0bf50)
          0 HASH(0x100d0c040)
            'attrib' => HASH(0x100d0c0a0)
              empty hash
            'content' => ARRAY(0x100d0c028)
              0 HASH(0x100d0be30)
                'content' => 37.859576
                'type' => 't'
            'name' => 'Latitude'
            'type' => 'e'
          1 HASH(0x100d0c148)
            'attrib' => HASH(0x100d0c1c0)
              empty hash
            'content' => ARRAY(0x100d0c118)
              0 HASH(0x100d0c0d0)
                'content' => '-122.291659'
                'type' => 't'
            'name' => 'Longitude'
            'type' => 'e'
          ...
```

It is the same data structure you would expect from XML::Parser::EasyTree. For this case, I find all of the dereferencing you need to do (i.e., $document

->[0]->{content}[0]->{content} . . . and so on) to be a bit tedious, so here's a quick way we can cheat. XML::Tiny's author also released a separate module called XML::Tiny::DOM that builds on XML::Tiny in a way that makes it *much* easier to get to the data we need. We'll lose some of the ::Tiny purity if we use it, but it means our code can look like this (pay particular attention to the clarity of the last two lines):

```
use LWP::Simple;
use URI::Escape;
use XML::Tiny::DOM;
use IO::Scalar;

# usage: scriptname <location to geocode>

my $appid  =  "{your Yahoo! API key here}";
my $requrl  =  "http://api.local.yahoo.com/MapsService/V1/geocode";

my $request = $requrl .
   "?appid=$appid&output=xml&location=" . uri_escape( $ARGV[0] );
my $RESPONSE = new IO::Scalar \get($request);

my $document = XML::Tiny::DOM->new($RESPONSE);

print "Latitude: " . $document->Result->Latitude . "\n";
print "Longitude: " . $document->Result->Longitude . "\n";
```

I feel compelled before we move on to the next module to let you know that some people are pretty negative about XML::Tiny's lack of XML-parsing rigor and handling of edge-cases. It certainly is the wrong module to use if pedantic parsing of XML is important to the task or the input data is at all complex. But in a case like the one above, it seems to do peachy.

The mention of complex data offers a segue to the next module: YAML::Tiny. YAML::Tiny's documentation explains its purpose like so:

> The YAML specification is huge. Really, *really* huge. It contains all the functionality of XML, except with flexibility and choice, which makes it easier to read, but with a formal specification that is more complex than XML.
>
> The original pure-Perl implementation YAML costs just over 4 megabytes of memory to load. Just like with Windows .ini files (3 meg to load) and CSS (3.5 meg to load) the situation is just asking for a YAML::Tiny module, an incomplete but correct and usable subset of the functionality, in as little code as possible.

(Note: we'll discuss the other large module examples mentioned above later in this column.)

Using it is as simple as using the standard YAML modules (again, from the doc):

```
use YAML::Tiny;
my $yaml = YAML::Tiny->read( 'file.yml' );
my $root = $yaml->[0]->{rootproperty};
my $one  = $yaml->[0]->{section}->{one};
my $Foo  = $yaml->[0]->{section}->{Foo};
...
```

Lest you think ::Tiny is somehow best for markup parsing only, let's look at two modules that handle markup generation. HTML::Tiny lets you write straightforward code to produce HTML documents, so code like this:

```
use HTML::Tiny;
    my $h = HTML::Tiny->new;

    # Generate a simple page
    print $h->html(
        [   $h->head( $h->title(';login Column') ),
            $h->body(
                [  $h->h1( { class => 'headline' }, 'First Section' ),
                   $h->p(
                      'This is a test.',
                      'Have you tried turning it off and on again?'
                   )
                ]
            )
        ]
    );
```

generates HTML like this:

```
    <html><head><title>;login Column</title></head>
    <body><h1 class="headline">First Section</h1><p>This is a test.</p>
    <p>Have you tried turning it off and on again?</p>
    </body>
    </html>
```

In a similar vein, if you parse or generate Cascading Style Sheets (CSS code), you might find Kennedy's CSS::Tiny module (mentioned above) useful for simple, quick work. It can parse a .css file, let you make changes, and write the file back out using much less overhead than CSS.pm.

The last markup-related module I'll mention is worth your attention not least for the gumption the author showed in even attempting to write it. Kennedy's Template::Tiny tries to give the Template Toolkit 2 distribution the ::Tiny treatment. Template Toolkit 2 (known as TT2 for short) is probably the single most popular templating engine in use in the Perl world today. I think you would be hard pressed to find a Perl Web framework that doesn't offer some way to use TT2 to generate content. Like other fully featured templating engines, it offers a panoply of ways to produce output based on a combination of static text and dynamic data. An example TT2 template (from the tutorial) looks like this:

```
    [% INCLUDE header %]

    People of [% planet %], your attention please.

    This is [% captain %] of the
    Galactic Hyperspace Planning Council.

    As you will no doubt be aware, the plans
    for development of the outlying regions
    of the Galaxy require the building of a
    hyperspatial express route through your
    star system, and regrettably your planet
    is one of those scheduled for destruction.

    The process will take slightly less than
    [% time %].

    Thank you.

    [% INCLUDE footer %]
```

Here you can see static text with placeholders that will be filled in with the appropriate content (in some cases the content of a variable, in others the contents of other files). You can also do fun stuff like:

```
<ul>
     [%  FOREACH page IN pages %]
          <li><a href="[% page.url %]">[% page.title %]</a>
     [%  END %]
</ul>
```

TT2 is so feature-full, there is an entire 592-page O'Reilly book on the subject. This means that creating a TT2 ::Tiny module is the equivalent of building a to-scale model of the QE2 in a bottle with the expectation that it should also be able to still transport a small number of passengers.

Given the author, I suspect the public Template::Tiny code is solid, but it is still in enough flux that I hesitate to recommend you use it. I would, however, recommend you follow Kennedy's blog (http://use.perl.org/~Alias/journal/) as he posts about the module's development and the engineering decisions he has made as he progresses. The first set of posts are at:

http://use.perl.org/~Alias/journal/39983
http://use.perl.org/~Alias/journal/39991
http://use.perl.org/~Alias/journal/40013
http://use.perl.org/~Alias/journal/40015

and for added fun, you can read about the JavaScript port of Template::Tiny at http://use.perl.org/~Alias/journal/40126.

::Tiny Programming

Lest you think the ::Tiny philosophy only applies to markup-related stuff, let's careen off in an entirely different direction and look at two programming-related ::Tiny modules. The first is the ::Tiny entry into an already pretty crowded field: class builders. If you are a Perl programmer with a heavy OOP bent (and yes, there are some of them out there), you know that Perl doesn't by default provide any shortcuts for the sometimes tedious process of creating accessors for data in the objects you define. The modules many people gravitate toward to handle this are something from the Class::Accessor distribution. Kennedy is very clear in his documentation about why Object::Tiny module is an improvement over those modules:

- Object::Tiny is 93% smaller than Class::Accessor::Fast.
- Class::Accessor::Fast requires about 125k of memory to load.
- Object::Tiny requires about 8k of memory to load.
- Object::Tiny is 75% more terse to use than Class::Accessor::Fast.
- Object::Tiny is used with the least possible number of keystrokes (short of making the actual name Object::Tiny smaller).
- And it requires no ugly constructor methods.

See the Object::Tiny documentation for the rest of the comparisons.

Using the module is as simple as:

```
package Foo::Bar;
use Object::Tiny qw{ foo bar baz };
```

and, lo and behold, when you create your Foo::Bar objects you get $object->foo, $object->bar, and $object->baz like magic. Super simple, super easy.

The other programming-related module worth mentioning given our limited space is Try::Tiny, which gives you the fairly standard try/catch exception-handling idiom other languages scoff at Perl for lacking. That's the one that looks like this:

```
try     { ... }
catch   { ... }
finally { ... };
```

The idea is you can run a block of code as a "try"; if it fails the error can be handled by the "catch" code and "finally" is run if either of those two blocks returns success. You can do this sort of error handling using eval() in Perl (and in fact, that's how "try" is run), but it turns out there are a number of fiddly details mentioned in the Try::Tiny documentation that make a bare eval() not as useful as you'd like for this idiom. Try::Tiny handles all of those details for you beautifully.

::Tiny To-Go

There are far too many interesting ::Tiny modules worthy of your attention to fit into one column. I can only show you two more, but I encourage you to go searching on CPAN for "::Tiny". The first of the two harks back to a February 2006 *;login:* column on configuration files. Config::Tiny is a minimalistic .ini-format config file reader and writer. Here's the sample example from the documentation (slightly abridged):

```
use Config::Tiny;
my $Config = Config::Tiny->new();
$Config = Config::Tiny->read( 'file.conf' );
my $one = $Config->{section}->{one};
$Config->{newsection} = { this => 'that' };    # Add a section
$Config->{section}->{Foo} = 'Not Bar!';        # Change a value
delete $Config->{_};                           # Delete a value or section
$Config->write( 'file.conf' );
```

If you need something to read and write config files only machines will touch, Config::Tiny works great. It is a less good option if you care about preserving comments, order, whitespace, etc., in your config file (because it doesn't). For that you'll want to seek out some of the other modules I mentioned in the February 2006 issue.

OK, last module. Along with Try::Tiny it is perhaps one of the most useful in this column. If you have ever had to capture the output of an external program for use in a Perl program, Capture::Tiny is the module for you. Before you do anything else, I'd highly recommend you read the slides to David Golden's talk "How (NOT) to capture output in Perl" at http://www.dagolden.com/wp-content/uploads/2009/04/how-not-to-capture-output-in-perl.pdf.

Read that talk even if you are not planning to use anything ::Tiny. In it he discusses the various ways people try to capture output and the drawbacks of most of the ways you thought were the right ones. By the end of the talk, you'll be begging for a module that just Does The Right Thing. Golden provides a happy ending to his talk by making Capture::Tiny available. It provides capture, capture_merged (STDOUT and STDERR merged), tee and tee_merged functions that do just what you would hope they would do and do it well. If there is one true way to capture output from another command, this is probably it. And it is ::Tiny to boot.

With that last nugget, I think it is time to take my leave. I hope you've gained a new appreciation for the smaller things in life. Take care, and I'll see you next time.

PETER BAER GALVIN

Pete's all things Sun: the "problem" with NAS

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

COMPUTER STORAGE HAS EVOLVED from Directly Attached (DAS) to Storage Area Networks (SAN). Along the way, Sun in 1984 invented NFS, and Network Area Storage (NAS) was born. Since then other NAS protocols have been added, most notably the Windows-based Server Message Block (SMB), a.k.a. CIFS. But throughout the history of storage, NAS has been regarded as poorly performing and unreliable compared to SAN and DAS. Certainly NetApp's creation of a NAS "appliance" helped move NAS from being a science project to a mainstream production solution, but in my opinion NAS is still underappreciated and underdeployed. Perhaps in light of the new generation of NAS appliances, that will change.

At a more philosophical level, it's worth asking, "What is SAN? What is NAS?" Fundamentally, they are storage arrays that make disk space available via varying protocols over varying interconnect media. For the most part, both technologies are available with Fibre Channel (FC), SATA, and SAS disks. Both have disks of varying speeds, capacities, and performance. Traditionally, SANs have been FC connected and NAS appliances connected via Ethernet, but many current products provide both interconnects—block transactions occur via FC or iSCSI and file transactions over Ethernet. A proof point of this merger of NAS and SAN is the FCOE protocol, which places Fibre Channel frames over Ethernet networks. Perhaps the most straightforward definition is that "SAN" is block-based storage and "NAS" is file storage, and that a given data center should chose which to use for any given application or function. After those decisions are made, it is easier to determine the best products to implement the resulting storage architecture. Now let's consider the problem with NAS as well as the solutions it can provide.

The "Problem"

Over the years I've seen many, many computing infrastructures. Back in the "old days" (say, the 1980s), we had servers and SANs for production, and NAS was pushed to the side. It was typically used for home directories and the storage of utility programs, if at all. In those cases, NAS storage was mounted to all servers as well as all workstations.

That helped NAS gain a reputation for unreliability—probably because any failure caused everyone to notice it, and failures were difficult to recover from (with hard mounts never timing out, for example, taking down all computing until the NAS server could be fixed). Also, many situations called for “cross mounts,” where servers would mount each other’s directories via NFS. If one server then failed, all servers would eventually end up hanging until the failed one recovered. NFS also had quirks like “stale file handles” that left a bad taste in the mouth.

So failures of NFS servers were quite painful to the computing infrastructure. Why did NAS servers fail as often as they did? Well, they were non-clustered, while their SAN brethren typically had more redundant components and automatic recovery from problems. Originally, a “NAS server” was just a general-purpose Sun server running NFS. SAN originally was and usually still is a purpose-built storage array. Also, they were and still are network-connected. Back in the day, there was typically one network connection to each workstation (and frequently between servers as well). That one link was used for NAS and non-NAS network traffic. Even if there was a separate network carved out for storage communication between the servers and NAS, it was rarely redundant. Multiple use and single points of failure meant NAS was more prone to failure than SAN. Thus the lingering impression that SAN is more reliable than NAS.

There is also an impression that SAN has better performance than NAS. First, consider the communications protocols. For SAN, the Fibre Channel medium carries SCSI protocols between servers and storage arrays. SCSI is (by definition) optimized for storage operations. TCP/IP is a general protocol used for everything from sending one character at a time (telnet, for example) to bulk file transfers (ftp and NAS). In addition, TCP/IP runs over a shared medium, so it has to deal with collision detection and recovery. The TCP/IP communications are therefore more chatty and less efficient than the equivalent SCSI commands (where there are equivalents). Also, the caching of NAS I/O is less effective than SAN, due to NAS storage being shareable. As one example, consider metadata caching. On a SAN, once a LUN is mounted, the mounting server “owns” that LUN. Over the course of I/Os it can cache all the data and metadata it needs, infinitely. With NAS, because other systems might be accessing the same directories and files, NAS clients must recheck with the NAS server periodically to see if any metadata has changed. Those timings can be modified via mount options but are typically measured in seconds, not minutes. If the NAS client detects that its cached data is invalid, the clients have to throw out the cached data and metadata and reload it in the worst case (depending on file open modes, for example). Thus the overhead of NAS operations is higher than SAN operations.

All of this adds up to NAS performance challenges. With NAS, a single user can seemingly cause more of a performance hit than on SAN. For example, again back in the ’80s, we would debug NAS performance problems by watching the network traffic and finding a user flooding the networking with NAS requests. Frequently, the problem would be a single user running a UNIX “find” command across some directory structures mounted via NFS. A single user running a single command could bring the NAS server to its knees. The equivalent operation across a SAN would be less onerous, most likely due to the large caches included in most SANS.

That Was Then, This Is Now

NAS is not just for sharing anymore and is past most of its adolescent problems. In fact, NAS is now quite mature, fast, and reliable. But NAS, in many data centers and in many instances, is still relegated to tasks of lesser importance. Tier-1 use of NAS (for non-stop production) seems to be rare, but shouldn't be. Consider the latest generations of two great NAS products: NetApp's FAS series of "filers," or NAS appliances, and the Oracle/Sun Storage 7000 line. Both product lines scale from small to very, very large capacities. And both scale up to very high performance, although that is harder to prove. The SPECsfs2008 benchmark (http://www.spec.org/sfs2008/) is one source of performance information about NAS servers, and there are many posted results, but Sun is not one of the contributors. Sun (rightly, in my opinion) considers it to be a severely flawed benchmark, but it's about the only thing we've got that shows comparative NAS performance. On-site testing of real environments is always the best indicator of performance but usually difficult to do and not commonly done. In testing in my company's lab, my colleague Sean Daly drove VMware to push 990.48 MB/s of throughput and 149,227 IOPS (I/O operations per second) from a NetApp filer. That is certainly a lot of performance. And both NetApp and Sun NAS servers can be configured as high-availability clusters, with fast failover in the case of component failures (with the NetApp failing over faster than the Sun 7000 in our testing).

Why, then, is NAS not taking the world by storm? In some ways it is, as indicated by the rapid growth rates of NetApp and Sun's storage group. But there are certainly many cases when NAS could and should be used but where DAS or SAN is used instead. The reasons for that are as varied as computing infrastructures and the managers that run them. In many cases it's a simple case of familiarity. Storage managers have more experience with SAN than NAS, and they go with what they know. In other cases it is for simplicity. Running one kind of storage, from one vendor, is simpler than running two kinds of storage solutions from one or two vendors (the existing SAN vendor or a new NAS vendor). And in some cases the lack of NAS use is based on previous painful experiences, or a lack of understanding of the state of NAS servers and their features. It is this last group that I'm hoping to address with this column.

The Case for NAS

If SAN storage arrays also have high reliability and high performance (for the most part), then why not just run SAN instead of NAS? Consider some of the more potent features of good NAS storage. Also consider that even though some of these features are available with SAN storage, they are frequently more expensive, require extra devices, or are much more limited than their NAS brethren.

- Snapshots—read-only point-in-time, fast, low-space-use file system copies—and clones, read-write versions of the same, are "magical" in their function and utility. When I first tested ZFS, for example, I was taking snapshots every minute of every day of every month for a year. I had thousands of snapshots, each representing the state of the file system at that minute. The power to undo and redo any file system changes is extreme and not used enough.
- Diskless booting allows servers to run as "field replaceable units," running interchangeably except for their knowledge of which remote boot disk image they are associated with. For this model to work, the servers must be configured similarly, and must all have access to all external storage units. That is certainly easier with NAS storage than with SAN storage. But consider combining diskless booting with cloning. A datacenter manager could create a "golden image" of a server operating system, configured

exactly as needed, and then clone it hundreds of times to make hundreds of identical boot disks (for hundreds of servers). When a change is needed, a new golden image can be created and cloned and the servers rebooted to use the new versions. Many versions of golden images (and boot disks) can be kept for revision control, testing, disaster recovery, and so on. This functionality is similar to that touted by virtualization vendors, but done at the disk level rather than the virtual disk level. Both have their place in the datacenter, but with diskless booting no virtualization (or virtualization license) is needed. To improve performance, you could consider using an internal disk for swap space, keeping swap traffic off of the network and the NAS array.

- Replication of snapshots allows disk-to-disk backups as well as easy disaster recovery site synchronization. When combined with diskless booting, a single NAS server replicating to a similar server in a remote datacenter "solves" the data part of disaster recovery. Set up a farm of servers at the remote site, and the compute portion is solved as well. That remote site can also be your disk-to-disk backup site, with production replicating to disaster recovery. Some environments are using such a scenario instead of backing up the data to disk, while some others only put tape drives in the remote site, and perform disk-to-disk-to-tape backup in that manner.
- Sharing is probably the most compelling feature of NAS over SAN. Home directories of users can be shared to all servers that the users log in to, giving them their environment across all servers. Less common but equally useful is the storage of applications on NAS. Those applications can be installed once and maintained in one location (with snapshots or other methods for revision control), and all servers can have access to the same versions of all applications. Also becoming more popular is the storage of application data on NAS. For example, Oracle happily recommends using NAS to store Oracle Database data. Even Oracle's RAC clustering can use NAS storage for the data, and it is actually much easier to set up that way than using SAN storage. As always, when in doubt check with your application vendors to see what they support. You might be surprised to find out that NFS is on the list.
- Ease of management is something rarely said about traditional storage arrays, although some newer arrays (such as 3PAR and IBM's XIV) are great improvements over their older counterparts. Tasks that take many steps and lots of time on a traditional SAN can take minutes or even seconds on NAS. Consider the pain of expanding the amount of storage available to a host on both a SAN and a NAS. Also consider standard, complicated tasks performed by your storage administrators. Compare the effort and risk (the more commands, the more likely a mistake) to performing the same task on NAS. If you don't have NAS on-site, consider a demonstration by a NAS vendor to show you the differences in administration.
- Deduplication is all the rage, and for valid reasons. It can reduce the amount of storage used by a given set of data, and, depending on the implementation method, it can maximize the use of caches by only storing the deduplicated block once in the cache. Likewise, it can decrease the amount of data replicated between data centers by only sending original blocks, not duplicates. And it is especially useful in environments, such as virtualization, where many copies of the same blocks are stored (operating systems and binaries). SANs have a difficult time including deduplication, and in many cases an external device is needed. Both NetApp and Sun NAS devices include free deduplication.
- Flexibility is the watchword with NAS, as most major NAS solutions (including the two being discussed here) can be used as SAN as well as NAS. These products provide both iSCSI and Fibre Channel connectivity. iSCSI

is useful for connections where NAS is not supported (e.g., the Microsoft Exchange datastore), and where the complexity and expense of FC cables, switches, and HBAs are not wanted. But where maximum performance and reliability via SAN storage from a NAS appliance is desired, the ultimate step of adding an FC SAN attachment between your hosts and your NAS appliance is available.

- Performance analysis and tuning are inarguably easier with NAS devices. Seeing what is happening at a file level is much more revealing than at the block level. There have been many instances in my debugging efforts where the SAN was a black box that we worked around, rather than a source of information useful in determining the cause of the problem. While both NetApp and Sun provide useful tools in their appliances, Sun has done an astonishing job of integrating DTrace into their device, providing never before available details (e.g., heat maps that depict the time each I/O request took to be satisfied).
- Cost can vary dramatically between SAN and NAS, and between vendors and configurations. Certainly a blanket statement such as "NAS is cheaper than SAN" cannot be made. But pricing out a NAS solution, in cases where NAS is a valid fit, is a worthwhile exercise. If possible consider the total cost of ownership over a period of time that suits your site's replacement schedule, say three or five years. Add into that the costs of software licenses, including host-side licensing (such as backup software, EMC PowerPath, and Veritas File System and Volume Manager). Frequently, soft costs are not considered or are considered unimportant, but if possible think about staff time as well.

Making NAS Work

NAS is not a panacea for all things ailing your datacenter. Although NAS performance can be very good, certain workloads can perform worse than very good SANs. Consider the total throughput of your solution, especially as limited by per-spindle IOPS abilities. Fewer large disks in SAN will provide fewer I/Os than a larger number of smaller disks in a NAS, for example. Make sure the I/O being provided by the device is sufficient for your needs.

Also, badly implemented technology will not perform as well or as reliably as well implemented technology. Both SANs and NASes need careful deployment planning, disk layout, and feature utilization. Consider especially mount options and block alignment during implementation. One other likely cause of admins thinking NAS is less reliable than SAN is the interconnect technology. FC switches and cables can only be used for storage connectivity. Ethernet can be used for host and storage connectivity. But those two tasks should not be shared. If the server to storage connection in NAS is treated as nicely as FC is, then NAS will run very well indeed. Certainly, for maximum reliability (and performance) dedicate a VLAN, and if possible two LANS (for redundancy), to NAS I/O. Do not use those networks for other purposes (even backups should be kept separate). Such segregation can go far toward an optimal NAS experience.

The choice of NAS solutions is of course important. I have already mentioned NetApp FAS filers and Sun's Storage 7000. Note that there is currently a patent lawsuit between the two companies. I don't believe that such legal actions should affect your decision-making process, as lawsuits rarely impinge on end users. Other commercial NAS solutions exist, and many are fine products. However, some are "one protocol ponies." Why settle for a device that can only provide data across one protocol, when many-protocol appliances are available? The trade-offs of simplicity versus utility (and cost) need to be considered, but rarely have managers been unhappy that they had too many protocols available to them.

Finally, rather than purchasing an appliance, many sites “roll their own” NAS services by using standard servers and SAN storage. I think many of those sites would be better off with an appliance, given the performance, reliability, feature sets, and ease of administration of appliances. Frequently, once an appliance is deployed in a data center, the datacenter managers find more and more uses for it and move datasets from the existing SAN to the new NAS. A roll-your-own approach might limit performance, reliability, and utility, artificially limiting the use of NAS in an environment.

Conclusion

Many SAN storage devices have many of the NAS features discussed here, but few have all of them. The combination of all of these features makes NAS a very useful, dare I say “compelling,” component of datacenter strategies. NAS is flexible, efficient, and can perform well and reliably. It can also be much easier than SAN to implement and administer. One of our clients recently replaced an EMC DMX 8000 (a high-end SAN) with a cluster of two NetApp FAS arrays. They are very pleased with the trade, citing improved convenience and good performance and reliability. They also note that their purchase of NAS, including three years of maintenance, cost less than renewing one year of maintenance on the DMX 8000. I suggest you consider the benefits your data center could enjoy with an increased use of NAS in production.

Special thanks to Adam Leventhal, Sean Daly, Jesse St. Laurent, and Paul Deluca for contributing to this column.

DAVE JOSEPHSEN

iVoyeur: pockets-o-packets, part 3

Dave Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and is 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

THIS MONTH BRINGS YOU THE THIRD and final installment in my series on Argus, the network flow monitoring and reporting framework. If you didn't catch my first two articles (see June and August issues of *;login:*), you should pull them out of your hamster cage and read what's left of them. I covered PCAP hardware, infrastructure, and I also flamed up on the entire database administration profession in general. In this final article on Argus, I'm going to cover the intricacies of the Argus client utilities, and I'll also probably flame up on the DBAs a little more.

So let me give you a feel for how I use Argus on a day-to-day basis. There are myriad reasons I find myself turning to Argus daily, but two spring to mind as especially common: complaints of network slowness, and daily security reports.

"Network slowness" is, as we both know, usually a PEBCAK [1] problem, but when I do want to get a quick feel for what "the network" is doing, I turn to ratop [2]. ratop is a great little utility that implements a network "top" command using Argus data-flows for input. You can point it at files with -r or -R, but I'll usually just go ahead and point it straight at one of our radium streams with -S. This way, I can log straight into the router and fire it up with:

```
ratop -S localhost:4300
```

and get live utilization info similar to this:

```
ratop -S 10.20.0.2:4300200/07/26.20:49:56 CDT
Rank  StartTime        Flgs  Proto  SrcAddr  Sport       Dir   DstAddr  Dport        TotPkts  TotBytes  State
  1   20:49:37.426974  e     tcp    10.20.34.1.40405     ->    10.20.5.1.ssh         27688    3045752   RST
  2   20:49:36.454273  e     tcp    10.20.1.5.57465      ->    10.20.0.2.4300        80       10832     CON
  3   20:49:39.094155  e     vrrp   10.100.1.20          ->    224.0.0.18            12       840       INT
  5   20:49:35.698224  *     llc    0:a:f4:20:60:85.stp  ->    1:80:c2:0:0:0.stp     9        540       INT
  6   20:49:37.779319  e     icmp   10.20.68.2           ->    10.100.1.10           3        342       URP
  7   20:49:42.364118  e     tcp    10.25.20.6.55616     <?>   10.20.5.1.ssh         3        258       CON
  8   20:49:43.094867  e     udp    10.100.1.20.42033    <->   10.100.1.10.domain    2        180       CON
  9   20:49:43.095310  e     udp    10.20.4.1.1679       <->   8.15.14.7.domain      2        180       CON
```

OK, this is pretty obvious stuff, but Argus gets even more impressive when you start getting into the data-mining clients, and among these, racluster [3] is king. But rather than bore you (more than I already have) with lists of options, let me tell you a story about our daily reports. We have several cron jobs that gather security-relevant information from various systems detailing things like people logging into things, files changing, and IDSes being paranoid, so these can be compiled and emailed out in a daily report format. Among these are a few racluster queries that I find useful. These are things like “chatty kathys” (the top ten bandwidth users):

```
racluster -r <yesterdays_file> -w - -m saddr - ‘src net 10.201.16.0/21’ \
| rasort -m bytes load - -s saddr bytes load  | head -n10
```

“town criers” (top ten broadcasters):

```
racluster -r <yesterdays_file> -w - -m saddr - ‘src net <heathen_subnet> \
and dst host <heathen_subnet_broadcast_addr>’  | rasort -m bytes load - \
-s saddr dport bytes load  | head -n10
```

and “the red light district” (top 10 destination ports by total bytes transferred):

```
racluster -r argus-2010-05-20.log -w - -m proto dport \
| rasort -m bytes load - -s dport bytes load  | head -n10
```

If you managed to stay awake through my last article, then some of these options should already be familiar to you, so I’ll go ahead and summarize them again to make absolutely sure you won’t stay conscious this time.

-r tells racluster the name of an input file. Popular alternatives are to recursively read a directory full of files with -R, or to read directly from an argus daemon or radium process with -S.

-w tells racluster to write out binary data format to STDOUT. If I had specified -w foo, it would have written to a file called foo instead. If no -w is provided, racluster will write human-readable text output. Some Argus tools like rastream have “special” -w options that can do variable expansion for things like writing to date-stamped files.

The story I want to tell you is about the last report I mentioned above (the red-light district). Usually this report looks something like this:

```
80      780506507   730813
22      684953405   675527
443     619322910   423802
8080    536904140   491149
...
```

This is a list of popular destination ports as measured by total byte count and, secondarily, by “load” or bits per second. We use racluster to get a list of all the ports and all the aggregated byte-counts and packet rates for those ports, and then we use rasort to sort the list and give us formatted output.

The -m switch is very important and a little confusing, because it performs related but very different functions in racluster and rasort. The -m switch to rasort is simple enough: it specifies what criteria you want the list sorted by. My chosen criteria in this case are twofold: primarily, I want the list sorted by byte-count, and secondarily, by rate. I’ll talk more about -m in racluster below. The -s switch does the same thing in both tools. It specifies the output format, which respects the order the arguments are passed on the command line, so the first column is the port number, followed by the byte-count, and finally the rate.

So one day, I opened my mail, and the red-light district report looked like this:

```
39756   1885065707   43076716
61589   1884953405   37598340
```

```
51295    1881932291    19342380
57683    1853690414    19121481
13487    1853657566    19197988
10242    1834900162    18292097
5735     1834731617    17534843
48909    1822365775    19418729
63838    1822191099    19795299
49242    1822184229    19288059
```

"Well that's odd," I thought while logging into the pcap server to verify. Sure enough, my top 10 destinations were all random high-number ports, and the byte counts were huge and yet eerily similar. Nearly two gigabytes per port traversing the router between the staging and dev subnets. My first question was, are these ports being used over and over again, like a virus with a list of ports? Or are each of these unique connections? Let's ask Argus:

```
> ra -r <yesterdays_file> - dst port 39756
   16:25:34.560010  e  D   tcp   10.20.49.21.19026  ->  10.20.33.21.39756
15681122 1885065707   FIN
```

The ra [4] client is the simplest of the Argus clients. It literally stands for "read argus," and its job is to read argus data and output it in human-readable format. The options given you should already recognize. I told it to read from my log file, and gave it a tcpdump-style packet filter with the first destination port on my list. In other words, I took the top destination port and asked ra to return whatever flows used that destination port on that day.

There is only one of them, so that 1885065707 bytes represents a single data connection. One flow to a high-number port looks less like a virus and more like the data channel of an ftp session. Now who would be silly enough to incur *my* wrath by using FTP on the network? Well, this query yields a hint to that question too. They're both database server IPs. A few more ra commands to the listed ports verifies that they're all single connections between the same two Oracle boxes.

Let's see if we can confirm whether this is in fact ftp, and while we're at it, let's see what else these boxes have been saying to each other. I want a list of destination ports that 49.21 has spoken to 33.21 on. Using ra to ask this question would yield every network connection these two hosts have made. So if these boxes had 50 ssh conversations, we would get 50 lines of output for ssh alone. Not what we want. We just want each protocol listed once, so if the boxes used ssh 50 times, we just want ssh mentioned once, kind of like piping the output to sort and then uniq. Argus has an elegant way of doing this in racluster:

```
racluster -r <yesterdays_file> -m proto dport -w - - src host 10.20.49.21 \
and dst host 10.20.33.21  | rasort -m dport -s dport trans bytes  | less
```

The Argus log file can be thought of as a pcap file, with a line for every network conversation. Just reading the file gets you exactly that. But if you want to, you can combine and consolidate these records using whatever metric you want. In this case, we want to combine all the connections that use the same port into a single record. To do this, we use racluster's -m switch. Passing -m proto dport, actually does two things: it first consolidates all of the records that use the same protocol (TCP, UDP, VRRP, etc.) and then further combines all of the records that use the same destination port. When we do this, all of the data we know about those individual records is preserved. For example, the byte counts for each individual ssh connection get added up to a total byte count for all the connections. Argus also keeps a connection counter (called trans) for us, so we know how many connections the record refers to. We sort the output by destination port number using rasort with -s dport. The output from the above command looks like this:

```
ftp        550         885138
sds          2          21922
5206         2          66124
5367         4      114308574
5509         4       45916440
sgi-es       2         132358
ininme       2          81566
openma       4      151217036
...
```

So these two boxes made 550 ftp control channel connections, which I think verifies our theory about the random high-number ports. Now that you (hopefully) understand racluster's -m switch, go back up and look at the other two reports, which both use -m saddr. Most racluster queries are doing basically the same thing: filtering out some subset of an archive of connections and then combining them based on the metric I'm interested in knowing about. The "chatty kathys" report filters hosts out of a certain subnet and combines them all on source address, so we can see how many combined bytes each source address sent. The "town criers" report does the exact same thing, just with a more specific filter (where the destination host is the network broadcast address).

My last question is, just how much data did these two boxes end up sending to each other? Let's ask Argus:

```
>racluster -r <yesterdays_file> -m daddr - src host 10.20.49.21 \
and dst host 10.20.33.21  -s bytes

363278259124
```

Yeesh. DBAs. There are a few things I want to note about this last command; First, I removed the -w -, because we wanted human-readable output directly from racluster (instead of piping it to rasort). If you forget to do this, you'll get binary gobbledy-gook output and probably hose your term. This is annoying at first, but it's a Pavlov [5] thing; you'll eventually learn to be aware of it. Second, racluster also supports formatted output (-s), as I mentioned above, so if you don't need sorted output, you can dispense with rasort. And last, if I were to back off the search filter and just specify "src host 10.20.49.21", I would get a list of every box 10.20.49.21 had spoken to, suitable for sorting by byte count (do you see why?).

Once you grok racluster's data aggregation concept, every use case immediately becomes kind of obvious. Want to know what Web sites your user base hits the most? Aggregate on destination address (dstaddr) and sort by bytecount (with a filter that excludes internal IPs). Want to know whose sending the virus du jour? Aggregate on destination port (dport). Argus makes getting data like this so easy it's fun. I can (and do) poke around at my Argus data for days (which, in my humble opinion, is a healthy and normal pastime for someone in our profession), but let's face it, eventually you're going to want graphs.

So before you run out and write a Perl script that ties racluster to rrdtool, you should know that the Argus guys already wrote one, and they included it in the client's tarball for you. It's called ragraph [6], works great, and is super easy to use.

Anyway, at this point, I do have a few more questions, but they're not for Argus. If anyone needs me, my clue-by-four and I will be over in the DBA area.

Take it easy.

REFERENCES

[1] PEBCAK: "Problem Exists Between Chair and Keyboard"; http://www.catb.org/jargon/html/P/PEBKAC.html.

[2] ratop: http://www.qosient.com/argus/.

[3] racluster: http://www.qosient.com/argus/.

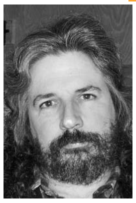
[4] ra: http://www.qosient.com/argus/.

[5] Pavlov: A man famous for being mean to dogs; http://en.wikipedia.org/wiki/Ivan_Pavlov.

[6] ragraph: http://www.qosient.com/argus/.

ROBERT G. FERRELL

/dev/random: airport security and other myths



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

rgferrell@gmail.com

I SPEND WAY MORE TIME THAN I WOULD consider optimum sitting in airports these days. Admittedly, optimum for me would be a big fat goose egg, because I hate flying since the demise of the "friendly skies," so I suppose that really isn't a content-rich statement. Get used to it.

As an aside, I am typing this on my MacBook Pro, which I very much like except for the wonky keyboard. What possessed Apple to think that no one would miss the "Home," "End," "Page Up," and Page Down" keys? Sure, you can approximate those functions using keypress combos that take six or seven fingers like some insanely difficult guitar chord, but at my age digitary gymnastics of that sort are problematic. I guess Cupertino was aiming at a younger target market, although you'd think they would have mapped everything to thumb buttons and triggers in that case. I am half-expecting the next generation of input devices to be equipped with accelerometers that require you to go into three-dimensional spasms to send a text message or check the sports scores. At least that will be entertaining to watch, albeit hazardous for unwary passers-by.

It used to be that airport wireless access points were restricted to a few well-defined locations with somewhat limited coverage. With the advent of portable hotspots like the Overdrive, however, the network topology of the average airport is evolving rapidly. I can fire up KisMAC on the aforementioned trusty MacBook (bearing in mind that OS X is mostly BSD and therefore perfectly acceptable for use by us UNIX geeks) and as I walk from gate to gate the number of ad hoc networks that pop up is impressive. I happen to be slouching about at BWI at the moment, returning from scenic Linthicum, and because I'm the kind of guy who doesn't like to be rushed, I still have two full hours before my flight back to the sun-soaked South is due to depart. That leaves me plenty of time to conduct WiFi reconnaissance and to reflect on the current status of network security, a term frequently and wholly inappropriately applied to the chaotic lawless frontier that is the Internets.

There are a number of parallels between network security and airport security that I can see. For one thing, the growing horde of security "experts" who spent thousands of dollars and dozens of hours in some cheesy workshop memorizing the answers to an exam and are thereby qualified to secure the most complex rete of interconnected heterogeneous systems ever conceived by sentient life (so far as we, the aforementioned sentients—and I employ the term generously—know) are eerily similar to the TSA. They provide a comforting illusion of competence and generate a warm fuzzy aura without providing any real security at all. It's pretty much just second-hand smoke and mirrors these days, whether you're boarding a 737 to Albuquerque or sending your credit card information over the Intertubes to a multibillion-taxpayer-dollar bailout recipient otherwise known as a bank. They won't let you bring a fingernail clipper on board an aircraft for fear you'll use it to overpower the flight attendants (by giving them killer manicures, presumably) and pry open the locked cockpit door with the little file, but apparently plastic explosives are OK so long as you tuck them in your underwear. If Semtex diapers aren't in your budget, just down a couple plates of the greasy bean and simulated cheese-like-substance–stuffed jalapeño appetizers at the restaurant across from your gate and you're good to go.

In the same vein, the little padlock that appears in your browser's status bar may be reassuring, but there's really not a lot of justification for that confidence. It means, ostensibly, that an encrypted SSL connection has been established with your remote host, but what it doesn't guarantee is that the remote host is who you think it is. Anyone with a valid certificate (which can be self-signed, by the way) can establish an SSL connection with you and, if they're clever, thoughtfully pass your information along to your intended destination after they get through with it, rendering the subterfuge very difficult to detect. This is called a "man-in-the-middle attack" by most of the industry, although I tend to refer to it as the "evil bucket brigade" because I'm not at all a well person. Call it what you will, it serves as yet another stark reminder of the illusory nature of security in an interconnected eSociety built on the confidentiality, integrity, and availability vacuum that is TCP/IP.

Returning to the airport security metaphor, let's talk about authentication. Having your boarding pass and driver's license available for TSA agents at the security checkpoint is sort of like authenticating, except that it's absurdly easy to circumvent. I love the way they shine the little light on your driver's license to give the impression they've been trained in how to spot clever forgeries for all fifty states, under the apparent assumption that no one but state governments has access to holographic printing. Even scarier, you can just go to work for one of the dozens of companies that provide support for the airport—many of whom conduct minimal background checks if any at all—and boom, you get to bypass even the rudimentary TSA butt-sniffing. While you're standing there getting probed, scanned, swabbed, and wanded, that guy in the coveralls driving the little catering truck with access to the delicate parts of the aircraft to which you are about to entrust your life may very well have a criminal record as long as the receipt for extra "fees" the airline has charged you in a desperate attempt to stave off the inevitable bankruptcy and or/merger looming in their future as their legacy of piss-poor business practices finally catches up with them. In infosec we refer to this as a "failure to apply discretionary access controls." In grammar we call it a "run-on sentence."

The airlines themselves could be considered system processes, and the gates I/O ports. That would make the terminals network segments, the trams connecting terminals bridging routers, your boarding pass the packet header, and you the actual payload. Despite what the airlines would like you to believe, air travel is definitely UDP. The encapsulation leaves much to be desired, as well. There's not even any error-checking to speak of, especially where baggage is concerned. If your packet fragments and fails to reach its destination, the payload is irretrievably lost: resend isn't an option. Latency is also a serious problem. Collision avoidance, fortunately, is pretty robust, at least in controlled airspace when the pilots aren't busy playing FarmVille on their laptops. I must confess that I live in constant fear that my TTL will expire between hops.

In the final analysis, however, it seems to me that the information security term that best applies to airports these days is "denial of service."

book reviews

ELIZABETH ZWICKY, WITH BRANDON CHING AND SAM STOVER

HACKERS: HEROES OF THE COMPUTER REVOLUTION

Steven Levy

O'Reilly, 2010. 487 pp.
ISBN 978-1-449-38839-3

I am profoundly conflicted about this book. It is a (mostly) sympathetic portrait of people who did interesting and often under-appreciated stuff in the early days of the PC and just before and after. It gives a clear picture of their value to the world and their values, which are not always mainstream, and it presents them as lovable if quirky. In the process, it often accepts their worldview uncritically. This is a community I am a part of, at least on the edges, and I like to see it appreciated. I do not think that uncritically accepting it is healthy. There is a fine line between celebrating difficult people and making it seem like you have to be anti-social to do any important work in computing, and I don't think this book always manages to stay on the good side of that line.

It is extraordinarily difficult to write books about real people, particularly big groups of real living people. Being true and honest and kind and inclusive without muddying up the story beyond all comprehensibility is an immense balancing act, and it's a balancing act this book manages really pretty well. The result does end up omitting a fair amount of important stuff that is present in hacker culture. This is a book about monogamous straight white guys, for instance, and while the culture itself is overwhelmingly white guys, it's not nearly so overwhelmingly straight or monogamous. That could be an accident of the people chosen, or it could be an artifact of trying to be polite, by some standards of politeness, particularly that current at the time the first edition was written. The book also ignores the fraught territory of deciding where the line is between merely being eccentric and actually having a condition that might be detrimental to your daily life to the extent where medical assistance could be very helpful. Again, this is difficult, private stuff, but, you know, much of one chapter is devoted to a multi-person quest for a single programmer to "get laid," so it's not as if the book confines itself to discussing public matters.

I didn't read it when it first came out, because it was a book that many people I knew loved and that I was pretty sure I'd hate for fundamentally feminist reasons. I didn't need to be fighting with my friends over it, but early in my career my feelings about being a woman in computing were pretty raw, for very good reason. And, indeed, even all these decades later, I gritted my teeth through most of the early chapters. It's not that women are never mentioned. No, that would actually be an improvement. Instead, the possession of a girlfriend or wife is used as a kind of indicator of level of socialization; if you have a woman, you are socially competent. If you don't, you aren't. (This is not, in my experience, accurate, in either direction.) And one of the interviewees presents the concept that women are so underrepresented as hackers that there must be a genetic cause, which is exactly as stupid as saying that Macintosh viruses are so rare that there must be a hardware cause. (In case you're wondering, Macintosh viruses are rare mostly for cultural/economic reasons—PC viruses pay better—and to some extent for software reasons.) Culture and genetics interact in complicated ways, but culture is vastly influential, as important to human behavior as software is to computer behavior.

But despite my teeth-gritting, and my very complex feelings about the book, I think it is a good book—as honest as it can probably get about what its chosen tiny part of computing history is about. As a slice of time, and as an introduction to the idea that really weird people can, in fact, end up reshaping the world, it's a gripping book.

If you want to know about computing history, do add in other books that cover more threads in the story; this one leaves UNIX off at Multics and picks it back up again roughly at Linux, which is a whole other tapestry taking place overlapping with the events described here. Peter Salus's *A Quarter Century of UNIX* is a very, very different book from *Hackers*, but it fills in some of that missing territory.

TEXTMATE: POWER EDITING FOR THE MAC

James Edward Gray II

Pragmatic Bookshelf, 2007. 182 pp.
ISBN 978-0-9787392-3-2

OK, enough with the sociological stuff. TextMate is a Macintosh editor in the spirit of Emacs, which is to say it is powerful, flexible, extensible, and programmer-friendly. (Also, it has a lot of Emacs key-bindings.) It's easy to overlook, but capable of many beautiful feats. Our Macs at work come with it installed, and although I'm no TextMate wizard, I use it, enjoy it, and have converted some of my colleagues. Some of them have been converted by cheap tricks (the impenetrable thicket of machine-generated JSON which we pasted into TextMate and had it auto-format as XML; auto-format is hardly innovative, but it saved our day) and some by actual elegance. One colleague was watching me write HTML and insert multiple links; the first one I didn't have the URL for, and TextMate did a nice job of setting things up for me, but nothing impressive. For the second one, I copied the URL from my browser first, and when I told TextMate to insert a link, it did the whole thing, fetching it from the paste buffer, looking up its title and inserting that as the link text, and preselecting it in case I wanted other linked text. At that point he said, "What is that, and where do I get it?"

TextMate has a number of tricks which are either automatic or easy to find, but fully getting your head around controlling, as the author phrases it, the team of magic ninjas it provides is a slow process which this book is very helpful with. If you find yourself using TextMate, you'll find the book helpful, and you will almost certainly want a copy if you're trying to extend your use of it.

THE JAZZ PROCESS: COLLABORATION, INNOVATION, AND AGILITY

Adrian Cho

Addison Wesley, 2010. 279 pp.
ISBN 978-0-321-63645-4

In general, the metaphors people use for thinking about business are based on sports or on war. These have several risks. For instance, sports metaphors translate poorly between cultures or to non-enthusiasts. Often a sports-unenthused audience may have only the vaguest idea what the speaker means. War metaphors often result in people being metaphorically exhorted to kill the customer or die in the attempt, which is not apt to result in a productive relationship.

I was therefore happy to find a book that used a metaphor which is inherently based in collaboration and pleasing the audience, rather than attacking it. (And amused to discover that the author likes to translate his insights into sports and war metaphors, just in case you needed something more familiar and macho. The temptation to stray back into social commentary is very high here.) Aside from appreciating the base metaphor, I also like the content a lot. It includes a number of insights that are common (hire good performers, enable them to lead as needed, watch the health of the team) and some that are not. The discussion of feedback loops and "hunting" is particularly rare. Even rarer, it discusses the good and bad points of its advice, instead of merely advocating One True Way.

This is a good, general theoretical overview of how to manage a team. It advocates a collaborative hacker-friendly management style that values excellence and transparency. It is, however, relatively abstract. It's a good conceptual basis, and usefully provides the metaphors and explanations you'll need to communicate this kind of management style to other managers. But it won't give you the nitty-gritty of how to implement it.

PERSUASIVE TECHNOLOGY: USING COMPUTERS TO CHANGE WHAT WE THINK AND DO

B.J. Fogg

Morgan Kaufmann, 2003. 265 pp.
ISBN 978-2-55860-643-2

When you think about it, most of us do, in fact, use computers as persuasive systems. System administrators try to convince users to do all sorts of things, from setting good passwords to submitting useful trouble tickets. Programmers want people to use their programs, register them, and file bug reports when they break, each of which requires persuasion. We use online dieting or exercise tools to try to persuade ourselves. And the book gives the example of an oscilloscope that increased its user satisfaction ratings significantly by rewording the error messages, which were brusque and annoying.

This book also covers a bunch of territory that may not be immediately obvious when you think about persuasion. For instance, it gives an interesting breakdown of roles that a computer takes in interactions, and ways that software behavior can work for and against those. (Social behavior from the computer is great! Except when you're trying to use it as a tool, at which point it should not be waving and offering helpful advice.) It also talks a lot about trust, which is necessary for persuasion but of course desirable for all sorts of other reasons.

There's a good bit of interesting information here, but at an academic book price and with a relatively slow start as it puts together an academic framework. If you're looking for specific instructions, this book isn't going to satisfy you (but then again, you don't have any other options; this is as techy as books about persuasion get).

NETWORK FLOW ANALYSIS

Michael W. Lucas

No Starch Press, 2010. 224 pp.
ISBN 978-1593272036

REVIEWED BY SAM STOVER

Having read and loved *Absolute OpenBSD* by this same author, I was really looking forward to *Network Flow Analysis*. Definitely not a disappointment, this book was exactly what I was hoping for. Combining a great writing style with lots of technical info, this book provides a learning experience that's both fun and interesting. Not too many technical books can claim that.

Chapter 1 describes flow fundamentals, and even if you know IP inside and out, it's probably not a bad idea to read it. One of the main tenets of "flows" is that each flow only describes unidirectional traffic, so for a single TCP session, you actually have two flows: from client to server and from server to client. Once you adjust to this way of thinking of flows, you'll understand how the flow analysis tools view the traffic, which is pretty important to have under your belt before heading off to the rest of the book. In addition, an overview of NetFlow history and versions provides some background of the how, why, and what netflow analysis can provide.

Chapter 2 talks about collectors and sensors, which are the two main components for gathering your NetFlow data. Hardware and software methods are discussed, and installation of the flow-tools software is shown. Once you have your collectors and sensors configured, you're ready for Chapter 3, which walks you through viewing flows. In any kind of enterprise environment you'll quickly see that, as with any other (firehose) data source, you'll want to be able to filter out the cruft to see what's really important. Chapter 4 describes filtering "primitives" and constructing useful methods for parsing your flow data. Chapter 5 deals with reporting and follow-up analysis. The default report is a great start, but this chapter shows how to modify it to suit your needs, as well as providing suggestions on some useful reports that you can use as a starting point for different types of network traffic: IP Address Reports, Traffic Size Reports, and BGP Reports, to name a few.

At this point, assuming you've been installing and configuring per the book examples, you have at a minimum a functional netflow analysis platform. Chapter 6 takes it a step further by introducing the Cflow.pm Perl module. Evidently, due to earlier flow analysis tool development, there is some confusion between Cflow.pm and cflowd (an obsolete flow analysis tool; this chapter will make sure that you get everything installed correctly. Once you do, you can start leveraging this Perl module to write your own analysis code. Not only that, but "the most user-friendly flow-reporting" tool, FlowScan, depends on it. This chapter will show you how to set up FlowScan and use it to start generating Web-based, graphical reports. Since FlowScan is somewhat non-customizable, you'll probably want to install the FlowView suite, and Chapter 7 will guide you all the way. You can think of FlowScan as being good for "normal" people and FlowView for network administrators. 'Nuff said.

Chapter 8 will gently help you set up gnuplot so that you can quickly create "impressive graphs of your network data" without wanting to kill yourself. What more could you ask for in a chapter?

The final chapter addresses NetFlow v9, which the current version of flow-tools does not accept.

Instead, the emerging successor to flow-tools, flowd, is introduced and explained. The author even says that he had considered using flowd as the core of the book, but it's still a little too new. As we've seen in the previous chapters, there is a lot of information you can get out of your network using flow-tools, but if you're interested in coding up your own reports and you need to handle NetFlow v9 traffic, then flowd is for you.

This is one of the best books I've read in a long time. It does precisely what it sets out to do: teach you about flow analysis and how to use the tools to set it up yourself. Lucas has a great sense of humor that isn't overwhelming and keeps the tone of the book moving along. If you are looking to learn about flow analysis or implement something in your network, you simply cannot go wrong with this book. If you don't care about network flow analysis, well, you should, so go get this book anyway. You won't be disappointed.

HIGH PERFORMANCE JAVASCRIPT

Nicholas C. Zakas

O'Reilly Media. 209 pp.
ISBN 9780596802790

REVIEWED BY BRANDON CHING

In my experience, most developers have only cursory JavaScript skills. Sure, most can write form validation and do some neat tricks with Jquery, but there seem to be very few developers who have actually mastered JavaScript. *High Performance JavaScript* is one of the books that can help you do so.

Don't let its relatively thin size fool you; Nicholas Zakas (and five other contributing authors) packed loads of insightful and valuable information into the pages of this very readable book. It covers not only the obvious performance topics of loading and execution but also DOM scripting, AJAX performance, algorithm and flow control, and even a chapter on regular-expression performance (which is really quite good). The chapter on DOM scripting really stood out. Extensive in its coverage and explanation, it is loaded with valuable nuggets of information.

While these types of topics can be dense even for experienced developers, the author's writing style is clear, consistent, and to the point. I often found myself at the end of a paragraph asking a question, only to have it clearly answered in the next paragraph. There are hundreds of code samples, performance metrics, and charts that demonstrate the more general point the author is trying to make at a clear, practical level, with concise explanations of why he is recommending something. This makes the book much more approachable to a wider audience of developers.

High Performance JavaScript is definitely intended for experienced Web developers. However, given the approachability of the writing and the continuity of the information, I think that even junior and mid-level developers would find immense value in having this book within easy reach. While it may be a bit of overkill, I can also feel confident in recommending this book to Web UI designers who might be responsible for the JavaScript-driven UI aspects of their sites. After just the first chapter, I was amazed at all the things I had learned, and I think many other developers would feel the same after reading this book. Very highly recommended.

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/

ACCESS TO VIDEOS from USENIX events in the first six months after the event: www.usenix.org/publications/multimedia/

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; 510-528-8649.

USENIX BOARD OF DIRECTORS

Communicate directly with the USENIX Board of Directors by writing to board@usenix.org.

PRESIDENT

Clem Cole
Intel
clem@usenix.org

VICE PRESIDENT

Margo Seltzer
Harvard University
margo@usenix.org

SECRETARY

Alva Couch
Tufts University
alva@usenix.org

TREASURER

Brian Noble
University of Michigan
brian@usenix.org

DIRECTORS

John Arrasjid
VMware
johna@usenix.org

David Blank-Edelman
Northeastern University
dnb@usenix.org

Matt Blaze
University of Pennsylvania
matt@usenix.org

Niels Provos
Google
niels@usenix.org

EXECUTIVE DIRECTOR

Ellie Young,
ellie@usenix.org

USA WINS WORLD HIGH SCHOOL PROGRAMMING CHAMPIONSHIPS—SNAPS EXTENDED CHINA WINNING STREAK

Rob Kolstad, USACO Head Coach

This year's four-member USACO elite high school programming team, sponsored by USENIX, nabbed the top team spot at the world programming championships (the International Olympiad on Informatics). For the first time, the United States has earned the undisputed world championship. The results were announced August 20, 2010, at the University of Waterloo in Waterloo, Ontario, Canada.

The USA programmers earned 3 of the top 13 spots among the 300 competitors from 83 countries vying for medals. USA Gold Medal winners included:

#4 **Wenyu Cao**, who will be a senior at Phillips Academy in Andover, Massachusetts, in the fall

#5 **Michael Cohen**, from Montgomery Blair, a magnet high school in Silver Spring, Maryland (now attending MIT)

#13 **Neal Wu**, from Baton Rouge Magnet High School in Baton Rouge, Louisiana (now attending Harvard)

Brian Hamrick, from Thomas Jefferson High School for Science and Technology in Alexandria, Virginia, earned a silver medal for 44th place; he has also matriculated at MIT.

China, Japan, and Russia tied for second place, with two gold and two silver medals. Bulgaria and the Czech Republic tied for fifth. Germany was seventh.

Head coach Dr. Rob Kolstad praised the team: "Our team trained for hundreds of hours over the past couple of years; some members competed in three dozen contests last year. They've done a terrific job transferring that training into competitive success." All four team members had international competitive experience at previous IOI competitions or the Central European Olympiad on Informatics (USA is the "western branch" of central Europe). They were selected from 15 USA candidates at this year's USA Invitational

Computing Olympiad (and selection camp), held at Clemson University in South Carolina.

USACO Educational Director Dr. Brian Dean noted, “This victory demonstrates that USA pre-college students can excel and win in tough international competitions where the Chinese have dominated for almost a decade.”

USACO’s mission to promote pre-college computing and recognize outstanding competitors is supported by: USENIX, the Advanced Computing Systems Association; Booz Allen Hamilton, the Strategy and Technology Consulting Firm; IBM, International Business Machines; and ITA Software, Innovative Travel Technology. USACO pursues this mission by providing training resources to over 200,000 registrants from over 80 countries, participating in six contests per year, with three divisions in each, that typically garner a total of 1,000 competitors at three different levels, as well as an annual training camp.

IOI 2010 sported new sorts of innovative tasks that rewarded exceptional levels of problem-solving creativity—just the thing USA students excel at. One problem required the identification of the written language used by various Web pages. Another required students to identify, not the shortest path through a corn maze, but the longest possible path, a unique twist on a standard problem.

USENIX’s long-term support is much appreciated by the competitors and coaches.

ROB AND BRIAN FLANKING THE TEAM: (FROM LEFT) DR. ROB KOLSTAD, USACO HEAD COACH; MEDAL WINNERS MICHAEL COHEN, WENYU CAO, BRIAN HAMRICK, AND NEAL WU; DR. BRIAN DEAN, USACO EDUCATIONAL DIRECTOR

conference reports

THANKS TO OUR SUMMARIZERS

2010 USENIX FEDERATED CONFERENCES WEEK

2010 USENIX Annual Technical Conference 62

Italo Dacosta, Paul Marinescu, Aleatha Parker-Wood, Dan Schatzberg, Xiao Zhang, Rik Farrow, John McCullough, Joshua Reich, Marc Staveley

USENIX Conference on Web Application Development (WebApps '10) 77

Pradeep Chalise, Rik Farrow, Pradeep Teregowda, Aiman Erbad, Thomas Moyer

3rd Workshop on Online Social Networks (WOSN 2010) 84

Saptarshi Ghosh, Christo Wilson

2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10). 91

Alva L. Couch, Joshua Reich, Rik Farrow, Malte Schwarzkopf

2nd Workshop on Hot Topics in Storage and File Systems (HotStorage '10) 100

Rik Farrow, Aleatha Parker-Wood

Configuration Management Summit . .104

Aleksey Tsalolikhin

2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '10) 106

Romain Cledat, Chris Gregg, Rik Farrow, James C. Jenista

2010 USENIX FEDERATED CONFERENCES WEEK

June 22–25, 2010
Boston, MA

This year, USENIX combined established conferences and new workshops into a week full of research, trends, and community interaction, with a completely customizable program. For more information about the events and the format, see http://www.usenix.org/events/confweek10/.

2010 USENIX Annual Technical Conference

June 23–25, 2010
Boston, MA

WELCOME, AWARDS, AND KEYNOTE ADDRESS: JOINT SESSION OF 2010 USENIX ANNUAL TECHNICAL CONFERENCE AND USENIX CONFERENCE ON WEB APPLICATION DEVELOPMENT

Summarized by Rik Farrow (rik@usenix.org)

Timothy Roscoe, program co-chair with Paul Barham of Annual Tech, said that 147 papers were submitted, slightly fewer than the previous year, due to competition from other conferences; after a thorough review process, 24 papers were accepted. Roscoe presented awards and checks for the two Best Papers: "LiteGreen: Saving Energy in Networked Desktops Using Virtualization," with Pradeep Padala of DOCOMO USA Labs accepting the award, and "ZooKeeper: Wait-free Coordination for Internet-scale Systems," with Benjamin Reed of Yahoo! Research accepting.

John Ousterhout, chair of WebApps, took over the podium. Ousterhout said that the size of the conference was a good beginning, with 80 attendees, 26 papers submitted, and 14 accepted. Ousterhout said that there have been three phases of the Web: the first, distributing documents; the second, as a platform for delivering apps; and phase three, the current one, which will see a complete turnover in the application development food chain. Ousterhout announced the Best Paper award, "Separating Web Applications from User Data Storage with BSTORE," by Ramesh Chandra, Priya Gupta, and Nickolai Zeldovich.

Clem Cole, President of the USENIX Board, took the stage to hand out two more awards. The USENIX Lifetime Achievement Award, a.k.a. "The Flame," went to Ward Cunningham, the inventor of the Wiki. The STUG award, which recognizes significant contributions to the community that reflect the spirit and character demonstrated by those who came together in the Software Tools User Group, went to the group who created MediaWiki, whose work includes a tool many of us use every day—Wikipedia. The award money was donated to the Wikimedia Foundation.

KEYNOTE ADDRESS

- ***Lessons of Scale at Facebook***
  Keynote by Bobby Johnson, Director of Engineering, Facebook, Inc.

  Summarized by Xiao Zhang (xiao@cs.rochester.edu)

Bobby Johnson explained how they address the technical challenges as the number of Facebook users grows explosively. In particular, he elaborated on three key perspectives: moving fast, server scaling, and client performance.

To be able to move fast, Facebook has a culture of making frequent small changes. Johnson commented that it was really easy to figure out what went wrong in production if you only changed one thing at a time and watched it closely over time.

The server infrastructure of Facebook is divided into Web Server, Memcache, and Database. Most of the scalability work falls into the Memcache layer, because it has to serve hundreds og millions of objects in a second. Johnson gave an example on how to dynamically scale the number of Memcache machines communicating with the switch to avoid packet dropping due to overload. He also pointed out most machine failures were due to software, and many failed machines run the same piece of buggy code. He told an anecdote of twenty machines leaking memory at the same rate.

Johnson introduced two projects to improve client performance. The first is called Big Pipe, which splits objects in a page and runs them in pipelines. By doing so, it allows priority content to be shown quickly and also benefits from parallelism. The second is a small JavaScript library core called PRIMER, which does bare-minimum things to make a page feel interactive during loading. Big Pipe and PRIMER share the property of dividing things into a fast path and a slow path.

In closing, Johnson talked about engineering culture at Facebook. One particular principle is that control and responsibility have to go together.

Marvin Theimer asked if all data had to stay in memory. Johnson replied that the social graph data is entirely indexed in memory, while pictures and videos are stored in disk. He also mentioned increasing interest in flash storage. Bill LeFebvre inquired about the problem of constantly increasing storage demand. Johnson said that Facebook does not plan to delete data and that the current solution is to buy lots of cheap hard drives. John Ousterhout asked whether PHP is the right language for Facebook. Johnson agreed that PHP is not a great language for running Web applications, although it is a fantastic language for writing them. And that's partially why Facebook has a compiler project to transfer PHP to C++ code. Johnson also emphasized that an interpreter language is critical for Facebook building things quickly. Ben Johnson asked about data consistency, and Johnson replied that Facebook cares about consistency and puts a lot of work there.

JUNE 23, 10:30 A.M.–NOON

Summarized by Joshua Reich (reich@cs.columbia.edu)

- ***DEFCon: High-Performance Event Processing with Information Security***
  Matteo Migliavacca and Ioannis Papagiannis, Imperial College London; David M. Eyers, University of Cambridge; Brian Shand, CBCU, Eastern Cancer Registry, National Health Service UK; Jean Bacon, Computer Laboratory, University of Cambridge; Peter Pietzuch, Imperial College London

Matteo Migliavacca presented the problem: event-stream processing needs strong security—this is of particular application in financial contexts. If flows are incorrect, this can lead to security violations (e.g., companies may see each other's trade data). Consequently, the authors propose tracking and controlling data flows. Their primary contribution is a decentralized event flow control, DEFCon, implemented in Java, where all data is tagged. For data tagged with an access security tag, one either needs to have access granted or the data needs to be declassified in order to be read.

Preventing nodes from peeking at data is actually rather tricky in practice, as there are many opportunities for information leakage (e.g., returning "access denied" provides information, and failure to respond may also do so). The DEFCon approach assumes that all units communicate through labeled events. This could be done using VM or OS-level mechanisms, but they would prove too heavy for low-latency environments. Instead, the authors use threads that share data in a single address space. They wrote an implementation using Java threads, but need to have them share immutable data objects, and thus some engineering design is called for. The authors show that with the right set of techniques the overhead can be made reasonably small.

Why use only one VM? For performance. Why not use features Java already has to divide flows? Currently existing features don't focus on label checking performance. Additionally, these approaches are generic, and they want to be as efficient as possible for their domain.

- ***Wide-Area Route Control for Distributed Services***
  Vytautas Valancius and Nick Feamster, Georgia Institute of Technology; Jennifer Rexford, Princeton University; Akihiro Nakao, The University of Tokyo

Currently, all traffic from a given data center uses only one path to the user, Vytautas (Valas) Valancius began. Yet different cloud apps have different requirements. Interactive applications need low latency and low jitter, while bulk-data applications need high throughput at low cost. Amazon EC2 has 58+ routing peers but picks only one route per user!

Today, if one does want to route flexibly one needs to obtain dedicated connectivity and numbered Internet resources, which are both difficult and expensive to set up. The authors proposed essentially building a BGP-level NAT Transit Portal. Each service has a virtual router through which all traffic flows. This virtual router essentially uncovers the

Transit Portal's info, allowing the virtual router to decide which path it would like to use for a given traffic flow (at least first hop).

In this setup each service has its own router (virtual or physical). Each router has a link to the Transit Portal, which emulates a connection to an upstream ISP (e.g., three links to a Transit Portal for three peered ISPs). This exposes a standard BGP router control interface. The authors have found it takes about 30 seconds to converge when a service router changes a path. Their system is currently deployed in academic settings, built on top of a regular router running custom software at three active sites.

Active experiments include BGP poisoning, IP anycast, and advanced networking class—students can use BGP.

The authors are also exploring advanced Transit Portal applications such as fast DNS and service migration (currently only available to large operators that have their own global network backbones).

The final challenge addressed by Valas was scaling. Here the Transit Portal needs to scale to dozens of sessions to ISPs and hundreds of hosted sessions, but standard BGP only chooses one peer to send to. Consequently, the authors have implemented separate routing tables for each peered ISP. They use virtual routing tables to shrink from 90 to 60 MB per ISP and schedule/send routing updates in bundles to reduce CPU usage.

Future work includes more deployment sites, making it accessible to testbeds (e.g., GENI), faster forwarding NetFPGA, OpenFlow, and a user-friendly interface for route control (running BGP is heavyweight right now).

Someone wondered whether this could have applications beyond the cloud. Valas responded that it is indeed more general. Have they considered abuse, security risks? Good question. These things have been seen in the wild (e.g., the YouTube Pakistan problem). They currently advocate that administrators regulate which paths users should be allowed to announce. How do they manage to negotiate between users and the ISP? By making the market more competitive and letting economic incentives prevail.

- ***LiteGreen: Saving Energy in Networked Desktops Using Virtualization***
  Tathagata Das, Microsoft Research India; Pradeep Padala, DOCOMO USA Labs; Venkat Padmanabhan and Ram Ramjee, Microsoft Research India; Kang G. Shin, The University of Michigan

  Won Best Paper Award!

Pradeep Padala began by saying that PCs waste much energy while idling but users do not like disruption. Also, manual methods for waking machines for remote access are cumbersome and, thus, automated energy saving methods are needed. Padala noted that much energy waste occurs during idle periods of less than three hours and this is the energy they focus on saving (their approach does save power for longer idle periods as well). The LiteGreen architecture calls for users to always run their OS inside a VM. This VM runs inside a hypervisor/VMM either locally (when the user is physically present or significant computation needs be done) or remotely (when the machine would have been idling). LiteGreen maintains instant availability and masks migration effects by using a combination of indirection (even when the VM is local users, log in through Remote Desktop) and live migration of VMs between the local machine and the remote LiteGreen server.

This setup requires that the user's PC, the LiteGreen server, and network storage server (data is no longer stored on local hard-drives) all be attached to a gigabit switch (the network storage could run on a separate backbone, of course). There is a several-second delay while live migration occurs and the Remote Desktop session transfers from remote to local VM instances (or vice versa).

The authors explore how idle should be defined/when VMs should be migrated, coming up with heuristics involving user activity and resource usage (both on the local machine and on the LiteGreen server). Finding good heuristics for this problem is still very much an open question. With their current methods, the authors found that on some machines very little energy could be saved, but for machines that slept soundly overnight, savings were quite significant. This does prove a bit problematic vis-à-vis the authors' goal of saving power on <3-hour idle periods.

Their prototype was built on top of Hyper-V and Xen. They found they could shrink VMs 8x by using just the working set. Moreover, they may get even larger consolidation ratios if the overlap between VM working sets is significant. For now they claim that 80 or so VMs could be supported by a single LiteGreen server.

How did they support the large amount of storage needed for all of their VMs? They only need to store the main OS image on the server and can use snapshots to reduce VM images even further. But what about user data? They use shared storage—e.g., NAS, SAN. It seems as if they've taken VDI and made it a harder problem—why not go for thin client, since they are running RDP anyway? This is different from thin clients. You need to have lots of servers for peak usage, but here they only keep idle VMs. What about scalability? They can support 100 users per machine. But don't idle Windows VMs use a lot of resource consumption compared to the VMs they've implemented on? With work they can get similar numbers. Do power savings also include server consumption? Yes. The server takes 250W, more or less static. Current servers aren't energy proportional.

JUNE 23, 1:00 P.M.–2:00 P.M.: INVITED TALK

■ ***Visualizing Data***
Ben Fry, Author and Consultant

Summarized by Marc Staveley (marc@staveley.com)

Ben Fry talked about his work in providing ways to understand through various methods of visualization the mountains of data that are being produced today. In his words, "Given a pile of data, how can we very quickly visualize it and mine through it to ask interesting questions?"

Fry is a cross-discipline practitioner combining graphic arts and computer science. He is the author (along with Casey Reas) of Processing, which is an open source programming language and environment for images, animation, and interactions. With Processing it is possible to quickly and easily generate an interactive image of "information that dances before your eyes."

Fry showed a number of examples to illustrate the power of Processing. One was a graphic of Fortune 500 companies over time that allowed the user to see the rise and fall of different companies and market segments by just mousing over their names. Another was a DNA browser that allowed the user to "look at the forest and the trees at the same time" by providing the user a way to expand segments on a DNA strand while still seeing the full chromosome for context. Fry also showed work where data was used to just provide a pretty picture that could, for example, be used as a magazine cover or illustration. One example was DNA strands spelled out on many planes.

A vibrant community has built up around Processing, with a user base that has grown to over 25,000 active members. Fry, of course, had a graphic that showed the activity of the user base over time. Processing, which is written in Java (so it works on Windows, Mac, and Linux), is an interpreted interactive visualization language that hides the complexity of graphic generation, while still providing a powerful set of primitives.

The community has contributed a large number of different libraries to the project to extend the power of Processing. There is even a port to JavaScript (processing.js) which allows Processing datasets to be visualized entirely in a Web browser.

To learn more about Processing, you can pick one of the available books, including Fry's *Getting Started with Processing*, which just came out for the nontechnical market. Or go to processing.org to read the wiki and download the environment.

An audience member noted that Processing and Apple Quartz Composer are similar. Fry replied that Quartz is all GUI programming–based (i.e., drag and drop boxes), while Processing is text programming–based, which he believes is more powerful for doing things the original designer didn't think of.

JUNE 23, 2:00 P.M.–3:00 P.M.

Summarized by Marc Staveley (marc@staveley.com)

■ ***Stout: An Adaptive Interface to Scalable Cloud Storage***
John C. McCullough, University of California, San Diego; John Dunagan and Alec Wolman, Microsoft Research, Redmond; Alex C. Snoeren, University of California, San Diego

John McCullough observed that there is a need to improve the performance of application server access to the storage tier in multi-tier Web architectures, especially when those applications are hosted in cloud environments where access to the storage tier may have competition from other users of the cloud.

When the storage tier is under high load, it is possible to achieve this improvement by batching storage requests from middle tier applications, thereby amortizing overhead costs over a number of storage requests. But there is a throughput vs. latency tension; when load is low on the storage tier, you want latency to dominate by batching only a small number of requests (or not batching at all), but when load is high, batching aggressively will increase aggregate throughput.

Stout is a storage interposition library that uses an adaptive algorithm to choose the batch size based on the current load on the storage tier. The algorithm runs independently in each middle tier application but adapts to give each application a fair share of the storage bandwidth. It does this by using the latency history of recent storage requests to adjust the batch size (similar to recent work in TCP congestion control).

Would using Stout on some of the middle-tier servers but not others still achieve fair sharing? The clients not using Stout would not achieve fair share, while those that do would still be able to improve their overall performance.

■ ***IsoStack—Highly Efficient Network Processing on Dedicated Cores***
Leah Shalev, Julian Satran, Eran Borovik, and Muli Ben-Yehuda, IBM Research—Haifa

Leah Shalev observed that TCP/IP is a major consumer of CPU cycles but wastes lots of those cycles on multi-processor machines with cross-calls and cache line misses (stalls). She claims that TCP/IP uses tens of thousands of CPU cycles for just hundreds of "useful" instructions per packet.

The problem with running the TCP/IP stack on a multiprocessor (including multicore) system is that using a single lock produces high contention, while using finer-grained locking has higher overhead and causes many cross-calls and cache line misses. She noted that using CPU affinity to keep the application on the same CPU as the TCP/IP stack for that application doesn't work in practice with multithreaded applications using multiple cores.

IsoStack runs as a single kernel thread isolating the network stack to a single dedicated CPU with a lightweight interconnect API between the rest of the kernel and the network stack on the single CPU. They produced the interconnect by splitting the socket layer with the front end in the application and the back end in IsoStack. They have achieved near line speed (10GiB/s) with a 25% CPU utilization on an IBM Power6 with eight cores.

An audience member asked whether this is a scalable solution as networks get faster but single cores do not. Is there a future bottleneck looming for IsoStack? Shalev replied that Receive-Side Scaling (explained in the paper) can be used to scale IsoStack to use multiple cores without the overhead of introducing any locks.

JUNE 23, 3:30 P.M.–5:30 P.M.

Summarized by Aleatha Parker-Wood (aleatha@soe.ucsc.edu)

- ***A Realistic Evaluation of Memory Hardware Errors and Software System Susceptibility***
  Xin Li, Michael C. Huang, and Kai Shen, University of Rochester; Lingkun Chu, Ask.com

Xin Li presented a survey of memory hardware errors, focusing on non-transient errors. The data was collected from 212 servers at Ask.com, with over 800GB of memory, monitored for nine months. In addition, they looked at data from PlanetLab machines, and 20 desktops from the University of Rochester. These results have been previously reported in USENIX '07.

One purpose of this work is to evaluate the efficacy of countermeasures such as ECC and Chipkill. These countermeasures are often expensive to add to a chip, and so the authors wanted to examine how often these problems occurred, as well as whether countermeasures were effective when applied. Since memory errors are rare, the authors used Monte Carlo simulation in order to step up error rates and evaluate the impact on software, with and without each of the countermeasures applied.

The authors were particularly interested in the effect on software running on faulty memory, since not all errors are accessed. In order to evaluate the effect, they injected faults into a virtual machine. To track memory accesses, they used a novel memory-tracking approach which relies on page access controls for coarse-grained tracking and then uses hardware watch points for faults within the page. They concluded that without error correction, 50% of non-transient errors cause errors in software, in the form of wrong output, software crash, or a kernel crash. When ECC is applied, the frequency is reduced, but some errors still creep through and are just as severe.

Mohit Saxena from the University of Wisconsin—Madison asked how the approach compared to 2-bit ECC and cache errors. Li said he was unfamiliar with the approach, but believed it was a weaker model than the Chipkill ECC. If it was widely available, he would look into its effect.

- ***The Utility Coprocessor: Massively Parallel Computation from the Coffee Shop***
  John R. Douceur, Jeremy Elson, Jon Howell, and Jacob R. Lorch, Microsoft Research

Jeremy Elson presented a utility-computing framework specifically designed for desktop applications working in high latency, low bandwidth applications for limited periods of time. A framework like this would allow highly parallelizable applications, such as software development, video editing, 3-D modeling, and strategy games to take full advantage of the computational power of the cloud. However, users and application designers are unlikely to want to install a new operating system or write highly specialized code to take advantage of this computing power. And users have highly heterogeneous systems, with different software and libraries, which the system should take advantage of.

To achieve a system with a low barrier to entry, they rejected manual replication of code, and software as a service. Instead, they suggest a remote file system, which requests files as needed from the client file system. To keep this from being prohibitively slow, they use a variety of techniques. First, they carefully relax consistency semantics, using task-open to task-close rather than file-open to file-close. This reduces the amount of data transferred. Second, they use a content-addressable storage (CAS) model to ensure not only that data can be re-used between runs of the software, but that users using the same libraries or software can leverage data from one another. On the first run, the parameters are sent to a distributor, and from there to worker processes. Workers request the files they need, such as libraries and binaries. Writers write to a temporary area and, on completion, the results are returned to the client. Subsequently, remote file hashes are checked against the local files to ensure that files are up to date, and differential compression is used to send changes.

Since all of the libraries and software are pulled from the client, there are no major OS compatibility issues or any need to manually update libraries on the cluster. One cluster can be shared across a variety of users and applications. The authors note that the only downside is a lack of shared memory. All IPC must be done through the file system.

Someone asked about what was required to persist between invocations, whether a file system was needed or whether computer time would need to be rented. Elson replied that all that was needed was a file system. Further, since the system used content-addressable storage, the file might already be cached from a different user. What about licensing issues, since the net result might be thousands of copies of Photoshop running on the cluster? A good point, but not one that Elson felt qualified to address. What about privacy issues? Cache sharing was not a vulnerability, since if you can name a file, you must already have a copy of it. Another audience member noted that the current model for cloud computing is to pre-allocate virtual machines, which are then billed by start-up cost. Did Elson think the charge model would change? Elson said that for the time being, it was best to assume that one client would be the only one using it. This would still be economical for many tasks, especially extended ones. However, he predicted that the cost model might change if there were enough people to amortize the cost of these clusters. Finally, an audience member noted that the target applications might benefit from using GPUs, or computing resources on a remote desktop. Elson replied that making things faster locally was always superior, but that it wasn't always practical to take a spare GPU to a coffee shop.

- ***Apiary: Easy-to-Use Desktop Application Fault Containment on Commodity Operating Systems***
  Shaya Potter and Jason Nieh, Columbia University

Shaya Potter presented Apiary, a framework for fault containment. Desktop applications are a common vector for exploitation. However, many of these applications have no reason to persist data or to interact with one another. One possibility would be to use an isolated VM for each application or instance of an application, in order to keep the impact of exploited applications to a minimum. But this represents a significant amount of overhead, both for the system and for the user. Instead, the authors propose a slightly smaller virtual system, known as a container. Containers can contain one or more applications and can either persist data between invocations, in an isolated file system, or be ephemeral. They retain the look and feel of the desktop. They are low overhead and quick to instantiate. They offer a lower degree of isolation than a full hardware VM, but are sufficient for most applications. If applications need to invoke external applications, such as a browser invoking a PDF viewer, an ephemeral container can be invoked for the duration of that session.

The system uses unioning file system concepts to manage packages. They introduced a new file system, known as the Virtual Layered File System (VLFS). VLFS turns packages into read-only shared layers. This allows different applications to depend on different versions of packages. Since layers are shared, a file system image for an ephemeral application can be created instantly by dynamically composing layers. Any file system changes are updated to the private layer, which isolates changes and makes malicious file system changes visible. The authors presented their system in a variety of case studies, and concluded that it introduces approximately a 10% overhead for 25 parallel instances running a suite of applications.

Catherine Zhang from IBM asked what would need to be changed to migrate to this system. Potter replied that you'd need to replace all of the packages with layers. The authors have a tool which converts packages into layers, but it's not very robust yet. John McCullough from UC San Diego asked how important it was to have the different layers for applications, and whether that was just to support conflicting versions. Potter replied that it also supports granularity. For instance, if a security hole is found in a library such as libc, it is better to be able to simply upgrade a single layer. Someone asked what happens when you don't want ephemeral behaviors, such as when a document is downloaded from the Web. Potter replied that files that are changed in an ephemeral process are persisted to the file system, but the container itself is deleted after use.

- ***Tolerating Malicious Device Drivers in Linux***
  Silas Boyd-Wickizer and Nickolai Zeldovich, MIT CSAIL

Silas Boyd-Wickizer presented SUD, a confinement system for Linux device drivers. SUD (not an acronym) is designed to convert existing kernel-space device drivers into drivers that can be run in user space. One of the major obstacles to this goal is the lack of modularity in the current driver interfaces. The kernel runtime cannot currently be used for drivers in a different protection domain.

To achieve their goal of user-space drivers with a minimum of rewriting, they emulate the kernel environment in user space using SUD User-Mode Linux (UML), which can be used to shadow necessary variables. In addition, they add proxy drivers to the Linux kernel, which allows reuse of the existing driver APIs. Proxy drivers and SUD-UML converts the existing Linux driver APIs into RPCs. The proxy driver is responsible for synchronizing shadowed variables before and after RPCs. Non-preemptable functions are implemented in the proxy driver to prevent the user-space driver from being preempted. SUD adds a hardware access module to the kernel to prevent drivers from doing real physical accesses which could be used to attack the system directly via hardware. By using I/O virtualization, the driver can be given direct device access while preventing attacks. This is implemented using the IOMMU capability of modern systems.

Wenji Wu from FermiLab asked how many times SUD copied from user space to the kernel, for instance, in the given example of packet transmission. Boyd-Wickizer replied that shared buffers in the user-kernel shared memory remove any actual copy operations in that example. How does the driver write to the actual registers for the hardware from user space? The memory is mapped using mmap. How is control passed from the proxy driver to the user-space driver, and does that need to be privileged? Silas replied that it did not need to be privileged. Xin Li from the University of Rochester asked how often device drivers were actually malicious versus simply a source of bugs. In general, drivers were not written to be malicious, but due to exploitation could become malicious over time. Li followed up, saying that this implied that the interface between the user level and the kernel level is fragile and that pushing the device driver outside the kernel wouldn't improve the situation. Boyd-Wickizer replied that this sort of isolation made it easier to restart the driver and keep it from crashing the kernel. An audience member noted that because the user-space drivers had a flag set to keep them from being swapped out, this would result in partitions in physical memory, which might make it hard to allocate large contiguous buffers. Boyd-Wickizer replied that since most of the DMA buffers were a few megabytes or smaller, this wasn't a major concern.

JUNE 24, 9:00 A.M.–10:00 A.M.: KEYNOTE ADDRESS

- ***Some Thoughts About Concurrency***
  Ivan Sutherland, Visiting Scientist at Portland State University

  Summarized by Dan Schatzberg (schatzberg.dan@gmail.com)

Ivan Sutherland opened the second day of the conference by discussing his design for an asynchronous computer. He has created the Asynchronous Research Center at Portland State University to work on this design, which he believes is achievable if we change two paradigms.

The first paradigm is that of sequential computing. When Maurice Wilkes ran the first program on EDSAC on May 6, 1949, the nature of the computing was a sequential order of operations. The cost of logic operations was much greater than the costs of communication between the operators. So it made sense to focus on the sequence of logic operations. But now the majority of the cost in the system is in communication. We currently don't have a vocabulary to configure communication. The details are hidden from the software. Sutherland then described his design, called Fleet. Fleet is a system designed to have configurable communication between the functional units. Programming is done by describing where data is sent to or from a functional unit. Sutherland claims that because the default is concurrent execution, programming the machine for concurrency is simpler.

The other paradigm is the use of a clock. It is not a necessary for a machine to have a clock. At one time it was useful for dealing with electrical noise, but now it creates power supply spikes. Because Fleet is designed so that functional units run when they have input to do so, there is no need for a clock tick for each execution. Everything runs concurrently (not just across "cores" but across all functional units).

Sutherland concluded his talk by saying that the system was still in its infancy. There is still much to do to make such a system really useful.

JUNE 24, 10:30 A.M.–NOON

Summarized by Dan Schatzberg (schatzberg.dan@gmail.com)

- ***Proxychain: Developing a Robust and Efficient Authentication Infrastructure for Carrier-Scale VoIP Networks***
  Italo Dacosta and Patrick Traynor, Converging Infrastructure Security (CISEC) Laboratory, Georgia Tech Information Security Center (GTISC), Georgia Institute of Technology

Italo Dacosta presented work done with Patrick Traynor on efficient large-scale authentication. He began by talking about the trade-offs among performance, scalability, and security. Some robust but computationally expensive security mechanisms are difficult to deploy in production environments, while others are more efficient but weaker and can be broken or abused. Session Initiation Protocol (SIP) is used for establishing, managing, and terminating sessions between at least two clients. It is generally associated with VoIP. Typically, only Digest authentication is used, because it is more efficient even though it is weak.

SIP Digest Authentication is a challenge-response protocol that uses cryptographic hash operations. The authentication works as follows: A user sends an invite request to a nearby proxy server. The proxy server asks the user for a hash of some secret stored on the database. The user responds and the proxy queries the database for the hash value to confirm that it matches, then sends the invite. The issue is that each time a user sends an invite, the database server must process a request and send it to the proxy. In testing, with no authentication their scenario could handle 24,000 calls per second. With authentication, they were brought down to just 1,160 calls per second.

The proposed solution is to cache temporary credentials created from hash chains to reduce the number of requests to the database. A hash chain is a sequence of one-time authentication tokens created by applying a hash function to a secret value multiple times. The server can cache the *n*th value in the chain. Then when the user sends an invite, the server can authenticate it by asking for the (*n*-1)th value in the chain, hashing it, and confirming it matches the original value. Then, on the next invite from the user, the server can ask for the (*n*-2)th value and so on. This only requires one database request initially and then none afterwards. The modifications required to implement this were relatively small and the cached credentials are only 134 bytes each. With Proxychain they were able to achieve 19,700 calls per second. Italo Dacosta can be reached at idacosta@gatech.edu.

- ***ZooKeeper: Wait-free Coordination for Internet-scale Systems***
  Patrick Hunt and Mahadev Konar, Yahoo! Grid; Flavio P. Junqueira and Benjamin Reed, Yahoo! Research

  Won Best Paper Award!

Benjamin Reed presented his work on a system for Yahoo! applications. The challenges involve lots of servers, users, and data. Requiring fault tolerance in such a system makes designing applications difficult. Reed discussed various distributed system architectures, some involving a master-slave relationship and others being fully distributed with a coordination service. Their system had a few requirements, including wait-free (slow processes cannot slow down fast ones), linearizable writes, serializable reads, client FIFO ordering, and client notification of a change before the change takes effect.

They designed a system with a very simple API that included only about 10 primitive instructions. A hierarchical namespace is designed where each node has data and children. Workers can get configuration when brought up and set a flag to be notified of a change. Administrators can change the configuration and then the workers receive the updated settings. Benjamin Reed showed how the API can be used to do leader election as well as locking.

The ZooKeeper Service is designed to have many servers with a copy of the state in memory. A leader is elected, the followers serve the client, and updates go through the leader. 2*f*+1 machines tolerate *f* failures. The service is open sourced at http://hadoop.apache.org/zookeeper.

- ***Testing Closed-Source Binary Device Drivers with DDT***
  Volodymyr Kuznetsov, Vitaly Chipounov, and George Candea, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Vitaly Chipounov presented work on debugging device drivers. Testing device drivers is difficult for many reasons. Using sample input, it is difficult to cover corner cases. Exhaustive path exploration is also inadequate, because drivers run in an environment, so symbolic analysis alone is not effective. Modeling the environment completely is difficult, too. Dynamically testing requires HW and cannot do multipath execution. Static testing requires the source code of the driver and a modified environment.

Chipounov proposes DDT as a solution. It executes the driver symbolically within a virtualized machine. The machine outputs symbolic values for each hardware request. The driver is then symbolically executed, with the system state forked on conditionals. If bugs are found, constraints are solved. Interrupts are also symbolic. It's not possible to call the kernel symbolically, so each call is returned with a random value that satisfies the constraints. With kernel API annotations, coverage can be increased.

With this exception, an OS-level checker can be run on multiple paths, and a VM level checker can run outside the machine. Detailed reports are output about bugs. Chipounov concluded with a demo for a reproducible blue screen on Windows XP SP 2 based on a bug in a Microsoft-certified closed-source driver.

JUNE 24, 1:00 P.M.–3:00 P.M.

Summarized by Marc Staveley (marc@staveley.com)

- ***A Transparently-Scalable Metadata Service for the Ursa Minor Storage System***
  Shafeeq Sinnamohideen and Raja R. Sambasivan, Carnegie Mellon University; James Hendricks, Carnegie Mellon University and Google; Likun Liu, Tsinghua University; Gregory R. Ganger, Carnegie Mellon University

Shafeeq Sinnamohideen gave a brief description of the Ursa Minor Storage System, a storage system designed to scale to thousands of storage nodes. Ursa Minor is split into data storage nodes (storing bulk file data) and metadata storage nodes (storing file attributes including the location of bulk file data on data storage nodes).

Ursa Minor needed a scalable metadata store that is consistent across all metadata servers. Since some operations can affect two objects (e.g., object rename and object create) whose metadata may be on two different servers, a mechanism was needed to maintain consistency across metadata server boundaries.

Sinnamohideen's team decided not to use a distributed transaction protocol (like Farsite) or a shared state with distributed locking protocol (like GPFS), since these seemed to be overly complex systems to handle an infrequent event. Instead, they decided to migrate all of the metadata objects needed for the operation to a single metadata server before applying the metadata change.

The authors noted that multi-object operations usually operate on objects that are close in the file-system hierarchy. So they decided to organize the store so that objects that are close in the filesystem hierarchy are handled by the same metadata server and therefore do not require object migration to have operations applied to them. Sinnamohideen showed that the latency added by this model does not adversely affect the overall system, since multi-object operations are so very rare. The team then measured the performance of the store with a modified version of SPECsfs97 (with multi-sever OPS at 100 times the observed usage) and showed that the system scales linearly with added metadata servers.

It was noted during the question period that this only works if metadata migration can happen quickly. In Ursa Minor the metadata is actually stored on the data storage nodes, so migrating metadata doesn't require actually moving the data, but just changing which metadata server is responsible for it.

- ***FlashVM: Virtual Memory Management on Flash***
  Mohit Saxena and Michael M. Swift, University of Wisconsin—Madison

Mohit Saxena noted that application memory footprints are ever-increasing but we don't always have the ability to just add more DRAM (e.g., power and DIMM slots limitations). Saxena presented a virtual memory subsystem for the Linux kernel which uses Flash memory as the backing store. He showed how they had modified the current VM subsystem to remove the disk optimizations, which are not needed for a Flash backing store. Saxena then went on to show how they handled the characteristics of Flash—for example, the need to erase pages before writing to them (it was noted that the SSD discard command is very slow, so FlashVM coalesces discards to amortize the cost of the command).

Their performance evaluation showed up to a 94% performance increase when there is pressure on the virtual memory subsystem. But Saxena believes that there is more work to be done in avoiding expensive discard operations.

Does their architecture interfere with the wear leveling that is being done by the SSD? Saxena did not believe so, since they are not doing any leveling themselves, but they do aggressively reduce the number of writes to the device so as to extend its life.

- ***Dyson: An Architecture for Extensible Wireless LANs***
  Rohan Murty, Harvard University; Jitendra Padhye and Alec Wolman, Microsoft Research; Matt Welsh, Harvard University

Matt Welsh believes that 802.11 (WLAN) is not suitable for the new applications and classes of traffic that it is currently being asked to handle. The inherent problems can be mitigated if the access points and clients are all cooperating to maximize aggregate throughput, unlike 802.11, where all decisions are made by the clients with no coordination between clients or the access points. Welsh also noted that changing 802.11 is a very lengthy process (802.11e took over six years to complete).

Dyson is an extensible WLAN system that uses a central controller to gather traffic data from the access points and clients and allows the IT administrators to set policies. The policies are short Python scripts that can, for example, cause all clients to associate the access point with the lowest load factor or separate VoIP traffic from bulk TCP traffic (which greatly reduces jitter). This combination of data gathering across all participants and policy implementation gives an elegant solution to current WLAN problems.

An audience member made the observation that this could all be done without the need for a central controller, since all the APs and clients are communicating with each other. Welsh agreed but thinks it would be much more difficult to get decentralized decisions working. Someone also asked what the overhead of the data gathering packets was, to which Welsh responded that it is very, very low, since most of the information can be piggybacked on standard 802.11 control messages.

- ***ChunkStash: Speeding Up Inline Storage Deduplication Using Flash Memory***
  Biplob Debnath, University of Minnesota, Twin Cities; Sudipta Sengupta and Jin Li, Microsoft Research, Redmond

Sudipta Sengupta noted that using deduplication to decrease the amount of data stored in enterprise backup systems can save a significant amount of storage. It can also save network bandwidth if the target is not on the local machine (which can be very important if the target is across a WAN).

But in order to run deduplication at line speeds it is necessary to have a scheme for quickly looking up the chunk fingerprint (in their case a 20-byte SHA-1 hash) in the database of previously seen chunks. The problem is that with current data stores this database is too large to keep in memory. Previous systems have used disk-based database schemes with heavy caching, but there are still performance challenges.

Sengupta's team devised a scheme to use Flash memory to hold the database. Their system uses Flash-aware data structures and algorithms and strives for low RAM usage to allow for large Flash databases. Chunk metadata (chunk length and location) is organized on Flash in a log-structured manner, with a cuckoo hash table of the chunks in RAM. They also have a metadata cache in RAM.

Sengupta compared ChunkStash with using Berkeley DB to store the database on hard disk and SSD, showing that they get a 25x (HDD) and 3x (SSD) improvement over using Berkeley DB.

An audience member asked if deduplication can be done offline. That is, copy all the data to secondary storage and then dedup the secondary storage in batch mode before moving off to tape. Sengupta replied that it could be done, but you lose one of the benefits of local deduplication, which is decreasing network traffic if the backup system is remote.

JUNE 24, 3:30 P.M.–4:30 P.M.: INVITED TALK

- ***Google Books: Making All the World's Books Universally Accessible and Useful***
  Jon Orwant, Engineering Manager, Google

  Summarized by Italo Dacosta (idacosta@gatech.edu)

The Google Books project is an example of Google's philosophy of organizing the world's information. The main goal of this project is to digitize the content of all the books in the world, organize it, and allow everyone to search it. Jon Orwant, the leader of the Google Books project, presented the motivation behind Google Books, the challenges faced by this project, and the benefits and possible uses provided by this service.

Orwant said that Google Books can be divided into two parts: the publishers half and the libraries half. Today Google works with approximately 30,000 publishers worldwide. While publishers want their books to appear in Google Books, they demand that only 20% of the books' content be displayed as text snippets. Surprisingly, only 10% of the books received by publishers are in digital format. As a result, Google has to digitize most of the books provided by publishers.

According to Google Books' weekly count, there are approximately 174 million books worldwide. From this total, 20% are in the public domain (out of copyright), 10–15% are in print (copyrighted), and the rest are books that are presumably copyrighted but out of print. The problem with the books in the last category is that they are only available in libraries. Therefore, to make these books more accessible to the public, Google began to scan books from libraries in 2005. Since then, Google has been working with more than 40 public and private libraries and has scanned around 12 million books. Orwant estimates that all the books in the world will be scanned in 10 to 15 years. In addition, Orwant mentioned that libraries benefit from this project, because they can obtain digital copies of the books for backup purposes for free because no money is exchanged during the process. However, Google has been the subject of several lawsuits from groups such as the American Association of Publishers and the Authors Guild regarding the fair use of the books' content. Orwant expects that a soon-to-be-approved settlement will allow Google Books to continue scanning books while providing additional benefits to libraries, publishers, and copyright holders.

The process followed by Google to digitize each book is conducted in seven steps. First, the book is obtained from the publisher or library and is scanned. Second, the book's scanned pages are enhanced using several image processing techniques (i.e., cropping, cleaning, de-warping). Third, optical character recognition (OCR) techniques are applied to obtain the text that will allow people to search the book's content. Fourth, the scanned book is analyzed to understand its structure (i.e., text flow, headers, footers, etc.). Fifth, the book is identified based on the metadata available from different sources. Sixth, the book is classified and indexed. In the seventh and final step, the digitized book is served in Google Books.

The process of adding books to Google Books faces several challenges such as: careful handling of library books; books in many different languages; multiple inaccurate, inadequate, and ill-formatted metadata sources; non-monograph books (e.g., boxed sets, series, and multi-volume works); the lack of unique book identifiers (e.g., ISBN); determining a book's contributors; and figuring out a book's structure (e.g., page numbers, publication year). To overcome these challenges, Google relies on different engineering and computer science techniques, as well as the creativity of Google's engineers (on their 20% time projects).

Finally, Orwant described how all the information gathered by Google Books represents a "corpus of human knowledge" and presented some examples of how to take advantage of this knowledge. He commented on the use of Google Books by researchers doing linguistic analysis (i.e., predicting the regularization of verbs and determining popular words in a particular decade). Also, Orwant described how Google Books could be used to test the "Great Man" hypothesis by determining if great ideas and discoveries could have been reported earlier in history by people in different cultures and places. These types of applications are possible because Google Books allows searching not only for phrases but also for concepts. In conclusion, Google Books exposes information that before was only available on library shelves, allowing everyone to ask questions that were not possible to be answered before.

During the Q&A, someone asked to what books the Google settlement applies. Orwant answered that the settlement applies to books scanned until May 5, 2009. The settlement also gives partial benefits to books scanned after that date and to future books. Orwant added that most of the settlement benefits only apply inside the US. Can the books covered by the settlement be scanned and sold by Google without the authors' permission? Copyright holders can decide if they want their books in Google Books or not. If a book is out of print and the author does not come forward, Google can sell the book and put the money in escrow until the author reclaims it. Orwant added that other companies as well as Google can sell the books. A short discussion on whether this was a fair practice followed. Another attendee asked if Google is planning to do the same with other forms of media. Orwant answered that it is a good idea but there are several technical and legal challenges associated with gathering information from other types of media.

June 24, 4:30 p.m.–6:00 p.m.: Work-in-Progress Reports (WiPs)

First three WiPs summarized by Aleatha Parker-Wood (aleatha@soe.ucsc.edu)

- ***Live Gang Migration of Virtual Machines***
  Umesh Deshpande, Xiaoshuang Wang, and Kartik Gopalan, State University of New York, Binghamton

Umesh Deshpande presented Live Gang Migration. Co-located virtual machines are often migrated for load balancing. Since VMs often share a lot of pages, this can result in many duplicate pages being sent across the network. Live Gang Migration identifies these identical pages and transfers only a single instance. Further instances are migrated by transferring a page ID to the remote machine. In their experiments, this resulted in 40% reduction in total migration time and 60% reduction in network traffic. Kai Shen noted that VM monitors already have a feature identifying identical pages, and Deshpande responded that there is a feature for sharing pages on the same host but that Live Gang Migration is for reducing duplicate pages during migration, a case which is not currently addressed.

- ***Remote Shadow I/O: A Framework to Achieve High Performance Remote I/O Using the Shadow Device State***
  Sejin Park and Chanik Park, POSTECH, Pohang, South Korea

Sejin Park presented Remote Shadow I/O. This work focuses on unmodified guest OSes running within a virtual machine (VM). Remote I/O is a significant amount of overhead for virtual machines. Currently, performing remote I/O to a hard drive requires the guest OS to go through VMExit Handler. However, in their analysis, 80% of disk I/O doesn't actually modify the disk, just reads or sets state in the file system. Only 20% of requests actually access disk. They propose to take advantage of this by maintaining a shadow device state in the hypervisor during the 80% of get/set disk I/O operations. When the 20% of real I/O occurs, updates are piggybacked into the write, in order to synchronize the state with the real disk state. Scheduling overhead is high, so the expectation is that 8.8% of performance can be improved for their test trace.

John McCullough from UCSD asked how this compared to paravirtualized devices such as for Windows. Park replied that they don't consider this to be a paravirtualized device. Dan Peek from Facebook asked whether there was extra latency that's added to a real request because of the additional changes, since the system has to replay the shadow device to the real device. Park replied that the real device has the same latency.

- ***Designing a Snapshot File System for Storage Class Memory***
  Eunji Lee, Seung-hoon Yoo, and Kern Koh, Seoul National University, South Korea; Hyokyung Bahn, Ewha University, South Korea

Eunji Lee presented a new snapshot filesystem concept. Storage Class Memory (SCM) is non-volatile and byte addressable. It is expected to be widely deployed by 2020. It will likely replace hard disk drives, due to high performance and low power consumption. The authors wanted to build a snapshot file system which exploits the properties of Storage Class Memory. Storage Class Memory has no seek time but has a limited capacity. Current algorithms optimize seek time by using extra capacity, using copy on write, for instance. For Storage Class Memory, the authors suggest that systems should reduce space usage rather than seek time, using a "write in place" snapshot policy. Rather than creating a new root and new data in a new location, they copy the old data into a new location and overwrite the existing location with the new data. Rather than mounting a new root, the system needs to do more work to recompute an old version, but this is rare. To access it, the system restores using copy on write, updating the new data back to the old data.

Someone asked if this system was optimized for rollback versus time travel. Lee replied that it was. Peter Desnoyers asked how the system was maintaining the copies it made and whether they were linked off the old block. Lee replied that the old data blocks are contained in a list, with a pointer in the old inode to the list.

Last three WiPs summarized by: John McCullough (jmccullo@cs.ucsd.edu)

- ***Multi-Client Proxy Server on a Low-Power Wireless Router***
  Amal Fahad and Kai Shen, University of Rochester

Mobile devices are often limited by their connection quality and battery life. Wireless gateways can potentially improve the experience for mobile devices by leveraging their improved network connectivity and dedicated power source. Potential activities include caching/prefetching, media transcoding, Web site customization, offloaded computation, and security functions. The main challenge is supporting such high-demand services on a low-power device. So far they have studied the Squid caching proxy and found that it has a modest latency increase over a desktop implementation for cache hits, but for cache misses the writes have higher latency because of the shortcomings of the compact-flash storage media.

- ***SSDAlloc: Hybrid SSD/RAM Memory Allocation Made Easy***
  Anirudh Badam and Vivek S. Pai, Princeton University

Flash storage provides cheaper and more power-efficient storage than DRAM. While most Flash does not support byte-level access, it is still useful for increasing working-set capacity. Current techniques either involve custom coding to SSDs, which is labor intensive, or using SSDs as a swap backing store, which is not very well suited to the medium. This work provides a calloc style interface to a runtime that keeps objects in memory when in use, maintains unused objects in a packed form in RAM for caching, and manages log-structured page storage on the SSD. This approach provides transparent access with a 2–6x performance gain over SSD-backed swap. Information about the project can be found at http://www.cs.princeton.edu/~abadam/ssdalloc.html.

- ***Jboa Minicluster (Just a Bunch of Atoms): New Techniques for HPC***
  Mitch Williams, Sandia National Lab

Sandia has a long history of building portable mini-clusters for HPC demonstrations and small scale simulations. Historically, mini-clusters have been constructed from Pentium 2, Pentium 3, Geode lx800, Core2Duos, Via C7, and, most recently, Atom processors. Most of the work focuses on virtual machines and software. The goal is to hit 50M VMs. The current 16-node cluster gets 3K VMs on lguest using oneSIS, Clustermatic, and VMatic. The current goal is to study botnet-spreading behavior on a simulated Internet. Currently, they hope to look at other platforms, potentially including cell-phone style platforms, because Atom is slow as a cluster node.

JUNE 25, 9:00 A.M.–10:00 A.M.: INVITED TALK

- ***Reconstructing Ancient Rome: 700 Years of IT and Knowledge Management***
  Maximilian Schich, DFG Visiting Research Scientist at BarabásiLab, Center for Complex Network Research at Northeastern University

Summarized by John McCullough (jmccullo@cs.ucsd.edu)

Documentation provides a fascinating view of our world. Today we have research projects that can construct 3D models of places like the Coliseum based purely on photos from Flickr. Beyond that we have Google Street View, which gives us views into even more obscure locations. These views give us a strong sense of what the world looks like today, but can we get a sense of what they looked like long ago?

Historical evidence provides only limited evidence. The best maps of ancient Rome include only one-third of the city, and it can be very hard to reconstruct what is missing. During the Renaissance there were many who documented ancient Rome, providing insight into ground plans and architecture—or at least part of it, as the documentation is heavy-tailed with concentration on the most popular monuments and little focus on anything else. The documentation that does exist has been through a remixing process.

The documentation process iterates through five steps: (1) study existing fragments or potential source documentation and surveys; (2) integrate the fragments, creating sketchy ground plans; (3) make a full reconstruction, which may have missing pieces; (4) re-fragment the reconstruction when publishing, losing the uninteresting parts due to the high cost of paper; (5) recombine the fragments, taking artistic liberties when putting them back together. This process repeats, losing more information, and introducing architectural pieces from one document to fill in gaps in another, or even making things up completely. Historically, we can observe the process of the "inductive surveyor" who adds documentation for monuments lacking any kind of source documentation.

This leads to a paradox of progress in modern archeology, as it tends to cite modern work and the ancient sources with little mention of the middle period. This falls in line with the practice of citing the original source rather than the place it appeared, but it is hard to know what the intermediate source may have introduced. Encyclopedias have collected the various historical documents. In the mid-20th century, researchers started putting together card indices to locate monuments and sculptures. The problem of citing the original, rather than the source used, persisted. In more modern forms, the card-indices were put in a database that gives you a UI to browse for documents associated with a monument, or monuments associated with a document, but provides little information on how they relate. Schich has used link-clustering to show cross-correlation between the Roman baths with maps and provide higher-level information than simple document queries and counts. Having full access to the data can be highly beneficial, because others may have a better idea for hot to interpret data than the simple structure a query UI can provide. The datasets are complex networks of complex networks, which are themselves part of larger networks. In many ways we are approaching high-throughput humanities: research databases have been on the order of thousands or tens of thousands, but now Google Books has scanned millions of books. Perhaps we can make a huge atlas of the humanities. For more information see http://schich.info.

An audience member, observing that we're drowning in data and that a lone person is inadequate, asked whether it would be more appropriate for a doctorate to be completed by teams. Schich responded that this question has come up before, as someone's life work might be reduced to two points on a line. There is enough complex overlap that we can't carve up the world into pieces for individual study. Someone else observed that there is a lot of aggregate data and asked whether there are ways to tag it with how valid it is and arrive at a probability of correctness. The trouble is that each person entering data has a different idea of the standard of correctness and you are back to the original problem. Ideally, we want to look at correlations and implicit citations and be able to toss out the junk.

JUNE 25, 10:20 A.M.–NOON

Summarized by Joshua Reich

- ***Sleepless in Seattle No Longer***
  Joshua Reich, Columbia University; Michel Goraczko, Aman Kansal, and Jitendra Padhye, Microsoft Research

Joshua Reich pointed out that idling PCs in corporate/enterprise networks waste significant amounts of power by idling. These machines generally have their OS settings disabling sleep because users and administrators want continuous and seamless access to these machines. Sleep-proxying systems were suggested as a solution to this problem over a decade ago. Yet they have not yet been deployed commercially. Reich argued that the key issue that need be considered here is the economic feasibility of the sleep-proxying system. The authors chose a sleep-proxying design for easy and economical deployment and maintenance. Their sleep-proxies reaction policy extends the best recommendations of previous work with their own customized improvements.

The reaction policy proposed by the authors is straightforward. Right before the client machine sleeps, it broadcasts a quick notification—informing the sleep proxy of the ports on which it is actively listening. The sleep proxy (which can be a lower-power, low-cost box—potentially even a client peer) then takes over, redirecting all traffic for the client to itself. It responds to IP resolution traffic, wakes the client only for incoming TCP connection attempts to the set of ports on which it had been listening, and ignores all other traffic.

Reich next shed light on the factors that impede the practical performance of sleep-proxying systems in real networks—identifying the twin problems of "crying babies" and "application-based insomnia." The first of these accounts for ~10% of lost sleep and is caused by other networked machines that attempt to connect to sleeping clients too often. The second accounts for ~90% of lost sleep and is caused by applications running on the host that prevent the host from sleeping in the first place. In both of these cases, it appears that IT servers and applications are the main troublemakers. The good news is that relatively low-cost approaches can likely be leveraged to schedule these applications in a coordinated fashion that will leave much more potential sleep time.

How much of the sleep savings you show came from your system as opposed to the default Windows sleep behavior? Reich said that in their environment it was 100%, as Windows sleep was disabled on all of their machines before their system was rolled out. In other environments, these savings would be reduced by one-third to 5%—depending on what proportion of the machines would have been sleeping. How does their system differ from the Apple sleep proxy system? Their system is geared to the home consumer and only works in their own closed ecosystem. In terms of reaction policy, they are quite similar (they support WiFi). However, the focus of the authors' work is on economic deployment and learning the lessons of such and their main finding is that the IT setup is really what you have to worry about.

How much of the sleep achieved was due to the particular setup at Microsoft? Wouldn't machines wake more elsewhere? Reich answered yes, that's one of the main reasons why they chose an extensible software-based approach instead of a hardware NIC-based approach—so they could do blacklisting, whitelisting, etc. However, these additional wake-ups would really come from scanning machines and they are focused on the corporate network, which tends to be firewalled pretty heavily, not on more open academic networks where this would be more of an issue. You could implement pretty much any reaction policy you'd like (although LiteGreen-style virtualization wouldn't work) using their framework.

- ***Wide-area Network Acceleration for the Developing World***
  Sunghwan Ihm, Princeton University; KyoungSoo Park, University of Pittsburgh and KAIST; Vivek S. Pai, Princeton University

Sunghwan Ihm pointed out that Internet access in developing regions is a scarce and expensive commodity. Web proxy caching has been proposed as a solution to this problem in the developed world. However, this solution isn't adequate for the developing world, where there is significantly greater diversity of demanded content (and thus much less cache-able content). So the authors propose a combination of Web proxy caching and WAN acceleration. In this scheme WAN accelerators sit in both the developed and developing world, with data being chunked together, compressed, and sent using much less bandwidth. Chunk metadata is stored in accelerator memory, while data is kept on disk.

There is a significant challenge here—small chunking has a high compression rate (less extraneous data is put in a given chunk) and puts little pressure on the disk (fewer cache misses) but puts much more pressure on the memory (since many more chunk IDs need to be stored). Large chunking has the opposite trade-off: better for memory, but it puts pressure on the network and disk. Consequently, the authors proposed multi-resolution chunking (MRC), which uses large chunks to ameliorate memory pressure and disk seeking and small chunks to achieve high compression rates. They generate these chunks efficiently by detecting the smallest chunks first and then making their way up (data contained in small chunks may also sometimes be encoded in larger chunks).

The authors also took advantage of the assumption that there will be many meshed machines in such a developing-world network. If this is the case, they can trade off between network (grabbing content from peer memory caches) and disk (grabbing it off one's own disk) to maximize efficiency. The authors evaluated their work with simulation experiments and a small testbed implementation.

Why not apply these techniques for the developing world to the developed world? Sunghwan agreed.

- ***An Evaluation of Per-Chip Nonuniform Frequency Scaling on Multicores (Short Paper)***
  Xiao Zhang, Kai Shen, Sandhya Dwarkadas, and Rongrong Zhong, University of Rochester

Xiao Zhang described the problem as applying DVFS to all cores on the same chip. However, not all cores are doing (or need be doing) the same amount of work. The authors proposed smart scheduling to facilitate per-chip frequency scaling, thereby saving power on the cores eligible to be run at lower frequencies. To do so, they group applications with a similar cache-miss ratio on the same chip. This way, applications with high cache-miss rates can be run at lower frequency (since the processor will most often be blocking for I/O anyway), while applications with low cache-miss rates can be run at higher frequencies. This also removes pressure on the cache (as applications with a high rate of cache misses are not continually knocking the cache lines of applications with lower rates of cache misses out of the cache). Likewise, it reduces pressure on the memory bus.

They evaluated their techniques on a 2-chip Intel 3GHz WoodCrest processor (two cores per chip, sharing a 4MB L2 cache) SMP running Linux 2.6.18 by running 12 SPEC-CPU 2000 benchmark applications. They found that their techniques performed reasonably well. Moreover, it appears that the power savings they experienced can be reasonably approximated using a relatively straightforward model. They then applied this model to develop frequency scaling policies that provided reasonable power savings.

Someone asked why the similarity grouping without using frequency reduction raises temperature. A CPU working at full blast will generate more heat than two CPUs sharing load. Someone else pointed out that their performance prediction model assumes that the behavior of other cores doesn't affect performance of the core that they are modeling. Doesn't that seem odd, given that lots of other resources are shared? They are looking at stand-alone applications. Having several applications running on other cores will affect things, but they think it is a second-order effect. This holds on an SMP-based machine, not on a NUMA-based machine.

- ***A DNS Reflection Method for Global Traffic Management (Short Paper)***
  Cheng Huang, Microsoft Research; Nick Holt, Microsoft Corporation; Y. Angela Wang, Polytechnic Institute of NYU; Albert Greenberg and Jin Li, Microsoft Research; Keith W. Ross, Polytechnic Institute of NYU

Jin Li raised the question of how to best select one of many remote locations from which to serve a Web-based content request. In order for a provider to direct users to the server it desires, it will use DNS redirection/reflection based on the

IP address of the client and insert that into the client's local DNS. This can be done using a geo-location database or using an anycast solution.

Yet, how do we pick the best remote site for a given client? Passive DNS-based measurement can be used, but this has many drawbacks, particularly that the performance of some of the clients is significantly degraded. So most CDNs have used active probing techniques. However, many clients (~45%) cannot be probed actively. So instead they use DNS traffic (DNS reflection) to trigger DNS queries from any LDNS server. Essentially, when an LDNS server that cannot be actively probed makes a query to a top-level DNS server, that server reflects the query to a collector node. Then the time between queries is measured and the network path performance inferred.

Using 17 DNS servers and 274 Planetlab nodes, the authors show that DNS reflection tracks within 6ms of ping.

Someone asked if they had thought about applying this technique to similar passive measurement problems. Li said that they have some other work in this area (e.g., measurements of clients to CDN providers).

JUNE 25, 1:00 P.M.–2:00 P.M.: INVITED TALK

Summarized by Rik Farrow (rik@usenix.org)

- ***RoboBees: An Autonomous Colony of Robotic Pollinators***
  Matt Welsh, Associate Professor of Computer Science at Harvard University

The idea started as hallway talk. Bee colonies around the country have been dying off, yet bees are essential pollinators of crops. The idea turned into a short paper, then a team was recruited. Brainstorming was followed by creation of an outline, division of labor, and a funding request for $10 million, which the NSF actually granted.

But that's not where Matt Welsh started his talk. Creating a colony of robotic bees is not just a CS project, as there are many problems to solve. The researchers broke the problem into three main areas: the brain, the body, and the colony, with different teams working on each area.

The body shares some aspects with actual bees: a pair of veined wings and small size. The veins and associated wing corrugations are important for flight. For muscles, piezoelectric actuators that require 200 volts but just tens of milliwatts of power are planned, with a flapping frequency of 230 hertz—very similar to bees. First flight has been achieved, but only when tethered to a power supply.

Power is a critical issue. Batteries will not work, because of size limitations, so Welsh said they plan on creating fuel cells tiny enough to fit on chips. There are existing micro-fuel cells, but they run at 200–500° C and require hydrogen for fuel.

The brain must interpret sensors, control flight, and follow instructions. Welsh explained how optical flow can be used with a simple 64x64 pixel sensor from Centeye: if you want to pass through an opening between obstacles, you want the optical flow to be equalized on either side of the opening. If the optical flow is getting uniformly larger, you are about to run into a wall.

They plan on using an ARM processor and accelerometers that can be turned on or off as needed. The program will model neural control, keeping things as simple as possible.

For the colony, they need a non-centralized organization but robustness as well. Welsh described using a high-level language to create a program that would be downloaded to robobees to get them to search, for example. For now, they are experimenting with Blade mCX micro-helicopters, with a goal of having 50 helicopters under radio control. Welsh showed a video of a computer controlling a micro-helicopter via radio, flying briefly then crashing. There is obviously a lot of work to do here.

Dan Peek of Facebook pointed out that plants and pollinators co-evolved, and that he just wanted to pass along that idea. Dan Klein commented on a news clip that Welsh showed toward the end of his talk. Fox News had called the program "a good example of wasting government funds as only 1.66 people were hired," and Klein wondered who the .66 person was. Welsh explained that the grant was funding 1.66 post-docs. Jitendra Padhye wondered why use flapping wings, and Welsh said that one of the other researchers believes that this is the most power-efficient design. Maximillian Schich worried about birds eating robobees, and Welsh agreed that it was important that they be able to find lost hardware.

JUNE 25, 2:00 P.M.–3:00 P.M.

Summarized by John McCullough (jmccullo@cs.ucsd.edu)

- ***An Analysis of Power Consumption in a Smartphone***
  Aaron Carroll, NICTA and University of New South Wales; Gernot Heiser, NICTA, University of New South Wales, and Open Kernel Labs

Aaron Carroll pointed out that smartphones have poor power consumption, as evidenced by how often we have to charge them. The situation is only getting worse as we add functionality without any fundamental increases in battery capacity. Current technology uses dynamic voltage and frequency scaling for computation, but in real systems the CPU doesn't use that much power. If we ask what does use power, the answer is often that nobody knows or vendors won't tell us.

To address this question, the authors instrumented an OpenMoko Freerunner to cover 98% of the components and measured the phone with a variety of benchmarks. In the suspended state the phone draws 69mW and at idle it draws 269mW. Half of the idle power is in the GSM chip, and the GPU, LCD, and backlight draw significant power. At full power, the backlight alone draws 400mW. RAM and CPU are actually fairly power-proportional. When browsing email, the GSM draw is half of 610mW. When playing back locally stored video, the CPU and RAM are dominant. When playing audio, display power is high because SD access goes through the GPU.

The authors looked at the more modern HTC Dream (G1) and Google Nexus One (N1) for validation, assuming the power breakdown is similar. The G1 and N1 have better idling power because of improvements in 3G over the older chipsets, and they found that the radios draw similar power even with the large differences in data transfer throughput. Computationally, DVFS provides energy benefits for the Freerunner and N1, but the G1 works better completing at full power and then sleeping. This generally shows that DVFS can still be effective, even though it has been eschewed lately. In general, the biggest consumers of power are the display, the cell radio, and, in some cases, the CPU. Power is not going to RAM, Audio, Bluetooth, or storage.

One of the audience members asked about variance in the LCD power based on displays. The author responded that there is variation from 14mW for white to 70mW with black for some displays, but that it varies by display technology and in some cases you get the opposite. Therefore you have to be sure to match a power-saving designed theme with your phone. Was the platform measurable because of its construction and could you measure other phones if you had schematics? The OpenMoko is measurable because of construction, but other platforms are likely to be hard due to routing through multi-layer circuit boards. While you can do some inference from coarse measurements, the authors wanted better accuracy.

- ***SleepServer: A Software-Only Approach for Reducing the Energy Consumption of PCs within Enterprise Environments***
  Yuvraj Agarwal, Stefan Savage, and Rajesh Gupta, University of California, San Diego

Yuvraj Agarwal said that buildings represent a large fraction of total power consumption. While the lighting, heating, and cooling are all duty-cycled well, the IT loads are typically not. This is particularly worrisome as the amount of power dedicated to IT is expected to continue increasing. The authors instrumented the UCSD CSE building and found that IT loads constitute 50–80% of the total building power even when most of the machines are idle.

Most modern PCs support sleep states that reduce power consumption to 1–2% of idle. This represents a huge potential for power savings, yet most people don't put their computers to sleep. The problem is that users or IT departments want to access the computers remotely or the users want to keep downloads running and maintain IM or VoIP presence. Unfortunately, sleeping computers can't provide that directly. There is wake-on-LAN, but it requires the magic "wake-up" packet to be sent from the local network segment and is typically a usability non-starter. You can use a sleep proxy that solves the usability problem of wake-on-LAN and can provide high-level filters, but it cannot handle stateful applications and users leave their computers running at full power for simple downloads or to update emails while they're out to lunch. The other end of the spectrum is full desktop VM-migration that allows the computer to run all of its applications on a server while the desktop sleeps. But that requires heavy technological buy-in, and the degree of power savings is tightly coupled to the scalability of hosting heavyweight VMs on servers. The authors offer an alternative called SleepServer that has most of the functionality of VM migration with the same cost of the sleep proxy.

The goal, then, is to be able to maintain presence transparently, match proxying demands for each sleeping PC, be highly scalable, address enterprise management, and be multi-platform. SleepServer addresses ARPs, ICMP, and DHCP directly while providing the ability to wake up on user-defined filters for traffic like incoming ssh or remote desktop requests. Stateful applications such as background Web downloads need application-specific "stubs" that receive current state when the machine goes to sleep and transfer new state back to the associated application when the machine wakes up. SleepServer is implemented using lightweight virtual machines that maintain the IP and MAC addresses of the machine with all associated VLAN encapsulation. The VMs are provisioned with 64MB of RAM and 1GB of storage, which has been shown to scale up to 250 virtual machines on a single 300W test server.

The authors have a deployment with 40 users, many of whom would not otherwise put their computers to sleep, due to remote access needs or needing at least one stateful application. In early tests they found that automatic sleep policies are much more effective than manual activation, due primarily to forgetfulness. When using the automatic sleep policy, overall power savings averaged to 70%. Widespread deployment in the department could be supported by two servers and potentially result in a cost savings of $60,000 per year. For more information and measurements, see http://energy.ucsd.edu.

Can SleepServer handle 802.1X authentication? They haven't looked into it. Another audience member inquired about the complexity of customizing applications and stubs for each image. Agarwal responded that most users are typically covered by a few stateful applications. What would be lost if stubs were removed? A number of the users would not participate in SleepServer without some of the features, regardless of whether they are used. How can you translate to stubs? You need to modify the applications, though in general there are only a few stateful applications that the users are concerned about. Could SleepServer be implemented in a lightweight manner, something closer to a honeypot? The necessary functionality could be implemented in software, but most of this functionality already exists in VM technology and there is an implementation trade-off.

JUNE 25, 3:30 P.M.–4:30 P.M.

Summarized by Paul Marinescu (pauldan.marinescu@epfl.ch)

- ***An Extensible Technique for High-Precision Testing of Recovery Code***
  Paul D. Marinescu, Radu Banabic, and George Candea, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Paul Marinescu started his presentation by arguing that current general-purpose software testing lacks the tools for testing error recovery code, as coverage information from various systems indicates. He then introduced a tool, LFI, that uses library-level fault injection to test error recovery code without making changes to the system under test.

Marinescu said that the real problem when doing fault-injection testing is finding good strategies to inject faults. He then focused on answering the when, where, and what to inject questions. He first introduced the notion of injection triggers, a mechanism that allows testers to specify with an arbitrary degree of precision when to inject. Then he showed a static analysis tool that can automatically find where to inject faults by choosing only the places where the return codes are not checked. Finally, he presented a different static analysis tool that can automatically infer possible error codes that an arbitrary library function can return.

The evaluation showed 11 new bugs LFI found in BIND, MySQL, Git, and PBFT, as well as the ability to improve line coverage of error recovery code from less than 5% to 35% for Git and 60% for BIND, entirely automatically, without writing new tests. LFI is open source software, available at http://lfi.epfl.ch.

How can LFI work without needing source code since some of its components were explicitly using source code information? Marinescu replied that source code or domain knowledge is not needed by LFI but can improve the results if available. How fast are the static analysis tools presented? The tools can analyze large systems (e.g., MySQL, libxml2) in a couple of minutes.

- ***Mining Invariants from Console Logs for System Problem Detection***
  Jian-Guang Lou and Qiang Fu, Microsoft Research Asia; Shenqi Yang, Beijing University of Posts and Telecom; Ye Xu, Nanjing University, P.R. China; Jiang Li, Microsoft Research Asia

Jian-Guang Lou argued that console logs are widely used by programs because (1) they are easy to use and (2) the free text format is very expressive. However, console logs are usually too big to manually parse in search of abnormal program behavior. The speaker proposed an automatic solution for interpreting log files. At its core, the solution relies on linear invariants based on the execution count of logging instructions. The linear invariants can be used to model control flow such as sequential execution, branching, or joining. Violations of these invariants indicate anomalies and also point to the place where the anomaly happened.

The problem is that automatically inferring the invariants for an arbitrary log file is NP-hard. Lou proposed a three-step solution for reducing the computational cost of the analysis: (1) the free text is transformed into structured text; (2) log entries are grouped according to the system variables they refer to; and (3) a hypothesis and testing algorithm is used to find the invariants. Several strategies including divide and conquer, early termination, and skipping are proposed to reduce the search space of invariant mining.

The evaluation consisted of searching for anomalies in Hadoop, CloudDB, and SharePoint log files. The approach was able to find anomalies in all the log files, out of which about 75% were caused by bugs.

Timothy Roscoe was interested in whether domain knowledge could be incorporated in the proposed algorithm. Lou said that is certainly feasible and could improve the accuracy of the analysis.

USENIX Conference on Web Application Development (WebApps '10)

June 23–24, 2010
Boston, MA

WebApps '10 shared the opening session and Keynote Address with the 2010 USENIX Annual Technical Conference: please see p. 63 for the report on that session.

JUNE 23, 10:30 A.M.-NOON

Summarized by Rik Farrow (rik@usenix.org)

- ***Separating Web Applications from User Data Storage with BStore***
  Ramesh Chandra, Priya Gupta, and Nickolai Zeldovich, MIT CSAIL

  Won Best Paper Award!

Ramesh Chandra pointed out that while some apps (e.g., Google mail) rely on a single online store, other applications require getting data from one site and doing something with it using a different site. Chandra used an example where a photo editing site needs to get on Flickr to gain access to a photo.

Their solution is BStore, moving data storage within the browser. BStore provides a single, simple (four call) API for storing data and is implemented in JavaScript. Back-end storage can be in the cloud (S3) or local. BStore provides security through tagging data. Only the principal or the user can tag data for sharing with another application. Tagging is used for more than access control, as files may be logically grouped using tags.

They ported several Web apps with a minimal amount of effort, adding about 5% to the size of the app. BSTORE uses postMessage for RPC within the browser, and currently only works in Firefox and Chrome. Performance is similar to using XMLHttpRequest.

Someone asked if BSTORE could use the browser cache for local store, and Chandra pointed out that the browser cache uses the same origin policy for security, so cannot be used. Dan Peek asked if they had tried to extend this to multiuser control, and Chandra said that tagging supports this. Also, they include version numbers with files, and that would prevent a file from being overwritten. Ben Livshits asked if they had thought about XSS or code injection. Chandra replied that each piece of code runs in its own window and process space, and they use an RPC built on top of post-Message.

- ***AjaxTracker: Active Measurement System for High-Fidelity Characterization of AJAX Applications***
  Myungjin Lee and Ramana Rao Kompella, Purdue University; Sumeet Singh, Cisco Systems

Myungjin Lee pointed out that applications are being migrated to the cloud using AJAX (Asynchronous JavaScript and XML). AJAX supports autonomous action on the client side, and this poses problems for measuring performance. The authors' goal is to characterize full application sessions, including flows/servers, request/response distributions, and inter-request time, as well as local operations such as dragging an icon into the trash or clicking on a map.

Their approach involves capturing X events using a program as well as capturing network traffic at the client side using tcpdump. They compared their approach, which collects both X events and network traffic, to a network-only approach, and tested both at four different bandwidth conditions. You can download the tool here: http://www.cs.purdue.edu/synlab/ajaxtracker/.

James Mickens wondered if they were using the DOM object ID in XML files. Lee pointed out that they are using an object ID as defined by users, since they have access to the X event that generated the actual event in the Web browser. Someone else asked how well their approach worked in richly dependent apps, and Lee said that there were limitations to their approach. John Ousterhout wondered if they had overfitted their data as shown in a graph in their paper. Lee said that they had picked one of four bandwidths to characterize the dominant bandwidth based on network traces, but did not use a sophisticated model to fit the curve.

- ***JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications***
  Paruj Ratanaworabhan, Kasetsart University; Benjamin Livshits and Benjamin G. Zorn, Microsoft Research

Ben Zorn stated that benchmarks, such as SunSpider and V8, do a poor job of capturing real-world Web application performance. JSMeter is a Microsoft Research project that instruments Internet Explorer, but is only currently available within Microsoft. The advantage of being able to instrument the browser itself, rather than running benchmarks on top of the browser, is that JSMeter can measure three areas of JavaScript runtime behavior: functions and code; heap-allocated objects and data; and events and handlers. Zorn pointed out that JavaScript is event-driven, and benchmarks run tight loops, allocate little data on the heap, and have few events.

Zorn displayed graphs showing the different heap behavior of several Web apps, including ones from Google, Amazon, eBay, The Economist, and Microsoft. JSMeter makes it easy to see different programming styles, such as how the heaps grow continuously with Gmail, while eBay wipes the heap clean with each new page. JSMeter also clearly shows that Google caches functions across reloads to speed performance. In Bing, you stay on the same page and the heap grows a lot over time, where Google focuses on using less memory.

Zorn went on to point out more shortcomings of V8 and SunSpider that make them poor benchmarks (and one that makes IE look much slower than Firefox and Chrome). He suggested that people visit their Web site and read their tech report: http://research.microsoft.com/en-us/projects/jsmeter/.

Does JSMeter deal with anonymous functions? JSMeter does monitor them, uses hashes to identify functions, and counts each invocation. Have they have noticed any changes in the way JavaScript is being used? JavaScript use is indeed a moving target and they expect its use will keep changing over time. Armando Fox mentioned that there are a number of code frameworks that generate JavaScript and wondered what framework writers could do to make the world better. Zorn said that the amount of code downloaded affected performance a lot, so why download code that is never going to be executed? He suggested staging code and lazily downloading code as needed. Someone else asked about how Web apps could better use browsers, and Zorn suggested that better tools need to be available. While JSMeter was an internship project in IE8, IE9 will provide tools that will be available to developers.

JUNE 23, 1:30 P.M.–3:00 P.M.

Summarized by Aiman Erbad (aerbad@cs.ubc.ca)

- ***JSZap: Compressing JavaScript Code***
  Martin Burtscher, University of Texas at Austin; Benjamin Livshits and Benjamin G. Zorn, Microsoft Research; Gaurav Sinha, IIT Kanpur

Ben Livshits from Microsoft Research introduced JSZap, a technique to efficiently compress JavaScript code based on Abstract Syntax Tree (AST) representation. Livshits claimed that 85% of downloads for rich Web applications is Java-Script code. An application like Bing Maps has more than

70,000 lines of code, which translates to more than 1MB of network traffic. This trend adheres to the conventional wisdom of moving more code to the client for responsiveness. However, execution cannot start without the code, so their approach aims to make the wire format lighter.

The main idea in JSZap is to send JavaScript code using an AST-based representation instead of sending the raw code and generating the AST at the client. JSZap uses Gzip as second-level compressor, following the wisdom that says if you can't beat them, join them. JSZap splits the JavaScript compression into three streams: productions, identifiers, and literals. JSZap proposes a number of techniques for compressing productions, such as frequency-based production renaming, differential encoding, predictable production elimination, and tree-based prediction by partial match. To compress identifiers, JSZap used a symbol table instead of a flat stream. JSZap achieved an average 10% improvement across the board. Preliminary results were verified using benchmarks with different sizes, application types, and a mix of machine-generated and hand-crafted code. Livshits claimed that AST-based representations can have broader applicability in caching incremental updates, unblocking HTML parser, correctness, and security checks.

Dan Peek from Facebook asked why the AST is the right abstraction level. Livshits explained that the language is standardized while lower-level representations (byte codes) are specific to browsers, and it is a small step to get the AST. Wanchun Li from Georgia Tech asked about performance trade-off and how much time it takes to parse the AST. Livshits explained that is future work and that they are not concerned about time on the server side. Jon Howell from MSR asked about other places where Gzip fails and the JSZap algorithm succeeds. Livshits mentioned tree-based predictive matching where structure matters, and places where the window of compression is small in Gzip. In a follow-up, Dan Peek asked about the ability to tweak Gzip settings. Livshits said they used the same settings in the browser, because some powerful compression algorithms are not supported by browser implementations.

- ***Leveraging Cognitive Factors in Securing WWW with CAPTCHA***
  Amalia Rusu and Rebecca Docimo, Fairfield University; Adrian Rusu, Rowan University

Amalia Rusu explained how they mapped their knowledge in the field of handwriting recognition and image analysis to improve Web security and CAPTCHA generation algorithms. CAPTCHA challenges are generated automatically on the fly without dependence on a database and are public, so the algorithm/code to build CAPTCHAs should be publicly accessible. They are used widely to differentiate between humans and machines and secure Web sites. CAPTCHAs are made harder for machines but this sometimes comes at the expense of human usability. The authors' work can improve the user experience of CAPTCHA while still maintaining or even improving the security, which Rusu refers to as usable security. The main contribution is to introduce structures and transformations in CAPTCHA challenges based on cognitive factors so that only machine recognition is hindered.

Some basic ideas which this work revolves around are: handwritten CAPTCHAs have unique characters that increase readability while making it harder for machine recognition, and tree-based structures are the basic structure for information visualizations so trees are familiar to people. They also leverage principles of cognitive psychology (such as symmetry, proximity, similarity, etc.) and geon theory in pattern recognition to transform handwriting. These could also be extended to the tree structure as well. So the CAPTCHA they envision is a tree structure with each node representing a handwritten word. To solve the challenge, the human/machine will answer a question such as what are the words connected with an edge, or give me all the pairs in this tree. So in order to pass the CAPTCHA you need to recognize the words, segment the information, and interpret it, which is where machines fail. In a preliminary evaluation, they tested with both machines and users with three state-of-the-art recognizers. Machines had a low recognition rate of 1–5%. In usability testing, the recognition rate for humans is more than 80%, and users gave these CAPTCHAs a difficulty level of 2 (5 is most difficult). These results show the feasibility of the handwritten tree-based CAPTCHAs.

James Mickens from MSR asked how you would attack this system. Rusu explained that the pre-processing phase before segmentation is used to attack CAPTCHAs. They created some custom attacks based on techniques such as occlusion. These attacks can make recognition for other transformations worse, so we cannot combine attacks for different kind of transformations. Dan Peek from Facebook asked if these techniques can enhance recognition neutral to the language. Rusu explained that, for their approach to work, they need to generate everything on the fly. So if they have a tool to generate the words based on IP addresses using the language common in the specific region, this might improve the CAPTCHA generation and make it language-neutral.

- ***Gulfstream: Staged Static Analysis for Streaming JavaScript Applications***
  Salvatore Guarnieri, University of Washington; Benjamin Livshits, Microsoft Research

Salvatore Guarnieri introduced Gulfstream, which helps with safe inclusion of third-party source code using staged static analysis. Two ways to have safe inclusion are run-time enforcement, which detects the behavior of code at run-time and prevents the damage, and static analysis, which analyzes the code before it is sent and, if it detects that something bad could possibly happen, prevents the page from being sent to the user. Static techniques are the focus here, and they usually require full source code. But JavaScript is usually streamed as you interact with the page. Updates are relatively small compared to the static page you are getting from the server initially. Gulfstream performs offline static analysis on the static page in the server and online analysis of the small updates in the client.

To understand the behavior of the program they use queries. For example, what does variable f refer to or is alert() ever called? To get the information, they use points-to analysis, which tells them what memory location f points to. They have two techniques for the points-to analysis: (1) datalog with an engine based on binary decision diagrams, which is fast for large programs and highly tuned but has a large start-up cost and is difficult to integrate in browsers; (2) graph-based flow analysis, which has a small start time but does not scale very well. This trade-off is reasonable when the updates are small. Gulfstream starts by normalizing the JavaScript, building the flow graph, and serializing the graph. Then Gulfstream performs points-to analysis and uses the results along with the flow graph to answer the queries. In evaluation they asked if Gulfstream is faster than non-staged (static) analysis using a representative benchmark. They simulated analysis on a diverse set of devices (PC and mobile) with different CPUs and network speeds. The main results are that slow devices benefit the most from Gulfstream because the analysis is CPU-intensive and is harder to perform in slow devices. A slow network can negate the benefits of the staged analysis if the CPU is fast enough to finish the full analysis before the staged analysis results finish transferring over the network. Large page updates don't benefit from Gulfstream due to analysis overhead (> 60kb).

Jon Howell from MSR asked why answering queries is slow—taking tens of seconds for large devices (which means minutes for small devices)—for both techniques. Guarnieri mentioned that they need to optimize the approach to yield timely results. Godmar Back from Virginia Tech asked whether the fact that all these applications were written to be modular can be exploited in the analysis. Guarnieri said that optimizations based on modularity might have malicious applications, so they were not considered.

JUNE 23, 3:30 P.M.–5:00 P.M.: WORK-IN-PROGRESS REPORTS (WIPS) AND POSTER PROMOS

Summarized by Pradeep Teregowda (pbt105@psu.edu)

- ***Detecting User-Visible Failures in AJAX Web Applications by Analyzing Users' Interaction Data***
  Wanchun Li, Georgia Institute of Technology

Wanchun Li pointed out that developers and administrators are not aware of user failures and issues with interfaces, which include AJAX and input elements in Web applications. He presented a method for detecting such failures of interfaces by identifying interactions such as repeated steps or missteps from user interaction data. He presented initial results for an AJAX application.

- ***Doha: Real-Time Support for Event-driven Web Applications***
  Aiman Erbad and Charles Krasic, University of British Columbia

Aiman Erbad presented Doha in the context of HTML5. The proposed HTML5 standards provide powerful tools for execution on the client side. Doha allows developers to effectively leverage workers on the client side. It does this by managing resources and coordination of Web workers so that event-driven Web applications can be supported in real time.

- ***MyEbay Research Web Service: An Application to Practice the Web Service***
  Shaun-inn Wu and Jian Huang, California State University San Marcos

Shaun-inn Wu presented an approach for involving students in building Web applications. MyEbay was built for students involved in undergraduate coursework. This service allows users to explore the depth of the Web application features and interfaces. MyEbay has been successful in involving students in Web application projects.

- ***A Seamless Online Application and Advising System for a Degree Program***
  Shaun-inn Wu, California State University San Marcos

Scheduling university courses presents several challenges, especially coordinating student choices and courses. Shaun-inn Wu and his group are building an online advice system for degree programs offered by the university. In scheduling courses, students are advised to contact other students following the same course pattern.

- ***Fine-Grained Isolation in Web Browsers Using Script Spaces***
  Amarjyoti Deka and Godmar Back, Virginia Tech

Godmar Back points out that the current state of browser isolation cannot support complex applications even with Google Chrome. Pages contain widgets executing third-party code; client-side extensions running content scripts on Web pages are particular examples. Script Spaces developed by his team provide an isolated execution environment for all parts of the Web page, separate namespaces, and access to CPU and memory. Script Spaces allows for fail-safe loading of pages without lockups. A prototype based on Firefox 3.0 has been developed.

- ***SaaS and Cloud Computing in Undergraduate Software Education***
  Armando Fox, University of California, Berkeley

Armando Fox discussed the use of cloud computing for a SaaS course. Adopting cloud computing saved resources and made it easier to finish assignments. The focus on SaaS using agile development also resulted in high-quality working prototypes by the end of the semester. Dedicated server resources would have consumed entire data centers and could now be used only when needed. The lifetime of projects also lasted more than the length of the course.

- ***xHunter: Tracking XSS Attacks on the Net***
  Elias Athanasopoulos, FORTH-ICS

Elias Athanasopoulos presented xHunter, a tool which allows researchers to track and analyze patterns of XSS attacks. Cross-site scripting (XSS) attacks are a significant security issue for Web applications. xHunter processes URLs, trying to build syntax trees using the provided URL. It marks those as suspicious that generate high-depth JavaScript syntax trees. Elias requested that users submit URLs to xHunter.

- ***A Case for PIQL: Performance Insightful Query Language***
  Michael Armbrust, Nick Lanham, Stephen Tu, Armando Fox, Michael Franklin, and David Patterson, University of California, Berkeley

Stephen Tu et al. found that many applications are moving away from traditional relational databases to key/value pair stores for scalability and performance guarantees. They propose PIQL, an expressive language for key/value stores. An aspect of PIQL adoption was challenging developers to think beyond the standard database design, forcing them to handle performance issues at design time. Applications developed with PIQL were demonstrated.

- ***A Performance-Monitoring Framework for Multi-Tier Web Applications***
  Chris McCoy, Northeastern University and Smarter Travel Media; Ryan Miller, Smarter Travel Media

Chris McCoy and Ryan Miller found it was a challenge monitoring profile information across multiple tiers, especially when they consist of heterogeneous systems and applications. The tool they developed allows administrators to monitor performance across the cluster with multiple tiers supporting better granularity than those provided by current tools. The data collected from the systems is displayed in a user-friendly interface.

- ***A Combined Autonomic and On-Demand Approach to Configuring Virtualized Development Environments***
  Ryan Miller, Smarter Travel Media; Matt Warren, Northeastern University and Smarter Travel Media

Ryan Miller said that developers in virtualized environments can save time and improve efficiency by adopting the proposed autonomic and on-demand configuration system. Such an adoption would reduce development time by enabling administrators to quickly deploy the virtualized environment without being constrained by complex configuration steps.

- ***Taking Control Away from Users . . . in a Good Way***
  Jon Howell, Microsoft Research

Jon Howell raised the provocative question of how much control should be vested with users. The contention was that, while simple choices are easy for the user to understand and answer, questions which are more involved tend to confuse the user. Such a model already exists in data centers and can be extended to desktops and Web browsers. Architectural changes and user interface issues were discussed.

JUNE 24, 9:00 A.M.–10:00 A.M.: INVITED TALK

- ***Gmail: Past, Present, and Future***
  Adam de Boor, Staff Software Engineer, Google

Summarized by Pradeep Chalise (pradeepchalise@gmail.com)

Adam de Boor started by outlining his talk: where Gmail came from, where it is now, what the Gmail team learned, and their plans for the future. Today, Gmail has expanded beyond mail to include video chat, voice chat, Gmail labs, and lots more. For Gmail to arrive at this stage, it had to undergo a lot of technical modifications since its launch back in 2004. At that time, Gmail was a slick Webmail application using AJAX, and it has continued to develop ever since. De Boor said that to fulfill Gmail's promise to its users, it has evolved into a single complex application supporting a lot of diverse functionalities.

De Boor wanted to provide some higher-level concepts regarding the macro architecture of the communication between the Gmail client and server, how the business logic is invoked and is used to reply to the client for chat, mail, etc. He explained some details of implementation of the Gmail client module and the Gmail server. Then he provided an introduction to mods, which are named code segments enabled on a per-user basis. He then combined the concept of modules and mods to give us the clear idea of what is possible and what is served. He also compared Gmail with Microsoft Windows features like video, chat, messaging, etc.

Gmail's future plans include dealing with service-oriented architecture, trying to make users feel that Gmail is like a desktop application. Gmail developers are now planning to use HTML5, which is exciting because it reduces the DOM by 30% and initial load time by 12%.

Finally, de Boor informed the audience about the lessons learned by the Gmail team through its experience: testing is vital, type-checking is important, it's beneficial to instrument everything and codify lessons learned in sanity tests.

Since Google heavily depends on JavaScript, are there any plans on the horizon for using any other language? Lots of programmers understand JavaScript and do simple things with it, so they will continue using it. There is no single application that doesn't use JavaScript right now.

JUNE 24, 10:30 A.M.–NOON

Summarized by Pradeep Chalise (pradeepchalise@gmail.com)

- ***Managing State for Ajax-Driven Web Components***
  John Ousterhout and Eric Stratmann, Stanford University

John Ousterhout started his speech by giving an introduction to the basics of Ajax and why and how Ajax-driven components cause problems. To deal with the problem that Ajax complicates Web applications is the idea of using reusable Ajax components that hide complexity. But this solution creates the problem of maintaining the state of the browser across Ajax requests. To solve this problem, his team proposes two possible solutions: using *reminders*, which store state on the browser, and using *page properties,* which store state on the server. He also emphasized that neither of these solutions is perfect.

Reminders are the collection of name-value pairs similar to the View State mechanism in ASP.net, except more granular and embedded in the page by server-side components. Reminders, however, have security implications since they store internal server state (potentially sensitive), requiring the use of encryption.

Page properties, which are name-value pairs specific to a page, are stored in session, are created during the initial page rendering, and are accessible/modifiable during Ajax requests. Page properties have no security issues but include extra overhead for saving properties. The biggest problem with Properties is garbage collection.

To demonstrate the work his team has done, Ousterhout provided trace-driven simulations showing a graph with broken pages per one thousand views in the Y axis and the Least Recently Used list length (per-user) in the X axis. He concluded that managing Web application state is hard, and neither reminders nor page properties are ideal but that garbage collection problems are less serious than security problems and that, overall, page properties are better.

- ***SVC: Selector-based View Composition for Web Frameworks***
  William P. Zeller and Edward W. Felten, Princeton University

William Zeller started his talk by giving an introduction to Selector-based View Composition (SVC) as a new programming style for Web application development. Developing a framework like SVC targeted for supporting both JavaScript and non-JavaScript browsers is worthwhile because there are still some browsers without JavaScript. He showed us how requests travel in a Model View Controller (MVC) architecture pattern from client to the controller and browser first without and then with SVC. Further, he went into details of the SVC-server side API. APIs are used to compose views together, and (CSS) selectors are used to identify the point of composition.

SVC allows developers to write/view/update code only once. He also said that the reasons for using selectors are that they are familiar to developers, are more common in JS frameworks, and are used in the HTML5 API. To give an idea of what SVC looks like, he provided some actual code samples, showing that they implemented nine actions with the possibility of easy extension. He also made clear how non-DOM actions are added to SVC.

Finally, he talked about the alternatives to SVC such as GWT, Cappuccino, and RIS, but none of them supports non-Ajax browsers, which are supported by the SVC model. He also said that extension of SVC could be done in terms of additional language support (Python instead of PHP) and additional client-side libraries. He concluded that SVC provides automatic progressive enhancement and compatibility with older browsers.

Further information about this paper can be found at http://svc.from.bz.

- ***Silo: Exploiting JavaScript and DOM Storage for Faster Page Loads***
  James Mickens, Microsoft Research

James Mickens started by discussing the process of interaction between a client and a server and how large round-trip times (RTT) are harmful and increase page load time. There are two general approaches to decreasing the RTT: (1) reducing Cascading Style Sheets and JavaScript in a page, but nobody will do that because it would decrease the page's fanciness; (2) inlining JavaScript and CSS, but doing so would make the browser cache useless.

As a solution to this problem, he introduced Silo, a system that leverages JavaScript and DOM storage to reduce both the number of HTTP requests and the bandwidth required to construct a page. Mickens evaluated the Silo protocol as having the advantages of being able to fetch an arbitrary number of JSS/CSS in two RTTs, with caching being restored and working on unmodified browsers. Silo exploits four key features: read/write, key+value, asynchronously fetching Web data, and overwriting a page's data.

Mickens said that slow pages cause anger and depression. To avoid this, we either have to reduce the number of objects or inline everything. Both options have their own problems. But Silo is an appropriate solution since it uses JavaScript and DOM storage to aggressively inline HTML, JS, and CSS and uses Cache with DOM storage instead of regular browser cache.

Someone asked if Silo is helpful for all kinds of Web sites, or are there any situations where you lose some functionality by using Silo? Wickens responded that delays may vary depending on how the Web site is designed.

JUNE 24, 1:30 P.M.–3:00 P.M.

Summarized by Thomas Moyer (tmmoyer@cse.psu.edu)

- ***Pixaxe: A Declarative, Client-Focused Web Application Framework***
  Rob King, TippingPoint DVLabs

Rob King presented Pixaxe, a client-focused Web application framework, supporting the model-view-controller (MVC) design pattern. The framework is designed for building Web interfaces with legacy code in mind. Pixaxe is built from several components, with each component being stand-alone. King presented the general structure of an example Pixaxe application and described each of the underlying components and how they work together to form the overall framework.

One of the main features of Pixaxe is that all of the processing is done on the client. The only operation the server is required to support is the ability to serve static files. The views developed for Pixaxe are valid XHTML, meaning the application developer can leverage existing knowledge of XSL transformations and XSLT macros. The logic in the views is written in the form of Jenner expressions. Jenner is one of the three components for Pixaxe. Furthermore, the Jenner component is a superset of the ECMAScript expression language (Esel), the second component of the framework. The final component of the framework is the parser/combinator library, named Kouprey. Kouprey is written in ECMAScript, and runs in all major browsers supporting JavaScript. King finished by highlighting the availability of the code at http://www.deadpixi.com and taking questions.

To illustrate the simplicity of the framework, King presented a complex example that pulled log files from a server and presented them to the client. The client was responsible for all of the rendering and processing, with the server only providing the code and data. The client stores all of the data in the model, which is then used by the view to generate the final output seen by the client.

One audience member wondered how complex an application had to become before Pixaxe was no longer a good framework to choose. King responded that the best target for Pixaxe was a single model with a single view and that more complex applications would probably be suited to other frameworks.

- ***Featherweight Firefox: Formalizing the Core of a Web Browser***
  Aaron Bohannon and Benjamin C. Pierce, University of Pennsylvania

Aaron Bohannon began the talk with a series of questions related to the behavior of browsers. The first question related to properties of the DOM, specifically the ownerDocument property and how the value is set. The second question related to altering the DOM to force script re-execution, noting that it is not possible to re-execute a script once it has been run. The final question discussed how event handlers can obtain access to the window that received the event, noting that the asynchronous nature of event handlers made the answer to this question unclear. Bohannon used these three questions to drive home a simple point: we don't fully understand all of the complex components of today's browsers and how they interact.

Featherweight Firefox is a model of the core functionality of the browser. Bohannon described the larger goal of this body of work to be a formal understanding of the security policies of browsers and how the enforcement happens. A formal model of the core functionality is required before formal security proofs can be presented. The model presented in the talk does not include any security features of the browser, such that the model can serve as a basis for modeling both current and proposed security features.

Bohannon highlighted several of the key features of the model, including the modeling of multiple windows and pages, mutable DOM trees, user inputs, event handlers, cookies, and network operations. Currently, the model does not have support for history, HTTP error codes and redirects, timeouts, or JavaScript and file URL handlers. Some of these features would require modeling the current security features of the browser, and are left out as a result.

The first questioner asked who the target audience of the model was. Bohannon sees the target audience as browser developers and researchers. Browser developers can use the model to gain insight into the browser, and researchers can use the model to develop new security mechanisms for browsers. What larger lessons were learned in doing this modeling? That browser behavior was found to be highly non-deterministic and standardizing the behavior would make things simpler. Why did the authors use small-step semantics? Small-step was chosen since it provides the most detailed information about the browser's internal workings.

- ***DBTaint: Cross-Application Information Flow Tracking via Databases***
  Benjamin Davis and Hao Chen, University of California, Davis

Benjamin Davis presented DBTaint, a system for providing taint tracking in Web applications. Davis began by describing current solutions for taint tracking and how they are insufficient. For example, systemwide tracking typically works at the process level, leading to marking all Web application processes as tainted as soon as they process any user input. The next approach was to examine solutions that perform taint tracking within a single process, such as Perl's taint mode. Davis argued that such systems don't allow for tracking information flow between the various applications that comprise a Web application.

DBTaint is a system that provides information flow tracking between the Web application and the database storing the data. DBTaint relies on the taint tracking systems present in languages like Perl and Ruby, as well as work being done to add taint tracking to Java and PHP. These systems allow the Web application to mark and track information as it enters the application and as it is processed. The database schema for an application is modified slightly to track the taint value of each value in the database. The prototype relies on the composite datatypes provided by PostgreSQL. The database interface is modified to work with these composite datatypes. The current prototype provides support for parametrized queries as well as queries constructed on the fly. The evaluation showed that the overhead was roughly 10%.

Does the system work with queries that are built on-the-fly, and not as parametrized queries? The current system does handle this; Davis provided a quick example showing that the taint tracking would require knowing which parts of the SQL statement were tainted. The second question related to handling implicit flows in the application. Davis responded that the current system only handles explicit flows. Had DBTaint led to the discovery of any flaws in the applications used in the evaluation? They did not discover any new flaws, but discussed several possible uses of DBTaint beyond information flow tracking, such as regression testing.

JUNE 24, 3:30 P.M.–4:30 P.M.

Summarized by Thomas Moyer (tmmoyer@cse.psu.edu)

- ***xJS: Practical XSS Prevention for Web Application Development***
  Elias Athanasopoulos, Vasilis Pappas, Antonis Krithinakis, Spyros Ligouras, and Evangelos P. Markatos, Institute of Computer Science, Foundation for Research and Technology—Hellas; Thomas Karagiannis, Microsoft Research, Cambridge

Elias Athanasopoulos opened the talk with justification for another anti-cross-site scripting (XSS) framework. He argued that each of today's frameworks failed in certain ways, including not being developer-friendly, unacceptable overhead, not being backwards-compatible, and not being DOM-independent. Their solution, xJS, addresses each of these issues and can defeat most XSS attacks. Athanasopoulos then introduced a new type of XSS attack called "return-to-JavaScript" attacks. He gave some examples of these attacks and showed how current techniques failed to prevent them.

xJS is a solution based on instruction-set randomization. The legitimate JavaScript in each page is passed through an isolation operator on the server that encodes the JavaScript. In order to decode the JavaScript on the client, the key is passed to the client. In their implementation, the isolation operator was the XOR operation followed by Base64 encoding. This encoded blob is then inserted where the script code would normally be and the client must run the isolation operator before being able to execute the script. Any injected code will be injected as plaintext and become muddled and not be understood by the JavaScript interpreter.

What JavaScript was protected in their implementation? The current implementation only encoded stand-alone JavaScript files, script tags in static files, and event handler code (e.g., onclick, onload, etc.) in static files. Handling dynamically generated files is future work.

- ***SeerSuite: Developing a Scalable and Reliable Application Framework for Building Digital Libraries by Crawling the Web***
  Pradeep B. Teregowda, Pennsylvania State University; Isaac G. Councill, Google; Juan Pablo Fernández R., Madian Kasbha, Shuyi Zheng, and C. Lee Giles, Pennsylvania State University

Pradeep Teregowda began with a description of SeerSuite and what services currently utilized SeerSuite. The SeerSuite framework is used to build digital libraries in an automated fashion. SeerSuite, unlike other digital libraries, does not rely solely on user submissions. Sites like CiteSeerX and ChemXSeer are currently available as examples of digital libraries built by SeerSuite. Scalability and reliability are two of the main design goals of the SeerSuite framework.

The architecture of SeerSuite was described, with a focus on how the framework automatically crawls the Web and builds a digital library. The Web crawler begins with a set of seed sites that it scans for links and documents. When documents are found, another component within the framework converts the document to plaintext and extracts the necessary information. The document metadata that was extracted is inserted into the database, adding the content to the digital library.

One audience member asked about semantic information provided by third parties. Teregowda responded that federation of services allows third parties to provide services and access available data.

3rd Workshop on Online Social Networks (WOSN 2010)

June 22, 2010
Boston, MA

SESSION 1

Summarized by Saptarshi Ghosh (saptarshi.ghosh@gmail.com)

Balachander Krishnamurthy opened WOSN 2010 by welcoming the attendees and thanking the contributors. He read out welcome messages from Prgram Chairs Bruce Maggs and Andrew Tomkins (who could not be present).

- ***Ghostbusting Facebook: Detecting and Characterizing Phantom Profiles in Online Social Gaming Applications***
  Atif Nazir, Saqib Raza, Chen-Nee Chuah, and Burkhard Schipper, University of California, Davis

Atif Nazir presented techniques to characterize and detect phantom profiles in online social gaming applications. He highlighted the fact that social gaming, platforms for which are provided by several popular OSNs, is now a billion-dollar industry, yet no one has previously studied the impact of social games on the underlying social graph. Highly engaging social games provide a tendency for some gamers to cheat by creating fake (phantom) profiles which contaminate the social graph. Whereas most OSNs presently rely on reports by users and manual inspection to detect phantom profiles, the objective of this work is to characterize phantom profiles in a gaming application and devise a supervised algorithm to detect such profiles.

The authors study the phantom profiles created in a social game played in Facebook, the "Fighters' Club" (FC) game, where two users start a fight and friends of either user can support them. A set of 13 OSN-and-game-biased attributes were tested to identify phantom profiles. However, the social-network-based properties (e.g., number of friends, number of networks joined, cumulative distribution of number of friends) were found to be similar for the genuine profiles as well as phantom profiles; hence statistical classification using such attributes are not effective in detecting phantom profiles, and game-based properties must be used.

Nazir reported observations on some of the game-based attributes: total number of fights defended is more for genuine profiles than for phantom profiles; average number of opponents in picked fights is observed to be higher for phantom users than for genuine users; phantom users instigate/defend fights of smaller duration. However, none of these attributes individually differentiates well between phantom and genuine profiles; so the authors propose to combine these attributes using a support vector machine classifier. Though the proposed technique achieves impressive results, the speaker's opinion is that the method is not yet perfect; it can be used for, at most, flagging of suspected phantom profiles, but not for automatic deletion of those profiles. In particular, one drawback this method suffers from is that the actual ratio of phantom profiles to genuine profiles in the entire population is unknown.

Someone asked whether a new "arms-race" between creators and detectors of phantom profiles could begin in the near future. The speaker's opinion was that only 5% of the phantom profiles are usually active, and out of these 5%, most will not likely make the effort to create phantom profiles if they find that effective methods for detecting phantom profiles are being used by the OSNs. A member of the audience drew attention to a similar paper on the use of structural analysis to detect malicious user profiles in eBay. Another member of the audience asked whether phantom profiles make social games interesting in some way, but Nazir did not think so.

- ***Diffusion Dynamics of Games on Online Social Networks***
  Xiao Wei, Jiang Yang, and Lada A. Adamic, University of Michigan, Ann Arbor; Ricardo Matsumura de Araújo, Federal University of Pelotas, Brazil; Manu Rekhi, LOLapps

In this presentation, Lada A. Adamic discussed propagation of games through invitations sent by users of OSNs to others. In contrast to the previous presentation, this work assumed all profiles to be genuine. The emphasis of this work is to analyze how social games can be designed to propagate efficiently along an OSN, i.e., how more people can be made to play a game. Also, the behavior patterns of gamers in sending out invitations to other users in the OSN were studied. A user is said to be a successful inviter if the user whom he invites actually joins the game. Some of the questions investigated in this work are (1) is there any correlation between how successful one is in inviting friends (to play the game) and how successful the friends are in inviting others? (2) should one invite all one's friends or only a few close friends? (3) can successful inviters be identified based on their profile attributes?

The data of two popular social games were studied using data provided by LOLapps (the company that created the games), both games having millions of Facebook users. They found that most users invite only a few friends; those who invite indiscriminately have a lower yield per invitation. Correspondingly, pacing the invites, sending repeat invitations to the same target, and inviting fewer friends at a time corresponds to successful invitations. On the other hand, factors such as the gender of the inviter or the size of the inviter's friend-network were uncorrelated; however, higher success in invitations is positively correlated with higher engagement (i.e., time spent) in the game of the inviter. Furthermore, dense cliques forming the underlying social network are readily absorbed into the games.

The properties of invitation cascades generated by the invitations sent by users were found to be wide and shallow and to have small-world properties, with many users receiving multiple invitations. It was also observed that the probability of a user joining a game depends on how many invitations they receive but not on how different the inviters are from one another. Adamic pointed out some future directions of research, such as investigating whether the games merely use the underlying social network to propagate or whether the games also cause the social network to grow. Other directions included characterizing large-scale invitation cascades.

Do users who send out more invitations get a higher yield in absolute numbers, even if the per-invite success is lower? Adamic said that this was true, but that games must take into account how much they annoy potential users, which could adversely affect adoption of future games. A member of the audience was of the opinion that most users of OSNs are likely to not use their official profiles for playing such games and sending invites; rather, they would create profiles specifically for the purpose of gaming. Another attendee asked whether the cliques that get absorbed into the game were cliques in the real world. Adamic said this was likely, to which the questioner asked about the profile similarity within cliques. Adamic replied that this would be an interesting measurement, since their work so far had looked at the correlation of profile similarity of inviters with invitation success, not within-clique profile similarity.

- ***Outtweeting the Twitterers—Predicting Information Cascades in Microblogs***
  Wojciech Galuba and Karl Aberer, EPFL; Dipanjan Chakraborty, IBM Research, India; Zoran Despotovic and Wolfgang Kellerer, DOCOMO Euro-Labs

Wojciech Galuba presented this study on the characteristics of information flows in Twitter. The authors report that the distribution of number of tweets received by Twitter users has a median of 552 tweets per day, which indicates an information overload. The motivation of the present work is to improve how information flows in Twitter. The authors look into the diffusion of URLs in Twitter (through tweets) and investigate whether it can predict with what probability a given URL will be mentioned (tweeted) by a given user. Such predictions can provide protection from information overload, e.g., by sorting incoming URLs by probability of re-tweeting.

Data was collected by querying the Twitter search API for the string "http". Power law distributions were observed for user activity, whereas log-normal fits were observed for the degree distribution of Twitter users in the social network. The authors studied information cascade digraphs formed in Twitter, where nodes in the digraphs represent users who mention a given URL and directed edges indicate the flow of a URL from one user to another through a tweet. Two types of information cascades were observed: re-tweet cascades, which are trees in their structure, and follower cascades ,which are directed acyclic graphs. Information cascades were also found to be shallow in their depth.

The authors also proposed a model to predict the probability of a given user mentioning a given URL. The model considers three factors for the prediction: influence of one user over another, external influences on users, and the vitality of the given URL. The input to the model is a time window of tweets, and the outputs are the probability values as stated above; the model optimizes the F-score (the harmonic mean of precision and recall). The proposed model was able to predict more than half of the URL-tweets with only 15% false positives. It is to be noted that the events predicted are only within one hop in the Twitter follower graph from the users that have already mentioned a URL.

Was spam considered in the model? Galuba said that they relied on Twitter itself to remove spammers, and each user in the data was verified later to ascertain whether the user profile still existed. Galuba also said that about one-fifteenth of the tweets gathered in the experiments had tweets in them. Was the presence of private profiles considered in the model? Galuba clarified that it was not considered. What is the relative importance of the three factors considered in the model (stated above)? Galuba's opinion was that the user-to-user influence seems the most important.

SESSION 2

Summarized by Saptarshi Ghosh (saptarshi.ghosh@gmail.com)

- ***Privacy Leakage in Mobile Online Social Networks***
  Balachander Krishnamurthy, AT&T Labs—Research; Craig E. Wills, Worcester Polytechnic Institute

This work, presented by Craig E. Wills, is a continuation of the authors' work presented in the 2nd WOSN in 2009. The objective was to study the potential leakage of personally identifiable information from OSNs to third-party aggregators. The widespread use of mobile devices has resulted in the popularity of mobile-OSNs (mOSN); personal information shared by users with a mOSN that is connected to a traditional OSN is also shared with that OSN, and this may cause additional concerns in leakage of such information. Most mOSNs have a "check-in" mechanism that establishes both a user's presence in the mOSN and the user's current location. Also, mobile devices have a unique device identifier and if this identifier is leaked to a third-party aggregator, it can be linked to a user's real identity. Such threats are usually not present in the case of a traditional OSN being accessed using a browser running on a computer.

The present work examined some mOSNs having roots as a traditional OSN and some OSNs which were not in existence before widespread use of mOSNs. Most of these mOSNs studied have device-specific applications that make the use of the data easier for users (e.g., Apple iPhone applications). However, only a minority of mOSNs provide any privacy settings via the smartphone and mobile applications. Many of the mOSNs studied are connected to the standard OSNs like Twitter, Facebook, and Flickr—comments posted on the mOSNs get reflected on these standard OSNs as well. The OSN identifier of the user posting the comment is often leaked from these mOSNs.

Wills reported that some type of private information, such as user location or device identifier, is being leaked by all the mOSNs that were studied. Such leakages may be quite difficult to prevent by the users. The authors classify information leakages to third parties from mOSNs into two classes—explicit leakage, via Request-URI or POST rquests, and implicit leakage, via the Referer or Cookie HTTP headers. Explicit leakages are difficult to prevent unless done on a per-server basis, while implicit leakages can be prevented by users. Moreover, users nowadays want to share their profile information across several OSNs, and this makes some of the personal information available to third-party aggregators.

What causes such information leakage? Is it careless programming, or is it part of the business model? The co-author of the paper, Balachander Krishnamurthy, said that the authors had contacted some of the mOSNs studied to inform them of the leakage and to seek explanations, but no satisfactory explanation was given; rather, one mOSN replied that such leakages are common. Have the authors tried to correlate what is public information in an OSN (i.e., what the users want to make public) to what was being leaked? It has been verified that some OSNs are actually handing over non-public information to third-party aggregators, allowing the real identity of users to be established or the tracking of Web sites visited by users.

- ***Don't Tread on Me: Moderating Access to OSN Data with SpikeStrip***
  Christo Wilson and Alessandra Sala, University of California, Santa Barbara; Joseph Bonneau, Computer Laboratory, University of Cambridge; Robert Zablit and Ben Y. Zhao, University of California, Santa Barbara

Christo Wilson presented methods to restrict or moderate access to user data in OSNs. User profiles in OSNs include information (such as birthday, location) that can be used for malicious purposes (e.g., by spammers to construct phishing mails). Researchers also collect and analyze large amounts of OSN data. In addition, there are firms which provide features for the very large-scale crawling of OSNs. OSN users are sometimes unable to defend themselves from such information gathering due to complex privacy settings used by some OSNs; most users are also too lazy to use these settings.

Wilson discussed some of the existing technologies to control crawling, but none of these technologies sufficiently controls access to OSN data by today's crawlers. Configuration files (e.g., robots.txt) tell crawlers how to behave, but compliance with the rules stated is voluntary. Requests can be filtered by servers based on HTTP Request Headers, but headers can be modified by crawlers. Servers also use IP-address-based tracking, but NATs and proxies create problems; for example, many genuine users using the same proxy as a malicious crawler can get blocked. Several OSNs do authenticate user accounts and track the number of requests sent by accounts, but the problem in this approach is that the URLs are session-independent; hence a malicious crawler can switch to another user account if one is blocked due to generation of too many requests.

To effectively moderate the crawling of OSN data, the authors propose SpikeStrip—a technique using server-side link encryption by the user's session key and a secret key generated by the server. If a crawler switches to another account, the session key changes and the server will no longer be able to decrypt URLs sent by the crawler using the new session key (since the encryption was done using the previous session key). Thus this method prevents session-switching by crawlers. Also, distributed crawlers will be prevented from sharing URLs, since the session keys will be different for each instance of the crawler. This method also enables strong rate limiting. The authors evaluate their proposed approach using two metrics—the server overhead caused by the method, and the effectiveness of the method in throttling crawlers. The proposed technique was implemented on an Apache server, and resulted in about 7% additional overhead on the server while very successfully throttling crawlers.

A member of the audience pointed out that a link-encryption-based technique has been commercially used by an OSN in Russia since 2007 to moderate access to data. Whereas the authors of the present work assume that a session key is linked to a user account, the aforementioned OSN server sends back new session keys with every request. Someone also commented that many OSN users wish to share links but link encryption makes this difficult. Wilson's opinion was that crawlers and users (people) are usually interested in accessing different sets of URLs: people are very often interested in accessing photos (images), for example, unlike crawlers; SpikeStrip could be used to selectively encrypt the links that crawlers are mostly interested in.

- ***Prediction Promotes Privacy in Dynamic Social Networks***
  Smriti Bhagat, Rutgers University; Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava, AT&T Labs—Research

This work, presented by Smriti Bhagat, discusses ways to anonymize OSN data in order to protect the identity and privacy of the users before the data is published for use by researchers and other third parties. For anonymization, a version of the OSN data must be published which protects the identity of users, yet is usable for analysis (e.g., studying communication patterns). Some commonly used techniques for anonymization are simple anonymization (replacing user names with arbitrary numbers, but easy to break), adding/removing edges to make neighborhoods similar (but this alters the social graph), and node mapping (grouping multiple users to hide the identity of any individual user). However, the focus till now has mostly been on anonymization of a static network, publishing a single anonymized instance of the network. The present work is on the problem of dynamic network anonymization—given a temporal sequence of graphs {G0, G1, . . . , Gt}, the output needs to be a (modified) sequence of graphs such that the likelihood of identifying the node that represents a given user or determining if a pair of users interact is small.

Bhagat clarified that naive approaches for dynamic anonymization—such as individually anonymizing each graph instance in the given sequence, or anonymizing only the first instance (G0) and inserting new edges as they appear—do not work well. The proposed method uses node mapping along with link prediction techniques; two users are grouped together only if there is no path of length two with at most one predicted edge between these two users. In simple terms, if there is a high probability of two users getting linked with each other in future, the proposed approach will keep these users in different groups. The authors defined a new metric called Edge Identification (EI) to measure privacy loss due to anonymization and show significant improvement in EI on using link prediction for social network datasets. The authors also showed that graph sequences anonymized by this approach effectively answer queries with insignificant increase in errors due to grouping with link prediction.

Is the proposed technique dependent on the window length? Bhagat clarified that the window length needs to be such that consecutive samples in the given sequence differ considerably with respect to number of new users and edges introduced in the samples. If certain structures get introduced in the social graph with time, would it be possible to identify the structures in the anonymized sequence? Although such structures might be identified in the anonymized sequence, due to the guarantees of the grouping approach, even if a few users in a group are identified, the privacy of other users in the group may be preserved. However, it is not possible to provide absolute guarantees on anonymization without assuming some attack models and background information.

SESSION 3

Summarized by Christo Wilson (bowlinearl@gmail.com)

- ***A Geometric Model for On-line Social Networks***
Anthony Bonato, Ryerson University; Jeannette Janssen, Dalhousie University; Pawel Pralat, West Virginia University

Anthony Bonato presented work that focuses on creating new graph models specially tailored to social networks. Typical random graph models are targeted to observed properties of random graphs, such as power-law degree scaling, small-world clustering, and local communities. It has also been observed that dynamic random graphs tend to exhibit densification power law (i.e., average degree increases over time) and shrinking distances (i.e., graph diameter decreases over time). The goal of this work is to create a good model for social graphs specifically, since few models are tailored specifically for this purpose. This model should evolve in a "natural" way, so that generative forces in the social network can be understood and used to predict links in the graph.The authors' previous work focused on a deterministic model exploiting transitivity in social relationships, called iterated local transitivity. This model exhibits many desirable properties but fails to generate power-law degree distribution. Hence, the authors moved on to geometric models that map OSNs, which live in social space, to Euclidean space. This process involves isolating user attributes that can be used to define a set of dimensions and then determining user placement in the space by looking at the commonalities between these elements. Once users have been embedded in Euclidean space, a volume can be drawn around them, and all other users inside this volume are linked to the central user.

The authors' current work on this topic involves combining two existing models, random geometric and Protean, to construct a new model called Geometric Protean (GEO-P). The key features of this model involve varying the size of a user's influence sphere based on a ranking function, iteratively removing and adding new nodes, then re-ranking. The benefits of this model are that it generates power-law graphs where average degree, diameter, and density are fully parametrized. The authors also note that the dimensionality of social graphs appears to be equal to log(n), although this has not been proven yet.

Questions from the audience mostly focused on the dimensionality of social graphs. Specifically, is there a way to map between the Euclidean dimensions and specific user attributes? As the authors' work is more theoretical in nature, they were unable to concretely answer these questions, instead saying that it's up to empiricists to solve those types of questions.

- ***Distance Matters: Geo-social Metrics for Online Social Networks***
Salvatore Scellato and Cecilia Mascolo, Computer Laboratory, University of Cambridge; Mirco Musolesi, University of St. Andrews; Vito Latora, Università di Catania and INFN

Salvatore Scellato presented a first stab at answering the question, what is the role of geography in the structure of OSNs, and how does it affect information dissemination? To answer this question, the authors gathered data from several social networks and mapped people based on their stated location. They could then examine users' friends in geographic space, to examine whether friendships occur between users who are close geographically, and classify users based on their preference for short- or long-distance relationships. The authors also aim to examine information dissemination with respect to user geography and how this may impact social application performance.

The authors constructed a new social graph type called a Geographic Social Network, where all users are placed in a 2D space, and social edges between users are weighted based on geographic distance. This leads to the invention of two new graph metrics: node locality (i.e., how close in space my neighbors are) and geographic clustering coefficient. These metrics are constructed to highlight nodes and groups of nodes that exhibit tight geographic clustering, and they include a scaling parameter to account for relative differences in geography (i.e., comparing user distances in a tight urban metropolis is not equivalent to the same task in a widespread, rural community).

The authors crawled Brightkite completely, and crawled snowball samples of FourSquare, LiveJournal, and Twitter. They then used the Yahoo Geocoding API to obtain GPS coordinates for users who gave their location via textual descriptions. The authors observe that users in location-based OSNs such as FourSquare and Brightkite have friends links geographically closer (1,200 km on average) than more general OSNs (Twitter was 5,000 km on average). On FourSquare, in particular, 60% of links are less than 1 km in length, showing that users tend to live very close to their OSN friends. There even exist a small number of hyperlocal users on FourSquare and Brightkite that have geographic clustering coefficients of 1, meaning they all live in exactly the same place.

Questions from the audience fell into two areas. The first concerned bias in the data due to people not uniformly tagging data (on Twitter) and people having geographic biases (i.e., users tend to follow people in their own country/culture). The authors agreed that these were difficult problems to deal with. The second area concerned time-based dynamics: does geography drive new friendships or do existing friends tend to migrate close together? Unfortunately, no location-based OSN reports the times that new friendship links form, and thus this can't be examined as of yet.

- ***Orion: Shortest Path Estimation for Large Social Graphs***
  Xiaohan Zhao, Alessandra Sala, Christo Wilson, Haitao Zheng, and Ben Y. Zhao, University of California, Santa Barbara

Xiaohan Zhao presented this work, focused on the creation of a new system that can perform node distance computations on massive social graphs in real time. There are many interesting and novel social applications that are enabled by today's huge social graphs. Examples include influence maximization for marketing campaigns, and social search. However, both of these examples, and many others, are predicated on being able to compute social distances in real time. This is impossible with today's graph distance algorithms, which usually run in O(VE) time.

The goal of this work is to design a new system that can scalably provide social distance information in O(1) time to enable real-time applications. Meeting this goal requires that some caveats be managed: specifically, the preprocessing time necessary to bootstrap the system, and the accuracy of generated results. The authors proposed to meet these goals through the construction of a novel Graph Coordinate System called Orion that embeds users from the social graph into a Euclidean space.

There are two approaches to embedding social graphs into Euclidean space. The first approach, modeling a physical system, requires too much preprocessing to be practical. The second approach, using landmarks, works better in practice. High-degree users are the best choice for the landmarks, and all other users can be placed by performing a constrained number of full-breadth first searches of the graph. The authors use an incremental global simplex downhill heuristic to ensure that preprocessing time for Orion remains linear on the size of the graph.

The authors evaluated Orion using four datasets crawled from Facebook regional networks. They show that a 300k node graph can be preprocessed in about two hours (an acceptable, one-time cost). They also showed that Orion calculates node distance measurements seven orders of magnitude faster than BFS (Breadth-first search), with only 0.2 relative error in distance calculations. The authors also show that Orion is suitable for calculating higher-order graph metrics, such as average path length (error > 0.3).

Questions from the audience were varied. One person wanted to know about recovering actual, hop-by-hop shortest paths from Orion rather than just overall distances, which the system doesn't currently support. Another question concerned dynamic graphs, which the presenter agreed was an interesting problem and the likely future focus of this work.

- ***The Effects of Restrictions on Number of Connections in OSNs: A Case-Study on Twitter***
  Saptarshi Ghosh, Gautam Korlam, and Niloy Ganguly, IIT Kharagpur, India

Saptarshi Ghosh discussed the effects of the limits imposed by many OSNs on the number of social friends each user may have. These limits are imposed mainly to control spam and to prevent undue strain on the OSN's infrastructure in supporting user-to-all-friends communication. In contrast to fixed limits in most OSNs, Twitter has imposed a "soft" limit on the number of people a user can follow (i.e., users' outgoing links are limited based on how many incoming links they have). While people often complain about these limits, research results suggest that the limits only actually affect hyper-active users. However, these limits do have ramifications on how the OSN evolves. This work seeks to probe those ramifications.

On Twitter, there is the concept of "follow spam," i.e., users who do nothing but follow others just to draw attention to themselves for nefarious reasons. This causes Twitter to limit the number of people one can follow to 2000, beyond which one can follow more only if one has a commensurate number of followers. The actual algorithm for this is secret (security through obscurity), but people have independently reverse-engineered two plausible algorithms. The authors performed BFS crawls of 1 million Twitter users in November 2009 (about one year after the limit was imposed) to observe users' in-degree and out-degree distributions. They observed that only 6.6% of users follow more than 2000 people, and almost all users have out-degree less than 1.1 times their in-degree; this result agrees with the conjectures regarding Twitter's secret follow-control algorithm. The authors also note that the 2000 follow-limit on out-degree causes a pronounced spike around 2000 in the out-degree distribution, as there are many users who exhibit the "follow spam" paradigm of following many but not being followed in return. The authors discuss a new social network growth model based on preferential attachment, which incorporates restrictions on user-degree. The proposed model can be used to analyze effects of different forms of restrictions on the topological properties of social networks as well as to design restrictions of varying rigidity.

Questions from the audience focused on potential deficiencies in the authors' crawling methodology. Specifically, BFS (and repeated BFSes from distinct starting points) are likely to bias gathered data towards high-degree nodes. The speaker agreed with this observation and said that they plan to collect unbiased samples of the Twitter social network, using methods such as Metropolis-Hasting random walk.

SESSION 4

Summarized by Christo Wilson (bowlinearl@gmail.com)

- ***Voice of the Customers: Mining Online Customer Reviews for Product Feature-based Ranking***
  Kunpeng Zhang, Ramanathan Narayanan, and Alok Choudhary, Northwestern University

This work, presented by Kunpeng Zhang, focuses on aggregating and automatically distilling textual user reviews of products online into concise and comparable feature-based rankings. Online shopping continues to increase its market share, and many of the top sites encourage users to review products they have purchased. User reviews are very important in influencing the decisions of new buyers, but it can be time-consuming to read all available reviews, especially since different people have different requirements when evaluating products. In this work, the authors propose using comparative and subjective evaluative sentences to construct graphs of product features which are weighted according to the products' relative strengths and weaknesses.

The authors look for two different types of statements when parsing user reviews: comparative sentences and subjective sentences. The first type of statement compares the relative merits of one product as compared to one or more other products in the same category. These statements form the edges of the product graph, with the weight being the positive or negative customer sentiment. The second type of statements are simply positive or negative declarations about one specific product. These statements are used to weight nodes of the product graph. The authors leverage a corpus of 2000 positive keywords, 4500 negative words, and 30 words of negation when performing their textual analysis. The authors use three techniques (keyword matching, part-of-speech analysis, and predefined patterns) to identify comparative sentences, at which point they apply refinements to ensure that statements conform to the requirements necessary for further analysis.

The authors targeted this work towards evaluation of cameras and televisions using data crawled from Amazon. Each product is present in multiple different weighted graphs, where each graph represents a particular product feature, i.e., a camera might be in lens, flash, and battery-life graphs. Once these graphs are constructed the authors leverage a novel ranking algorithm, pRank, to calculate the importance of each product n for each feature f. This allows them to isolate the most important feature in terms of the overall product quality (importance function). The statistics of feature sentences also tell what features were most talked about (relative feature fraction). pRank compares favorably with reviews generated by experts in each of the two measured product categories, as opposed to simple metrics like Customer Star Rating and SalesRank, which fail to correlate.

Questions from the audience focused on extensions to this work. Specifically, could the weight of individual reviews be added to the system, say to up-moderate knowledgeable experts and down-moderate trolls? The author stated that they are currently working on using data on reviewer fidelity from Amazon to extend their techniques to incorporate this additional information.

- ***I rate you. You rate me. Should we do so publicly?***
  Chun-Yuen Teng, Debra Lauterbach, and Lada A. Adamic, University of Michigan

Chun-Yuen Teng presented this paper, which focused on quantifying the biasing effects of public visibility of user reviews in online systems. Many Web 2.0 sites use recommendation/reputation systems to help users evaluate products and each other, but are these social ratings honest? Specifically, does the design of these sites, and whether review sources are public or private, affect the quality of resultant reviews? To answer these questions, the authors crawled reviewer data from Amazon and Epinions and obtained the entire (anonymized) review database from CouchSurfing.

The most immediate effects of site design observed by the authors were on review reciprocity. Positive reviews left in public places were likely to generate positive reviews for the initial reviewer as well. However, making reviews/reviewers public (and thus leaving open the threat of public retaliation) also has the effect of causing few overall negative reviews to be left at all. For example, the ratio of positive to negative reviews on CouchSurfing is 2500:1, and 70% of vouches are reciprocated. The same is true on Epinions, where public reviews are likely to be positive, while anonymous reviews are more negative. Real name (as opposed to anonymous) reviewers also tend to be more thorough, as these reviewers leave longer comments on Amazon.

Other social factors seem to affect reviews as well. Men tend to be more egalitarian with their ratings, while women tend to rate other women more highly on friendship and trust. Age, geographical separation between reviewers, and cultural homophily also all influence review scores between people. Overall, the authors' takeaway advice was not to take online reviews at face value; there are many hidden social effects that warp and skew reviews.

Questions from the audience focused on additional sources of bias in the review data. The authors were asked if temporal dynamics affect reviews (i.e., incidental mood swings) and about review reproducibility (i.e., if one person reviews the same thing twice, will the reviews be consistent?). The authors acknowledged that these were difficult questions that were hard to quantify with the available data.

- ***Measuring Online Service Availability Using Twitter***
  Marti Motoyama, University of California, San Diego; Brendan Meeder, Carnegie Mellon University; Kirill Levchenko, Geoffrey M. Voelker, and Stefan Savage, University of California, San Diego

Marti Motoyama explained the potential for using Twitter to automatically detect outages affecting online service providers. Twitter is ideal for this sort of study because of the social structure of the network, the real-time nature of its messages, and the rapid dissemination of information throughout the community. People also provide a varying set of real-world vantage points on service conditions and may notice localized and transient failures that other automated up-time measurement tools (e.g., ping) may miss. In effect, this enables researchers to treat the OSN as a human-sensor network. The importance of measuring service outages will only increase as more services move to the cloud; hence the need for this work. As a motivating example, the authors note that it took several traditional news media outlets over one hour to publicize news of the Gmail outage that occurred on September 1, 2009, whereas users on Twitter started messaging about the outage within minutes of its occurrence.

The authors crawled Twitter for data and performed textual analysis to look for outage events. They focused on the bi-gram "is down" as the primary indicator of service outages, with the hashtag "#fail" being a secondary indicator (mostly because it is a commonly used tag, hence leading to a noisy signal). They examine the two words surrounding service names during known outages and verify that the word preceding the phrase "is down" effectively denotes the name of the affected service. In order to filter out noise (e.g., people tweeting about non-existent outages), the authors use an exponentially weighted moving average to determine the incidental volume of tweets about each service. The stream of tweets is divided into five-minute intervals, and any service that exceeds a moving threshold for two consecutive intervals generates an outage alarm. In order to tune the parameters, the authors created a validation set by looking for outages through search engines and manually inspecting maintenance blogs. Subsequently, they tried different combinations of parameters and picked the ones that produced the lowest F-scores.

In order to validate their methodology, the authors looked at eight service disruptions that occurred in 2009, for which they were also able to find an article or blogpost approximating the start time of the outage. Their automated system managed to detect all eight test events, although the time to detect the event sometimes lagged 10 to 50 minutes behind the actual start of the outage. They noted that news articles may be imprecise in their assessment of the actual outage start times. The authors observed that detection times could be improved by expanding the set of warning bi-grams, e.g., to detect tweets such as "anyone having problems with Gmail" instead of just "Gmail is down."

Running the automated analysis on the entire corpus of crawled tweets resulted in the detection of 894 outage events affecting 245 entities. The authors determined that of these 245 entities, 59 were false positives, mostly associated with sporting events (where the phrase "is down" makes sense). Of the top 50 experimentally discovered outages (as determined by the volume of tweets containing "is down"), the authors manually verified 48. Interestingly, nine of these outages were of Twitter itself. One possible explanation may be that third-party applications queue tweets during the outages and push the updates out when Twitter comes back online. Out of 50 random outages sampled by the authors, 35 were manually verifiable. These included outages of major services such as World of Warcraft and Netflix.

One person was concerned about using the system to measure public sentiment, which Motoyama agreed would be possible. Another question was about attacks, i.e., would it be possible to spam/astroturf Twitter and fool the sensor? This seems unlikely, given the large number of users on Twitter, but is not impossible. Lastly, there was a question about using tweets that are geo-tagged to further refine the locations of failures. The author thought this would be a very valuable feature, but presently this isn't feasible given the small percentage of tweets that are geo-tagged.

2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10)

June 22, 2010
Boston, MA

Foreword by Alva L. Couch (couch@eecs.tufts.edu)

For those of us who attended both HotCloud '09 and '10, there has already been a transformation. In 2009, the workshop was dominated by definitions and struggles, including carefully defining the kinds of clouds and struggling with the oxymoron we call "cloud security." In 2010, surprisingly, the community has made some peace with the invariant properties of clouds, and the workshop was overwhelmingly solution-centered, with many small problems addressed and at least a hint that the grand-challenge problems of cloud privacy, security, isolation, and predictability may have technical solutions. More profoundly, ideas have matured, and it is reasonable to expect that many of the papers presented in 2010 will appear shortly in full paper venues. As scary as cloud security and privacy issues may be in isolation, clouds still have the potential to transformationally improve security and privacy of data for many small enterprises, whose non-cloud security practices are currently inadequate. In short, while 2009 painted a fairly bleak picture with rather stormy clouds, 2010 looked forward to a relatively sunny future, with a few cirrus clouds unobtrusively floating high in the sky.

PERFORMANCE AND POWER

Summarized by Joshua Reich (reich@cs.columbia.edu)

- ***Seawall: Performance Isolation for Cloud Datacenter Networks***
  Alan Shieh, Cornell University and Microsoft Research; Srikanth Kandula, Albert Greenberg, and Changhoon Kim, Microsoft Research

Currently, applications can't guarantee performance after migrating to the cloud. Globally shared resources, such as network capacity share, are probably the most difficult to control, and this is when other applications running in the cloud are well behaved. When malicious users are present (e.g., DNS attacks from cloud tenants) this problem becomes even worse.

Alan Shieh presented Seawall, whose goals are to isolate tenants, controlling their share of the network, while still utilizing network capacity effectively. Additional constraints are that tenant code is untrusted and system churn should be minimized as VMs come and leave. Possible solutions include counter-based flow level AC (can't handle sufficient number of flows), QCN (not yet standard, requires network upgrades, solves a somewhat different problem), and end-to-end bandwidth restrictions (easy to subvert).

Seawall works by having all traffic flow through a "Seawall port" on the hypervisor. It requires no central controller (each host runs its own rate controller); is enforced by "Seawall ports," sitting on the packet forwarding path (one per source-dest pair); and detects congestion on src-dest paths, using direct feedback, AIMD.

Practically, this approach needs to keep overhead low—in particular, the main bottleneck is the Seawall port, which interacts with every packet. Encapsulation might be one approach, but this adds lots of overhead and breaks header format optimizations in switches and hash-based load balancing. The authors suggest the possibility of "bit stealing," using spare bits from existing headers—e.g., IP-ID field and some other bits from the constant part of the Seq#—and although this seems like a bit of a kludge, they claim it produces good performance in their simulator validation experiments.

Someone asked about the Amazon security group, and Shieh answered that Seawall can add that directly to the Seawall port. Another person suggested speculative network capacity use (e.g., spot pricing). Shieh noted that they haven't addressed this directly, although it doesn't appear that this would be a difficult extension. Someone asked about the CPU time required for Seawall ports, and Shieh said they spent a lot of time optimizing the code, but 40% of the core cycles are still needed to implement this. Centralized rate limiting? They have considered other distributed feedback loops but believe centralized rate limiting won't scale.

- ***Performance Profiling in a Virtualized Environment***
  Jiaqing Du, EPFL, Switzerland; Nipun Sehrawat, IIT Guwahati, India; Willy Zwaenepoel, EPFL, Switzerland

Jiaqing Du said that clouds are built on lots of different hardware/software configurations, which provides significant opportunities for performance profiling/tuning in the virtualized environment. The problem is that, currently, profilers need to interact with underlying hardware quite a bit, and doing this is significantly more difficult in a virtualized environment. XenOProf is the only existing VM profiler, and it only works for Xen and requires admin access. Profiling in the guest can be done but doesn't tell one much about how the system is truly performing.

At a very high level most profilers work by keeping track of event counters and interrupts. The authors propose that these need to be exposed to guests through the standard PMU interface. Of course, one still needs to determine whether the VM should track that as CPU-switched—only in-guest execution is accounted to the guest—or domain switched, which includes other tenants on the same machine.

The authors implemented this method for KVM, using Intel VT extensions. Their experiment consisted of pushing packets to a Linux guest, running OProfile (in the guest), and monitoring instruction retirements.

How long will it take manufacturers to provide hardware support for virtualization? Du said they believe it will take a while, since there are many different hardware/software combinations. Thoughts about instrumentation-based profiling? These should work directly in the guest without the need for virtualization-specific techniques. How does this differ from XenOProf? Is there a more generic methodology? XenOProf does system-wide profiling and requires admin access. We've provided more options and don't require admin access.

- ***The Case for Energy-Oriented Partial Desktop Migration***
  Nilton Bila and Eyal de Lara, University of Toronto; Matti Hiltunen, Kaustubh Joshi, and H. Andrés Lagar-Cavilla, AT&T Labs Research; M. Satyanarayanan, Carnegie Mellon University

Currently PCs waste significant amounts of power while idling. Eyal de Lara said that sleep modes aren't used because users and IT expect always-on semantics, such as background apps and remote access. The authors aim to provide always-on semantics with the benefits of sleep mode. Potential approaches include migrating a VM to a consolidation server (like LiteGreen), but the problem is that VMs are very large and lots of memory will be needed on the consolidation server. Instead, the authors suggest just migrating the working set on that machine using a page-fault-based method.

When a machine goes to sleep a small VM is started on the consolidation server. Every time some memory access is attempted for memory not already present on the consolidation server, the server wakes the client, grabs those pages, and proceeds. The client falls back asleep shortly thereafter. The authors claim this allows for many images on the same server and relatively little network overhead.

However, in order for this to work well, sleep time needs to be long enough and the size of the memory footprint needs to be sufficiently small. The authors have conducted a feasibility study for their proposed methodology, using Snowflock. They have not actually implemented any of the migration code. They examined four workloads: login, email (every 10 minutes), IM client (messages every couple of minutes), and a multitasking workload (PDF, email, spreadsheet, file-browser, IM) and track the resulting page faults. Since they aren't prefetching (although an actual implementation would almost certainly do so), some of their results aren't always very good. However, their overnight test looks like things work nicely in low-usage settings.

The authors estimate a consolidation ratio of 9:1: that is, one can serve nine PCs on one PC, so roughly an order of magnitude more PCs could be consolidated onto a high-quality server. The remaining challenges include device methods that will avoid some of this power-cycling, including proactive methods, CAMs, and addressing policy questions as to when to migrate VMs back and forth.

What about disk accesses? De Lara replied that this requirement is an order of magnitude less than memory. Why do people need machines overnight? They don't, but IT likes these machines to be awake. We believe we leverage smaller sleep times. Have you considered partial wakeup modes? That would require massive stack changes. Wake-on-LAN is a very simple version of this.

- ***Energy Efficiency of Mobile Clients in Cloud Computing***
  Antti P. Miettinen and Jukka K. Nurminen, Nokia Research Center

Antti P. Miettinen pointed out that one of the major problems for mobile devices using cloud applications is battery life. One could do almost all of the processing locally (processor intensive) or do the processing remotely, essentially using the phone as a thin-client (radio intensive). The authors sought to understand the trade-off between computing and communication in terms of mobile phone power use.

Different cores have different energy profiles but, interestingly, mobiles actually use the least efficient cores. This is because mobile phone cores are selected for high peak performance on single-threaded applications. The straightforward solution is to leverage dynamic voltage and frequency scaling (DVFS). Essentially this means using cores that support lower power draw at decreased clock speeds.

With respect to radio power draw, the authors find that 3G radios are much more efficient at high bit rates, while WiFi is more forgiving of lower-bit-rate traffic. Consequently, particularly for 3G traffic, smooth traffic is handled with less power efficiency than bursty traffic. Moreover, if the core is set to a lower clock cycle (power saving on the CPU) while doing data transfer over the network, more power may be used overall, since the data rate of the 3G card will be slowed down accordingly (drawing more radio power).

As a very rough rule, the authors observed 1000 cycles core/one byte of radio parity. Of course, everything depends on the particulars. With a PDF viewer, the authors show power can be saved by running remotely on their hardware. There are lots of challenges: when to transition between thin and thick client operation on mobiles, tools for managing these transitions, energy-aware middleware, energy optimized protocols for thick clients, etc.

How did you do the PDF viewer offload? Miettinen replied that they ran evince on the server and exported x11, but this isn't an optimal thin client protocol, so they could do better (e.g., crawling the PDF is a big mess using x11). Someone else wondered if they had considered more complex apps (e.g., OCR, image processing, auto-translate). They hope that people will come up with these so they can study them. Current apps more or less prune out the computation.

ECONOMICS AND PRICING

Summarized by Alva L. Couch (couch@eecs.tufts.edu)

- ***CloudCmp: Shopping for a Cloud Made Easy***
  Ang Li and Xiaowei Yang, Duke University; Srikanth Kandula and Ming Zhang, Microsoft Research

Cloud computing involves difficult business choices. Cloud providers describe their services in incommensurate units, and it can be difficult to compare one service against another for a potential use. CloudCmp attempts to aid in this decision by combining service benchmarks with load profiles to create predictions of cloud performance without deployment. Speaker Ang Li concentrated on the first part: how to collect and utilize accurate benchmarks. This includes three kinds of measurement: storage and retrieval time, computation time, and network latency. Network latency was determined by measuring the time to query distributed data centers via PlanetLab. Studies of three unnamed cloud services indicated that different services have different performance strengths: one excelled at quick storage and retrieval, while another excelled at computation. The audience was concerned about why the three services were anonymous; Li responded that all three service agreements include prohibitions of reverse engineering, but that in one, disclosure of performance data was prohibited. Someone was also concerned that the measurements were made over short time periods and did not account for changes in background load.

- ***Distributed Systems Meet Economics: Pricing in the Cloud***
  Hongyi Wang, Microsoft Research Asia; Qingfeng Jing, Shanghai Jiao Tong University; Rishan Chen, Peking University; Bingsheng He, Zhengping Qian, and Lidong Zhou, Microsoft Research Asia

Pricing in the cloud is complex and—like pricing on first-generation timesharing servers—can vary with load. Pricing is fair if there is a balance between customer satisfaction and provider satisfaction. Hongyi Wang utilized 1/cost as a customer satisfaction index, and profit/cost (return on investment, or ROI) as a provider satisfaction index. By running instances of Postmark (storage-intensive), Parsec (compute-intensive), and Hadoop (communication-intensive) tasks on EC2, he depicted both fairness and variation in fairness due to load. He concludes that optimality points for customer and provider are often quite different: for Postmark, two VMs maximized customer satisfaction, while four VMs maximized provider satisfaction. Not surprisingly, satisfaction indices also varied with load, which he interprets as a kind of unfairness.

Someone questioned whether this kind of analysis will lead to fairer pricing from providers.

- ***See Spot Run: Using Spot Instances for MapReduce Workflows***
  Navraj Chohan, University of California, Santa Barbara; Claris Castillo, Mike Spreitzer, Malgorzata Steinder, and Asser Tantawi, IBM Watson Research; Chandra Krintz, University of California, Santa Barbara

A "spot instance" in EC2 is a server instance that is activated when the current (volatile) market price for service exceeds a predetermined (constant) customer-specified bid value. Spot instances are thus transient, are started when the market price lowers below the bid, and are terminated when the market price increases above the bid. Market prices are determined based upon supply and demand. Because of their transient nature, spot instances are cheaper to use than "premium" instances that do not go online and offline based upon demand.

Navraj Chohan and his coauthors studied the use of spot instances during three kinds of Hadoop computations: word-count, sorting, and Monte-Carlo computation of π. Data for the computation was kept on ("premium") Hadoop HDFS nodes that were always running, while spot instances were allocated as extra workers. Even though spot instances are cheaper to use, using spot instances can be more expensive (and slower) than using only "Premium" instances, because of the time required to discover and correct the effects of terminated "spot" worker nodes.

Someone asked whether running spot instances with a high bid is equivalent to having a premium node, and whether market prices are periodic. Chohan responded that market prices show no predictable periodicity (and discussion groups on the Web report that spot instances can be volatile even if the bid price is always high enough, because Amazon can terminate them for reasons other than market price fluctuations, e.g., for maintenance).

- ***Disaster Recovery as a Cloud Service: Economic Benefits & Deployment Challenges***
  Timothy Wood and Emmanuel Cecchet, University of Massachusetts Amherst; K.K. Ramakrishnan, AT&T Labs—Research; Prashant Shenoy, University of Massachusetts Amherst; Jacobus van der Merwe, AT&T Labs—Research; Arun Venkataramani, University of Massachusetts Amherst

Disasters happen even to clouds. Tim Wood and his team considered how clouds should handle disastrous conditions. Business objectives for disaster handling include "recovery point objectives" (how much data can be lost), as well as "recovery time objectives" (how long it takes to resume processing of requests). Wood made some cost comparisons between cloud-based recovery and traditional co-located recovery. He estimated that recovery infrastructure for a RUBiS site with four servers and 99% uptime would cost $10,373 per year with co-located physical recovery servers and $1,562 if recovery servers are instead located on EC2.

As a second example, a data warehouse allows one to balance the cost of cloud recovery against recovery-point optimization, because one only pays for the servers when they are running to store recovery points. Open questions include how the provider can maximize profit (due to a similar mismatch between customer and provider satisfaction to that presented in another paper above), how many resources the customer should commit to disaster recovery, how to deal with correlated failures (e.g., regional outages), and how one should resume regular processing after a disaster is mitigated.

Someone asked whether the disaster recovery model included regional distribution of disaster recovery resources, and Wood responded that it did not. Would the mechanism continue to work if "everyone" did their disaster recovery this way? More resources would have to be available in the cloud for this to be practical.

- ***CiteSeer^X: A Cloud Perspective***
  Pradeep B. Teregowda, Bhuvan Urgaonkar, and C. Lee Giles, Pennsylvania State University

The Web site CiteSeer^X is supported by 22 physical servers, stores greater than 1.6 million documents and greater than 30 million citations, and responds to about 2 million hits a day. It includes a Web crawler, as well as software components for document conversion and ingestion, data storage, query response, and maintenance services. Pradeep Teregowda discussed the options for moving this site into the cloud, either in whole or in part. He considered two options, including Amazon EC2 and Google AppEngine. Moving an application to the cloud requires both data refactoring and code refactoring.

Data refactoring seems to be cost-effective, while the practicality of code refactoring remains unknown. In particular, the cost of moving code to AppEngine remains unknown. He concludes that for now, hosting static content and query response in the cloud is cost-effective, while the practicality of moving other software components is as yet unknown. Re-factoring the main software components may make cloud hosting of the entire application cost-effective.

Someone asked how CiteSeer^X will handle load changes, and Teregowda answered that, for now, the plan is for the cloud to handle only peak loads, while physical infrastructure will continue to handle non-peak loads.

NEW PROGRAMMING MODELS AND USAGE SCENARIOS

Summarized by Malte Schwarzkopf (malte.schwarzkopf@cl.cam.ac.uk)

- ***Spark: Cluster Computing with Working Sets***
  Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica, University of California, Berkeley

Spark is a framework for cluster computing optimized for iterative applications with fixed working sets, with datasets reused across multiple parallel operations. Conceptually, Spark builds upon frameworks such as MapReduce and Dryad, but improves their limited performance on iterative jobs such as those found in many machine learning and data-mining applications.

Spark is written in Scala and integrates with the Scala language for easy programmability. It defines the notion of a resilient distributed dataset (RDD), which can be obtained from the Hadoop Distributed File System or parallelized arrays. RDDs can be transformed using map and filter functions and are cached across operations. The main difference from previous models is that RDDs are defined to be persistent even across iterations within the driver program. Spark supports parallel operations of reduce, collect, and foreach on RDDs and supplies shared variables (accumulators and broadcast variables) to allow for data to be communicated. For example, consider the use case of log mining: the logs would be RDDs, and there would be a driver node (running the main driver program) as well as a set of workers. The driver creates a distributed dataset, executes some filter operations on it, and then calls a cache() function. At this point, the RDDs exist as lazy objects on the cluster. On the first iteration of the algorithm, they are then disseminated to and cached on worker nodes. Further iterations will then hit the version cached in memory on the worker nodes, thus speeding up the information.

RDDs maintain fault-tolerant semantics as each RDD maintains lineage information that can be used to reconstruct lost partitions (RDDs). This is done by going back to the beginning and reapplying the filter and map operations (this restricts the programming model to disallow certain side effects, but provides fault tolerance).

In the evaluation, the authors compared Spark to the Hadoop open-source implementation of MapReduce and found that Spark outperforms Hadoop by a factor of up to 20 on the second and subsequent iterations, while running about 50% slower on the first iteration (Hadoop: ~127 seconds per iteration, Spark: 174 seconds for first iteration, 6 seconds for further iterations). There was no time for questions, due to a demo that interactively reproduced the above results.

- ***Turning Down the LAMP: Software Specialisation for the Cloud***
  Anil Madhavapeddy, University of Cambridge; Richard Mortier, University of Nottingham; Ripduman Sohan; University of Cambridge; Thomas Gazagnaire, Citrix Systems R&D; Steven Hand, University of Cambridge; Tim Deegan, Citrix Systems R&D; Derek McAuley, University of Nottingham; Jon Crowcroft, University of Cambridge

Anil Madhavapeddy described the implementation of a "fat-free" operating system that reduces the level of dynamic abstraction in computer systems. The authors point out that typical application stacks of modern operating systems consist of a large number of abstraction layers on top of one another, with dynamic languages adding another layer (language runtime) and virtualization another (hypervisor). Although undeniably convenient, these layers exist mainly for compatibility reasons and ensure legacy compliance. For example, as was pointed out, the POSIX layer was designed to run compiled and linked C applications, but not dynamic languages with garbage collection, such as Java or .NET code. Furthermore, the number of abstraction layers means that there is a greater potential for security vulnerabilities.

Madhavapeddy contended that the emergence of virtualization has produced a unique opportunity to remove as many compatibility abstractions as possible, as hypervisors provide a "stable" virtual hardware platform to develop against. This facilitates not only the removal of unnecessary abstractions, but also means that small research efforts that would previously have been doomed to fail due to the complexity of operating systems implementation can succeed. In addition to moving towards writing applications on top of the "bare metal" provided by virtualization architectures, the authors also propose to use the opportunity to introduce a strong and statically typed implementation language in order to avoid security vulnerabilities and improve code quality. Their language of choice is OCaml, which has strong static typing, is extensible, and has a simple runtime system introducing minimal overheads.

As a motivating benchmark, the authors reimplemented SSH and DNS in OCaml, and the measured performance is on par with the best, optimized existing implementations (OpenSSH and Bind, respectively). When turning on memoization in their DNS implementation, the authors even managed to reach an order of magnitude performance improvement over Bind. Coming from these motivating measurements, they have started implementing MirageOS, a new operating system that embraces the philosophy outlined above. MirageOS features zero-copy I/O and gets rid of multicore concurrency issues by only ever running one application. Application-level concurrency is provided by multiple VMs running on the same host system and communicating using message passing.

The authors evaluated an early alpha version of MirageOS in terms of SQL performance and memory usage and found that the performance is improved dramatically compared to a Linux-based setup, while the memory footprint remains the same. In further work, the authors are planning to investigate the use of MirageOS for highly specialized Web servers that can run at very high performance while using less resources than current setups. There was no time for questions.

- ***Scripting the Cloud with Skywriting***
  Derek G. Murray and Steven Hand, University of Cambridge Computer Laboratory

Derek Murray introduced Skywriting, a programming language for cloud computing. Existing frameworks such as MapReduce and Dryad have pioneered a managed approach in distributed computing, freeing programmers from having to concern themselves with the low-level details of message passing, synchronization, and fault tolerance. However, in doing so, they restrict the programming model to one that corresponds to a map-reduce paradigm (MapReduce) or a finite, static DAG (Dryad) and thus have difficulties modeling certain data flows. For example, unbounded iteration and recursive computation are impossible to express in either in any way other than submitting a series of jobs. Ideally, as the authors assert, one would like a truly universal programming model for cloud computing, allowing any Turing-complete program to be expressed.

With Skywriting, the authors have designed a programming language that can express any Turing-complete program. Its syntax is similar to JavaScript, but it is capable of expressing functional constructs such as lambdas. It is an interpreted coordination language and can delegate the actual computation to external high-performance code. Contrary to MapReduce and Dryad, it supports spawning tasks dynamically (using the spawn() primitive), as well as expressing data-dependencies through futures that can be dereferenced in the style of C pointers. In this way, Skywriting can express data-dependent control flow such as unbounded iteration to convergence.

The authors evaluated Skywriting by performing a microbenchmark that tested the job creation overhead and found that it performs much better than Hadoop, which incurs up to 30 seconds of overhead (Skywriting: ~2 seconds). In order to evaluate Skywriting on real workloads, the authors implemented the Smith-Waterman string matching algorithm in Skywriting and ran it on a variety of cluster configurations. They found that the best performance for two strings of length 100,000 was achieved with 400 tasks on 20 workers with a speedup factor of about 2.5x compared to linear, but also that the system scales to cluster sizes of hundreds of nodes.

Anil Madhavapeddy asked whether Skywriting has an issue that needs fixing and how one would hook it into an existing cluster. For the former, Murray said that a possible weakness is that Skywriting does not natively support efficient data motion for MapReduce-type workflows (although it does support MapReduce in principle) and that it is a research prototype and much of the code is unoptimized (e.g., the task dispatch queue is single-threaded). For the latter question, Murray responded that Skywriting can easily be deployed onto a cluster using a set of scripts and that binding code to allow Hadoop or Dryad workers to be tied in with a Skywriting cluster is currently in development.

- ***Toward Risk Assessment as a Service in Cloud Environments***
  Burton S. Kaliski Jr. and Wayne Pauley, EMC Corporation

Formal risk assessment is a necessity for many commercial outfits nowadays, but tends to be a highly time-consuming service. There are a number of well-established standards, but all assessment is still done manually by humans. Wayne Pauley asserted that as a consequence of this, traditional risk management strategies fail when applied to the cloud, which is a fast-paced, dynamic environment that is also geographically diverse and on-demand. Endpoint devices can be anything and resource pooling means that it is impossible to tell in advance what resources are going to be shared with. Subcontracting at the cloud provider and the challenge of having to meter and monitor customers while avoiding leaking private data through observation of usage characteristics are further issues.

The authors introduced the notion of "risk assessment as a service," in a manner akin to the automated credit ratings in use today. Various different models are possible to imagine: self-assessment by the cloud provider, third-party audit, or consumer assessment involving internal and external agents. To facilitate this, the authors have designed a risk assessment architecture for the cloud, with a risk monitor and a set of agents as its core components. The risk monitor is supplied with information from a variety of agents (both with the cloud and with customers) and also receives information from external auditors and definition lists.

After having sketched the architecture, Pauley now sees their future work in figuring out what sensors are needed with the agents, what implementation language to use for the system, and working out how customers and providers can react to the assessment in an automated fashion. Furthermore, they plan to undertake an evaluation of the effectiveness of automated assessment versus traditional (manual) methods, as well as of the actual trust assurances given by the automated measurements.

How much money should be spent on security assurances and can risk be measured in a monetary form? Pauley replied that some equilibrium needs to be found; currently, there is not that much intellectual property in the cloud, but as we start moving more information into it, the importance of determining the value of individual assets in order to assign risk thresholds and monetary value to it will grow.

- ***Information-Acquisition-as-a-Service for Cyber-Physical Cloud Computing***
  Silviu S. Craciunas, Andreas Haas, Christoph M. Kirsch, Hannes Payer, Harald Röck, Andreas Rottmann, Ana Sokolova, and Rainer Trummer, University of Salzburg, Austria; Joshua Love and Raja Sengupta, University of California, Berkeley

Christoph Kirsch described using virtualized flying vehicles carrying sensors as information acquisition nodes that can be sold using a similar elasticity model to other resources in cloud computing. The authors are prototyping this concept using an autonomous quadrocopter, called JAviator, as their hardware platform. The quadrocopter is controlled by a computer and must be controlled this way, as the platform is unstable without computerized control. The quadrocopter acts as a "cyber-physical server," having an IP address and a geographic location, plus the capability to move about and carry sensors. Multiple quadrocopters can form a "cloud." Kirsch went on to introduce the notion of a "virtual vehicle" that can migrate between physical vehicles (which can run multiple virtual vehicles at the same time). One use case for this is as follows: imagine there are a number of flying vehicles that follow predefined routes. If someone now were to require a vehicle that describes a flight route that none of the physical vehicles describes individually but which a combination of them could cover, transparently migrating a virtual vehicle between different physical vehicles enables us to provide this.

In the experimental setting, real vehicles with real sensors (Webcam, laser, ultrasonic, gyroscope, accelerometer, etc.) exist, as well as real servers that are mounted onto real vehicles. The latter are still works-in-progress and will provide a powerful small form-factor server more powerful than an embedded system. Virtual vehicles can run on those physical vehicles and servers and have access to virtual sensors (also still a work-in-progress—current efforts are focused mainly on the Webcam) as well as virtual processors (which provide real-time guarantees required for flight control). Migration of virtual vehicles between different physical vehicles needs to be really quite fast (~10ms), due to the short reconnaissance times the physical vehicles experience. Furthermore, there is the notion of "virtual actors," which substantiate themselves in the form of virtual vehicles and which can pilot real or other virtual vehicles.

The current research effort is split between Salzburg (virtualization infrastructure) and Berkeley (collaborative control), with joint work on the programming language. In the virtualization infrastructure, an "Earliest Deadline First" (EDF) scheduler was added to Xen to support temporal in addition to spatial isolation, and future work is concerned with power isolation, migration, and tracking real and virtual vehicles. The collaborative control problem is mainly concerned with the allocation of real vehicles to virtual vehicles under consideration of mutable flight plans (read-only flight plans for physical vehicles are easy, but read-write flight plans for virtual ones are hard due to potentially conflicting interests of virtual vehicles). The programming language in use is the Collaborative Sensing Language (CSL), which will specify dynamically changing missions of virtual vehicles. The key challenge here is said to be the handling of concurrent and dynamically changing sets of real and virtual vehicles.

How flexible are virtual vehicles and are there any other cyber-physical systems that the authors know about? Virtual vehicles are quite flexible, but of course ultimately must conform to the limits of real vehicles that can only fly in certain ways (in addition to other challenges). To the second question, "anything that moves" is potentially a cyber-physical system; however, the applications considered in this case require certain computational power, which excludes some options (e.g., very lightweight sensor nodes).

SECURITY AND RELIABILITY

Summarized by Rik Farrow (rik@usenix.org)

- ***A First Look at Problems in the Cloud***
  Theophilus Benson, University of Wisconsin—Madison; Sambit Sahu, IBM Research; Aditya Akella, University of Wisconsin—Madison; Anees Shaikh, IBM Research

Benson explained that they studied three years of support data of an Infrastructure as a Service (IaaS) cloud provider. The data was from a support forum where a new thread was treated as a trouble ticket, and resolution could come from other users or the IaaS support group. They used an automated information retrieval algorithm (Lemur) for extraction of problem clusters, then selected a subset of these clusters for manual analysis.

The authors found that most problems with the cloud service fell into five categories: image maintenance, connectivity, performance, virtual infrastructure, and application-related. Over the three-year period, image maintenance problems declined as better APIs and tools appeared for dealing with images. Problems with virtual infrastructure increased whenever new features, such as cloud storage, appeared. Over time, fewer operator interventions occurred as users became better able to solve their own problems as the online support database grew in size.

Benson gave an example of a problem that required the cloud support to intervene. A user complained of losing connectivity with her instance, and it turned out the VM instance was running out of memory and killing the ssh daemon. Benson suggested that cloud providers expose more information to their users without divulging infrastructure details while avoiding storage overhead by collecting more data/logs. He also suggested adding more user controls.

John Arrasjid suggested that they should have test plans from the provider's developers ready when new releases come out. Benson agreed, saying that would be a good way to decide which tools to expose to users. Christina Serbin asked about a spike in a graph that occurred in March 2009. Benson replied that this was an anomaly.

- ***Secure Cloud Computing with a Virtualized Network Infrastructure***
  Fang Hao, T.V. Lakshman, Sarit Mukherjee, and Haoyu Song, Bell Labs, Alcatel-Lucent

Fang Hao laid out their goals: isolation transparency (see only the user's own virtual network), location independence (locate anywhere in the data center), easy policy control (change policy settings for cloud resources on the fly), scalability (restricted by total resources available), and low cost (use off-the-shelf whenever possible). VLANs have been used to provide network isolation, but this solution has problems. The VLAN ID field only supports 4096 VLANs, and hypervisors must be configured to map VLANs to particular VMs, a potential weakness, as hypervisors can be attacked.

Their solution involves adding Forwarding Elements and a Central Controller. FEs are Layer 2 routers that enforce forwarding of packets to a particular edge network and interface. FEs attach edge networks to the core networks and route packets between core networks. Users decide which VMs can communicate, and the Central Controller configures the FEs.

Hao described an attack that used traceroute to determine domain 0 addresses and looked for other numerically close addresses with a short roundtrip time. Their solution prevents this attack, because only addresses that are part of a virtual network can be seen, since the FEs control the forwarding of packets.

Someone asked about the delay imposed when a host first issued an ARP for an address, an operation that the Central

Controller must handle. Hao responded that this occurs in the data center, so latency will be low. Also, this only occurs once, for the first packet. The same person asked about how they came up with the Layer 2 mechanisms, and Hao answered that it is basically a hash reuse process, and that they really didn't evaluate performance. Wayne Pauley asked if removing traceroute increased the number of support requests, and Hao answered that traceroute is still available, but can only be used to view the customer's own virtualized network.

- ***Look Who's Talking: Discovering Dependencies between Virtual Machines Using CPU Utilization***
  Renuka Apte, Liting Hu, Karsten Schwan, and Arpan Ghosh, Georgia Institute of Technology

Renuka Apte described a system for uncovering relationships between VMs by examining CPU usage. Knowing which VMs have dependencies is useful when it comes to migrating VMs, as you don't want to move related VMs too far apart, where "far" has to do with network latency.

They use xentop to monitor CPU utilization, at a frequency of once a second with a window of 300 seconds. Both of these values can be adjusted, but this is what they found worked well in their experiment. Then they used k-means for clustering spikes in CPU utilization. The value of k must be supplied by the user but should match the number of applications sharing dependencies. They found they could identify dependencies with 91% true positives and 99% true negatives. They ran three applications in their test, with multiple instances of RUBIS and one of Hadoop, with one master and three slaves.

Someone asked what the meaning of false positives was, and Apte answered that it meant identifying a particular dependency that didn't exist. This would be harder to measure in the real world, where dependencies are not known in advance. Someone else suggested looking at network traffic instead of CPU load, and Apte pointed out that VMs can share the same physical system and not have any external network traffic. You would also need to clean up traffic traces to remove any non-significant traffic when resolving dependencies.

- ***A Collaborative Monitoring Mechanism for Making a Multitenant Platform Accountable***
  Chen Wang, CSIRO ICT Center, Australia; Ying Zhou, The University of Sydney, Australia

The speaker for this presentation was held up because of visa issues, so the session chair made a brief summary of the paper. The authors use Merkel B-tree, which is authenticated so you can present evidence back to the cloud provider. The goal is to provide clients of a multitenant service, such as force.com, with a means of providing evidence that SLAs have not been met, for example.

PANEL

Summarized by Alva L. Couch (couch@eecs.tufts.edu)

- ***Barriers to Cloud Adoption and Research Opportunities***
  Moderator: Erich Nahum, IBM T.J. Watson Research Center

  Panelists: Albert Greenberg, Microsoft Research; Trent Jaeger, Pennsylvania State University; Orran Krieger, VMware; Prashant Shenoy, University of Massachusetts Amherst; Ion Stoica, University of California, Berkeley

Trent Jaeger discussed the misconceptions common to cloud security. The cloud provider thinks of security as "guards" around the user's data, which is processed within the cloud, while a security expert thinks of data as something that should be encrypted whenever it is stored in the cloud. Neither of these is practical. We need an intermediate approach in which data is secure in the cloud but computation can still occur inside the cloud. The key to this approach is some form of transparent security where data remains both secure in the cloud and available to applications. This requires a number of security measures, including proofs of host "correctness."

Orran Krieger discussed the role of clouds in "fungible" computing. A "fungible" asset is a commodity whose provider can be freely changed without impact. If clouds become fungible, then cloud providers can compete with host applications without incurring refactoring costs. Fungible computing is a transformational paradigm. A new company or startup can use the cloud as a cost-effective way of experimenting without capital investment, in the sense that no capital equipment is either purchased or managed to create, for example, a new Web site. When a company fails, its images are deleted, so there is little overhead incurred for failure.

There are two competing paradigms for clouds: vertically integrated clouds like Amazon, AppEngine, Azure, and IBM, and the cloud "marketplace" of a plethora of VM hosting services. Krieger hopes that the "marketplace" wins. For this to happen, we need common abstractions for writing virtualized applications, as well as practical methods for federating services to be used by the applications.

Krieger envisions future transformational features, including a "follow-me-anywhere" desktop, laptops with remotely administered system administration and security features, and even the ability to be "a sysadmin for your mom." No one has even started tackling the "tough problems."

Prashant Shenoy discussed three challenges of cloud computing, including economics, manageability, and network/cloud interoperability. The cloud argument is that leasing is cheaper than owning. But in fact, this is built into outsourcing agreements and one who outsources IT does not interact with cloud economics directly. Those who do must deal with new challenges, including resource provisioning. There remains a need for an economically justifiable private cloud model.

Second, there is a need for enterprise management tools such as IBM Tivoli and HP OpenView to understand and be able to manage the cloud, as one way to make private clouds economically justifiable for large enterprises.

Third, there is a need for coordination between cloud management and the network in which the cloud functions. Optimizing cloud application configuration requires also adjusting network configuration, including available bandwidth. Enterprise management tools such as IBM Tivoli and HP OpenView must be extended to manage the cloud and should also be able to tune the cloud, applications, and network as one integrated task.

Ion Stoica discussed the need for meaningful service level agreements (SLAs) in selling and consuming cloud services. SLAs provide the customer with "one throat to choke" when things go wrong, and they serve as a contract and point of accountability between customer and provider. In this context, research opportunities include achieving high utilization (for the provider) in the presence of interactive applications, providing appropriate isolation between applications, and the ability to scale up and down. The "holy grail" is that an application can safely assume that it is running in isolation.

A second issue is "multi-datacenter support" for cloud applications. To achieve appropriate availability and scalability, we need a shared API for applications that can be provided at several places to support migration. Research challenges include intelligently choosing locations for an application, designing an appropriate API for interactions with the data center, and assuring data consistency, no matter what happens.

Finally, the cloud needs to provide appropriate and usable notions of data security that allow applications to safely execute across administrative domains. Research opportunities include how to construct privacy-preserving queries, how to utilize encrypted data, and how to leverage test-case management (TCM) in the cloud environment.

Moderator Erich Nahum then pointed out that the panel was way too much in agreement. In response, Krieger claimed that the whole security paranoia about clouds is overblown. In fact, the same security problems affect non-cloud systems and customers don't seem to care. Trent Jaeger pointed out (in agreement) that in the 1990s we had agent computing and sent code "into the wild" to be executed. We again have a mental hurdle to leap, in writing code to be executed on systems one does not own.

Sambit Sahu asked about trusted security structures. What can one do to analyze security when one cannot examine the software stack in the cloud? Jaeger asked how one is even going to know what is running. Krieger responded that you can know what you are doing, but not what is running beside you.

Someone then asked what utility computing regulations should be. Krieger responded that providers must be regulated, but some of the things that turn utilities into monopolies will not happen in this case. There will be a market for third-party auditors to ensure compliance. Shenoy pointed out that for clouds, the main issue is not going to be regulation but, rather, compliance with standards, driven by customer needs.

Someone asked about the new kinds of security attacks that arise from "inside" the cloud. Krieger predicted that the larger enterprises are not going to have this problem, because they are going to be running their own private clouds. The smaller enterprises, which cannot justify private clouds, are going to be the ones at risk. The need is to provide security to smaller enterprises that—in fact—they would not be able to afford to provide for themselves "in a million years."

Someone responded that the inside attacker has access. Krieger responded that in the future, the attacker won't have access, and that there are technical solutions to this problem. Krieger pointed out that the cloud is awesome for Web apps that need to scale, but—as a whole—looks more like a 24/7 enterprise application with nearly constant load and potentially low (10%) utilization, due to the need to respond to demand changes. We would like to think of it as an 80% utilization, but this may not be realistic.

Someone next asked how one migrates to the cloud. Shenoy responded that the key is to understand your own cloud usage strategy. Krieger responded that he is not sure whether there is a problem for larger enterprises, because enterprise data centers are already virtualized and anything that will run there will run in the cloud. The appropriate migration strategy is to make the cloud look like the enterprise from which the application was migrated. Shenoy pointed out that virtualization is already ubiquitous outside the cloud. Stoica said that getting out of the cloud is straightforward; one just duplicates the API outside, and only when it becomes cost-effective.

A member of the audience admitted to some remaining confusion about how to define the cloud. Krieger reminded us of the story of the blind men and the elephant. It is not that interesting to define the term "cloud" as any more than we have done to support the Web so far. Even the very simple model of IaaS—if it can be standardized—is going to be transformative.

2nd Workshop on Hot Topics in Storage and File Systems (HotStorage '10)

June 22, 2010
Boston, MA

DON'T WORRY, YOUR DATA IS SAFE WITH FLASH

Summarized by Rik Farrow (rik@usenix.org)

- ***Removing the Costs of Indirection in Flash-based SSDs with Nameless Writes***
  Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison; Vijayan Prabhakaran, Microsoft Research

Remzi Arpaci-Dusseau began with a quote attributed to Butler Lampson: "All problems can be solved . . . by another level of indirection." He went on to list the many uses in operating systems of redirection, such as virtual memory, RAID, and VMMs. But indirection introduces performance issues as a side effect. Arpaci-Dusseau said that the target of this research is the Flash Translation Layer (FTL), an abuser of indirection.

FTLs use indirection because writing to flash requires writing only to erased pages, and erasing a page takes milliseconds, not microseconds. FTL hides this latency by writing to a log. Arpaci-Dusseau presented the authors' main idea: nameless writes. Instead of attempting to write to a particular block on a flash device, the data is written to the device. On completion of the write, the device returns the physical location of the block written. Someone asked about handling wear-leveling, and Arpaci-Dusseau responded that the device would upcall into the client, updating the physical location. Randal Burns vehemently disagreed, saying that causes all sorts of problems, and Arpaci-Dusseau agreed with Burns. But then he said he still thought this is a good idea. He then pointed out that every write cannot be nameless, as there must be some known beginning address. The device must also be willing to share some low-level information.

Chris Small pointed out that all performance problems can be solved by removing indirection as a corollary to the opening quote. Then Small worried that making flash too dumb might cause problems. Arpaci-Dusseau agreed that there must be some information stored within the flash, for example, for wear-leveling. Peter Desnoyers didn't consider wear-leveling the big issue, but instead thought that garbage collection was more of a problem. Arpaci-Dusseau said he worried about this too, but didn't have a solution for this yet. Someone suggested adding new interfaces that handle resource allocation.

- ***Depletable Storage Systems***
  Vijayan Prabhakaran, Mahesh Balakrishnan, John D. Davis, and Ted Wobber, Microsoft Research Silicon Valley

Vijayan Prabhakaran pointed out that, traditionally, space is the major constraint in storage, but in SSDs, the primary issue is the number of erasures. Ideally, the lifetime of a device would be the product of the size of the device times the number of erase cycles, but wear-leveling reduces this in practice. When using flash, write patterns also influence wear: for example, sequential block-sized writes compared to small random writes.

In response, Prabhakaran suggested that we need depletion-aware resource management. This would allow predictable replacement based on the lifetime of a device, a way to charge users for usage and to compare designs that reduce depletion, and to deal with new attacks against devices that reduce lifetime. Prabhakaran listed two challenges: many layers in file systems, such as caching, journaling, schedulers, and RAID; and media heterogeneity, such as SLC vs. MLC with different performance and erasure limits. Their solution is to introduce a VM that isolates applications from the device, minimizing the layers before issuing writes to an SSD.

Dan Peek from Facebook wondered if doing this would hide important details from the application writing that need to be exposed. Prabhakaran agreed that this was the right way to do things, but wondered what metrics should be exposed to applications. Peter Desnoyers continued on this theme, using Intel's high-end SSD, which has 80GB of flash but only exposes 64GB, as an example. Desnoyers wondered how much information needs to be exposed. Prabhakaran said that as a community, we need to provide a set of techniques to expose this data. Someone else asked if they had disabled caching, and Prabhakaran said that they had tried their experiments both ways, with caching enabled and disabled.

- ***How I Learned to Stop Worrying and Love Flash Endurance***
  Vidyabhushan Mohan, Taniya Siddiqua, Sudhanva Gurumurthi, and Mircea R. Stan, University of Virginia

Vidyabhushan Mohan explained how stress events affect both the retention and endurance of flash. Flash memory can only be written to after being erased, so a stress cycle consists of write (program)/erase cycles (P/E). Most research on flash use utilizes manufacturer datasheets to calculate endurance, but recent papers on NAND flash chip measurements hint at a much higher endurance. An important but overlooked factor in flash endurance is a recovery process which occurs during the time between stress events and allows partial healing of a memory cell.

The authors designed a simulation that takes the time for recovery into account, while modeling the device physics and applying write traces from four different server applications: EXCH, LM, RADIUS, MSNFS. Using these traces, they could calculate the amount of recovery time between stress events and use this to calculate the number of P/E cycles under each workload. What they found is that endurance could be increased by two orders of magnitude with recovery times on the order of a couple of hours, and this occurred with all their example workloads. Their conclusion was that SSDs are durable enough to support enterprise workloads, although this should be examined using real enterprise workloads.

Remzi Arpaci-Dusseau said he had hundreds of questions, but asked just one: is there a measurable difference with bandwidth and performance? Mohan responded that you can see better performance with newer (less stressed) memory. Arpaci-Dusseau then asked, "Add capacity in the SSD to spread out the load more?" Mohan said yes, this improves both performance and endurance. Someone asked if they had talked to vendors about this, and Mohan said they had. He then asked if endurance also depends on implementation of the hardware, and Mohan said yes. The same person asked about the vendor endurance numbers, and Mohan said they are worst-case estimates, created by testing SSDs in ovens. Peter Desnoyers said that this is exciting work, and he wondered if adding error detection could extend SSD life even further. Mohan said that bit error rate does increase after a few million cycles.

OUT WITH OLD (RAID)

Summarized by Aleatha Parker-Wood (aleatha@soe.ucsc.edu)

- ***Block-level RAID Is Dead***
  Raja Appuswamy, David C. van Moolenbroek, and Andrew S. Tanenbaum, Vrije Universiteit, Amsterdam

Raja Appuswamy presented a modular file system stack called LORIS, which inverts the conventional file system/RAID stack, moving RAID-like file multiplexing into the logical layer, instead of the block layer. This allows the physical layer to implement parental checksumming on *all* blocks, rather than RAID being allowed to propagate corruption into the parity blocks.

LORIS divides the software system into a fully modular stack. At the physical layer, metadata caching, checksums, and on-disk layout are handled. Above that is a logical layer, which handles RAID and logical policy storage, and has a mapping file which maintains all of the policy information, such as the RAID level, the stripe size, and the file identifier. The caching layer is responsible for caching data, as usual. Finally, the naming layer handles POSIX call processing and manages directories.

LORIS uses a unique ID and a set of attributes for each file, which are fully shared between all layers. Any layer can get or set an attribute, and files are referred to by their common ID. Because of this shared infrastructure, any layer can set policy information. Because RAID is in the logical layer, it can be file aware, rather than block aware, and therefore can implement these policies in an intelligent fashion. LORIS offers a clean stack abstraction which allows more intelligent error handling for RAID, and allows other components of the stack to be swapped out at will, opening up new possibilities for filesystem designers.

Chris Small from NetApp asked how this was different from what NetApp currently does. Appuswamy replied that LORIS has the ability to isolate changes in the file system to a single layer, rather than being tightly integrated throughout the stack. Remzi Arpaci-Dusseau asked why they were stopping before the hardware layer, noting that there's a lot of abstraction that goes on in the hardware level and they're moving away from the common interface. He asked what kind of interfaces they would like to see at the lower layer. Appuswamy replied that because LORIS is a pure stack, any number of the layers could be moved into the hardware. For instance, hardware could move to an object-based interface without disrupting the stack.

- ***Mean Time to Meaningless: MTTDL, Markov Models, and Storage System Reliability***
  Kevin M. Greenan, ParaScale, Inc.; James S. Plank, University of Tennessee; Jay J. Wylie, HP Labs

Kevin Greenan presented a new reliability metric, called NOrmalized Magnitude of Data Loss (NoMDL). Greenan argued that mean time to data loss (MTTDL) is a metric which is meaningless and misleading. In the authors' opinion, a good reliability metric should be calculable, meaningful, understandable, and comparable. In other words, it should be generable using a known and understood method (such as closed form equations, or simulation), relate to real world systems, and be possible for system owners to understand and use to compare systems.

MTTDL is easy to calculate, since it relies on Markov models, which can be calculated in closed form. However, Greenan noted some flaws in the meaningfulness of the model. For instance, since Markov models are memoryless, the model completely ignores hardware aging. Every time a hard drive is replaced, the model assumes that all remaining hardware is in perfect condition. This does not accurately reflect reality. Likewise, MTTDL often is applied in a sector-failure-agnostic way. Even if sector failures are accounted for, Markov models do not describe the "critical mode" of a system, where additional sector failures during rebuild may cause data loss, depending on their location. Since the probability of data loss declines continually over the rebuild period, it is challenging for a Markov model to describe. Finally, MTTDL is a metric which only answers the question, "When will I lose data?" and not "How much data will I lose?" The authors argue that the latter is a more useful and important question to answer.

Greenan proposed NOrmalized Magnitude of Data Loss (NoMDL) as a replacement for MTTDL. NoMDL avoids some of the flaws of MTTDL and aims to answer the question "How much data will I lose?" by relying on Monte Carlo simulation, a popular statistical technique. The authors have built a framework for modeling drive failure and made it available for other researchers to use. It uses a "mission time" (such as the 15-year expected lifespan of a system) and a number of simulation iterations to return an expected amount of data lost at the end of the time span. Comparing it to other metrics, Greenan noted that their system is the only one which is system-agnostic and provides a magnitude of failure.

Randal Burns from Johns Hopkins noted that an answer such as 14 bits is still not meaningful, because systems lose data in large chunks or not at all. Greenan replied that the simulator can actually return a histogram of data loss. Randal retorted that the final output was still a metric. Jim from EMC said that a lot of people associate MTTDL with lifetime, which is clearly inaccurate. Michael Condict from NetApp noted that Greenan was arguing against MTTDL for a single device, and he asked why the regenerative model was bad for a whole system. He suggested that they just change the model to say that the device is halfway through the lifespan. Greenan replied that that still wouldn't be accurate, because the system as a whole is aging, not just an individual device. Empirical testing suggested that just artificially aging the device in the model yielded unrealistic results.

- ***Discussion Panel***

The session chair, Arkady Kanevsky, kicked off the discussion panel by asking whether RAID was even relevant any more, given that failures are now known to be highly correlated. Greenan replied that block-level RAID is dead because of rebuild time. The window of vulnerability is getting bigger and bigger. Distributed RAID will make more sense, because the system can spread the load out. Appuswamy replied that by imparting semantic knowledge to the RAID layer, many things become possible. Search-friendly name schemes require a full rethinking of RAID, which requires an abstraction layer.

Ric Wheeler, addressing Greenan, noted that soft errors in hard drives are found via pro-active scanning, which offsets second drive failure problems. Greenan asked whether a higher-level process which was checking the errors would have the information to fix the errors. Wheeler replied that the information was available at the block level, since things like trim commands have a notion of which blocks are alive.

Jiri Schindler asked Greenan to give an argument that his model was a more general one than the one used by Elerath. Greenan replied that Elerath's model was specific to Weibull distributions, where their package allows them to plug in distributions. He also noted that Elerath was focused very specifically on a RAID 4 array, where theirs can take an arbitrary erasure code.

Michael Condict from NetApp noted that one of the benefits of MTTDL was the ease of calculation and asked Greenan what the inputs to his model were and whether his model was easy to calculate. Greenan said that the model was available already and that MTTDL didn't allow anything except a Markov model. Condict then asked whether he could input more intelligent data into the model if it was available. Greenan noted that it was possible to apply a Markov model in that way, but that it was very difficult and somewhat inaccurate. They chose simulation because it was more accurate and required less work for the systems engineer versus creating a very complex Markov model.

SCALING UP, VIRTUALLY

Summarized by Aleatha Parker-Wood (aleatha@soe.ucsc.edu)

- ***KVZone and the Search for a Write-Optimized Key-Value Store***

  Salil Gokhale, Nitin Agrawal, Sean Noonan, and Cristian Ungureanu, NEC Laboratories America

Nitin Agrawal presented Alphard, a write-optimized local key-value store, and KVZone, a benchmarking tool for testing key-value stores. There are a variety of existing key-value stores, but benchmarking tools for key-value stores significantly lag behind development, and there has been no head-to-head comparison. The authors needed a low latency local key-value store to back their content-addressable file store, HydraStore, and therefore set out to benchmark existing key-value stores to find a suitable candidate.

KVZone is a benchmark specifically optimized for testing local key-value stores. It generates key-value pairs based on a specified set of properties. Key lifetimes and a mix of operation probabilities, such as a workload which is 60% reads, 20% writes, and 20% deletes, can be specified. The rate of requests can be specified in terms of either throughput or latency. Finally, KVZone can take an already existing key-value state to warm up the key-value store, in order to evaluate a particular real-world situation.

The results of their testing suggested that even the most performant of key-value stores operated at less than 40% of their raw device throughput. This was inadequate performance for HydraStore, which led them to create their own key-value store, Alphard. Alphard was specifically created with local, write-intensive workloads in mind and was optimized for SSDs. Some of the optimizations include direct I/O, block-aligned I/O, and request coalescing, as well as including metadata with key-value pairs to maximize the effectiveness of single writes. Alphard uses a logically centralized queue, with multiple physical queues and worker threads, in order to bring operations as close as possible to one synchronous I/O per key-value operation. Alphard achieves very close to device performance under their write-intensive workload.

Chris Small from NetApp noted that comparing a multithreaded KVS versus single-threaded KVS was not an apples-to-apples comparison. Agrawal replied that they were specifically focused on the properties of each key-value store under the required workload, rather than redesigning existing key-value stores. Ric Wheeler asked about the trade-offs for durability and whether Alphard was durable. Agrawal replied that they did care about durability. Writes are persistent to the media, and mirrored. Mirroring requires a 3–5% overhead. Were there any insights about key-value stores in general that could be distilled from the authors' work, and why had they chosen an asynchronous interface? The asynchronous interface gave them the ability to coalesce operations into a single I/O. Also, since they were dealing with an asynchronous interface to the device, it preserved the semantics of the FS.

- ***Rethinking Deduplication Scalability***
  Petros Efstathopoulos and Fanglu Guo, Symantec Research Labs

Petros Efstathopoulos presented a highly scalable system for deduplication, designed to scale to one hundred billion objects, with high throughput. The authors were willing to sacrifice deduplication performance in order to achieve near-raw-disk performance.

The conventional approach to scaling deduplication performance has focused on using larger and larger segment sizes. However, this reduces the quality of deduplication and creates problems for reference management. The larger the segment size, the more catastrophic a deletion error or a lost reference is. In addition, the system still needs to be fast. The authors propose to use a sub-sampling technique, which they call progressive sampling.

The system creates a sample index, which is maintained in memory. The sampling rate is a function of the memory size, the size of each segment entry, and the total number of segments. When memory is plentiful, everything is indexed. However, as the system runs low on space, the sampling rate is progressively reduced. A purely random sampling strategy will result in decreased performance, so the system uses a fingerprint cache to take advantage of locality. In addition to the sample index, a full deduplication index is created and checkpointed to disk. Finally, the authors propose using SSDs for a fingerprint index, allowing memory to be used purely for caching and bloom filters which summarize the SSD index.

The final challenge in deduplication is reclaiming resources. Reference counting is simple, but challenging to make resilient in the face of failure. A reference list makes it possible to identify which files use which segments, but doesn't handle lost updates and is prohibitively expensive to scale up. Mark-and-sweep is another popular garbage collection technique, but this has a workload proportional to the capacity of the system, which is too slow at the petabyte scale. The authors propose a group mark-and-sweep, which improves the performance. The system tracks changes to a group and re-marks changed groups. If nothing has changed since the last iteration, mark results are saved and reused. This results in a workload which is a function of the work done since the last mark-and-sweep, rather than the size of the system.

Michael Condict from NetApp asked how the system decided which fingerprints to keep and which to discard to disk. Efstathopoulos replied that they just picked every *n*th segment. Condict suggested that they consider Extreme Binning as a complement to their work. Efstathopoulos noted that Bhagwat et al. were using Extreme Binning as a method for identifying super-segments. Condict noted that this method might improve the chances of the system having a hit. Dutch Meyer from the University of British Columbia asked how the system determined what constituted a group for their mark-and-sweep approach. Efstathopoulos replied that groups were composed of one or more backups of a system. Finally, Meyer asked if they had tried reference counting as a heuristic on their cache. Efstathopoulos replied that they had tried a variety of heuristics, and concluded that the overhead wasn't worth it.

- ***TrapperKeeper: The Case for Using Virtualization to Add Type Awareness to File Systems***
  Daniel Peek, Facebook; Jason Flinn, University of Michigan

Daniel Peek from Facebook presented TrapperKeeper, a method for extracting rich metadata from files without requiring file type creators to write plugins for every file system and search system. Rich metadata is the holy grail for designers of search systems. Unfortunately, extensions follow a long-tailed distribution, and it is uneconomical for either search systems or applications to support every file type in existence. Popular file types are well supported, but less popular file types are unlikely to be.

TrapperKeeper utilizes the already implemented behavior of applications to parse files, in order to capture metadata. It runs applications in a virtual machine environment. By opening a dummy file and then taking a snapshot at the moment of the open() call, the system can guarantee that the application is about to parse a file. When parsing behavior is needed, the VM can be restarted, and a real file can be substituted for the dummy file that was about to be invoked. From the application's point of view, this is seamless.

The next challenge is using the application to extract key-value pairs. However, most applications implement the accessibility APIs bundled with operating systems. By leveraging these and applying a variety of heuristics, the system can automatically detect tables, labels, and so on. Alternatively, the user can do manually guided extraction, which the system will cache for later use on other files of that type.

Someone raised a number of open questions about the system, which Peek noted were valid future work areas. For instance, what if the application has no accessibility support or does not expose metadata? What if the application needs external information, such as configuration files, in order to parse the input? One audience member suggested that a hybrid approach might be best, where plugins are used for the most common file types, but TrapperKeeper is used for the long tail.

▪ ***Discussion Panel***

The session chair, Ric Wheeler, started the discussion by asking each of the panelists how scalable they would like their systems. Peek replied that he worried both about system performance and human scalability. He wanted to avoid duplication of effort, such as multiple users creating parsing behavior for the same file types. Agrawal replied that he wanted to push the limits of Alphard and make it effective as a scalable store, as well. Efstathopoulos noted that scalability doesn't always rely on a new idea. The design principles are well known, but not always applied. Systems often aren't built with those in mind; sometimes the system designer has to go back and build it right later on.

Someone asked Efstathopoulos whether his system was processing directed acyclic graphs for garbage collection or was just a single level deep. Efstathopoulos replied that they had a flat space for garbage collection, where the storage group container has an ID and containers have chunks.

Someone asked Agrawal what the guarantees were that Alphard provided, from the time the client uses the system until the data is safely on disk. The questioner noted that coalescing writes just made matters worse and that there was an opportunity for something to go wrong while a request was in the queue. Agrawal replied that the actual interface didn't return until the data was safely committed, so while the system was asynchronous in implementation, the interface was, in fact, synchronous.

Another audience member asked whether the move to key-value stores would inhibit or help accessibility of rich metadata. Peek replied that right now there was no specialized file system handling for indexes, so a key-value store would have little impact. Efstathopoulos replied that there was a constant tug-of-war between specialized and general file systems. Agrawal added that system designers should think about what they actually want in a file system and design around it, rather than vacillating between extremes, as system designers realize they're missing key pieces each time they jump on a new technology.

Finally, someone asked about the differences between usage for key-value stores versus databases, noting that databases offer a many-to-many relationship, where key-value stores are strictly one-to-one, or one-to-many. Agrawal replied that he normally only used one or two keys, and that in practice data is often sharded across multiple databases, such that complex join operations, while possible in theory, are rarely used in practice.

ALL ABOARD HMS BEAGLE

No reports are available for this session.

▪ ***Fast and Cautious Evolution of Cloud Storage***
Dutch T. Meyer and Mohammad Shamma, University of British Columbia; Jake Wires, Citrix, Inc.; Quan Zhang, Norman C. Hutchinson, and Andrew Warfield, University of British Columbia

▪ ***Adaptive Memory System over Ethernet***
Jun Suzuki, Teruyuki Baba, Yoichi Hidaka, Junichi Higuchi, Nobuharu Kami, Satoshi Uchida, Masahiko Takahashi, Tomoyoshi Sugawara, and Takashi Yoshikawa, NEC Corporation

▪ ***Discussion Panel***

Configuration Management Summit

June 24, 2010
Boston, MA

Summarized by Aleksey Tsalolikhin
(aleksey.tsalolikhin@gmail.com)

On Thursday, June 24, USENIX hosted the first Configuration Management Summit, on automating system administration using open source configuration management tools. The summit brought together developers, power users, and new adopters. There are over a dozen different CM tools actively used in production, and so many choices can bewilder sysadmins. The workshop had presentations of four tools, a panel, and a mini-BarCamp. This summary covers the four tool presentations and includes some brief notes on the BarCamp.

▪ ***Bcfg2***

Narayan Desai thinks of configuration management as an API for programming your configuration. Bcfg2's job is to be configuration management "plumbing"—it just works.

Centralized and lightweight on the client node, each server can easily handle 1000 nodes.

Bcfg2, pronounced be-config-two, uses a complete model of each node's configuration, both desired and current. Models can be compared (with extensive reporting on differences), or you can designate one node as *exemplar* and its configuration will be imposed on other nodes.

To facilitate learning, the Bcfg2 client can be run in dry-run (no changes, print only), interactive (are you sure you want to do this?), and non-interactive modes.

Bcfg2 supports extensive configuration debugging to help the sysadmin get to the bottom of things quickly, with full system introspection capability (why is Bcfg2 making the decisions that it is?).

Strengths: Reporting system. Debugging.

Weaknesses: Documentation (new set of documentation is coming out now, but still weak in examples). Sharing policies between sites is not easy; group names need to be standardized first.

- ***Cfengine***

Mark Burgess explained the underlying philosophies of Cfengine:

- Promise theory: Files promise to be there, packages promise to be installed, processes promise to be running—or not, etc. Cfengine is the promise engine to fulfill those promises.
- Convergence: Describe an ideal state and Cfengine will get you there, as opposed to a roadmap/log of system changes necessary to bring a system to a configured state from a known starting state—Cfengine will get you to an ideal state from a known or unknown state.
- Self-healing: Assume the environment is hostile and entropy exists, and take measures to continuously check and restore the integrity of the system.
- Pragmatism: Environments can be participated in but not controlled; constrain rather than control the environment; cooperation will take you further than enforcement.

Strengths: Highly multi-platform: runs on very old and very new systems, the full gamut—underwater unmanned vehicles, Nokia cell phones, and supercomputer clusters; lightweight (1.9 MB footprint). The only prerequisites are Berkeley DB library and crypto-library. Cfengine has the largest user base—more companies using it than all the other tools combined. Resilient—able to continue operating under degraded conditions (e.g., if the network is down and server is unreachable, node agents will used the cached policy; Chef and Puppet run the same way). Secure—Cfengine has a healthy paranoid streak (assume they're out to get you) and an impressive security record (only three serious vulnerabilities in 17 years). Commercial version addresses knowledge management—ISO standard topic maps, etc.

Weaknesses: Hard to get started because there is a lot to learn.

- ***Chef***

Aaron Peterson, Opscode Technical Evangelist, presented Chef, primarily a configuration management *library* system and *system integration* platform (helps integrate new systems into existing platforms).

Chef is data-driven. Configuration is just data. Enable infrastructure as code to benefit from software engineering practices such as agile methodologies, code sharing through github, release management, etc. You manage configuration as *resources* (files, packages, processes, file systems, users, etc.), put them together in *recipes* (lists of resources), and track it like source code to configure your servers. *Cookbooks* are packages of recipes. Chef has been out since 2009.

Chef grew out of dissatisfaction with Puppet's non-deterministic (graph-based) ordering. Sequence of execution in Chef is tightly ordered.

Strengths: Cloud integration (automating provisioning and configuration of new instances). Multi-node orchestration. Reusable policy cookbooks and highest degree of recipe reuse between sites (compared to the other three tools).

Weaknesses: Attributes have nine different levels of precedence (role, node, etc.) and this can be daunting.

- ***Puppet***

Michael DeHaan explained that Puppet grew out of dissatisfaction with Cfengine 2. Puppet has a centralized model: a server detects deltas from the desired configuration and instructs the node agent to correct them. Chef works the same way.

Puppet's internal logic is graph-based. It uses decision trees and reports on what it was able to do and on what failed (and everything after it). Manual ordering is very important, as decision trees will be based on it. Ordering is very fine-grained.

The Puppet language is a datacenter modeling language representing the desired state. The Puppet language is designed to be very simple and human readable. This prevents you from inserting Ruby code but it also makes it safer (prevents you from shooting yourself in the foot). However, you can still call external (shell) scripts. Also, an upcoming version (2.6) will support programming in a Ruby DSL.

The server gets the client to tell the server about itself. These are *facts* in Puppet. The configuration policies are the *manifests*. The server compares the facts to the manifests and, if necessary, creates instructions for the clients on the managed nodes to move from what is to what should be. These instructions are encoded as a JSON catalog.

Strengths: Large community of users (over 2000 users on the Puppet mailing list).

Weaknesses: The Puppet server right now is a potential bottleneck, which is solved by going to multiple servers. Execution ordering can be non-deterministic but reports will always tell you what succeeded and what failed, and order can be mandated.

- ***BarCamp***

A BarCamp is an informal colloquium where the audience members take turns presenting to the audience (http://en.wikipedia.org/wiki/BarCamp).

There were a total of five 15-minute presentations from the audience during the final part of the summit. Matt Richards presented "Converting an Ad-Hoc Site to CM: The Story," narrating a successful Cfengine deployment with resulting increase in stability and uptime. Aaron Peterson gave a Chef demo. Michael DeHaan presented "Cobbler: Automated OS Installs," a Linux installation server. David Pullman presented "Cfengine: Complexities of Configuring Different Operating Systems." Finally, Michael DeHaan presented "Func—Attack!!!—Your Systems!!!"—Func is a distributed one-time command or query tool for Red Hat systems.

You can find a much more detailed report at http://www.verticalsysadmin.com/config2010/.

2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '10)

June 14–15, 2010
Berkeley, CA

JUNE 14, 8:30 A.M.–10:00 A.M.

Summarized by James C. Jenista (jjenista@uci.edu)

- ***Towards Parallelizing the Layout Engine of Firefox***
  Carmen Badea, University of California, Irvine; Mohammad R. Haghighat, Intel Corporation; Alex Nicolau and Alexander V. Veidenbaum, University of California, Irvine

Multicore is ubiquitous and the browser is becoming a thin client to run a wider range of applications. Carmen Badea argued, therefore, that it is worthwhile to explore the parallelism in browsers.

Badea explained that Firefox was chosen because it is open source and has the second highest browser market share. They profiled Firefox and discovered that 40% of the test execution time was spent in the layout engine and 32% of that time is devoted to CSS rule matching. This led to a parallelization effort of the CSS rule matching subsystem. After giving a brief background for CSS, Badea explained that CSS rule matching executes when a user loads a new page or a page is interactively updated. The Mozilla Firefox page load tests and Zimbra Collaboration Suite (ZCS) were employed as benchmarks to profile the CSS rule-matching system; descendant selector rules were executed most often, and the vast majority resulted in a non-match.

They decided to parallelize the common descendant rule case of non-match by executing rules for batches of ancestors of an element concurrently. When there is a rule match, some of the work is speculative and therefore discarded, although Badea argued that the profiling data implies this case is infrequent. Dan Grossman asked how many rules are being matched in the parallel implementation, and Badea answered that only one rule is matched at a time. She explained that the code base is factored this way, although future work could explore a parallel implementation that matched many rules at once.

The parallel CSS rule matcher was tested in seven configurations for ZCS and in 12 configurations for the Mozilla test pages. Badea noted that more than two threads did not perform well and hypothesized that future Web pages with richer CSS may benefit from more than two threads. For Mozilla pages, the end user's perceived speedup was as high as 1.8 times, and for ZCS as high as 1.6 times. They attribute the better speedups for the Mozilla page load tests to more complexity in layouts, as well as to the fact that ZCS is a more JavaScript-oriented benchmark suite.

Badea was asked if she believed there will be fewer improvements for such a parallel CSS rule matcher as Web pages have more and more JavaScript. She answered that more JavaScript doesn't exclude more complex layouts. Can an early match result in a longer execution time than the single-threaded version? It's possible, but the profiling data suggests this is a rare occurrence. What behaviors caused the worst speedups? Badea explained that Web pages with few ancestor elements did not trigger the parallel rule matcher, but suffered from added preprocessing.

- ***Opportunities and Challenges of Parallelizing Speech Recognition***
  Jike Chong, University of California, Berkeley; Gerald Friedland, Adam Janin, Nelson Morgan, and Chris Oei, International Computer Science Institute

Adam Janin said that the goal of their work is not for the sake of parallelism specifically, but, rather, to improve speech recognition accuracy, throughput, and latency. Janin then offered a scenario to drive his presentation; he had recorded the speech of a meeting with an iPhone on the table. The systems they developed should process the audio and allow browsing and retrieval of useful information such as querying who was speaking at a given time, finding audio segments by words spoken, and finding segments by speaker. Janin broke down the system and made a clear distinction between speech recognition that extracts words and diarization that identifies the speaker.

Then Janin built an argument for developing a parallel software implementation. Current technologies scale easily along any resource axis; still, state-of-the-art systems are 100 times slower than real time to achieve the best results. Specialized hardware has gotten mixed improvements, so general parallel software may be the answer.

Perceptual models of the inner ear, Janin explained, are used to compute features of audio. The combination of features usually improves results for noisy conditions, so current systems typically select two to four cochlear representation variants. Janin asked, when we have more resources, why not add many more representations? He explained that the representations are filters fed to a neural net, which prompted a question: is the system similar to deep neural nets? Janin answered yes, he would call it deep learning, but with no unsupervised step. He then highlighted that the many streams and dense linear algebra required all have an obvious parallel structure.

Their experiments included both a 4-stream and a 28-stream configuration. Janin indicated that the 4-stream setup improved accuracy by 13.3% on a Mandarin conversational task, and the 28 streams improved accuracy by 47% on a read digits tasks (e.g., phone numbers, zip codes). When asked if the system is commercially viable, Janin answered that the noisy number input audio is an artificial test, and current systems can do well for reading numbers over the phone under normal noise conditions. Another questioner asked whether there is a diminishing return for adding streams. Janin responded that they don't know, but he certainly believes it. The data supports it, although he said the brain is thought to be processing hundreds of millions of audio interpretations at once.

Janin then moved from their improvement of accuracy to the improvement of throughput. They pipelined the speech recognition system and improved the throughput of the inference engine, which looks up the closest utterance from a language. Janin described how the machine-learning model generates a complex, static graph of state transitions to implement the inference. The online system, he explained, does a time-synchronous beam search over the graph, only keeping the best hypotheses. When asked how big the graph is, Janin answered that there are one million states, four million arcs, and thirty-two bytes for each node. They reported an 11-fold speedup overall—an 18-fold speedup for the compute-intensive transitions, and a 4-fold speedup for communication-intensive hypothesis merging.

The next segment of Janin's talk covered how they improved latency and accuracy for online diarization. This might be useful, Janin explained, to identify who is speaking in real time during a distributed meeting. An attendee asked if training data is needed. Janin responded that none is needed for the speakers or even the language. Their strategy is to begin the offline diarization as soon as the meeting starts and hand off models for each speaker to the online system as they improve over the course of the meeting. Janin reported the error rate drops about 7% by parallelizing this implementation over eight cores.

In response to several requests to characterize the challenges of developing the parallel software, Janin replied that designing the parallel algorithm was more challenging than the implementation. New parallel tools could certainly help, especially any that might bring in new programmers. David Padua asked if there is a way to measure the progress in the field of speech recognition. Janin gave details of the US government's annual challenge. Janin's analysis was that the accuracy of systems entered has slowly improved to about 50% word error rates, which he said is quite good for many applications, but that the major progress in the field has been to accomplish previously hard tasks much more easily.

JUNE 14, 10:30 A.M.–12:30 P.M.

Summarized by Chris Gregg (chg5w@virginia.edu)

- ***A Balanced Programming Model for Emerging Heterogeneous Multicore Systems***
  Wei Liu, Brian Lewis, Xiaocheng Zhou, Hu Chen, Ying Gao, Shoumeng Yan, Sai Luo, and Bratin Saha, Intel Corporation

Brian Lewis talked about how computer architecture is becoming more heterogeneous and how to improve programming models for such systems. More and more programmable accelerators are being designed into computer systems, and this talk focused on them. GPUs are either discrete or integrated on-die with CPUs, in which case they share computational resources. Low-level languages that exist today (OpenCL, CUDA) focus on coarse-grained offloading of parallel computation, but do not fully take advantage of CPU capabilities. The authors want to improve programmer productivity and extend the range of applications that can be easily programmed.

David Padua asked, "Is the limitation for fine-grained processing a factor of language, or of hardware?" Lewis answered that both were relevant, to an extent. It is low-level, which does not lead to high-level breaking up of tasks. There was another question about why parallel languages weren't yet meeting our needs; Lewis answered that it is mainly because they are still relatively low-level. They want a balanced programming model, to enable fine-grained computation using all cores, with better support for task and data parallelism, load balancing, and dynamic reconfiguring.

Lewis talked about the importance of shared virtual memory and the need for lightweight atomics and locks, which will allow better coordination between the CPU and GPU. The discrete Larrabee implementation has shared memory that supports release consistency and ownership rights, which allows the CPU and GPU to both work on the same data. There is an OS on both sides, leading to VM page protection, which helps with consistency. The shared memory CPU-integrated graphics has a device driver, and there isn't an OS to handle page faults. It doesn't detect updates using page faults, but it exploits shared physical memory, meaning there is no data copying.

Nicholas Matsakis asked, "Is there a model for how the data should be shared?" Lewis said that the keyword "shared" marks shared data. Keywords, used for offloading functions, are elaborated on in the paper. Timothy Roscoe asked, "Have you looked at what would happen if you ran multiple applications across this system?" Lewis answered that they did not look at that for this paper, but that is the end goal of the work.

- ***Collaborative Threads: Exposing and Leveraging Dynamic Thread State for Efficient Computation***
  Kaushik Ravichandran, Romain Cledat, and Santosh Pande, Georgia Institute of Technology

Romain Cledat started by discussing parallelism in general and how it can be improved. Parallelism today relies on threads, which is splitting up data or tasks. Current models include TBB and CnC, which leads to a natural parallelism. However, threads still use locks and barriers and transactional memories. They share data through shared memory, but do not have knowledge about their "role" in the computation nor the overall state of the computation. Current models break up a computation, and the distribution of work is done just in time. The state of the computation is not taken into consideration. The threads work independently and do not have higher-level semantic knowledge. Performance of HPC problems has dependencies that are greater than simply how the work is split up.

Cledat then discussed useful semantic state, determined by the programmer, to influence scheduling. Byn Choi asked, "Isn't scheduling the threads the role of the task scheduler, and are you trying to make the threads do this in a distributed manner?" Cledat said they are trying to do more than scheduling and were trying to use the meta-information for more than just this problem.

Cledat turned the talk over to Kaushik Ravichandran, who discussed the system the authors created, specifically the semantic state taxonomy. Similar sub-problems are clustered together in a tree structure. The sub-problems are hierarchical, incremental, and approximate. This results in good lookup time, without having to build from scratch. With this information, they can re-use results, orient a computation, prioritize sub-problems, and select cores appropriately. Sean Halle asked, "Is the compiler doing this, or the app programmer?" Ravichandran answered that the programmer designates certain information and the run-time uses it.

Ravichandran provided an example of a sum of subsets, showing that a large of amount of redundancy can be exploited. The programmer can more aggressively parallelize this problem by specifying a similarity metric, which the system will use to make the best use of previously computed values. For the second example, K-Means, objects can share data between localized points. The speedup comes from making fewer comparisons than the original algorithm, sharing results from the closest neighbors.

- ***Structured Parallel Programming with Deterministic Patterns***
  Michael D. McCool, Intel

McCool discussed how people parallelize applications and the structures they use. In particular, he described a total of 16 different fundamental parallel programming patterns. A *parallel pattern* is a commonly occurring combination of task distribution and data access. Many programming models support a small number of patterns or low-level hardware mechanisms. However, a small number of patterns can support a wide range of applications, deterministically. A system that directly supports the deterministic patterns on a lot of different hardware architectures can lead to higher maintainability, and application-orientated patterns can lead to higher productivity.

Sean Halle asked, "Should patterns not have hardware details?" McCool answered, no, he would rather find more abstract patterns, and specifically functional programming patterns. There are structured programming patterns for serial computation, and we can add a number of parallel patterns to this list for a number of different, fundamental patterns.

Sean Halle asked, "Do you want your application talking to the runtime?" and McCool replied that yes, although you don't want to over-constrain the runtime. You do, however, want communication between the two. He continued his talk by discussing partitioning, which is very important; you're breaking an input collection into a collection of collections. This is useful for divide-and-conquer algorithms. There is also the issue of boundary conditions. Another pattern is stenciling, which applies a function to all neighborhoods of an array. There are also fused patterns that can be useful in specific conditions. Examples include: gather = map + random read; scatter = map + random write. Scatter is tricky, because you need to watch out for race conditions. It would be nice to find a deterministic scatter, and the best solution is "priority scatter," which prioritizes the elements as they would have happened in a scalar scatter.

McCool finished with "the bottom line," trying to create a taxonomy of good practices for parallel programming. Are these the right patterns? Is there a smaller list of primitive patterns? How important are nondeterministic patterns? Sarita Adve asked about determinism and isolation, and McCool answered that the merge-scatter pattern came closest to matching.

JUNE 14, 12:30 P.M.–2:00 P.M.

Lunches on both days included tables labeled with questions for discussion. You can find the results of these discussions and some comments at http://www.usenix.org/events/hotpar10/tech/techLunches.html. (I found the results fascinating and interesting in themselves.—The Editor)

JUNE 14, 2:00 P.M.–4:00 P.M.

Summarized by James C. Jenista (jjenista@uci.edu)

- ***Separating Functional and Parallel Correctness using Nondeterministic Sequential Specifications***
  Jacob Burnim, George Necula, and Koushik Sen, University of California, Berkeley

Jacob Burnim identified nondeterministic interleavings as a major difficulty when reasoning about the functional correctness of a parallel program. He proposed that a programmer-generated nondeterministic sequential artifact could decompose the effort into the questions of parallelism correctness and functional correctness. The key, Burnim explained, is that the programmer should annotate intended nondeterminism and then a system can check that the parallelization adds no more nondeterminism.

As an example, Burnim introduced a branch-and-bound code and asked the attendees to consider the sequentially expressed code as a parallel version by adding a few parallel constructs; is the parallelization correct? Burnim offered an interleaving that shows that the parallel answer may be different but correct. Burnim hypothesized that a specification in between the sequential and parallel codes is needed to express the allowed nondeterminism and then provide a framework for proving the correctness of the parallelization.

Their artifact is a nondeterministic sequential (NDSEQ) expression of the code. Burnim introduced the nondeterministic for loop as an element of the NDSEQ which runs one iteration at a time but in any order. Burnim pointed out that there are still interleavings to avoid some prunings that the parallel version can express but the NDSEQ cannot. As there is intended nondeterminism in the example, Burnim introduced the use of if(*) to instruct the NDSEQ to choose either branch. Burnim claimed the modified NDSEQ expressed the intended nondeterminism in the parallel version, and that the NDSEQ could generate a given parallel interleaving. A question was raised about whether the parallel algorithm was suboptimal, which Burnim conceded, but he stated that it is reasonable and apt for the illustration of their work.

Once the NDSEQ is provided, Burnim continued, the parallel correctness and functional correctness can be proved separately. He interjected an argument that the correctness of the parallel version is undecidable and the correctness of the NDSEQ is decidable, offering another justification for the effort of creating the NDSEQ. Then Burnim demonstrated the correctness of the parallelism with a proof by reduction, consisting of the rearrangement of parallel interleavings matched against the NDSEQ. Burnim was asked if the proof works for nested loops. He said that the correctness of the inner loop can be proved, then replaced with a sequential version to prove correctness of the outer loop.

Their future work will include automating the proof for real benchmarks. Burnim suggested that their approach might be applied to other model checking techniques. He also suggested that instead of a static system, their work might be integrated in a debugger to consider the correctness of a parallel trace, where a bug might be classified in relation to the parallelism or the functional correctness.

An attendee asked how to detect when the NDSEQ is incorrect. Burnim answered that there are two cases: when the NDSEQ is too strict, the situation is manageable, as the system could report parallel interleavings that the NDSEQ cannot express to aid NDSEQ improvement; when the NDSEQ is too weak, Burnim conceded that it becomes a difficult problem. Several people asked Burnim about the possibility of language solutions to avoid needing a correctness checker. Burnim answered that when you get correctness for free, language solutions are good, but some computations, such as types with a lot of guarantees, are hard to express without sufficient nondeterminism.

- ***Synchronization via Scheduling: Managing Shared State in Video Games***
  Micah J Best, Shane Mottishaw, Craig Mustard, Mark Roth, and Alexandra Fedorova, Simon Fraser University, Canada; Andrew Brownsword, Electronic Arts Blackbox, Canada

Micah Best introduced their work as a fruitful technique for synchronizing threads via scheduling in the video game domain, a domain in which performance and responsiveness are high priorities. Though it was not the subject of his talk, Best covered the Cascade project, which expresses a video game engine as a dataflow task graph. Their work integrates with Cascade, he explained, and attempts to ease the burden of managing task-shared state off the developer through static analysis and new synchronization techniques at runtime.

Best described how static analysis of the Cascade mark-ups identifies constraints between tasks. The constraints are potential conflicts, such as access to elements of a collection, and their work uses a runtime strategy to determine the actual constraints. Best stated that the scheduler uses task-constraint profiles to synchronize access to shared data.

Best moved the discussion to a method of expressing constraints in binary. References and members, he said, are simply expressed. He continued with the expression for arrays which occur frequently in video game kernels and require some analysis of indices. The hardest cases are forms of indirection and will be addressed in their future work.

The constraints identified by static analysis are passed through Bloom filters to produce a fixed-length bit string. The bit string is a constraint signature for the task; Best added that signatures are cheap to calculate and compare. A task may run when its signature has no conflict with running tasks, although signature comparisons may produce false positives but will never show a false negative. In response to a question about user control over the signatures, Best responded that users may tune the construction parameters through Cascade.

Best characterized their scheduling algorithm as generational. Tasks are batched by using logical-OR on their signatures until no more tasks may be added without conflict. A batch forms a generation and is sent to a core while the next generation is batched.

They tested their work by adding Cascade mark-up to Cal3D, a library for animating character models where separate animations may be blended and applied to the same model. When the application of multiple animations have a state conflict, the system synchronizes access; otherwise animations may be applied concurrently. Best then presented the experimental setup; a workload of four models with eight animations was executed on a two-processor Xeon totaling eight cores.

Their results were compared to a natural implementation as a baseline, which Best defined as one written by a non-expert, competent programmer. The baseline implementation applies animations in a straightforward way without requiring synchronization. Best displayed an activity graph from Cascade that showed banding effects in core usage because there was not enough work while waiting for the next animation. With signatures and then a partitioning strategy they obtained better core utilization. Best highlighted an important result by presenting an expert-tuned version of the benchmark that had the highest utilization. He concluded that they had pursued parallelism too aggressively; a method for finding the right amount of parallelism for given overheads is future work.

Someone from Toshiba asked how they could encourage adoption. Best answered that adoption is always a concern for new parallel languages. One approach is to convince programmers the benefits of the new language are undeniable and always be sure the language is addressing the true problems facing the programmer. Nicholas Matsakis asked how much work Cal3D was to port. Best replied that the work was completed in a few weeks but noted that porting the Cube 3D code was much more difficult. He attributed this to the well-written source for Cal3D as opposed to messy source for Cube 3D and concluded that bad code is hard to parallelize.

- ***Get the Parallelism out of My Cloud***
  Karthikeyan Sankaralingam and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

Karthikeyan Sankaralingam asked whether the current degree of focus on parallelism and multicore is out of proportion to the number of applications for the research. Implementing parallel software is complex, Sankaralingam said, and by asking if real developers or users even want it he stirred up a hornets' nest.

Sankaralingam painted a future computing environment in which notebooks and smartphones offload computation to the cloud, and the average programmer can easily deploy and maintain software in the cloud. He argued that a small number of experts can implement the lower layers of the cloud for multicore architectures, while the average programmer or user device sticks with a few-core model.

Their work addressed three myths that Sankaralingam hypothesized are steering research away from improving the cloud environment and toward an overemphasis on multicore and parallelism. The first myth Sankaralingam covered was that hardware drives software. He argued that programmers historically spent significant software effort to achieve efficiency with hardware, but the major hardware problems are now solved. Now, he continued, programmers must be productive and demand high-level languages to express programs with as little code as possible, and therefore software is currently either decoupled from or even driving hardware.

Sankaralingam moved on to the second myth: multicore will be everywhere. He presented a graph describing the relation of performance to energy and explained that technology scales the curve, but by only so much, and ultimately the number of cores on a handheld device is limited. Sankaralingam conjectured that the limit will be about 10 cores. An attendee asked what he meant by a core; Sankaralingam said he meant a programmable processor. He concluded his discussion of this myth by pointing out that the mobile device may not need multicore, because from its perspective it gets free performance from the cloud without paying energy.

The third myth Sankaralingam identified was that everyone should become a parallel programmer. Sankaralingam called parallel programming a great challenge that may even disrupt the curriculum and suggested it should be left to the experts. The average cloud application parallelizes over many clients in the cloud without being a parallel program, he said.

Sankaralingam summarized their work as an argument to rethink the role of parallelism and then opened for questions by taking off his jacket, revealing a bull's-eye emblazoned t-shirt. Krste Asanović stated that productivity will always be important, and Sankaralingam agreed but used Jango as an example of programmers never even seeing the underlying SQL base. Someone from Qualcomm disagreed that cloud computing will become dominant, because distance to the tower doesn't follow Moore's Law, but devices are following it. Sankaralingam agreed that latency is a hard problem in cloud computing, but offered an anecdote. Sankaralingam had mounted a remote file system while traveling to the workshop, with virtually no impact on his environment; already, he said, the latencies are not so apparent to the end user. Sean Halle began by saying that Sankaralingam was very brave, and Sankaralingam replied that his advisor, Remzi Arpaci-Dusseau, is responsible for the things you disagree with. Halle pointed out that there are 200,000 iPhone applications and asked if Sankaralingam believed the iPhone successor will be single-core. Sankaralingam answered no, but continued by claiming that a mobile device will never have 100 cores for the average programmer to deal with. Sarita Adve asked who the PC members were who accepted this paper, as she wanted to talk with them later.

JUNE 14, 5:00 P.M.–8:00 P.M.: POSTER SESSION

Posters below summarized by Romain Cledat (romain@gatech.edu)

The poster session included all the talks in the program, as well as the papers reported here.

- ***A Principled Kernel Testbed for Hardware/Software Co-Design Research***
  Alex Kaiser, Samuel Williams, Kamesh Madduri, Khaled Ibrahim, David Bailey, James Demmel, and Erich Strohmaier, Lawrence Berkeley National Laboratory

In this work, the authors developed high-level language implementations of key kernels in HPC. They then implemented each kernel in C. The kernels cover the seven Dwarfs presented in the Berkeley vision. A tech report as well as the full code in C will be released soon. Note that all implementations are sequential. Contact: ADKaiser@lbl.gov.

- ***Contention-Aware Scheduling of Parallel Code for Heterogeneous Systems***
  Chris Gregg, Jeff S. Brantley, and Kim Hazelwood, University of Virginia

This work looks at how best to choose where a program needs to run: on the GPU or on the CPU. The assumption is that most kernels will prefer the GPU but it depends on whether the GPU is busy, the input size, the runtime of the baseline, the historical runtimes for the program, etc. Contact: chg5w@virginia.edu.

- ***Capturing and Composing Parallel Patterns with Intel CnC***
  Ryan Newton, Frank Schlimbach, Mark Hampton, and Kathleen Knobe, Intel

This work extends the CnC model by introducing modules which encompass an entire graph as a single step. This allows better reusability of code and modular building. CnC also introduces more schedulers for the tuning experts. The TBB scheduler is still the base scheduler, but there are now schedulers to specify task priorities, ordering constraints, and locality. Contact: ryan.r.newton@intel.com.

- ***General-Purpose vs. GPU: Comparison of Many-Cores on Irregular Workloads***
  George Caragea, Fuat Keceli, Alexandros Tzannes, and Uzi Vishkin, University of Maryland, College Park

This work presents a PRAM-on-chip vision with a full vertical integration from the PRAM model to the hardware implementation. XMT is the PRAM abstraction and XMTC is the C-like language built on top of it. The PRAM model provides speedup in many cases, as well as ease of programming. Furthermore, there is no need to reason about race conditions. This model has been tried in classes and people get it very quickly. Contact: {gcaragea,keceli,tzannes, vishkin}@umd.edu.

- ***Leveraging Semantics Attached to Function Calls to Isolate Applications from Hardware***
  Sean Halle, INRIA Saclay and University of California, Santa Cruz; Albert Cohen, INRIA Saclay

The need for continuity with past systems in parallel programming makes function calls very attractive (similar to OpenGL). Indeed, big changes are expensive and take time, and people feel comfortable with the way they were doing things before. After the code has been written to integrate the parallel function calls, a specializer can produce different implementations for each call depending on the platform. The code is therefore isolated from the platform. Furthermore, this specialization step happens after the main development cycle, which means that there is more time to do it right. Another important aspect of the model is the use of program virtual time to easily detect scheduling errors. The final aspect of the model is the use of interfaces to implement paradigms such as "divide work." The application implements an interface "how to divide" which the runtime can call with the number of chunks to produce, depending on the target platform. Contact: seanhalle@yahoo.com.

- ***Enabling Legacy Applications on Heterogeneous Platforms***
  Michela Becchi, Srihari Cadambi, and Srimat Chakradhar, NEC Laboratories America

The goal of this work is to enable the re-targeting of legacy applications to heterogeneous systems. The system uses libraries to catch certain system calls, and each platform can have its own library which implements the calls differently depending on the platform. Contact: mbecchi@nec-labs.com.

- ***OpenMP for Next Generation Heterogeneous Clusters***
  Jens Breitbart, Universität Kassel

This work is an extension of OpenMP. It works on shared memory systems and adds PGAS-like semantics for distributed memory systems. In that case, the runtime will seek to over-saturate the system to hide latencies. Annotations are done just as in OpenMP. Contact: jbreitbart@uni-kassel.de.

- ***Energy-Performance Trade-off Analysis of Parallel Algorithms***
  Vijay Anand Korthikanti and Gul Agha, University of Illinois at Urbana-Champaign

Energy is becoming a big issue: as performance increases, energy increases quadratically. For embarrassingly parallel applications, increasing the number of cores is good, as it results in better time and a quadratic decrease in energy. The problem, however, lies in the energy required to communicate. There is a sweet spot that optimally trades off communication energy and core energy. Two metrics are introduced: energy scalability under iso-performance and energy bounded scalability. The goal of this work is to determine the optimal number of cores based on the input size. Contact: vkortho2@illinois.edu.

- ***Prospector: A Dynamic Data-Dependence Profiler to Help Parallel Programming***
  Minjang Kim and Hyesoon Kim, Georgia Institute of Technology; Chi-Keung Luk, Intel Corporation

This work introduces Prospector, a profiling approach to dynamically determine data-dependencies. This greatly improves auto-parallelization. The main contribution of this work is the implementation of efficient compression of the profiling data. This produces much better results than Intel Parallel Advisor, for example. Contact: minjang@gatech.edu.

- ***Bridging the Parallelization Gap: Automating Parallelism Discovery and Planning***
  Saturnino Garcia, Donghwan Jeon, Chris Louie, Sravanthi Kota Venkata, and Michael Bedford Taylor, University of California, San Diego

This work introduces pyrprof, which is a profiler for parallelism. It relies on the idea that potential parallelism is the ratio of work and the length of the critical path. Pyrprof ranks regions of code based on their parallelism potential and reports this information back to the user. The programmer can provide feedback to improve the accuracy of the system. Pyrprof will soon be publicly available. Contact: http://parallel.ucsd.edu/pyrprof.

- ***Checking Non-Interference in SPMD Programs***
  Stavros Tripakis and Christos Stergiou, University of California, Berkeley; Roberto Lublinerman, Pennsylvania State University

This work is like Lint for CUDA. It uses an SMT solver to determine if there are interferences in blocks separated by __syncthreads. Contact: chster,stavros@eecs.berkeley.edu, rluble@psu.edu.

- ***Molatomium: Parallel Programming Model in Practice***
  Motohiro Takayama, Ryuji Sakai, Nobuhiro Kato, and Tomofumi Shimada, Toshiba Corporation

This framework allows easy parallel programming of platforms such as TVs. Mol is a C-like language that borrows characteristics from Haskell (functional and lazy evaluation). It describes the parallelism present. It is compiled to a bytecode. Atom describes the platform code (the target is mostly Cell). Contact: motohiro.takayama@toshiba.co.jp.

Posters below summarized by Rik Farrow (rik@usenix.org)

- ***DeNovo: Rethinking Hardware for Disciplined Parallelism***
  Byn Choi, Rakesh Komuravelli, Hyojin Sung, Robert Bocchino, Sarita Adve, and Vikram Adve, University of Illinois at Urbana-Champaign

The key concept here is that by creating disciplined software, problems in designing hardware will become simpler. They have written Deterministic Parallel Java as an exemplar. DPJ allows partitioning the heap into named regions, and language constructs define data dependencies between regions. Cache coherency becomes easier as the relationships between cache lines are spelled out in software, and message passing can be used for updating invalidated cache lines. Contact: denovo@cs.illinois.edu.

- ***Lock Prediction***
  Brandon Lucia, Joseph Devietti, Tom Bergan, Luis Ceze, and Dan Grossman, University of Washington

The authors wrote a trace generator that wrapped around the pthreads library to collect calls of lock acquisition functions. They investigated the PARSEC benchmark suite of multithreaded programs and performed offline analyses of the traces to predict the next thread to acquire a given lock. Using a handful of models for lock transitions, they tested the accuracy of each model against the traces for different programs. Each program had different lock acquisition characteristics, but past work has shown that accurate lock acquisition prediction does improve code performance. Their *most frequent transition* predictor model worked the best in general. Contact: http://sampa.cs.washington.edu.

- ***Resource Management in the Tessellation Manycore OS***
  Juan A. Colmenares, Sarah Bird, Henry Cook, Paul Pearce, and David Zhu, University of California, Berkeley; John Shalf and Steven Hofmeyr, Lawrence Berkeley National Laboratory; Krste Asanović and John Kubiatowicz, University of California, Berkeley

In Tessellation, applications and OS services are assigned to *Cells*, an abstraction that contains parallel software components and supplies resource guarantees. A two-level scheduler separates global resource allocations from local scheduling and resource usage. A policy service determines how resources are allocated to each Cell, and application-specific schedulers, such as Lithe, are responsible for scheduling threads within each Cell. At the global level, gang-level scheduling ensures that components within a Cell are available during scheduled runtime. Contact: yuzhu@eece.berkeley.edu.

- ***Processes and Resource Management in a Scalable Many-core OS***
  Kevin Klues, Barret Rhoden, Andrew Waterman, David Zhu, and Eric Brewer, University of California, Berkeley

ROS provides a new process abstraction, the manycore process (MCP). With MCP, there is only one kernel thread per process, rather than per thread, and cores provisioned to an MCP are gang-scheduled. Traditional system calls are asynchronous and non-blocking, and processes are notified before a core or other resource is revoked. Resources include anything that can be shared in a system: cores, RAM, cache, on-and off-chip memory bandwidth, access to I/O devices, etc. Contact: brho@eecs.berkeley.edu and yuzhu@eece.berkeley.edu.

JUNE 15, 8:30 A.M.–10:00 A.M.

Summarized by Chris Gregg (chg5w@virginia.edu)

- ***Dynamic Processors Demand Dynamic Operating Systems***
  Sankaralingam Panneerselvam and Michael M. Swift, University of Wisconsin—Madison

Sankaralingam Panneerselvam started by discussing the symmetric chip multiprocessor and why it does not support sequential workloads well. He then went on to show that the asymmetric chip multiprocessor satisfies diverse workloads well, but not as well as we would like. A dynamic multiprocessor, however, is flexible enough to adapt to the right configuration based on need. Dynamically variable processors lead to better performance with merging resources and shifting power and also lead to better reliability, because of the ability to have redundant execution.

Geoff Lowney asked why we need to reconfigure the OS. Panneerselvam said that an unexpected processor shutdown can lead to thread execution stopping (in the case of a lock, for instance) or other stalls. He then described Linux HotPlug, which allows dynamic addition or removal of a processor. This allows for partitioning and virtualization, and for physical repair of the processor. It can be used for long-term reconfigurations, which assumes that the processor will never come back online, and all relevant systems are notified.

Dan Grossman asked, “When you say ‘short-term reconfiguration,’ what time frame are we talking about?” Panneerselvam answered, “Milliseconds.” Performance is good for virtualization, but too slow for rapid reconfiguration. Next, Panneerselvam described the “processor proxy,” which is a fill-in for an offline processor. The proxy does not actually execute threads, but ensures that everything else continues. Proxies are not a long-term solution, but if the reconfiguration is long-term, it is better to move to a stable state. To do this, a "deferred hotplug" happens, which means that a CPU that is currently proxied is removed. A "parallel hotplug" can also happen, which is the reconfiguration of multiple CPUs. These methods provide greatly improved performance.

Timothy Roscoe asked, "Why is Linux the right OS to try this in? How much of this is about monolithic kernels?" and there was a long discussion about the role of the monolithic kernel. Panneer Selvam said that if we assume the virtual case, or a hypervisor, we want a number of things added to the OS in order to handle it. The OS wants to know about the changes, and we can implement those changes in Linux, in the monolithic kernel. Monolithic kernels aren't going away soon.

- ***Design Principles for End-to-End Multicore Schedulers***
  Simon Peter and Adrian Schüpbach, ETH Zurich; Paul Barham, Microsoft Research, Cambridge; Andrew Baumann, ETH Zurich; Rebecca Isaacs and Tim Harris, Microsoft Research, Cambridge; Timothy Roscoe, ETH Zurich

Simon Peter described the scheduler in Barrelfish, an experimental operating system. He started by asking why having two applications, one CPU-bound and one a barrier application, are a problem for OpenMP, in particular on a 16-core system. The barrier application shows decreased performance as the number of barrier threads increases. This is because of the increasing cost to execute the barriers. This situation works fine for a small number of threads, but eventually the performance drops significantly. Their approach mitigates this with gang scheduling and smart core allocation. Peter proposed an end-to-end approach, involving all components that can cut through classical OS abstractions, focusing on OS and runtime integration.

Peter then described the five design principles Barrelfish implements. First, he discussed time-multiplexing cores that offer real-time quality of service for interactive applications. David Patterson said, "There is a possibility in the future that we cannot turn on and off cores at will, because of power issues. Is it still worthwhile to use time-multiplexing instead of space-multiplexing?" Peter answered that actually, this is a perfect case for time-multiplexing, because you might have to time-multiplex the cores that you do have access to.

Peter then discussed scheduling at multiple timescales. He described the need for a small overhead when scheduling, because synchronized scheduling on each time-slice won't scale. This is implemented in Barrelfish with a combination of techniques, including long-term placement of applications on cores, medium-term resource allocation, and short-term per-core dispatch. David Patterson then asked, "What is the problem this is trying to solve?" Peter replied that they are trying to decouple things so we don't have to reschedule all the time. Barrelfish has phase-locked gang scheduling, which decouples schedule synchronization from dispatch. There may be a future re-sync necessary, but this happens at coarse-grained time scales.

Peter then outlined the "system knowledge base" in Barrelfish, which contains a rich representation of the hardware in the system. The OS and applications use this database. David Patterson asked, "Why did you go with a system knowledge base, which seems like a central bottleneck? Why didn't you make it a runtime database?" Peter answered that the hardware discovery information, boot-time micro-benchmarks, etc., go into the database. The data can be comprehensively queried, and the applications can use the database effectively. The centralized database was their first attempt, and it will be improved in the future.

JUNE 15, 10:30 A.M.–12:30 P.M.

Summarized by Rik Farrow (rik@usenix.org)

- ***OoOJava: An Out-of-Order Approach to Parallel Programming***
  James C. Jenista, Yong hun Eom, and Brian Demsky, University of California, Irvine

Jim Jenista described how they had created a version of Java that can add parallelism to serial programs. In this work, they added a single language construct, the reorderable block, or *rblock*, that designates portions of code that can be executed out of order. Rblocks can be executed as soon as all dependencies are satisfied.

The OoOJava compiler builds graphs between parent and child blocks and safely determines all data dependencies automatically. Jenista admitted that their implementation has several limitations, including a single exit point from each rblock. Dan Grossman immediately asked about exceptions, and Jenista answered that they use a subset of Java with no exceptions. He went on to describe a simple code example with two rblocks and explained the tree of dependencies that would be created, then walked, during execution. This graph shows that heap dependencies are properly handled, that all writes to a memory location occur in the same order.

Someone asked about virtual functions, and Jenista replied that they make a summary of all possible methods and combine them. Another person wondered if they had threads. Jenista answered that their subset has no threads, exceptions, global variables, or reflections. But, given a serial program, OoOJava creates a parallel program out of it.

David McCool asked how many lines of code this required, and Jenista said several thousand. They convert Java into C code that is compiled, resulting in a decent speedup. The code is available at http://demsky.eecs.uci.edu/compiler.php and includes other research features as well. David Padua wondered what happens if the compiler fails, and Jenista answered that the compiler reports that to you and suggests changes.

▪ ***User-Defined Distributions and Layouts in Chapel: Philosophy and Framework***
Bradford L. Chamberlain, Steven J. Deitz, David Iten, and Sung-Eun Choi, Cray Inc.

Brad Chamberlain described Chapel, a new language that supports parallelism. Chapel is part of the DARPA-led High Productivity Computing Systems program. The language is designed to improve the programmability, robustness, and performance of parallel programs and targets both multi-core and commodity cluster systems. You can download the source code from http://sourceforge.net/projects/chapel/.

Parallelism and data locality are driving concerns in Chapel. Chapel includes notation for arranging data in arrays and how data parallel operators should be implemented. McCool asked about their strategy for vector instructions, and Chamberlain responded that they haven't created vector compilers, but generate C code.

Chamberlain then explained domains and domain mappings. Domains takes lists of indices, and domain maps specify how the data will be accessed—for example, with a blocking factor or by tiling. An example mentioned a *zippered* domain map, and someone asked what "zippered" meant. Chamberlain explained that you would use zippering to suggest to the compiler which iterator to use when you have two domains with different layouts.

Chamberlain said that Chapel includes a user-defined domain-map framework and that at Cray they use this framework themselves. They don't want to have an unfair advantage using a tool that is publicly funded. McCool asked if the compiler can convert nested multiple arrays, and Chamberlain answered, not currently, but there are default domain maps you can use to support this explicitly. Dan Grossman asked if using Chapel avoids static analysis, and Chamberlain said that you still have to do this yourself. Grossman said that you expose this, but Chapel does not understand it, and Chamberlain agreed. He said that you want to implement the right domain maps whenever possible.

One goal of Chapel is not to impose arbitrary limitations. They do want to support separation of roles, with parallel experts writing domain maps and others using them. Chapel does support both CPUs and GPUs. They have compared Chapel to CUDA and gotten the same performance using a smaller code base. Sarita Adve asked about loads and stores, and Chamberlain responded that they support normal C indexing and that memory consistency is incredibly relaxed. The programmer is responsible for arranging copying data between main memory and the GPU.

Grossman asked if the goal of Chapel is to become popular or develop new language structures, and Chamberlain answered that either would be satisfactory. The main goal is to make users more productive. Grossman asked about status. Chamberlain said that performance is not good enough yet, but please try Chapel and provide feedback. You can find the slides for this presentation at http://www.usenix.org/events/hotpar10/tech/slides/chamberlain.pdf.

▪ ***On the Limits of GPU Acceleration***
Richard Vuduc, Aparna Chandramowlishwaran, Jee Choi, Murat Efe Guney, and Aashay Shringarpure, Georgia Institute of Technology

Richard Vuduc started his talk with a quote: always compare your results with scalar, unoptimized Cray code, as this will make your code look good. He then said that his paper was more of a survey, perhaps a story. The story begins with Scott Klasky posing a question: should I port my application to GPUs? A quick literature search turns up amazing speedups, 30–100 times faster than running on a modern CPU. What Vuduc and his fellow researchers found was something very different.

Vuduc pointed out that current GPUs are bandwidth-bound, as they sit on the PCIe bus. A related issue has to do with memory access patterns. McCool asked if working-set size matters, and Vuduc said that has some influence. Even with the GPU on the same die as the CPU, there could still be bandwidth issues. Patterson asked if he was suggesting a second memory, and Vuduc pointed out that you might need to keep GPU memory even in the on-die version.

The bottom line is that with code properly tuned to run on a multicore system, like a Nehalem, the big exciting differences fade away. The authors tried three different scientific computations: (1) iterative sparse linear solvers, (2) sparse Cholesky factorization, and (3) the fast multipole method. Geoff Lowney asked how much work was involved in tuning, and Vuduc said that someone spent perhaps one month of work, producing roughly twice the number of lines of code, in tuning one application. Lowney then wondered if the NVidia GPU code also represented tuned code and Vuduc said they were well tuned, with NVidia's cooperation.

In the sparse matrix and fast multipole methods, the issue is clearly bandwidth related. Andrew Bauman asked if pipelining would help, and Vuduc said that a student is working on that. By tuning code, they found that a multiple core version on Nehalem was only 10% slower than a dual GPU version. In summary, one GPU is roughly equal to one CPU. If you look at power, CPUs are better. Someone asked why it was easier to gain so much speedup on GPUs? Vuduc answered that it isn't really, that it took an equal amount of effort to tune and prepare code for either GPU or CPU.

JUNE 15, 2:00 P.M.–4:00 P.M.

Summarized by Romain Cledat (romain@gatech.edu)

▪ ***Gossamer: A Lightweight Programming Framework for Multicore Machines***
Joseph A. Roback and Gregory R. Andrews, The University of Arizona, Tucson

Gossamer is a framework for annotating existing applications to make them parallel. Roback first presented the 15 annotations that compose Gossamer. The annotations are meant to encompass as large a domain as possible and support task spawning through constructs such as fork, parallel, divide and replicate, memory synchronization with join, barrier, atomic, buffered, copy, ordered, shared, and the map-reduce paradigm. Roback then illustrated the annotations with a variety of well-known examples. In the n-queens problem, he showed how fork and join could be used. In bzip2, the presenter also demonstrated the "ordered" keyword, which allows a serialized join in the order of spawning.

Roback briefly described the source-to-source translator that is used to compile down the annotations and generate bookkeeping artifacts. For example, the translator tries to limit the number of locks required to enforce "atomic" sections by finding the best middle ground between one global lock and one lock per variable.

Roback then described the runtime involved. The application-level threads are referred to as "filaments" and are stackess and stateless, making them extremely lightweight. The filaments share the stack of the thread they are running on. A member of the audience asked if, once placed on a thread, a filament had to run till completion. Roback said yes, at this time, as the threads are stackless, but the authors are exploring medium-weight threads that could be interrupted. David Padua asked about the producer-consumer paradigm and Roback said this was also future work. Recursive and task filaments are scheduled in a round-robin fashion: iterative filaments are scheduled in groups to maximize cache locality, and domain decomposition filaments are scheduled statically with one filament per processor.

Results were presented that demonstrated the very low overhead of the system. The super-linear speedup in the matrix-multiplication benchmark is due to the fact that when the benchmark runs on two sockets, it has a larger L2 cache. Results also showed Gossamer comparing positively to Cilk and OpenMP in most situations.

In conclusion, Gossamer is a simple portable framework and the translator is available as a stage in the compilation process and can therefore be simply plugged into GCC or ICC.

David Patterson asked if they were thinking about trying out larger applications (like the Dwarfs and the implementation presented at this year's HotPar). Roback answered that the goal was to try to fit as many applications as possible. David Padua asked why OpenMP was so much slower at times than Gossamer, since the approach seemed similar. It's because the task implementation in OpenMP is currently not very good.

- ***Reflective Parallel Programming: Extensible and High-Level Control of Runtime, Compiler, and Application Interaction***
  Nicholas D. Matsakis and Thomas R. Gross, ETH Zurich

Matsakis presented the concept of "reflective parallelism," which he describes as a program's ability to reason about its own schedule at runtime. Consider, for example, two tasks "A" and "B." Questions that could be answered with reflective parallelism are: "Do A and B always run in parallel?" and "Must A finish before B starts?" The results from queries should return results that hold true for all executions and the program should be able to dynamically modify the schedule by adding scheduling constraints. Matsakis stated that reflective parallelism could be used for many things, from schedule visualization to testing frameworks to data-race detection. In this paper he focused on data-race detection.

Matsakis then exposed the big problem with current threads: they construct their schedule through primitives such as "start," "join," and "wait," but the schedule is therefore never explicit until after the whole program has executed. Even after the program has run, it is nearly impossible to analyze the schedule and come up with assertions that are always true. Reverse-engineering the program to build the schedule is also risky.

Matsakis then introduced his answer to these problems: make the schedule a first-class entity in the program with the use of intervals where their use can express the schedule through declarative methods. The three concepts captured by the model are: (1) intervals that represent an asynchronous task or group of tasks; (2) points that represent the start and end of intervals (the point right before an interval and right after) on which "HappensBefore" relationships can be specified; and (3) locks that can be held by intervals to specify a constraint without imposing an order. Alexandra Federova asked how this was different from TBB, and Matsakis responded that although a task-graph existed in TBB, it was more low-level with reference counts and was thus not a first-class entity.

Matsakis then briefly described the scheduling model where a "ready()" method expresses to the runtime that an interval is ready to run. "HappensBefore" relationships can be added dynamically, but they cannot be removed, which guarantees monotonicity and makes scheduling easier.

Finally, Matsakis defined how reflection can be used to specify "guards" on data objects. Guards can evaluate a condition based on information gleaned through reflection to determine whether the object they are guarding can be accessed. Many of these conditions can be known at compile time, but even if they cannot, they can be quickly evaluated at runtime and warn the user of any data-race.

In summary, the intervals framework, available at http://intervals.inf.ethz.ch, allows users to specify access conditions using information reflected back about the schedule.

Geoff Lowney asked how the system handles the case where there is no guard on an object. Matsakis responded that the framework mandates a guard for all fields. Another person asked how to know if this is the right way to proceed. Matsakis said it was a tough question to answer but that they had tried this model in undergrad classes with success. Finally, a member of the audience asked how many of the checks were dynamic. Matsakis answered that many checks could be done statically and, as is the case for most type systems, some small restructuring of the program can expose even more static checks.

- ***Task Superscalar: Using Processors as Functional Units***
  Yoav Etsion, Barcelona Supercomputing Center; Alex Ramirez, Barcelona Supercomputing Center and Universitat Politècnica de Catalunya; Rosa M. Badia, Barcelona Supercomputing Center; Eduard Ayguade, Jesus Labarta, and Mateo Valero, Barcelona Supercomputing Center and Universitat Politècnica de Catalunya

In this talk Etsion presented the idea of extending out-of-order instruction pipelines to tasks to aid in exposing the operations that can execute in parallel and manage data synchronization. Indeed, for many years out-of-order pipelines have been managing parallelism in a sequential stream of instructions. Although ILP does not scale well, due to the problems of building a large instruction window (difficulty with building a global clock, as well as the limited scalability of dependency broadcasts) and the unpredictability of control-paths, Etsion believes that out-of-order task parallelism may work better.

The presenter then moved on to explain the StarSS programming model, where tasks are modeled as abstract instructions. A master thread spawns the various tasks encountered, which are dispatched to the worker processors. A runtime dynamically resolves dependencies and constructs a task-graph. It is important to note that the task-graph can get very complicated very quickly but that StarSS can build it and exploit it efficiently.

The need to do the task-decoding and scheduling in hardware is due to the high latency of software (between 700ns and 2.5us).

The model of execution is very similar to that of out-of-order instruction execution: tasks are decoded and pushed to reservation stations. Data dependencies are taken care of in the same way as for instructions. Etsion showed results that demonstrated that parallelism could be uncovered in many scientific applications.

Etsion explained that task parallelism will scale more than ILP, for a variety of reasons. Firstly, broadcasts do not have to be used, since the latencies involved are much higher. Dependencies can therefore be dealt with using point-to-point communication, which is much more scalable. Secondly, there is no need for a global clock. Thirdly, the multiplex reservation stations allow multiple tasks in the same data structure, making the representation much more compact. Tasks also are not speculative, although the authors are looking at task predication.

As future work, the authors wish to exploit locality-based scheduling and also to gather tasks using a similar kernel and package them off to a GPU. They also wish to explore which instruction-level optimizations can be applied.

David Padua asked if tasks can interrupt each other. At this point they cannot, but nothing verifies that this is the case. Another audience member asked how energy-efficient the model is. It is difficult to predict, although it seems to be more efficient than having a dedicated big core decode and schedule the tasks.

REAL SOLUTIONS FOR REAL NETWORKS

Each issue delivers technical solutions to the real-world problems you face every day.

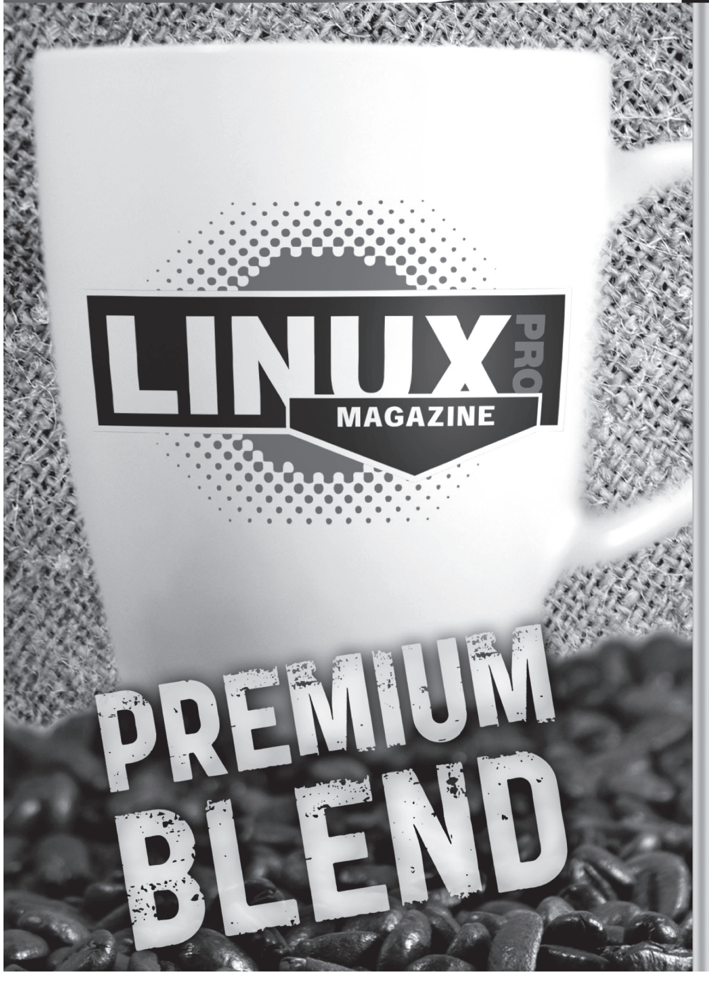
Learn the latest techniques for better:

- network security
- system management
- troubleshooting
- performance tuning
- virtualization
- cloud computing

on Windows, Linux, Solaris, and popular varieties of Unix.

FIND ADMIN MAGAZINE ON A NEWSSTAND NEAR YOU!
OR ORDER ONLINE AT *www.admin-magazine.com/subs*

Supplies are limited so order your copy today!



LINUX PROS READ
LINUX PRO

Enjoy a rich blend of tutorials, reviews, international news, and practical solutions for the technical reader.

Subscribe now to receive:

3 issues
+ 3 DVDs
for only

$3.00!

www.linuxpromagazine.com/trial

USENIX

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

NOV. 7–12 2010

24TH LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE

SAN JOSE CALIFORNIA

LISA '10

UNCOVERING THE SECRETS OF SYSTEM ADMINISTRATION

Program includes:
Unraveling the mysteries of Twitter infrastructure, legal issues in the cloud, huge NFS at Dreamworks, and a keynote address by Tony Cass, *CERN*

Join us for 6 days of practical training plus a 3-day technical program including invited talks, refereed papers, workshops, a vendor exhibition, and more.

REGISTER AT WWW.USENIX.ORG/LISA10/LG BY OCTOBER 18 AND SAVE!

Stay Connected... http://www.usenix.org/facebook/lisa10 http://twitter.com/LISAConference