

login:

APRIL 2013

VOL. 38, NO. 2



Sysadmin

↻ **Theia: Visual Signatures for Problem Diagnosis in Large Hadoop Clusters**

Elmer Garduno, Soila P. Kavulya, Jiayi Tan, Rajeev Gandhi, and Priya Narasimhan

↻ **Interview with Cory Lueninghoener: HPC**

Rik Farrow

↻ **Analyzing Network Traffic with Chimera**

Jonathan Springer, Kevin Borders, and Matthew Burnside

Columns

Practical Perl Tools: What's Up, Perldoc?

David N. Blank-Edelman

Some Easily Overlooked, But Useful Python Features

David Beazley

Nagios XI (Part 3)

Dave Josephsen

/dev/random: Cause and Effect

Robert G. Ferrell

Book Reviews

Elizabeth Zwicky, Mark Lamourine, and Rik Farrow

Conference Reports

LISA '12: 26th Large Installation System Administration Conference

LISA '12: Advanced Topics Workshop

LISA '12: Real World Configuration Management Workshop



UPCOMING EVENTS

HotOS XIV: 14th Workshop on Hot Topics in Operating Systems

May 13–15, 2013, Santa Ana Pueblo, NM, USA
www.usenix.org/conference/hotos13

2013 USENIX Federated Conferences Week

June 24–28, 2013, San Jose, CA, USA
www.usenix.org/conference/fcw13

USENIX ATC '13: 2013 USENIX Annual Technical Conference

June 26–28, 2013
www.usenix.org/conference/atc13

ICAC '13: 10th International Conference on Autonomic Computing

June 26–28, 2013
www.usenix.org/conference/icac13

HotPar '13: 5th USENIX Workshop on Hot Topics in Parallelism

June 24–25, 2013
www.usenix.org/conference/hotpar13

UCMS '13: 2013 USENIX Configuration Management Summit

June 24, 2013
www.usenix.org/conference/ucms13

Feedback Computing '13: 8th International Workshop on Feedback Computing

June 25, 2013
www.usenix.org/conference/feedback13

ESOS '13: 2013 Workshop on Embedded Self-Organizing Systems

June 25, 2013
www.usenix.org/conference/esos13

HotCloud '13: 5th USENIX Workshop on Hot Topics in Cloud Computing

June 25–26, 2013
www.usenix.org/conference/hotcloud13

WiAC '13: 2013 USENIX Women in Advanced Computing Summit

June 26–27, 2013
www.usenix.org/conference/wiac13

HotStorage '13: 5th USENIX Workshop on Hot Topics in Storage and File Systems

June 27–28, 2013
www.usenix.org/conference/hotstorage13

HotSWUp '13: 5th Workshop on Hot Topics in Software Upgrades

June 28, 2013
www.usenix.org/conference/hotswup13

USENIX Security '13: 22nd USENIX Security Symposium

August 14–16, 2013, Washington, D.C., USA
www.usenix.org/conference/usenixsecurity13

Workshops Co-located with USENIX Security '13

JETS: *USENIX Journal of Election Technology and Systems*

A new “hybrid” journal + conference in conjunction with EVT/WOTE '13
www.usenix.org/conference/jets
Submissions due: April 17, 2013

EVT/WOTE '13: 2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections

August 12–13, 2013
www.usenix.org/conference/evtwote13

CSET '13: 6th Workshop on Cyber Security Experimentation and Test

August 12, 2013
www.usenix.org/conference/cset13
Submissions due: April 25, 2013

HealthTech '13: 2013 USENIX Workshop on Health Information Technologies

Safety, Security, Privacy, and Interoperability of Health Information Technologies
August 12, 2013

www.usenix.org/conference/healthtech13
Submissions due: May 6, 2013

LEET '13: 6th USENIX Workshop on Large-Scale Exploits and Emergent Threats

August 12, 2013
www.usenix.org/conference/leet13
Submissions due: May 8, 2013

FOCI '13: 3rd USENIX Workshop on Free and Open Communications on the Internet

August 13, 2013
www.usenix.org/conference/foci13
Submissions due: Monday, May 6, 2013

HotSec '13: 2013 USENIX Summit on Hot Topics in Security

August 13, 2013

WOOT '13: 7th USENIX Workshop on Offensive Technologies

August 13, 2013
www.usenix.org/conference/woot13
Submissions due: May 2, 2013

LISA '13: 27th Large Installation System Administration Conference

November 3–8, 2013, Washington, D.C., USA
www.usenix.org/conference/lisa13
Submissions due: April 30, 2013

;login:

APRIL 2013 VOL. 38, NO. 2

EDITORIAL

- 2** Musings *Rik Farrow*

OPINION

- 6** Is DevOps the Future of Sysadmin? *Mark Burgess*

SYSADMIN

- 8** Theia: Visual Signatures for Problem Diagnosis in Large Hadoop Clusters
Elmer Garduno, Soila P. Kavulya, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan
- 14** Cory Lueninghoener on Managing HPC Clusters: An Interview
Rik Farrow
- 16** Wireless Means Radio
David Lang

FILESYSTEMS

- 20** Improving the Performance of fsck in FreeBSD
Marshall Kirk McKusick

SECURITY

- 24** Analyzing Network Traffic with Chimera
Jonathan Springer, Kevin Borders, and Matthew Burnside

COLUMNS

- 30** Practical Perl Tools *David N. Blank-Edelman*
- 34** Some Easily Overlooked But Useful Python Features
David Beazley
- 39** iVoyeur *Dave Josephsen*
- 42** /dev/random *Robert G. Ferrell*

BOOKS

- 45** Book Reviews *Elizabeth Zwicky, with Mark Lamourine and Rik Farrow*

CONFERENCE REPORTS

- 48** LISA '12: 26th Large Installation System Administration Conference (LISA '12)
- 71** LISA '12: Advanced Topics Workshop
- 75** LISA '12: Real World Configuration Management Workshop



EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Rikki Endsley
rikki@usenix.org

COPY EDITOR
Steve Gilmartin
proofshop@usenix.org

PRODUCTION
Arnold Gatilao
Casey Henderson
Michele Nelson

TYPESETTER
Star Type
startype@comcast.net

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

www.usenix.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 a subscription to *;login:*. Subscriptions for non-members are \$90 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.

©2013 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.



Rik is the editor of `;login:`.
rik@usenix.org

Arguing with Mark Burgess is easy. When I read his article in this issue, I found myself agreeing with some parts and disagreeing with others. System administration has changed in many ways, but some things will remain the same.

In his opinion piece, Mark makes a point that he has made before: that sysadmins need to become *business relevant*. Although I understand his point, I also doubt that any sysadmin (who wasn't fired) was ever business irrelevant.

The Dark Ages

I began working as a sysadmin in 1984, managing single systems for a handful of people. I co-wrote (with Becca Thomas) one of the first books on system administration [1], starting at a point when the people who were actually doing the work at UC Berkeley had never even heard of the term “system administration.” This was the same tack taken by Becca Thomas earlier when she wrote the first wildly successful tech book about UNIX [2]. Becca had looked at the logs created by process accounting to discover which UNIX tools people most commonly used at UC Berkeley and wrote about them. I went to UCB to collect data on what sysadmins were actually doing, but as there was no way to distinguish what ordinary users and sysadmins did via process accounting (besides using `su`), I interviewed sysadmins.

Not so surprisingly, some things really haven't changed. Those sysadmins were managing user accounts, managing printer queues, and dealing with disk issues. One issue that has largely gone away were techniques for preventing disk volumes from filling up. But others, such as backup strategies or migrating users to balance load, are still common today.

To support my research, I consulted as a sysadmin for small companies using UNIX. In those days, most of the work involved AT&T's System V, as you could not run UNIX without a license from AT&T. I also worked for a software development company that would get different UNIX-based workstations for the purpose of porting their software. Many of these computers were BSD-based, which, then as now, meant differences in pathnames, directory hierarchies, and even the formats of configuration files. Thanks to included online documentation (man pages), getting systems set up and ready for work was possible, with some head-scratching. The real oddballs were HP's HPUX, IBM's AIX, and Apollo's DomainIX, in increasing order of weirdness.

At this time, networks consisted of serial lines. I became an expert at installing RS-232 connectors and still have a breakout box (used for debugging RS-232). Offices had serial-connected terminals, and the software company used UUCP to copy files between computers, which wasn't as bad as it might seem. Sure, having a communication speed of 9600 to 19200 baud makes it seem like file transfers would take forever: a one megabyte transfer could take 20 minutes! But files were smaller then, too.

The Business

The point of this history is not how primitive things were. Things were interesting, but they did work. The real point is that none of this would have been done *unless I was performing some task required by the business that had hired me*. None of it was business irrelevant.

Having written that, I will readily admit that things have changed. Not just that communications are blazingly fast, disks amazingly huge, and computers incredibly faster and cheaper. Rather, we now have so many systems to manage. Instead of managing a couple of office systems, or tens of workstations, some people now manage clusters of thousands, even tens of thousands of servers. So doing what I did in the 1980s just doesn't scale; you cannot sit in front of a terminal and use the command line to configure thousands of servers. You need automation.

Although I had moved on to security by the late 1980s, I really began to see the changes when I researched patch management for NASA in 2002. I had heard that the San Diego Supercomputing Center had centralized patch management, so I flew out there to take a look. There were just four people managing nearly 300 UNIX workstations, and seven people managing 400 Windows desktops and a few servers. The UNIX folks were using Cfengine, my first real exposure to Configuration Management (CM), whereas the Windows sysadmins were using Microsoft's proprietary image management system, SMS [3].

This is where I find myself really agreeing with Mark. If SDSC had opted to build their own CM tools, I doubt that things would have worked out so well. The real difficulties were that not all systems were the same. And it's these differences that will always ensure that there are sysadmins in the loop. Some differences include having to run different patch levels, or even different versions of OS software to support applications, while others have to do with differences in things as diverse as hardware used and management style. And by management, I do mean the bosses.

The Lineup

Mark Burgess is first up, with his article about the future of sysadmin. You will notice that Mark is wondering whether DevOps is the future, not the relatively narrow focus that I decided to pick on. I've argued privately with Mark before (and recently), but a lot of what I've fired off to him has more to do with his choice of words than the ideas behind them. Mark has both thought and written a lot about this topic, and his article deserves careful reading.

Garduno et al. have produced an article based on their award-winning LISA paper about a means of visually monitoring Hadoop clusters. They convincingly argue that having visual consoles that convert the output of Hadoop logs into different presentations makes understanding where problems lie much simpler. For example, they show how you can tell the difference between node (hardware) issues and problems with a particular MapReduce job.

I interviewed Cory Lueninghoener, a strong advocate for CM, but not about CM, but rather about supercomputers. While one

might think that a Hadoop cluster is a type of supercomputer, supercomputers are really quite different. They use specialized hardware, software, and operating systems, and focus on computation rather than data crunching, what Hadoop clusters are designed to do. Cory provides interesting insights into his world, where CM goes beyond being just a management tool, but also a form of audit.

David Lang presents his research on setting up WiFi networks at the SCALE Linux conference. While most of us will not be doing anything like this, David, who is also a licensed radio operator, explains that setting up multiple access points (APs) *and* having them function efficiently relies on understanding how radios work. David has some simple, down-to-earth tips for how best to configure multiple APs.

Kirk McKusick writes about the changes he has made to the FreeBSD 8 fast filesystem (FFS) and fsck. Kirk was attending FAST 2013, and the very first paper was about changes to the Linux ext3 filesystem and fsck that could greatly speed up filesystem checking. Kirk was inspired, and went off that very afternoon and wrote patches to take advantage of some of the ideas presented.

Springer et al. explain the tool they have created to analyze network traffic, Chimera. We typically use IDS software, like Snort or Bro, for monitoring network traffic. The trouble with these tools is that they are either too simple (Snort) or hard to program (Bro) to analyze patterns of traffic that extend across millions of packets. Chimera helps with this through the use of an SQL-like programming language that outputs optimized code that works with Bro. The authors hope that their tool will make it easier for non-specialists to extract useful information from network logs.

David Blank-Edelman educates us about Perl documentation. I know, sounds boring. But it's not. David explains the cool things you can do using perldoc, as well as adding perldoc annotations to the Perl code you create.

Dave Beazley has decided not to take another deep dive into Python. Dave has been working on the *Python Cookbook* [4] and shares some of his own surprises based on his work revising the book. As always, I sit with the Python prompt waiting, trying out the tricks that Dave exposes.

Dave Josephsen finishes his three part series on Nagios XI, a commercial Nagios product that answers most of the criticisms made by some people against Nagios. Dave begins with his usual rant, this time against "religious battles," before pointing out just how XI solves the objections many people have to using Nagios.

Robert Ferrell muses about cause-and-effect, as in, why aren't more people aware of the harm that ignoring this causes? Although texting while driving can produce spectacular results (and I don't mean increases in Twitter followers), Robert also

takes a hard look at how many people respond to having someone point out security vulnerabilities to them. We are, after all, a society that discretely points out unzipped zippers, so why men get their panties in a bunch when unzipped Internet applications get pointed out is beyond both Robert and me.

Elizabeth Zwicky has written wonderful reviews of five books for this issue. She begins with one of her favorite topics, learning statistics, then covers several books on programming, and a new Donald Norman book. Mark Lamourine covers a Maker book about the Raspberry Pi. The more I learn about the Pi, the more interesting it sounds. I've also written a review of a book about learning vim, the replacement for vi.

We finish up this issue with summaries from LISA 2012. USENIX has started posting summaries to the *login*: portion of the Web site as soon as they are ready, instead of waiting until the next print issue of *login*: comes out. This can mean that summaries will appear weeks before they would have back when we waited for the issues to appear in the mail, or show up on the Web site [5].

The pace of software development has sped up enormously, with the big data companies, like Google and Facebook, pushing out new versions of their software every couple of weeks. These updates would not be possible without the existence of something that gets called DevOps, and it is totally unlike the world of sysadmin that I researched back in the mid-'80s.

But not every company or organization is like Google or Facebook. Not everyone manages to be an amazing programmer or infrastructure engineer (to borrow a term from Mark). For many businesses, a slower, steadier pace works well enough. And that's a good thing, as the supply of geniuses is pretty limited. While it is important to follow best practices, it is just as important to realize the limitations of the actual workforce, and not expect that everyone will be a heroic programmer or sysadmin.

Editor's Note: With the April 2013 issue, we have changed to a two column layout for articles and columns. Two columns has been the standard layout for conference reports for many years now. This new layout gets rid of most of the white space present in the wide margins, and leaves us with room for more articles and columns without increasing page count, that is, using more paper.

The April issue is also the one I have the hardest time filling every year. Print magazines have long lead times, and the time for writing April articles falls during December and early January. Few people are willing to write over the Christmas holidays, and once January begins, they are busy catching up after a week (or more) spent dealing with the holidays.

I also underestimated the effect of the layout changes, as my goal is to produce a great magazine everytime, while not going beyond a set page length. I plan on taking advantage of the additional space in future issues.—*Rik*

References

- [1] R. Thomas, R. Farrow, *Unix Administration Guide for System V* (Prentice-Hall, 1989): <http://www.amazon.com/Unix-Administration-Guide-System-V/dp/0139428895>.
- [2] R. Thomas, *A User Guide to the Unix System*, 2nd edition (McGraw-Hill Osborne Media, 1984).
- [3] SMS, now, System Center Configuration Manager: http://en.wikipedia.org/wiki/System_Center_Configuration_Manager.
- [4] A. Martelli, A.M. Ravenscroft, D. Ascher, *The Python Cookbook* (O'Reilly Media, 2005): <http://www.amazon.com/Python-Cookbook-Alex-Martelli/dp/0596007973>.
- [5] *login*.: <https://www.usenix.org/publications/login>.

SAVE THE DATE!



LISA '13

27th Large Installation
System Administration Conference

Sponsored by USENIX
in cooperation with LOPSA

NOVEMBER 3-8, 2013 | WASHINGTON, D.C.

www.usenix.org/lisa13

Come to LISA '13 for training and face time with experts in the sysadmin community.

LISA '13 will feature:

6 days of training on topics including:

- Configuration management
- Cloud Computing
- Distributed Systems
- DevOps
- Security
- Virtualization
- And More!

Plus a 3-day technical program:

- Invited Talks
- Guru Is In sessions
- Paper presentations
- Vendor Exhibition
- Practice and Experience reports
- Workshops
- Posters and Lightning Talks

New for 2013! The LISA Labs "hack space" will be available for mini-presentations, experimentation, tutoring, and mentoring.

Check out the Call for Participation! Submissions are due Tuesday, April 30, 2013.

www.usenix.org/lisa13/cfp

Stay Connected...



twitter.com/usenix



[www.usenix.org/facebook](https://www.facebook.com/usenix)



www.usenix.org/youtube



[www.usenix.org/linkedin](https://www.linkedin.com/company/usenix)



www.usenix.org/gplus



www.usenix.org/blog

Is DevOps the Future of Sysadmin?

Bemoaning the Failures of the Sysadmin Profession

MARK BURGESS



Mark Burgess is the CTO and Founder of CFEngine, formerly professor of network and system administration at Oslo University College and the principal author of the Cfengine software. He is the author of numerous books and papers on topics ranging from physics to network and system administration and including fiction. mark.burgess@cfengine.com

Last year, Doug Hughes and Tom Limoncelli had the foresight to choose “DevOps” as the theme of the LISA ’11 conference. For some at the time, this was a controversial choice. Indeed, for a long time it had not been clear what DevOps even was about. Like “cloud,” DevOps had been suspiciously vaporous, vaguely connected to Web operations, and there was brewing skepticism that there was anything new, more a crowd of novices re-learning old lessons; however, Ben Rockwood’s excellent keynote at LISA ’11 changed that, in many minds.

For the first time, in my view, Ben convincingly linked DevOps to a topic that has been close to my own heart for several years: the extent to which system administration is business relevant in the modern world (actually a topic introduced by my friend Claudio Bartolini of HP research as early as 2006). For the first time, I realized that this grassroots movement in the IT world was saying the two important things:

- ◆ System administration practice, as a culture or profession, is holding us back from doing business fast enough.
- ◆ There is a way for the profession to metamorphose from larva into butterfly by getting sysadmins out of their dungeons and integrating them into the business value chain.

From Skill to Discipline

Since I started writing about sysadmin some 20 years ago now, I tried to crystallize the essence of sysadmin as a discipline, and even usher it with science in the direction of engineering. My views often polarized people—they tended to love or hate the message, because at the scale of the 1990s one could often get away with clinging to the old ways—manual command-prompt legerdemain for any ailment. Today, however, necessary scale and complexity are business imperatives that are forcing the new ways into even the most conservative industries.

When I look at system administration over my career, I see a profession that has simply failed to move forward in those 20 years. The identity and values of the system administrator are basically the same as they were when I started in the field: a pretty closed world of “Do It Yourself,” and then do it over again, fighting dragons by command line. Those who managed to embrace the modern architectural methods of science and engineering built the new super-sized IT-based companies of today. I elevate them to the status of infrastructure engineers, masters of the available tools of predictable automation and modeling. They moved from reinventing every wheel, to a mature commoditization of infrastructure. Some of them are even selling this infrastructure as a service today for the benefit of others.

Recently I have manned myself up to level these fairly harsh accusations in public, and braced for an onslaught of unmitigated flames and hostility. But surprisingly it didn’t come. Remarkably, most sysadmins I say this to ruefully acknowledge that this is the case. True enough, some technologies have changed, certain tools have come and gone, but the basic methodology of do-it-yourself technology quilt-work still pervades a majority of sysadmin practice. Sysadmins need a new identity that doesn’t involve remaking wheels for every

occasion. Imagine if each time a company needed to expand, everyone picked up tools and began building furniture for the new employees instead of going to IKEA? Well, we still do this with computers.

A Profession

Organizations such as SAGE and LOPSA seemed to lose their way, too; by trying to “unionize” the profession, they effectively sent the message that sysadmins just felt poorly treated and underrepresented when they could have led the march to modernize practices and be the heroes of IT emancipation. In fact, the profession as a whole simply failed to adapt to the needs of the rapidly expanding IT industry. Perhaps, if sysadmins had taken on the mantle of responsibility for integrating into business processes, that might have led to their rising up the pay-scale automatically. But system administration has remained, for many, an introverted gaming occupation. Now it needs to become a more disciplined engineering profession. And history is in danger of repeating itself with a new generation of junior admins and impatient developers working with the cloud, or with new scripting frameworks for automation.

Developers and Sysadmins Work Together

The term DevOps was coined by Patrick Debois while working as a consultant helping to deploy applications. He observed that Developers and Operations people were often mistrustful of one another, and that this often led to delays and problems. The answer was to promote a culture of inter-departmental cooperation. After all, the languages of programmers and sysadmins are not that far apart.

When DevOps came along and saw the clash of old and new, they publicly shook both parties by the lapels, saying: act like specialists who respect each other. Developers represent business value, and need to make rapid changes to cope with modern online commerce. Sysadmins (operations) experts know more about security and configuration and how to do the job properly. So folks, work together (damn-it)! By working closely, sysadmins become the heroes who deploy quickly and developers learn how

to write for real-world systems instead of merely dumping their code onto sysadmins with a “Deploy this!”

The pace of change is picking up in the industry today. System administration in the old sense (caught in a poverty trap of firefighting and lurching from crisis to crisis, because of lack of holistic thinking) will become extinct because business can't afford it. It will be replaced by a smaller core of infrastructure engineers who can think in larger terms than following “how-tos,” or they will go to the IKEA of infrastructure—the rapidly improving cloud providers.

The future is not really about replacing humans with machines; it is about respecting the role of humans, their cooperation and their creativity. People should not be logging onto computers by hand to debug and diagnose alarms, any more than farmers should be reaping the harvest with a scythe. DevOps says, if you lay out your corn properly, I can get it to market ten times as fast. That is what makes the industry go around, and the industry will pay a premium for it.

Summary

System administration cannot survive in its present form forever. As technologies go mainstream, jobs change. Look at the Internet business, which has gone from being a specialized engineering task to a commoditized split between a few “architects” and a lot of “cable guys” who install the box. Automation will change the face of IT over the next ten years in a similar way.

So what can we learn from DevOps? Well, it did not set out to become a panacea model for operations, or even be an alternative to system administration—but it is evolving faster than system administration into a respectable, modern engineering culture for business optimization. The term system administration is already falling from favor as a concept. Site reliability engineer is, after all, a more compelling title.

There is something to learn from DevOps. There is time to mold it, to make a difference. So, never mind the name. Change it, if you like, but listen to what it is saying. We need it.

Theia: Visual Signatures for Problem Diagnosis in Large Hadoop Clusters

ELMER GARDUNO, SOILA P. KAVULYA, JIAQI TAN, RAJEEV GANDHI,
AND PRIYA NARASIMHAN



Elmer Garduno is the Principal Big Data Architect at UPMC's Technology Development Center. He holds a Master's degree on very large information systems from Carnegie Mellon University. Elmer's main focus is on large scale language technology systems. elmerg@cs.cmu.edu



Soila Kavulya is a PhD student at Carnegie Mellon University. Her research focus is primarily on problem diagnosis in large distributed systems such as Hadoop and Voice over IP systems. She is also interested in fault-tolerance in safety-critical embedded systems, and holds a patent for a fault-tolerant propulsion-by-wire system. spertet@ece.cmu.edu



Jiaqi Tan is a PhD student at Carnegie Mellon University. His research focus is on problem diagnosing in MapReduce and other distributed systems through log-analysis and visualization. He is also interested in program-analysis techniques for understanding software and ensuring security. tanjiaqi@cmu.edu



Rajeev Gandhi is a Senior Systems Scientist at Carnegie Mellon University. His research focuses on problem diagnosis in large-scale distributed systems and video streaming. Rajeev was previously involved in the H.264 video-compression standardization and received a Motorola Outstanding Performance award in 2002 in recognition of his contributions to global standardization. In 2000, Rajeev received his PhD from the University of California, Santa Barbara. rgandhi@ece.cmu.edu



Priya Narasimhan is an Associate Professor at Carnegie Mellon University. Her research interests lie in the fields of dependable distributed systems, embedded systems, mobile systems, and sports technology. Priya is also the CEO and Founder of YinzCam, Inc., a Carnegie Mellon spin-off company focused on mobile live streaming and scalable video technologies to provide sports fans with the ultimate in-stadium mobile experience at NHL/NFL/NBA games. priya@cs.cmu.edu

Visualization tools play an important role in summarizing large volumes of data by revealing interesting patterns such as trends, gaps, and anomalies in the data. Users can leverage visualization tools to identify problems in their programs quickly. In this article, we present novel visualizations that help users diagnose problems in Hadoop applications. These visualizations allow users to identify problematic nodes in the cluster quickly, and distinguish between different classes of problems.

Hadoop is a popular open source system that simplifies large-scale data analysis. Large companies like Twitter use Hadoop to store and process tweets and log files [6]. A typical Hadoop deployment can consist of tens to thousands of nodes. Manual diagnosis of performance problems in a Hadoop cluster requires users to comb through the logs on each node—a daunting task, even on clusters consisting of tens of nodes. We have developed a visualization tool, Theia (named after the Greek goddess of light), that helps users distinguish between application-level problems (e.g., software bugs, workload imbalances), which they can fix on their own, and infrastructural problems (e.g., contention problems, hardware problems), which they should escalate to the system administrators.

Each Hadoop job consists of a group of Map and Reduce tasks. Map tasks process smaller chunks of the large data set in parallel and use key/value pairs to generate a set of intermediate results, while Reduce tasks merge all intermediate values associated with the same intermediate key. Theia leverages application-specific knowledge about how MapReduce jobs are structured to generate compact, interactive visualizations of job performance. Theia generates three different types of visualizations: one at the cluster-level that represents the performance of jobs across nodes over time, and two others at the job-level that summarize task performance across nodes in terms of task duration, task status, and volume of data processed.

We describe how Theia works, and use actual problems from a production Hadoop cluster to illustrate how our visualizations can provide users with a better understanding of the performance of their jobs and easily spot anomalous nodes.

Generating Visual Signatures of Hadoop Job Performance

We implemented Theia using a Perl script that gathered data about job execution from the job-history logs generated by Hadoop. These logs store information about the Map and Reduce tasks executed by each job, such as task duration, status, and the volume of data read and written. We store this information in a relational database, and generate visualizations in the Web browser using the D3 framework [1].

<i>Visual Signatures of Problem Classes</i>			
	Application problem	Workload imbalance	Infrastructural problem
Time	<i>Single</i> user or job over time	<i>Single</i> user or job over time	<i>Multiple</i> users and jobs over time
Space	Span <i>multiple</i> nodes	Span <i>multiple</i> nodes	Typically affect <i>single</i> node, but correlated failures also occur
Value	Performance degradations and <i>task exceptions</i>	Performance degradation and <i>data skews</i>	Performance degradations and <i>task exceptions</i>

Table 1: Heuristics for developing visual signatures of problems experienced in a Hadoop cluster

We developed visual signatures that allow users to spot anomalies in job performance by identifying visual patterns (or signatures) of problems across the time, space, and value domain. Table 1 summarizes the heuristics that we used to develop visual signatures that distinguish between application-level problems, workload imbalances between tasks from the same job, and infrastructural problems. These heuristics are explained below:

1. *Time dimension.* Different problems manifest in different ways over time. For example, application-level problems and workload imbalances are specific to an application; therefore, the manifestation of a problem is restricted to a single user or job over time. On the other hand, infrastructural problems, such as hardware failures, affect multiple users and jobs running on the affected nodes over time.
2. *Space dimension.* The space dimension captures the manifestation of the problem across multiple nodes. Application-level problems and workload imbalances associated with a single job manifest across multiple nodes running the buggy or misconfigured code. Infrastructural problems are typically limited to a single node in the cluster. However, a study of a globally distributed storage system [2] shows that correlated failures are not rare, and were responsible for approximately 37% of failures. Therefore, infrastructural problems can also span multiple nodes.
3. *Value dimension.* We quantify anomalies in the value domain by capturing the extent of performance degradation, data skew, and task exceptions experienced by a single job. Application-level and infrastructural problems manifest as either performance degradations or task exceptions. Workload imbalances in Hadoop clusters can stem from skewed data distributions that lead to performance degradations.

Detecting Anomalous Nodes

We detect anomalies by first assuming that under fault-free conditions, the workload in a Hadoop cluster is relatively well-balanced across nodes executing the same job—therefore, these nodes are peers and should exhibit similar behavior [4]. Next, we identify nodes whose task executions differ markedly from their peers and flag them as anomalous. Aggregating task behavior

on a per-node basis allows us to build compact signatures of job behavior because the number of nodes in the cluster can be several orders of magnitudes smaller than the maximum number of tasks in a job.

We compute an anomaly score using a simple statistical measure known as the z-score. The z-score is a dimensionless quantity that indicates how much each value deviates from the mean in terms of standard deviations, and is computed using the following formula: $z = [(x - \mu)/(\sigma)]$, where μ is the mean of the values, and σ is the corresponding standard deviation. We compute z-scores for each node based on the duration of tasks running on the node, the volume of data processed by the node, and the ratio of failed tasks to successful tasks. For the cluster-level visualization, we estimate the severity of problems by using a single anomaly score that flags nodes as anomalous if the geometric mean of the absolute value of the z-scores is high, i.e., $\text{Anomaly-Score} = (|z_{\text{task_duration}}| * |z_{\text{data_volume}}| * |z_{\text{failure_ratio}}|)^{1/3}$.

Visualizations and Case Studies

Theia generates three different visualizations that allow users to understand the performance of their jobs across nodes in the cluster. The first visualization is the anomaly heatmap, which summarizes job behavior at the cluster-level; the other two visualizations are at the job-level. The first job-level visualization, referred to as the job execution stream, allows users to scroll through jobs sequentially, thus preserving the time context. The second job-level visualization, referred to as the job-execution detail, provides a more detailed view of task execution over time on each node in terms of task duration and amount of data processed. We analyzed the jobs and problems experienced by Hadoop users of the 64-node OpenCloud cluster for data-intensive research [5] over the course of one month. We use actual problems experienced by users of the cluster to illustrate our visualizations.

Anomaly Heatmap

A heatmap is a high-density representation of a matrix that we use to provide users with a high-level overview of jobs execution at the cluster-level. This visualization is formulated over a grid that shows nodes on the rows and jobs on the columns, as shown

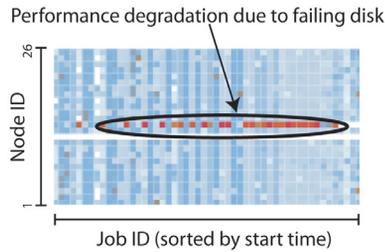


Figure 1: Visual signature of an infrastructural problem using an anomaly heatmap shows succession of anomalous jobs (darker color/red) due to a failing disk controller on a node.

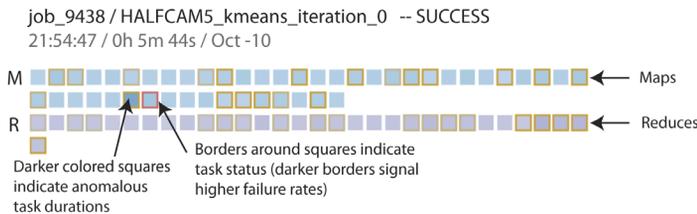


Figure 2: The job execution stream visualization compactly displays information about a job's execution. The header lists the job ID, name, status, time, duration and date. The visualization also highlights anomalies in task duration by using darker colors, and task status by using yellow borders for killed tasks and red borders for failed tasks. The nodes are sorted by decreasing amount of I/O processed.

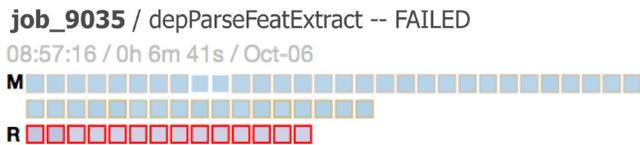


Figure 3: Visual signature of bugs in the Reduce phase. Failures spread across all Reduce nodes (solid dark/red border) typically signal a bug in the Reduce phase.

in Figure 1. The darkness of an intersection on the grid indicates a higher degree of anomaly on that node for that job. By using this visualization, anomalies due to application-level and infrastructural problems can be spotted easily as bursts of color that contrast with non-faulty nodes and jobs in the background.

Figure 1 displays the visual signature of an infrastructural problem identified by a succession of anomalous jobs (darker color) due to a failing disk controller on a node. The data density of the anomaly heatmap is around 2,900 features per square inch on a 109 ppi (pixels per inch) display, using 2x2 pixels per job/node, which is equivalent to fitting 1200 jobs x 700 nodes on a 27-inch display.

Job Execution Stream

The job execution stream, shown in Figure 2, provides a more detailed view of jobs while preserving information about the context by showing a scrollable stream of jobs sorted by start time. In addition to displaying general information about the job (job ID, job name, start date and time, job duration) in the



Figure 4: Visual signature of data skew. A node with anomalous task durations (darker color) and high volume of I/O (nodes are sorted by decreasing order of I/O) can signal data skew.

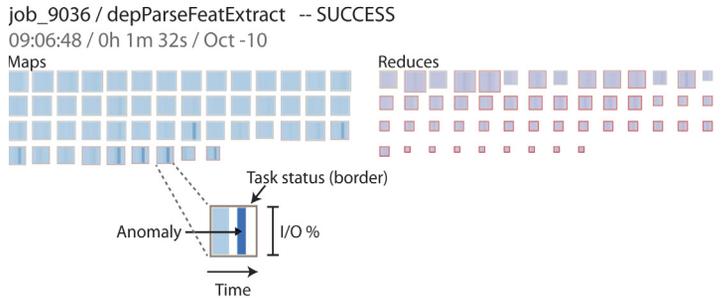


Figure 5: The job execution detail visualization highlights both the progress of tasks over time and the volume of data processed. Each node is divided into five stripes that represent the degree of anomalies in tasks executing during the corresponding time slot; the size of the square represents the proportion of I/O processed by that node.

header, this visualization uses variations in color to highlight anomalous nodes.

Because the application-semantics of Map and Reduce tasks are very different, we divided nodes into two sets: the Map set and the Reduce set. We enhanced the representation of each node with a colored border that varies in intensity depending on the ratio of failed tasks to successful tasks, or the ratio of killed tasks to successful tasks; killed tasks arise when the task scheduler terminates speculative tasks that are still running after the fastest copy of the task completed. Killed tasks are represented using a yellow border, which is overloaded by a red border if there are any failed tasks.

The job execution stream visualization allows us to generate signatures for application-level problems, which manifest as a large number of failed tasks across all nodes in either the Map or Reduce phase (see Figure 3). Workload imbalances and infrastructural problems tend to manifest on a single or small set of nodes in the system. For example, the dark left-most node in Figure 4 shows a node whose performance is slower than its peers due to data skew.

Job Execution Detail

The job execution detail visualization provides a more detailed view of task execution and is less compact than the job execution stream. The job execution detail visually highlights both the progress of tasks over time and the volume of data processed as shown in Figure 5. Nodes are still represented as two sets of squares for Map and Reduce tasks; however, given that there is

Theia: Visual Signatures for Problem Diagnosis in Large Hadoop Clusters

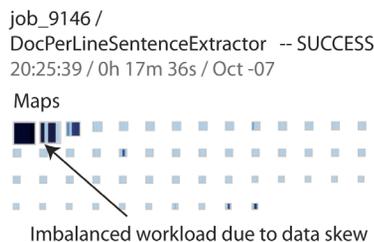


Figure 6: Visual signature of data skew. A single node with high duration anomaly (darker color) and high amount of I/O (larger size) can signal data skew.

additional space available because we are only visualizing one job at a time, we use the available area on each of the squares to represent two additional variables: (1) anomalies in task durations over time by dividing the area of each node into five vertical stripes, each corresponding to a fifth of the total time spent executing tasks on that node; darker colors indicate the severity of the anomaly while white stripes represent slots of time where no information was processed; and (2) percentage of total I/O processed by that node, i.e., reads and writes to both the local file system and the Hadoop distributed file system (HDFS); larger squares indicate higher volumes of data.

Figure 6 shows the visual signature of a data skew where a subset of nodes with anomalous task durations (darker color) and high amounts of I/O (first nodes in the list, large square size) indicate data skew. In this visualization, the data skew is more obvious to the user when compared to the same problem visualized using the job execution stream in Figure 4. Infrastructural problems such as the failed NIC (network interface controller) in Figure 7 can be identified as a single node with high task durations (darker color) or failed tasks (red border), coupled with a low volume of I/O (small square size), which might indicate a performance degradation.

All of our visualizations are interactive, and they provide access to additional information by using the mouse-over gesture. By hovering over the failed node in Figure 7, a user can obtain additional information about the behavior of tasks executed on that node. For example, a user can observe that 50% of the tasks executed on this node failed.

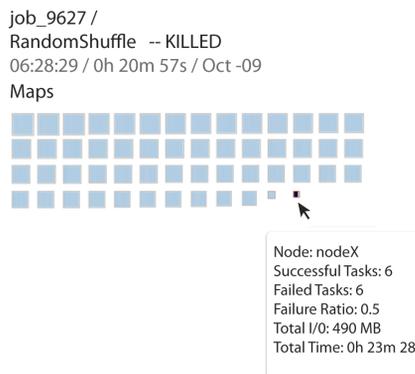


Figure 7: Interactive user interface. This job execution detail visualization shows degraded job performance due to a NIC failure at a node. Hovering over the node provides the user with additional information about the behavior of tasks executed on that node.

Evaluating the Effectiveness of Visualizations

We generated our visualizations using one month’s worth of logs generated by Hadoop’s JobTracker on the OpenCloud cluster. During this period, 1,373 jobs were submitted, comprising a total of approximately 1.85 million tasks. From these 1,373 jobs, we manually identified 157 failures due to application-level problems, and two incidents of data skew. We also identified infrastructural problems by analyzing a report of events generated by the Nagios tool installed on the cluster. During the evaluation period, Nagios reported 68 messages that were associated with 45 different incidents, namely: 42 disk controller failures, two hard drive failures, and one network interface controller (NIC) failure.

We evaluated the performance of Theia by manually verifying that the visualizations generated matched up with the heuristics for distinguishing between different problems described in Table 1. Table 2 shows that we successfully identified all the application-level problems and data skews using the job execution stream (similar results are obtained using the job execution detail). Additionally, the anomaly heatmap was able to identify 33 of the 45 infrastructural problems (the problems identified by the job execution stream are a subset of those identified by the heatmap). We were unable to detect four of the infrastructural problems because the nodes had been blacklisted. We hypothesize that the heatmap was unable to detect the remaining

Type	Total problems	Diagnosed by heatmap	Diagnosed by job execution stream
Application-level problem	157	0	157
Data-skew	2	2	2
Infrastructural problem	45	33	10

Table 2: Problems diagnosed by cluster-level and job-level visualizations in Theia. The infrastructural problems consisted of 42 disk controller failures, two hard drive failures, and one network interface controller (NIC) failure. The infrastructural problems diagnosed by the job execution stream were a subset of those identified by the heatmap.

eight infrastructural problems because they occurred when the cluster was idle.

Conclusion

Theia is a visualization tool that exploits application-specific semantics about the structure of MapReduce jobs to generate compact, interactive visualizations of job performance. Theia uses heuristics to identify visual signatures of problems that allow users to distinguish application-level problems (e.g., software bugs, workload imbalances) from infrastructural problems (e.g., contention problems, hardware problems). We have evaluated our visualizations using real problems experienced by Hadoop users at a production cluster over a one-month period. Our visualizations correctly identified 192 out of 204 problems that we observed. More details about Theia can be found in our USENIX LISA 2012 paper [3].

References

- [1] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-Driven Documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, 2011, pp. 2301-2309.
- [2] D. Ford, F. Labelle, F.I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in Globally Distributed Storage Systems," *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (Vancouver, CA, October 2010), pp. 61-74.
- [3] E. Garduno, S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "Theia: Visual Signatures for Problem Diagnosis in Large Hadoop Clusters," *USENIX Large Installation System Administration Conference (LISA)* (San Diego, CA, December 2012).
- [4] X. Pan, J. Tan, S. Kavulya, R. Gandhi, and P. Narasimhan, "Ganesha: Black-Box Diagnosis of MapReduce Systems," *SIGMETRICS Performance Evaluation Review* 37 (January 2010).
- [5] Parallel Data Lab. Apache Hadoop: <http://wiki.pdl.cmu.edu/opencloudwiki/>, September 2012.
- [6] The Apache Software Foundation, PoweredBy Hadoop: <http://wiki.apache.org/hadoop/PoweredBy>, September 2012.



Join other members of the sysadmin community looking to stay ahead in this dynamic industry. Become a part of the SIG today.

Are you getting the most out of the sysadmin community?

We can help . . .

Created by and for system administrators, this USENIX SIG serves the system administration community by:

- Offering conferences and training to enhance the technical and managerial capabilities of members of the profession
- Promoting activities that advance the state of the art or the community
- Providing tools, information, and services to assist system administrators and their organizations
- Establishing standards of professional excellence and recognizing those who attain them

LISA members enjoy a number of benefits, including:

- Discount on registration for LISA, the annual Large Installation System Administration Conference
- Access to the large and growing online library of Short Topics in System Administration books—see reverse for the complete catalog
- A free Short Topics in System Administration book every year: Your choice of any book in print
- Access to the LISA SIG Jobs Board, including real-time email notification of new jobs posted and the ability to post resumes
- The option to join LISA-members, an electronic mailing list for peer discussion and advice
- Student programs, including grants to help fund conference attendance, low membership fees, and a university outreach program
- And more!

Find out more at www.usenix.org/lisa

Cory Lueninghoener on Managing HPC Clusters

An Interview

RIK FARROW



Rik is the editor of ;login:
rik@usenix.org



Cory Lueninghoener is the lead of the HPC Architecture Team at Los Alamos National Laboratory. He has co-hosted the LISA Configuration Management Workshop for the

past few years and is always looking for more efficient ways to manage acres of computers with only a handful of people.

cluening@gmail.com

I first met Cory during a LISA conference, when he volunteered to write an article about getting started with configuration management (CM) for ;login: [1]. I could tell he was passionate about CM from his article, and through his participation in the Configuration Management Workshop, and the LISA short topics book [2] that he had written with Narayan Desai.

I knew that Cory worked at Los Alamos National Labs on high performance computing (HPC) clusters, and that got me even more interested. I've learned a lot about modern clusters through working with people on Hadoop projects, but I thought for sure that HPC was very different from a Hadoop cluster. And it turned out that I was right.

Rik: I am guessing that you didn't start out by managing large numbers of HPC clusters. Tell us a little bit about how you wound up in this position?

Cory: You're right! I started out managing just my Linux desktop machine when I was in college, and I expanded that by a couple orders of magnitude as a summer student with Argonne National Laboratory's Mathematics and Computer Science Systems Group. I started managing HPC systems as a grad student, with a few little clusters. That number grew as I worked with Argonne's Leadership Computing Facility and Los Alamos National Laboratory's High Performance Computing Division. Now I'm on a team at Los Alamos that manages about 20 HPC resources ranging in size from around 20 nodes up to around 10,000 nodes.

Rik: You mention being a grad student, and I am guessing that you weren't in grad school to become a sysadmin. I'm curious about what you were studying before you entered the challenging realm of riding herd on multiple clusters.

Cory: My Master's degree is in computer science, focusing a lot on machine learning and artificial intelligence. But my advisor was a computational chemist, and that's where my interest in high performance computing started: he was just starting to move from large single-system image machines to clusters and had funding to buy some small clusters. I quickly found that I was more interested in helping manage the clusters and work with a wide variety of scientific users than find my own specific project to run on them, and that's what brought me to where I am today.

Rik: More and more people are managing clusters, used either as nodes for VMs, Hadoop (MapReduce), or grid computing. How is managing HPC clusters different?

Cory: Today's HPC clusters are generally tightly coupled systems that consist of a collection of compute nodes, a dedicated high-speed network, one or more global file systems, and a handful of support systems. A queueing system runs on top of the hardware that users submit jobs to. These jobs can range in size from a few tens of processors up to hundreds of thousands of processors, and jobs may run for only a few minutes or for a day or longer. Nodes running these jobs are generally dedicated to a single job at a time, giving each job its maximum resources.

Managing HPC systems can be tricky—once a job is running on a set of nodes, we want to do as little administration work on the nodes as possible. Tightly coupled HPC jobs usually cannot survive a single node reboot, so any needed reboots need to be scheduled while a node

is completely idle. Changing software on a node running a job is also avoided, as an unexpectedly heterogeneous job environment can wreak havoc on a job. Even subtle changes can affect jobs: simply running a monitoring script too frequently and without coordination across the cluster can introduce enough operating system jitter to affect large running jobs.

Rik: What is the role of CM in managing HPC? Is it any different from what other people are using CM for, based on your experiences at the Configuration Management Workshop?

Cory: The role of configuration management on an HPC system is very similar to any other site. We use our configuration management tools to push out regular updates, install user-requested software, ensure compliance, and ease system management in general. Some of the details differ, however. For example, I mentioned earlier the importance of ensuring identical software environments on all of our compute nodes. Whereas an administrator of a network of desktop systems may expect his systems to be all unique, or an administrator of a VM hosting cluster may not care what kernel version is running on her systems, we require that all of our systems be as identical as possible. Here at the lab, we also use configuration management as a sort of self-documenting system.

We are able to hand new administrators a copy of our central configuration repository, consisting of all of our modified configuration files, local packages, and anything else that makes one of our systems unique. This quickly gives them a focused view of our configuration and how it differs from a default installation. We can do the same with auditors to demonstrate the level of control we have over our systems and our knowledge of what we are currently running.

Rik: Do your HPC systems run standard operating systems, which presumably makes them easier to manage?

Cory: In general, yes. Using standard Linux distributions makes things easier for a lot of people: users are familiar with developing code on standard distributions; vendors support compilers, debuggers, and other development tools on specific distributions; and administrators are much more familiar with standard distributions than company-proprietary operating systems. Cluster vendors will often customize the distributions, though, to support the cluster's high-speed interconnect, provide more compilers or other tools than are available by default, or improve the user experience in other ways.

Rik: You mentioned that HPC systems are broken into three roles: management, execution engines, and storage. Does storage get managed separately, or are all three categories handled as if they were a single system?

Cory: Yeah, that is a pretty standard way to split up a system. While management systems like login nodes, resource management nodes, and boot servers are almost always treated as part of the cluster, the storage can be very site-specific. On smaller clusters, a storage system consisting of a single node that serves NFS exports to the compute nodes is a common simple solution. In that case, treating the storage server as another management node is the easiest approach. On larger systems, having a dedicated storage cluster that serves up file systems like GPFS, Panasas, or Lustre is more common. These systems are generally treated as separate resources and managed as more independent entities, especially if they serve their file systems to multiple clusters.

Rik: In HPC, I've learned (from listening to talks at LISA) just how critical the network is in maintaining good performance. How do you monitor the network?

Cory: Really intense HPC jobs do a lot of communication between each of the nodes in the job, so a reliable high speed network is indeed critical. The amount of network monitoring that can be done on a cluster depends on what interconnect is used. Infiniband is a popular interconnect on commodity clusters, and OpenFabrics provides a number of tools to help monitor connectivity, speed, errors, and other aspects of an Infiniband network. We use these tools to monitor our networks and automatically disable nodes that are not performing optimally. Higher end systems, such as Cray's MPPs and IBM's Blue Gene systems, use custom high speed networks that are designed to require less monitoring and provide more information when a problem does occur. Either way, keeping the high speed network working is an important part of running an HPC system.

References

- [1] Cory Lueninghoener, "Getting Started with Configuration Management," April 2011, vol. 36, no. 2: <https://www.usenix.org/publications/login/april-2011-volume-36-number-2/getting-started-configuration-management>.
- [2] Narayan Desai and Cory Lueninghoener, *Configuration Management with Bcfg2*: <https://www.usenix.org/lisa/books/configuration-management-bcfg2>.

Wireless Means Radio

DAVID LANG



David Lang is a staff IT engineer at Intuit, where he has spent more than a decade working in the Security Department for the Banking Division. He was introduced to Linux in 1993 and has been making his living with Linux since 1996. He is an Amateur Extra Class radio operator and served on the communications staff of the Civil Air Patrol California Wing, where his duties included managing the statewide digital wireless network. He was awarded the 2012 Chuck Yerkes award for his participation on various open source mailing lists.

david@lang.hm

Wireless means radio, and it sounds obvious when it's stated, but sysadmins are normally not trained in the radio field. If the radio side doesn't work, you have no chance of the network working. In this article, I explain the problems you are facing, so that it's possible to build a reliable wireless network that will support hundreds to thousands of people using cheap commodity equipment.

Why do conference and school wireless networks always work so poorly? As IT professionals we are used to the network layer "just working" and fixing things by changing configuration files. This mind-set, combined with obvious but wrong choices in laying out a wireless network, frequently leads to a network that works just fine when tested with a small number of users, but that then becomes unusable when the crowds of users arrive. This is at its worst at technical conferences, where there are so many people, each carrying several devices, all trying to use the network at the same time, and in schools where you pack students close together and then try to have them all use their computers simultaneously.

Is this a fundamental limitation of wireless? While it is true that there are some issues that cannot be solved, there are a lot of things that the network administrator can do to make the network work better.

I have been running the wireless network for the Southern California Linux Expo (SCALE) since 2010, and this article is based on the results of the past five years' worth of SCALE conferences and the resulting paper that I presented at LISA in 2012 [1]. At the 2012 SCALE, we had 1965 attendees with 1935 unique Mac addresses on the network and 875 devices connected at peak.

The key thing to recognize when building a wireless network is that the network is primarily radios, and only secondarily digital. This doesn't mean that getting the radio side of things right will guarantee that your network will work, but it does mean that getting it wrong will guarantee that your network will not work.

The Problems

The 2.4 GHz band (b/g) has 11 channels assigned in the US, but they overlap and, as a result, you can only use three of the channels at once without problems. Three channels are really not enough as you want to leave a channel "unused" for a substantial distance between each area that is used. The rule of thumb is that if you plan to have an access point cover an area with a radius of 50 ft, you don't want to have another access point using the same channel within 200 ft. With only three channels, you can't even do this in two dimensions, let alone three, and will have to have the access points much closer together.

The 2.4 GHz band is also used extensively by other equipment, including cordless phones, cordless microphones, Bluetooth, and even microwave ovens. While the 802.11 protocol is designed to be resistant to interference from these things, they can cause packets to be corrupted, which results in retries.

Most mobile radio services suffer from the "hidden transmitter" problem. In simple terms, this is where you have three stations in a line: the station in the middle can hear stations on

each side, but the stations on the outside cannot hear each other. This prevents the stations on each end from avoiding transmitting when the one on the other end is already transmitting. When both sides transmit at the same time, the receiving station in the middle gets confused and can't make out either signal, causing both to have to retransmit the packet. In voice communication this is annoying; in digital communication, this causes everything transmitted by both stations to be garbled and both stations will have to retransmit their data.

Excessive power levels can add to the hidden transmitter problem. It is common to think that if you can't get through, turn up the power, but if only one side turns up the power, it seldom improves communications. This is because wireless networks are two-way conversations; if only one side gets louder it doesn't increase the range in which the conversation can take place but, instead, causes the stronger signal to go further and interfere with other stations.

The WiFi protocols have evolved over time, with new modes being created that squeeze more data into a given amount of airtime. In most cases the newer, higher speed modes are more sensitive to interference, so the protocol includes fallbacks to slower modes when the data is not getting through. If the problem is outside interference, weak signal and similar problems, this works very well, but if the problem is an overload of the available airtime, the result is that each station transmitting takes longer to send its signal, which makes it more likely that a hidden transmitter or other interference will corrupt the packet, resulting in retries.

802.11 has a fair amount of housekeeping traffic to let all stations in the area know that they exist and to maintain the connection to the access point. This traffic eats away at the time available and is frequently required by the spec to be transmitted at the lowest supported speed [2].

802.11n can be a benefit or a problem. The fact that it can transmit more data in a given amount of airtime can reduce congestion, but enabling the high bandwidth (dual channel) mode will require that two adjacent channels be allocated to it. Also, if the equipment is configured to operate in pure n mode, the b/g equipment will not recognize that there is a station transmitting and so will go ahead and attempt to transmit their packet.

Inappropriate use of high-gain antennas can be a problem as well. Unlike turning up the transmitter power, improved antennas help to both transmit and receive the signal. But if they are used incorrectly they will cause the station using them, in covering a larger area, to interfere with, and be interfered with by, more stations.

Mesh networks (access points connected to other access points via wireless links) require that the packets be transmitted over

the radio more times, and as a result are almost always the wrong thing to use in a high-density environment.

Multi-radio enterprise access points seldom help and frequently hurt because there are already not enough channels to avoid overlapping coverage. They create large amounts of bandwidth through a single access point but decrease the overall system bandwidth by causing more interference with other access points.

Retries can also be caused by problems on the digital side of things.

The bufferbloat phenomenon [3], where the delays in getting packets to their destination can result in the packets timing out before they arrive, can also result in packets being retransmitted.

The typical collapse of wireless networks results from the combination of:

- ◆ Retries (frequently due to hidden transmitters or other interference)
- ◆ Fallback to slow speeds
- ◆ Wasted packets (due to bufferbloat and other problems)

The collapse isn't gradual; it's a performance cliff. Things work fairly well with minor slowing until you run out of airtime. At that point devices are retransmitting their data and transmitting slower (and therefore lowering the available bandwidth), and almost all useful communication just stops. If you are monitoring the network, everything will look just fine, like you are transmitting a lot of data, well below design capacity (and well below the levels you were running prior to the collapse).

The Solutions

The solution is to get as many access points into the area as you can without causing interference. To do this, you want to have each access point cover as *small* an area as possible.

First, you need to know what you are up against. Do a site survey to find out what the situation is.

- ◆ Where are the network and power jacks? I've had cases where they were eight feet apart.
- ◆ What other WiFi signals are in the area, and what channels are they on? Good tools to use are WiFi analyzer on Android or Kismet on a laptop.
- ◆ What interference is there in the area (usually not as critical as looking for WiFi signals)? My-Spy spectrum analyzer can see all signals, not just WiFi signals.
- ◆ What effect do the walls have on your signal? Movable partitions tend to block the signal more than traditional walls due to the metal mesh in the partitions.

Wireless Means Radio

Here are some suggestions:

Bring an AP that you can plug in and then find out where you can hear it.

Encourage the use of 5 GHz channels. There are far more of them so you can have more radios covering a given amount of floor space without interference, resulting in more bandwidth per user. In 2012 at SCALE only approximately 20% of devices were using 5 GHz, even though it had three times the capacity available.

Turn power down on 2.4 GHz to allow for more access points without overlapping footprints.

How low? At SCALE in 2012 we had the APs set to 4 mw output.

Take advantage of things that block the signal for you. In addition to walls (see above), make use of the human body, which is mostly water, which absorbs 2.4 GHz signals. Put access points low so that the crowds will prevent their signal from going as far as they normally would.

Use advanced antennas carefully. They can help you cover an area that doesn't have power or network for a fixed AP, and can help prevent interference with other areas.

Digital Issues

You should use one SSID for each band (e.g., SCALE24, SCALE5). Using a different SSID on each AP lets advanced users select the best AP for them, but it prevents roaming to a closer/better AP; if users move around they are going to have to switch SSIDs frequently. One SSID per band allows users to select the band to operate on, but then let their device use the closest AP.

Run DHCP on a central server. This similarly allows access points to act as bridges for mobile devices to roam from one AP to another without having to get new IP addresses.

Enable wireless isolation. Unless you really need the mobile devices to talk to each other, this would avoid IP-level broadcasts' many retransmissions on wireless networks.

Lengthen the beacon interval. This reduces the amount of house-keeping traffic, while lengthening the time it takes for devices to learn that the network is there or notice new APs as they move. Changing this from fractions of seconds to seconds is unlikely to cause any real problems.

Disable slow speeds. If you can disable the 802.11b speeds entirely, you avoid a significant amount of overhead. There are very few devices today that don't support at least 802.11g. If you can control what devices are in use and make sure they are all 802.11n capable, you can disable 802.11g as well.

Use APs that allow you to replace the default firmware with a Linux-based firmware that you can really configure. In 2011 we used DD-WRT on the access points, but found that it did not give us the control that we wanted, and in 2012 we used OpenWRT and were happy with it.

Disable connection tracking. Connection tracking can be a very significant overhead on the CPU and RAM of the AP. Connection tracking also doesn't work when an established connection migrates to a different AP, so it's both expensive and ineffective. Disabling connection tracking may require recompiling the kernel, but is well worth it.

If possible, disable all firewalling on the AP. Do that work upstream in order to leave as much CPU and memory available for processing packets (including doing encryption if enabled).

Set short inactivity timers. You don't want APs spending resources trying to track devices that have moved or been turned off.

Adjust kernel network buffers. The Linux wireless stack includes quite a bit of buffering inside it, so setting the kernel buffers for the wireless interfaces very low helps minimize the possibility of excessive latency. There is some recent work in this area, but it does not yet deal with the buffers inside the wireless stack [4].

Set up monitoring. If you don't know what's going on you can't fix it.

One year we had a serious problem with people turning off the power on access points or unplugging them. Without monitoring you won't know when something goes wrong. An AP that is still powered but not on the network is far worse than one that is completely dead, as user devices keep trying to connect to it.

With hundreds to thousands of users, you will never have enough Internet bandwidth to satisfy everyone. So you should implement normal site bandwidth-saving tools such as blocking streaming sources, and implement QoS traffic shaping to provide fairness between users.

Additionally, packet timeouts and bufferbloat latencies are more likely the most hops any one network connection has, so you can avoid a lot of problems if you can have the users connecting to a local machine that then acts as a proxy. Running a Web caching proxy server or a local mirror for popular distro repositories both saves you Internet bandwidth and changes the user connections from long distance to local connections. The number of people who opt to do system updates at large events is surprising.

Once you understand what problems you are facing on the radio side of the equipment, you can plan accordingly and build a reliable wireless network that will support lots of users, and do it using commodity equipment.

References

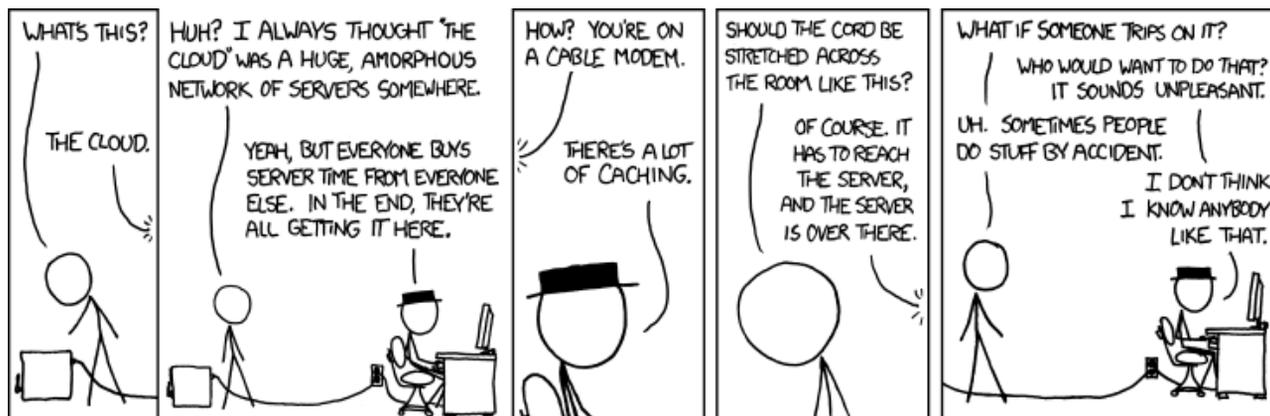
[1] <https://www.usenix.org/conference/lisa12/building-wireless-network-high-density-users>.

[2] Any broadcast traffic (such as SSID broadcasts, connection requests, etc.) must be transmitted at the lowest speed supported so that devices that only support that speed and not higher ones will still be able to decode the message.

[3] In an attempt to prevent packet loss, and with memory becoming vastly cheaper over time, buffers on network devices have become very large. If there is significant congestion and the buffers stay full for an extended time, packets can sit in the buffers so long that by the time they arrive at their destination they have already timed out and a replacement packet has been sent.

[4] <http://www.bufferbloat.net>; <https://lists.bufferbloat.net/listinfo/cerowrt-devel>.

xkcd



xkcd.com

Improving the Performance of fsck in FreeBSD

MARSHALL KIRK MCKUSICK



Dr. Marshall Kirk McKusick writes books and articles, teaches classes on UNIX- and BSD-related subjects, and provides expert-witness testimony on software patent, trade secret, and copyright issues, particularly those related to operating systems and filesystems. His work with Unix and BSD development spans over four decades. It begins with his first paper on the implementation of Berkeley Pascal in 1979, goes on to his pioneering work in the '80s on the BSD Fast File System, the BSD virtual memory system, the final release of 4.4BSD-Lite from the UC Berkeley Computer Systems Research Group, and carries on with his work on FreeBSD. mckusick@mckusick.com

While listening to the presentation of the first paper at FAST '13, “ffsck: The Fast File System Checker” [1], I immediately wondered whether I could implement some of the ideas in FreeBSD. The researchers' goal was to reorganize the Linux ext3 filesystem and to rewrite its filesystem checker so that a complete check of the filesystem could be done more quickly. With the addition of a couple of hundred lines of code, I was able to implement both the improvements to fsck and the layout policy in the FreeBSD filesystem (FFS).

Although the thrust of the paper was to make changes to the layout of the filesystem to enable fsck to run more quickly, some of the changes resulted in a reduction in performance of the filesystem. As I am unwilling to accept a reduction in filesystem performance solely for the purpose of speeding up fsck, I chose to consider only on the subset of their changes that improve both.

Implementation

The paper describes changes that the researchers made to the on-disk layout of the filesystem. Getting folks to change to a different filesystem format that is incompatible with the existing filesystem format is difficult. So, in my implementation, I was not willing to change the filesystem format beyond using one of the spare fields in the superblock to tune the layout policy. Even with these limitations, I was able to get an impressive improvement in fsck's running time and some small improvements in filesystem performance.

In FFS (the Fast File System), the disk space is broken up into groups of contiguous blocks called cylinder groups similar to the ext3 block groups. The first block of each cylinder group contains the cylinder group descriptor that includes a map showing the free and allocated blocks and a map showing the free and allocated inodes in that cylinder group. Following the cylinder group descriptor are blocks that contain the metadata (inodes) for the files in that cylinder group. The organization of an inode is shown in Figure 1. The remainder of the cylinder group is made up of blocks that contain the indirect blocks and data blocks for the files and directories contained in the filesystem. An inode may reference blocks in one or more cylinder groups in the filesystem, although the policy is that small files have their blocks allocated in the same cylinder group in which the inode resides. For details, see Chapter 8 of McKusick & Neville-Neil [2].

The key idea in the paper [1] is to reserve a small area in each cylinder group immediately following the inode blocks for the use of metadata, specifically indirect blocks and directory contents. It requires that metadata be allocated in this area and does not allow data blocks to be allocated in this area. Thus, the paper has a long discussion of how to size this area. If it is improperly sized, the filesystem will report as being full when it in fact still has plenty of available space since it reports a filesystem full error when either the metadata area or the non-metadata area fills up.

The FFS separates the allocation of data blocks and inodes into two distinct layers: policy and implementation. The policy layer is responsible for picking what it views as the ideal place to

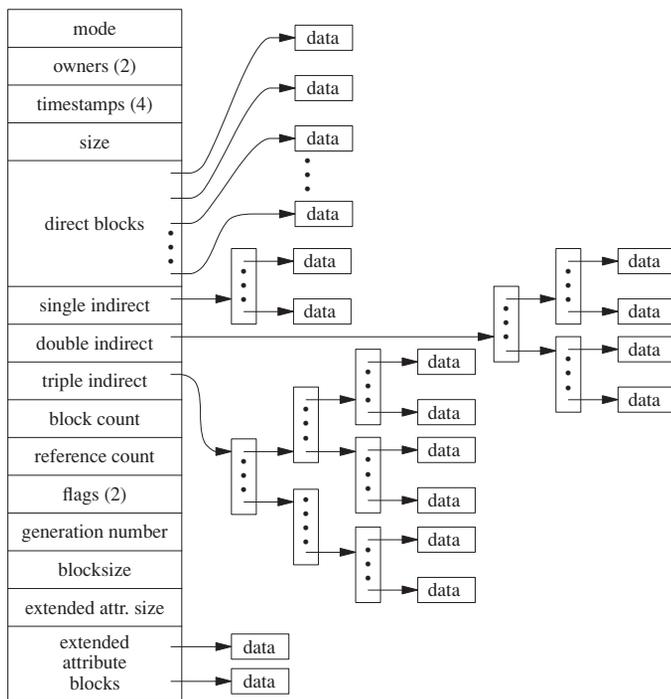


Figure 1. The structure of an inode

allocate the inode or the data block. For example, when asked to allocate a block for a file, the policy layer will usually ask for the block that immediately follows the previously allocated block.

The implementation layer is responsible for managing the allocation bitmaps and ensuring that resources do not get double allocated. Thus, the policy layer does not have to worry about requesting an already allocated block. If the implementation layer finds that a requested block is already allocated, it simply scans through the map to find the closest available free block. The result of this separation is that once the implementation layer is working properly, filesystem designers are free to try out whatever hare-brained policy ideas that they want without fear of corrupting the filesystem. In the case of FFS, the implementation layer was written and debugged in 1982 and has not been changed since. Further refinements to the filesystem have been done at the policy layer.

Following these design principles, I chose not to change the filesystem layout or the implementation layer. Instead I chose to implement it entirely as a new policy. Specifically, the new policy is to hold about the first 4% of the data blocks in each cylinder group for use of metadata. The policy routines preferentially place metadata in the metadata area and everything else in the blocks that follow the metadata area. In my implementation, the size of the metadata area does not matter as it is just used as a hint by the policy routines. If the metadata area fills up, then the metadata just gets put in the regular blocks area and vice versa.

And this decision happens on a cylinder group by cylinder group basis (e.g., some cylinder groups can overflow their metadata area whereas others do not overflow it). For filesystem performance, having the metadata in the same cylinder group as its inode is usually better than pushing it to the metadata area of another cylinder group as is done by the design in the paper.

Another area where I chose to take a different approach than the paper is in the allocation policy for the first indirect block of the file. The BSD fast filesystem tries to place the first (single) indirect block inline with the file data (e.g., it tries to lay out the first 12 direct blocks contiguously followed immediately by the indirect block followed immediately by the data blocks referenced from the indirect block). One of the performance slowdowns in the paper occurs for files that spill into only the first part of their first indirect block. The slowdown comes from moving this first indirect block to the metadata area, thus causing two extra seeks when reading it. To avoid this slowdown, I do not change the layout of the first indirect (leaving it inline). Only the second and third level indirects along with the indirects that they reference are moved to the metadata area. The nearly contiguous allocation of this metadata close to the inode that references it noticeably improves the random access time to the file as well as speeding up the running time of fsck. Also, as noted in the paper, the disk track cache is frequently filled with much of a file's metadata when the second level indirect block is read, thus often speeding up even the sequential reading time for the file; however, in limited testing I did not see statistically significant differences in sequential reading times.

Putting the contents of directories in the metadata area gives a similar speedup to directory tree traversal because the data is a short seek away from where the directory inode was read and may already be in the disk's track cache from other directory reads done in its cylinder group.

The final observation that I plucked from the paper specifically for speeding up fsck is to save an in-memory copy of the cylinder groups during pass1 so as not to need to re-read them in pass5. This nearly doubles the memory footprint of fsck, so if memory runs short (e.g., its mallocs begin to fail) this cache is released as needed to make room for other allocations.

Results

I have been testing on an Intel Quad-core CPU running at 2.83 GHz with 2 Gb of memory and a 2 Tb Western Digital 7200 rpm testing disk running FreeBSD 8.3-STABLE (Subversion revision r246915M). Filesystems are created with their default settings: 16 K blocks, 2 K fragments, soft updates, and 4% of the data blocks held for metadata. For these tests, the filesystem is 75% full mostly populated with big files (to exaggerate the metadata effects). In each case a new filesystem was created and all the data copied into it so that the new layout could have maximal

Improving the Performance of fsck in FreeBSD

effect. There are few files and hence little directory information, so the benefit to the running time for directories is minimal in these tests. I am currently running tests on a more conventionally populated filesystem.

Fsck times are also better as the filesystem has not been aged; however, aging effects in the FFS filesystem tend to be a lot less noticeable than in others because of its use of dynamic block reallocation. Notably, the Harvard folks found that I/O performance dropped off by only about 10% after 10 months of simulated aging [3]. Also, fsck times are low because of the small number of files in the filesystem and hence the smaller number of inodes needing to be inspected. Finally, a technique similar to the metadata compression discussion in the Ma, et al paper has been in use in fsck for the directory metadata since 1988, which cuts down on running time.

Executive summary on running time of fsck:

- ◆ Baseline before any changes: 284 seconds (4 min 44 sec)
- ◆ Storing second and third level metadata (and their referenced indirect blocks) but not first indirect block in the metadata area: 135 seconds (2 min 15 sec)
- ◆ Adding directory data blocks to metadata area: 134 seconds (2 min 14 sec)
- ◆ Caching cylinder group blocks in pass1 to avoid the need to read them in pass5: 84 seconds (1 min 24 sec)

In Appendix 1 [4] are the summary statistics for each run. I/O listed as “Double Level Indirect” includes all double-indirect blocks referenced from inodes and all the single level indirect blocks below them. Similarly, “Triple Level Indirect” includes all triple-indirect blocks referenced from inodes and all the single and double level indirect blocks below them. The key observation is that whereas the number of I/Os of each type of data remain similar from run to run, the percentage of time for reading the metadata has dropped dramatically.

I ran just a few tests on the speed with which data could be read from or written to files. Random read times improved a bit. The remaining tests were not statistically significantly different. More thorough tests would need to be run to get a reasonable idea of whether it makes any difference; first results imply no degradation and some hints at improvement.

Conclusions

This work has once again shown the power of separating the filesystem layout policy routines from the implementation routines. I was so excited by the possibilities presented by the FAST '13 paper that I skipped lunch after hearing the presentation so I could try implementing it in FFS. By the time the 90 minute lunch break was over, I had fully written the 100 lines of changes (half of which were comments) to the block layout policy routine

to implement the reserved metadata area. And I had no fears of bringing it up on my primary server to test it out because I knew that at worst I would get some badly laid out files; certainly I was not running the risk of corrupting my filesystems.

By retaining the same on-disk format, I did not need to make any changes to fsck. The stock fsck just ran faster because of the new layout of metadata. I did need to make about 100 lines of changes to fsck to add the caching of cylinder groups between pass1 and pass5; however, that was a trivial change and one that will provide equal improvement whether or not the new filesystem layout is in use. The vast majority of my time has been spent measuring the effects of the changes and writing this paper. Having spent time writing or tuning fsck for the past 30 years, I never would have guessed that so much improvement in running time could come from fsck for so little effort.

The lesson to be learned is that separating policy from implementation is an important design principle when architecting software systems, especially when they are mission-critical systems. The policy layer allows new ideas to be implemented and tested quickly. Once validated, those ideas can be deployed without danger of compromising the integrity of the system.

I commend the authors of the paper for their work. Unfortunately the filesystem on which they worked is not separated into policy and implementation layers, so they had to make several thousand lines of changes in areas where bugs would compromise the filesystem integrity. The monolithic architecture led to a great deal more effort on their part than would otherwise have been necessary. Finally, the scope of the change and the possibility of destabilizing a production filesystem will make it far more difficult for them to get their changes accepted back into the mainline code base.

References

- [1] A. Ma, C. Dragga, A. Arpaci-Dusseau, & R. Arpaci-Dusseau, “ffsck: The Fast File System Checker,” 11th USENIX Conference on File and Storage Technologies (FAST '13), <http://www.usenix.org/conference/fast13/ffsck-fast-file-system-checker> (February 2013).
- [2] M. K. McKusick & G. V. Neville-Neil, *The Design and Implementation of the FreeBSD Operating System*, Addison-Wesley, Reading, MA (2005).
- [3] K. Smith & M. Seltzer, “A Comparison of FFS Disk Allocation Algorithms,” Winter USENIX Conference, pp. 15-25, <http://www.eecs.harvard.edu/margo/papers/usenix96-ffs> (January 1996).
- [4] Appendix 1: <http://www.usenix.org/publications/login/april-2013-volume-38-number-2>.

ORACLE®

LABS

Research Partnerships

The Mission of Oracle Labs is to identify, explore, and transfer new technologies that have the potential to substantially improve Oracle's business. Oracle Labs researchers look for novel approaches and methodologies, often taking on projects that are high risk, uncertain, or difficult to tackle within a product development organization. Oracle Labs research is focused on real-world outcomes: our researchers aim to develop technologies that will someday play a significant role in the evolution of technology and society. We maintain a balanced research portfolio, including exploratory research, directed research, consulting and product incubation.

The External Research Office at Oracle Labs invests in research collaborations that narrow technology gaps, explore and expand new technologies. Oracle Labs hosts undergraduate and graduate student interns, postdocs, and visiting professors to partner with leading researchers in onsite collaborations. Oracle Labs also funds off-site research through partnerships with research institutions worldwide. In the past two years, Oracle Labs funded 50 universities and 90 students worldwide. Last year alone, Oracle Labs hosted 63 interns.

Areas of Research

VLSI Circuit Design

Hardware / Software Co-design

Query Processing on Extreme Scale-out Architectures

Processor Design

Integrated Circuit Packaging

Domain-Specific Language Design and Implementation

Managed Language VM technologies

Silicon Photonics

Information Retrieval and Machine Learning

Operating Systems in the Cloud

HW Application Accelerators in the Network

Persistent Programming Languages

Program Analysis for Correctness & Vulnerability Detection

Graph Analytics

Scalable Concurrency

Locations

Based at Oracle headquarters in Redwood Shores, CA, Oracle Labs also has research centers in Boston, MA, Austin, TX, San Diego, CA, Linz, Austria, Cambridge, UK, Vancouver, CA, and Brisbane, Australia. Senior individual researchers work from remote locations all over the world, from Germany to New Zealand. The geographic spread allows Oracle Labs to take advantage of a tremendous pool of scientific and engineering talent and enables Labs researchers to collaborate with colleagues from a wide range of industries and universities.

For more information on how to establish a research collaboration, visit <https://labs.oracle.com>, or contact Marie-Therese Ellis at marie-therese.ellis-house@oracle.com.



Analyzing Network Traffic with Chimera

JONATHAN SPRINGER, KEVIN BORDERS, AND MATTHEW BURNSIDE



Jonathan Springer is a Managing Engineer at Reservoir Labs. Jonathan started out developing functional language implementations at the University of Illinois, where he received his PhD in computer science. After spending time working on workstation compilers at Hewlett Packard, he joined Reservoir, where he has led numerous projects developing compilers, static analysis tools, and language runtimes.
springer@reservoir.com



Kevin Borders is a Computer Security Researcher and Senior Software Engineer. His research interests include large-scale stream processing and automated behavioral analysis. Kevin completed his PhD at the University of Michigan in May 2009, where his thesis topic was protecting confidential information from malicious software. kevin.borders@gmail.com

Matt Burnside is a Researcher at the National Security Agency. burnside3@lnl.gov

The increasing frequency and complexity of network-based attacks is generating a correspondingly high level of interest in intrusion detection systems (IDS), which detect and filter these attacks. A variety of languages such as Snort and Bro have been developed to program an IDS to recognize specific threats, but these languages cater to specialists. We are developing a new IDS language, Chimera, that is more accessible to analysts and system administrators due to its adoption of the familiar SQL syntax.

Intrusion detection systems (IDS), operating, for example, at the switch level or as a transparent “bump on the wire,” must cope with an ever-changing landscape of threats, which requires that they be very flexible. This flexibility is realized by programmability: a programming language serves to customize the IDS to look for traffic of interest. As a result, the power of an IDS is constrained by the choice of language as well as by its physical capabilities such as throughput rate.

A general-purpose programming language is not ideally suited to the task of telling IDS systems which traffic to report or filter. Core elements of the problem domain such as operating over a stream of traffic and deconstructing packets lend themselves to special syntax that gives the language user a lot of leverage. In selecting a syntactic model, the IDS programming language needs to balance a number of factors that sometimes trade off against each other. Some of these are

- ◆ Expressivity: how well properties of interest can be described in the language;
- ◆ Efficiency: how well the description can be realized with the IDS’s capabilities;
- ◆ Accessibility: how easily the user can make use of the language’s power.

Although the first two factors are commonly considered, less thought is often given to the third. Snort [7], for example, aims to be lightweight. Its rules are easy to write and efficient to check but are limited in their capabilities. Individual packet properties can be examined, but correlating packets to investigate properties at the level of the protocol is difficult.

Bro [6] chooses to be more expressive, able to recognize protocol-level structure and to recognize richer patterns in the traffic stream. One cost of this expressivity, however, is the additional demands on the user. Writing a Bro script is more akin to a traditional programming task (albeit aided by domain-specific support), and this can limit its audience. Furthermore, performance of these scripts is dependent on subtle implementation design decisions where small changes to a script can dramatically affect performance of the whole IDS.

Although there is overlap, the audience of programmers is fundamentally different from the audience of network operators and analysts. We would like to make the power of a language like Bro more accessible to this latter pool of people, who have the domain expertise to know what they are looking for but want better tools to express their desires. This audience needs a better programming idiom. We have selected SQL as this idiom, and created the language Chimera [2] to make use of it. We have implemented Chimera with a compiler that translates to Bro, allowing us to take advantage of Bro’s expressive power and mature infrastructure.

The Chimera Language

Chimera's use of SQL structure allows it to express complex, stateful queries about data streams in a straightforward manner. We choose SQL because it is familiar to many users and uses a high-level, word-based syntax to describe the structure of data and how it is to be manipulated. Its application to processing network traffic is not exact, though, and requires some adaptation.

SQL operates over tables, in which the rows are records and the columns are fields in those records. In the network analog, packets are rows. The structure of a packet, at the IP and TCP levels as well as the application protocol level, decodes into columns. In Chimera, we do not speak in terms of packets, however, but in terms of tuples, an abstraction from relational algebra that facilitates application to generalized data flows, which may or may not be packets. Tuples are simply typed, multipart records. Unlike SQL, where table data is uniform, Chimera provides native support for variable-length records via list and map types (useful, for example, for SMTP mail headers).

The notion of a flow of data is itself a departure from SQL. Whereas we are used to thinking of an SQL query as operating over a table of fixed size, Chimera operates over streams of indeterminate length. Some SQL operations are naturally defined in terms of the cross product of all rows, an operation that doesn't make sense for a stream. Since we do not have infinite memory, we must design our operations to account for the fact that we can remember a limited number of tuples.

Beyond the principal differences just described, the SQL model and lower-level features such as its expression language fit our problem domain well. We illustrate this and introduce the language details through examples below.

Basic Queries

While there are several top-level commands in SQL, including those to create and update a data set, almost the only one of interest to Chimera is the query operation, introduced by the SELECT construct.

```
SELECT <exp> AS <name> [, AS <name>]* [modifiers]
```

A variety of the familiar SQL clauses may be used in a SELECT query, and we survey those in the next section.

As a first taste of Chimera, consider the program in Listing 1.

```
SELECT
  $.get('packets').first().get('srcip') AS srcip,
  $.get('headers').first().get('User-Agent') AS agent
FROM http
```

Listing 1: A basic Chimera query

This informational query consists of one SELECT with a FROM clause to indicate what data stream to process. There are a number of protocol parsers built into Chimera; these cover HTML, SMTP, DNS, and other common protocols. All that is needed to access the parsed stream of objects is to refer to the correct pre-defined stream. Each is a stream of tuples, all of which conform to a record structure with a specific set of named fields.

The main body of the SELECT—the lines beginning with \$.get in this example—are a comma-separated list of data items that are returned as the result of the select query. These data items can optionally be named with an AS clause, with these names used in other clauses attached to the SELECT (though none exist in this case).

Each of the two data items is constructed by code drawing from Chimera's rich expression syntax. In this case, the code performs a sequence of operations, evaluated from left to right. The stream produced by the protocol parser is accessed by referring to the special token \$. The first function call, the method get('packets') operates on the stream to obtain the raw list of packets. The result of this operation is a list, from which we pick off the first item via a second function call first(). The object we obtain is a Map, which maps names to values as in a tuple from the stream. We pick out the source IP address with another “get” call, get('srcip'). The second data item is constructed in just the same way, except by referring to the list of HTTP headers provided by the protocol parser and picking out the “User-Agent” header.

The “get” operation is so common that Chimera supports a shorter equivalent, [*field*]. An expression [*srcip*] will perform a get('srcip') function call. In addition, if the object it operates on turns out to be a list rather than a record, it applies a first() operation. Finally, if a method is not applied to any object (no dot operator), it is treated as implicitly referring to the stream as with \$. Thus, our example can be rewritten more concisely, as in Listing 2.

```
SELECT
  [packets].[srcip] AS srcip,
  [headers].[User-Agent] AS agent
FROM http
```

Listing 2: Variant form of first Chimera query

Arranging Information

With only the SELECT construct, we cannot do much data processing beyond retrieving structured data from the network packet stream. Often we want to filter and rearrange a stream to get a more concise or pertinent result. This can be done with additional modifier clauses supported in conjunction with a SELECT.

WHERE { boolexp }

The WHERE clause can be used to filter the result according to a Boolean expression. The { boolexp } is evaluated for each tuple in the stream, and only those for which the result is true are retained.

GROUP BY { exp } UNTIL { boolexp }

The GROUP BY clause operates a little differently from its SQL counterpart. Because we have a stream of input data, we cannot process an entire table at once and must consider when exactly to bundle an incoming group as a unit for processing. Controlling this “window” of processing is key to keeping execution efficient and timely. The GROUP BY clause combines like tuples according to { exp } until { boolexp } becomes true, at which point it emits the group of tuples and starts another.

Listing 3 shows an example of a query that uses the additional features discussed above.

```
SELECT count_distinct([aip]), [name]
FROM dns_rr
WHERE [aip] != NULL
GROUP BY [name]
UNTIL GLOBAL
([packets].first().timestamp() -
 [packets].last().timestamp()) > 86400
```

Listing 3: Query to list distinct IP addresses per domain name

The goal of this query is to list the distinct IP addresses per domain. It starts with the DNS protocol stream; a special form that has been decoded into individual columns (or tuples) by Chimera is provided by the dns_rr token. Tuples without an IP address are dropped by the WHERE clause. The tuples are grouped by like domain names by the GROUP BY clause, and chunked to a 24-hour window (the GLOBAL keyword indicates that the boolexp refers to the global stream rather than the element being processed). When the window of the GROUP BY has expired, the name and a count of the distinct IP addresses are constructed (utilizing a call to a built-in function count_distinct) and returned.

Working with Multiple Streams

So far, we have the ability to do detailed inspection and manipulation of a single stream. Often, however, we want to be able to correlate information learned across streams, or perform multiple manipulations of the same stream. Chimera supports this through JOIN and CREATE VIEW syntax.

JOIN { stream } ON { exp } EQUALS { exp }

A JOIN combines two streams into one. Chimera joins are required to be equi-joins, meaning { exp } expressions may compare for equality only. There are still many different ways to perform the combination. Chimera understands the standard LEFT/RIGHT/FULL, EXCLUSIVE, and OUTER dimensions. Note that not all combinations of these modifiers are supported in the current implementation.

Additionally, Chimera makes an efficiency-related distinction relevant to streams. When matching elements from the left and right streams, storing them is necessary (Chimera uses a hash table for this purpose). By default, Chimera orders the join so that left-side tuples will only match later right-side tuples, meaning only left-side ones need to be stored. The UNORDERED keyword can be used to get the traditional, symmetric behavior (at the cost of also storing right-side tuples).

CREATE VIEW { name } AS { select }

Unlike the above constructs, CREATE VIEW is not a modifier to a SELECT, but rather a top-level construct in its own right. The purpose is simply to save the results of some query by assigning a name to it.

Now we have the tools to construct complex queries that correlate across multiple streams. Consider the problem of spam detection. One way to approach this would be to write an analytic that keeps an eye out for new mail transfer agents (MTAs), and if one is seen that transmits a large amount of mail in a small amount of time, report it. We can write a query that operates over the SMTP-parsed stream, looking for MTAs in the “Received” header. For 24 hours after a new one is seen, keep a count of the number of distinct recipients from that MTA. If the amount exceeds some threshold (say 50), emit a tuple reporting this.

This query is complex in that it requires not only understanding the protocol, but keeping state on the history of traffic and correlating the new MTAs with the recipient count. Listing 4 gives an implementation in Chimera.

```
CREATE VIEW mtasmtp
AS (SELECT headers AS headers,
      [packets].timestamp() AS time0,
      [headers].find('RECEIVED').sub_regex('^.*by +', '')
      .sub_regex('.*$', '') AS mta
FROM smtp
WHERE [headers].find('RECEIVED') == /.*/.by .*/ );
CREATE VIEW mtasmtp_unique
AS (SELECT headers, mta AS mta, time0 AS time0
FROM mtasmtp
WHERE unique([mta]));
SELECT
```

```

merge([b].[headers].find('TO').split_regex(', '),
      [b].[headers].find('CC').split_regex(', '),
      [b].[headers].find('BCC').split_regex(', ')
      ).iterall{count_distinct($)}
  AS recipient_count,
[a].[mta]
FROM
  mtasmtpl_unique AS a JOIN mtasmtpl AS b
  ON [mta] EQUALS [mta]
WHERE [b].[time0] - [a].[time0] < 86400
GROUP BY [a].[mta]
UNTIL [recipient_count] > 50

```

Listing 4: Spam detection Chimera script

To start, we create two subsidiary queries with the CREATE VIEW construct. The first creates a stream `mtasmtpl`, which is a view of `smtpl` in which we have extracted the MTA from the “Received” line as well as a timestamp and the headers. The second view is created by filtering `mtasmtpl` down to unique MTAs using a Chimera built-in function `unique()` on the `mta` field that we constructed in the previous CREATE VIEW. With these two views, we are ready to construct the core query via SELECT. The two views are joined, performing the key correlation between MTA and recipients mentioned above. Only tuples within the one-day window are retained. We then extract specific recipients from all relevant headers (To, Cc, and Bcc) and feed those into a total count. This is used to trigger a new group, leading to the query output.

Related Work

We are not the first to combine SQL with a streaming data model, nor even to apply this to network traffic analysis. STREAM [5] and Aurora [1] are seminal works in this area. Research into windowed querying [4] and load shedding [8] has also been done. These efforts informed the present work, and Chimera builds on them in a few ways. Chimera adds support for structured datatypes, and operations such as SPLIT mediate between structured values in the expression language and the domain of tuples manipulated by the query language. Chimera also innovates in its support for windows, offering the UNTIL trigger for aggregates and the WINDOW condition for joins. Finally, of course, Chimera provides a translation to an external framework, Bro.

Another project that aims to support network traffic analysis using an SQL query language is Gigascope [3]. Gigascope is a vertically integrated platform where the query language is tied to the implementation platform. Chimera is designed to be platform-agnostic, and we are developing implementation targets other than Bro as well as stream sources other than network traffic. Gigascope’s query language also shares the limitations of the streaming SQL work noted above with respect to windows and to structured data.

Looking Forward

We have covered just the core features of Chimera, but there is more in the query language, the expression language, and the built-in library of functions and protocol parsers. Additional details are provided in our symposium paper [2]. We have also set up a site, www.chimera-query.org, which tracks the latest news and updates to the language and implementation.

Chimera is in its early stages yet. More experience is needed at the language level in order to assess it from a practical usability standpoint. There is no substitute for people writing queries to determine what works well and what weaknesses need to be addressed. On the implementation side, while we have a preliminary compiler to Bro, there are still missing features and much more testing needs to be done.

Our goal is to release the implementation under an OSI-approved license. We believe that this software will be especially attractive to those who use or might consider Bro, as the two can coexist, allowing different interfaces to a common installation. Our hope is to foster an ecosystem around Chimera so that the power of the IDS can be utilized more readily by system administrators and analysts.

References

- [1] Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik, "Aurora: A New Model and Architecture for Data Stream Management," 2003.
- [2] Kevin Borders, Jonathan Springer, and Matthew Burnside, "Chimera: A Declarative Language for Streaming Network Traffic Analysis," Proceedings of the 21st USENIX Security Symposium, 2012.
- [3] Chuck Cranor, Theodore Johnson, Oliver Spataschek, and Vladislav Shkapenyuk, "Gigascop: A Stream Database for Network Applications," *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, 2003, pp. 647-651.
- [4] Jin Li, David Maier, Kristin Tuft, Vassilis Papadimos, and Peter A. Tucker, "No Pane, No Gain: Efficient Evaluation of Sliding-Window Aggregates over Data Streams," 2005.
- [5] Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Manku, Chris Olston, Justin Rosenstein, and Rohit Varma, "Query Processing, Resource Management, and Approximation in a Data Stream Management System," Technical Report 2002-41, Stanford InfoLab, 2002.
- [6] Vern Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," 1999.
- [7] Martin Roesch, "Snort-Lightweight Intrusion Detection for Networks," *Proceedings of LISA '99: 13th Systems Administration Conference*, USENIX, 1999.
- [8] Nesime Tatbul and Stan Zdonik, "Window-Aware Load Shedding for Aggregation Queries over Data Streams," 32nd International Conference on Very Large Data Bases, 2009.

USENIX Board of Directors

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

PRESIDENT

Margo Seltzer, *Harvard University*
margo@usenix.org

VICE PRESIDENT

John Arrasjid, *VMware*
johna@usenix.org

SECRETARY

Carolyn Rowland
carolyn@usenix.org

TREASURER

Brian Noble, *University of Michigan*
noble@usenix.org

DIRECTORS

David Blank-Edelman, *Northeastern University*
dnb@usenix.org

Sasha Fedorova, *Simon Fraser University*
sasha@usenix.org

Niels Provos, *Google*
niels@usenix.org

Dan Wallach, *Rice University*
dwallach@usenix.org

CO-EXECUTIVE DIRECTORS

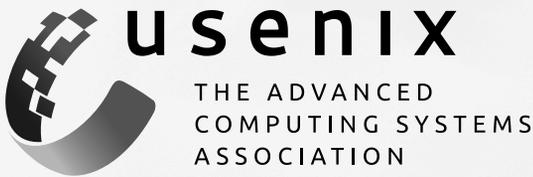
Anne Dickison
anne@usenix.org

Casey Henderson
casey@usenix.org

Who We Are

Since 1975, the USENIX Association has brought together the community of engineers, system administrators, scientists, and technicians working on the cutting edge of the computing world. Our mission is to:

- Foster technical excellence and innovation
- Support and disseminate research with a practical bias
- Provide a neutral forum for discussion of technical issues
- Encourage computing outreach into the community at large



What We Do

We offer services to the advanced computing systems community in the following ways:

- Conferences (technology sharing, community building, and educational training)
- Academic Programs & Good Works (for students, academics, and the community)
- Publications (journals, proceedings, and books)
- Membership (provides benefits and participation in shaping the industry)
- LISA (a SIG for system administrators)



Why Students Should Join

USENIX offers programs tailored especially for students, including:

- **Student Discounts:** We keep membership dues and conference registration fees at an affordable low rate for full-time students. Students can join USENIX for \$50 a year and LISA, the SIG for system administrators, for \$30 a year.
- **Conference Grants:** USENIX works with corporate partners to provide financial assistance to students to attend USENIX conferences, covering registration and helping with expenses.
- **Unparalleled Networking Opportunities:** Students who attend USENIX conferences and contribute to mailing lists have the chance to mingle with leaders in their field. Take the opportunity to chat with industry experts during the many conference Guru Is In sessions, the "hallway track," and evening events.
- **Publish Your Work:** A must-have for technology students wanting to stay ahead of the curve. CiteSeer ranks our proceedings among the top ten in highest impact for computer science. Proceedings are published for each event and are immediately available to student members. USENIX also offers a Best Student Paper Award at many events. The awards are cash prizes awarded to the best paper for which a student is the lead author at the USENIX event. Keep an eye out for our Calls for Papers (CFPs), and feel free to submit a paper!
- **Special Discounts:** USENIX offers its members discounts on everything from *Linux Journal* to No Starch Press and O'Reilly books, and more.

www.usenix.org/students

Practical Perl Tools

What's Up, perldoc?

DAVID N. BLANK-EDELMAN



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

For the past million or so columns, we've taken a look at how to use Perl to do X or how to use Perl to do X by communicating with such and such Web service. Every once in a while I think it is good to step back and talk about how to use Perl, period. This is going to be one of those columns where we go back to some of the basics that you may have missed as you zoomed right to the "high priestess of Perl" status you now hold. We're going to talk a bit about documentation in the Perl world, both how to consume and create it.

Rockin' the perldoc

If you looked back at all of the classes I've taught over the years with Perl content in them, I think you could make a safe bet that there would be someone in every class who has never heard of the "perldoc" command. If you really wanted to cash in, you would bet that a large majority of the people in the room who already knew about the command didn't realize everything it can do. Want to take that bet? Read on.

"perldoc" is a command that ships with Perl. It is designed to show you various parts of the Perl documentation installed on your system. The documentation it can display includes all of the text documentation that ships with Perl plus the manual pages for the core modules and any modules you've installed.

So, for example, if you wanted to see the manual page for the File::Spec module, you could type

```
perldoc File::Spec
```

and perldoc would find the documentation, run it through a converter to convert it into man page format, hand the man page to whatever your system uses to display them (e.g., nroff -man), and then show it to you in your favorite pager. This works fine for the larger Perl documentation sections as well:

```
perldoc perl5dsc
```

(Wait, you mean you didn't know that Perl shipped with such excellent doc as the Perl Data Structures Cookbook and perlperf, the Perl Performance and Optimization Techniques tome? Well, you best type "perldoc perl" right now and then come back to this article in a few hours after you've read some of the good stuff you'll find. For a more verbose version of that listing, try "perldoc perltoc".)

A very reasonable question you might have about this command is "Why not just type 'man File::Spec'?" It's a good question because for the core modules, on many default installations of Perl, this will indeed work. perldoc is preferable for at least two reasons:

1. perldoc will find documentation within a copy of Perl that wasn't installed in the default place ("man" won't find it unless you changed your MANPATH), and
2. unless your Perl was installed carefully with this in mind, non-core modules may install their documentation in a different place or with a different suffix than "man" expects.

But perhaps the best argument for perldoc over man is about to be revealed when we look at the cool stuff it can do.

Not to give away the best hint first, but I don't think I would find coding in Perl as easy as I do if it wasn't for the `-f` flag to perldoc. The `-f` flag lets you look up the doc for all of the many, many Perl built-in language functions. Can't remember what the order of the arguments of the `split()` is? Type:

```
perldoc -f split
```

and you'll see

```
split /PATTERN/,EXPR,LIMIT
split /PATTERN/,EXPR
split /PATTERN/
split Splits the string EXPR into a list of strings and
returns that list. ...
```

along with all of the rest of the documentation on that function. A similar flag, `-v`, helps you look up the documentation for a dizzying array of predefined variables in Perl. So let's say you were reading someone else's code and you run into the `$(` variable. If you didn't want to shake your head sadly and say, "Kids these days, with their wacky emoticons, I just don't understand them..." you could instead type

```
perldoc -v '$('
```

and you'd see

```
$_REAL_GROUP_ID
$_GID
$( The real gid of this process. ...
```

For people just starting out with Perl, it can be helpful to type commands such as

```
perldoc -v '%ENV'
```

to see what the `%ENV` hash is and what it does.

Beginners may be aware that there exists a substantial nine-part FAQ about Perl and how to use it, but I'd dare say that they probably don't know they can search it using perldoc's `-q` flag. If you type `perldoc -q {something}`, it will search for that something (using a regular expression search, btw) in the questions text from all of the sections of the `perlfaq`. If I typed `perldoc -q mail`, for example, it would show me the answers to the following questions:

- ◆ What mailing lists are there for Perl?
- ◆ How do I parse a mail header?
- ◆ How do I check a valid mail address?
- ◆ How do I return the user's mail address?
- ◆ How do I send mail?
- ◆ How do I use MIME to make an attachment to a mail message?
- ◆ How do I read mail?

I may have listed my most used perldoc hint first, but I have saved the most surprising for last. Very few people know about the `-l` and the `-m` flags to perldoc. Here's where they come in handy: anyone who has done any substantial amount of Perl programming has had to go look at the Perl source to a module they are using. You do this for any number of reasons: sometimes it is sheer curiosity for how something has been implemented; sometimes we're struggling to figure out how to use a module and have to resort to the source code for guidance; other times we need to better understand an object it defines and so on.

The first step toward consulting the source code of a module is finding where it lives on disk. This can be done using the `-l` flag. To return to the very first example, if we wanted not only to see the documentation for `File::Spec`, but where it was installed, we could type

```
perldoc -l File::Spec
```

and find out this path on OS X's Mountain Lion release:

```
/System/Library/Perl/5.12/darwin-thread-multi-2level/
File/Spec.pm
```

There's a bunch of auxiliary info we're getting back from this little command, including just where modules are installed on the system and the version of Perl in play (or at least the versioned directory presumably associated with that version).

But that's just *where* the file is located; even cooler still is to run perldoc using the `-m` flag:

```
perldoc -m File::Spec
```

```
package File::Spec;

use strict;
use vars qw(@ISA $VERSION);
```

```
$VERSION = '3.31_01';
$VERSION = eval $VERSION;

my %module = (MacOS => 'Mac',
              MSWin32 => 'Win32',
              os2 => 'OS2', ...
```

Why yes, that is the actual source of the module. With one command we can easily see the source of (the main file of) a module. Very handy sometimes!

perldoc from Orbit

perldoc on your machine works great for providing the documentation for Perl things installed on that machine, but what if you wanted to consult the documentation for a different version of Perl? A lovely resource for that sort of thing is the Web site <http://perldoc.perl.org>, which has the full doc sets for 16 versions at last count and offers a usable Web interface to boot.

If you fall in love with that Web interface and can't bear to be without it even when you are disconnected from the Intertubes, Jon Allen, the site's creator, offers a module to help you run an HTTP-served version of the doc on any machine. Perldoc::Server will provide a Catalyst-based Web application that can be started up just by typing "perldoc-server". Perldoc::Server will then run a tiny Web server by default on port 7375 ("PERL" on a phone keypad, explains the doc).

So far it appears all of the command line stuff we've talked about displays documentation for things that have been installed locally. That seems kind of limiting. Perhaps you'd like to see the documentation for something you haven't installed on that machine. Pod::Cpandoc will do this for you. When you install it, it provides a command "cpandoc," which can stand in for perldoc if you'd like. If you type

```
cpandoc SomeModule
```

it will display the locally installed doc (just as perldoc would do), and if it can't find it there, it will fetch it right from CPAN. And in case you are curious, the perldoc flags I was crowing about above still work. If you type

```
cpandoc -m SomeModule
```

and SomeModule isn't installed, it will still let you read the source for that module by grabbing it from CPAN. cpandoc even slips in a flag perldoc doesn't have: -c. This flag will show the Changes (i.e., a changelog) for a module if it has one.

Good Documentation Starts at Home

I'd like to switch gears now and move away from how to consume documentation to the question of how to create good documentation for the Perl code you write. The first thing you'll want to do

is take a quick look at the Pod (Plain Old Documentation) documentation with a command like "perldoc perlpod". The reason why I say "quick" is I find that reference page to be a bit overwhelming if you've never seen Pod before. Glance over it, maybe make note of the sections on how to embed Pods in Perl Modules and Hints for Writing Pod, but don't get nervous. Pod is described in the doc as "a simple-to-use markup language used for writing documentation for Perl, Perl programs, and Perl modules," and it really is. I think the easiest way to learn Pod is to pick a simple module or command that has been marked up, look at the source, and basically copy what you see there.

For example, if we looked at the source for the cpandoc command line script, we'd see:

```
#!/usr/bin/env perl
use Pod::Cpandoc;
exit( Pod::Cpandoc->run() );

__END__

=head1 NAME

cpandoc

=head1 DESCRIPTION

See L<Pod::Cpandoc> and L<Pod::Perldoc>.

=cut
```

The first part loads the module and calls a function to start it running. But that's not the interesting part for our discussion. After the executable code, there is a marker of __END__ to let Perl's parser know that it has finished finding any code it should read in. From that point on, we see Pod format doc with two headings (=head1), a little bit of body text, and a =cut command to indicate the end of that Pod block. Here we are seeing Pod at the end of the Perl code, but it is also designed to be interleaved with executable code so that the doc is right next to the code it documents. When used with care, this programming style can be used quite effectively. If you find you like the idea of combining code and doc and you'd like to see how far the idea can be taken, I'd encourage you to check out Knuth's work on literate programming (and in case you are curious, I was going use Pod and literate programming in the same sentence until the Wikipedia entry slapped me down hard).

Rather than dwelling on Pod for the entirety of this section, I'd like to end with a look at a spiffy documentation-related module that has actually shipped with Perl since the 5.6 days back in 2000. The Pod::Usage module comes with a pod2usage() function that can do magic if you've embedded Pod documentation

in your code. `pod2usage()` knows how to find the USAGE and related sections of your Pod documentation and spit them out at a given level of verbosity. You can programmatically decide whether it will show just the USAGE text (i.e., the SYNOPSIS) or even the whole man page. To see all this in action, let's look at the recommended use sample code from the documentation:

```
use Getopt::Long;
use Pod::Usage;

my $man = 0;
my $help = 0;
## Parse options and print usage if there is a syntax
## error, or if usage was explicitly requested.
GetOptions('help?' => \$help, man => \$man)
    or pod2usage(2);
pod2usage(1) if $help;
pod2usage(-verbose => 2) if $man;

## If no arguments were given, then allow STDIN to be
## used only if it's not connected to a terminal
## (otherwise print usage)
pod2usage("$0: No files given.")
    if (@ARGV == 0) && (-t STDIN));
__END__

=head1 NAME

sample - Using Getopt::Long and Pod::Usage

=head1 SYNOPSIS

sample [options] [file ...]

Options:
  -help      brief help message
  -man      full documentation
```

```
=head1 OPTIONS

=over 8

=item B<-help>

Print a brief help message and exits.

=item B<-man>

Prints the manual page and exits.

=back

=head1 DESCRIPTION

B<This program> will read the given input file(s)
and do something useful with the contents thereof.

=cut
```

Here we can see a more complete Pod example topped off by calls to `pod2usage`. If this script gets called with a `-man` switch, it will show the entire manual page. If it is called with a `-help` or `-?` switch, only the SYNOPSIS section will be printed. This is similarly printed if the script doesn't receive the input it expects (i.e., is called with no arguments) as a way of demonstrating how `pod2usage()` can help provide useful error messages. I think it is a nice touch for a script to be able to supply its own documentation if asked.

So go, document lots. Take care and I'll see you next time.

Some Easily Overlooked But Useful Python Features

DAVID BEAZLEY



David Beazley is an open source developer and author of the *Python Essential Reference* (4th Edition, Addison-Wesley, 2009). He is also known as the creator of Swig (<http://www.swig.org>) and Python Lex-Yacc (<http://www.dabeaz.com/ply/index.html>). Beazley is based in Chicago, where he also teaches a variety of Python courses. dave@dabeaz.com

For the past eight months, I've been locked away in my office working on a new edition of the *Python Cookbook* (O'Reilly & Associates). One of the benefits of writing a book is that you're forced to go look at a lot of stuff, including topics that you think you already know. For me, the *Cookbook* was certainly no exception. Even though I've been using Python for a long time, I encountered a lot of new tricks that I had never seen before. Some of these were obviously brand new things just released, but many were features I had just never noticed even though they've been available in Python for many years.

So, in this article, I'm going to take a tour through some of these easily overlooked features and show a few examples. Most of these features are extremely short—often one-liners that you can start using in your code. There's no particular order to the discussion; however, I do assume that you're using the latest version of Python, which is currently version 3.3. Many of the features presented will work in older versions, too.

Checking the Beginning and End of Strings

Sometimes you need to check the beginning or end of a string quickly to see whether it matches some substring. For example, maybe you've written some code that checks a URL like this:

```
# Check a URL for HTTP protocol
if url[:5] == 'http:' or url[:6] == 'https:':
    ...

# Alternative using a regex
if re.match('(http|https):', url):
    ...
```

Sure, both solutions “work,” but they're not nearly as simple as using the `startswith()` or `endswith()` method of a string. Just supply a tuple with all of the possible options you want to check. For example:

```
if url.startswith(('http:', 'https:')):
    ...
```

Not only does this solution involve very little code, it runs fast and it's easy to read; however, you only get that benefit if you know that you can do it in the first place.

Tricks with `format()`

While I was teaching a training course a few years back, somebody pulled me aside to test me on their favorite job interview question for Python programmers. The problem was to write code that formatted an integer with the thousands comma separator properly placed in the right positions. I can only assume that he wanted me to write a solution like this:

```
>>> x = 1234567890
>>> print(', '.join(reversed([str(x)[::-1][n:n+3][::-1]
                             for n in range(0, len(str(x)), 3)])))
1,234,567,890
>>>
```

Such problems are so much easier to solve if you just use `format()` like this:

```
>>> print(format(x, ','))
1,234,567,890
>>>
```

Ah, yes. That's much nicer. `format()` also works in ways that you might not expect with certain sorts of objects. For example, you can use it to format dates:

```
>>> from datetime import datetime
>>> d = datetime(2012, 12, 21)
>>> format(d, '%a, %b %d %m, %Y')
'Fri, Dec 21 12, 2012'
>>> format(d, '%a, %b %d, %Y')
'Fri, Dec 21, 2012'
>>> print('The apocalypse was on {}'.format(d))
The apocalypse was on 2012-12-21
>>>
```

Faster Date Parsing

On the subject of dates, I've recently learned that the common built-in function `strptime()` is dreadfully slow if you ever need to use it to parse a lot of dates. For example, suppose you were parsing a lot of date strings like this:

```
s = '16/Oct/2010:04:09:01'
```

The easiest way to parse it is to use `datetime.strptime()`. For example:

```
>>> import datetime
>>> d = datetime.strptime(s, '%d/%b/%Y:%H:%M:%S')
>>> d
datetime.datetime(2010, 10, 16, 4, 9, 1)
>>>
```

If you didn't know about such a function, you might be inclined to roll your own custom date parsing function from scratch. For example:

```
import calendar
months = {name:num for num, name in enumerate(calendar.
month_abbr)}

def parse_date(s):
    date, _, time = s.partition(':')
    day, mname, year = date.split('/')
    hour, minute, second = time.split(':')
```

```
    return datetime(int(year), months[mname], int(day),
                    int(hour), int(minute), int(second))
```

Here's an example of using the above function:

```
>>> d = parse_date(s)
>>> d
datetime.datetime(2010, 10, 16, 4, 9, 1)
>>>
```

More often than not, creating your own implementation of a function already built in to Python is a recipe for failure; however, not so in this case. It turns out that the custom `parse_date()` function runs nearly six times faster than `strptime()`. That kind of improvement can be significant in programs that are performing a lot of date parsing (e.g., parsing dates out of huge log files, data files, etc.).

One of the reasons `strptime()` is so slow is that it's actually written entirely in Python. Because it has to do a lot more work, such as interpreting the format codes, it's always going to be slower than a custom-crafted implementation aimed at a very specific date format.

New Time Functions

Not all is lost in the time module, however. Python recently picked up new timing-related functions. For making performance measurements, you can use the new `time.perf_counter()` function. For example:

```
import time
start = time.perf_counter()
...
end = time.perf_counter()
print('Took {} seconds'.format(end-start))
```

`perf_counter()` measures elapsed time using the most accurate timer available on the system. This eliminates some of the guesswork from benchmarking as common functions such as `time.time()` or `time.clock()` often have platform-related differences that affect their accuracy and resolution.

Similarly, the `time.process_time()` function can be used to measure elapsed CPU time. For example:

```
import time
start = time.process_time()
...
end = time.process_time()
print('Took {} CPU seconds'.format(end-start))
```

Last, but not least, the `time.monotonic()` function provides a monotonic timer where the reported values are guaranteed never to go backward—even if adjustments have been made to the system clock while the program is running.

All three of these time-related functions are only usable for working with time deltas. That is, you use them to compute time differences as shown. Otherwise, the value returned, although having a unit of seconds, doesn't have any useful meaning and may vary by platform.

Creating a File Only If It Doesn't Exist

Suppose you wanted to write to a file, but only if it doesn't exist. This is now easy in Python 3.3. Just give the 'x' file mode to `open()` like this:

```
>>> f = open('newfile.txt', 'x')
>>> f.write('Hello World')
>>> f.close()
>>>
>>> f = open('newfile.txt', 'x')
Traceback (most recent call last):
  File "", line 1, in
FileExistsError: [Errno 17] File exists: 'newfile.txt'
>>>
```

Although it's a simple feature, this saves you from first having to test like this:

```
import os.path
if not os.path.exists(filename):
    f = open(filename, 'w')
else:
    raise FileExistsError('File exists')
```

System Exit with Error Message

When writing scripts, it is common to follow a convention of writing a message to standard error and returning a non-zero exit code to report a failure. For example:

```
import sys

if must_die:
    sys.stderr.write('It failed!\n')
    raise SystemExit(1)
```

It turns out that all of the above code, including the import statement, can just be replaced by the following:

```
if must_die:
    raise SystemExit('It failed!')
```

This writes the message to standard error and exits with a code of 1. Who knew it was that easy? I didn't until recently.

Getting the Terminal Width

Sometimes you'd like to get the terminal width so that you can properly format text for output. To do this, you can try to fiddle around with environment variables, TTYs, and other details.

Alternatively, you could just use the new `os.get_terminal_size()` function. For example:

```
>>> import os
>>> sz = os.get_terminal_size()
>>> sz.columns
108
>>> sz.lines
25
>>>
```

On the subject of formatting text for a terminal, the `textwrap` module can be useful. For example, suppose you had a long line of text like this:

```
s = "Look into my eyes, look into my eyes, the eyes, the eyes, \
the eyes, not around the eyes, don't look around the eyes, \
look into my eyes, you're under."
```

You can use `textwrap.fill()` to reformat it:

```
>>> import textwrap
>>> print(textwrap.fill(s, 70))
Look into my eyes, look into my eyes, the eyes, the eyes, the
eyes,
not around the eyes, don't look around the eyes, look into my
eyes,
you're under.

>>> print(textwrap.fill(s, 40))
Look into my eyes, look into my eyes,
the eyes, the eyes, the eyes, not around
the eyes, don't look around the eyes,
look into my eyes, you're under.
```

Interpreting Byte Strings as Large Integers

Recently, I was working on a problem where I needed to parse and manipulate IPv6 network addresses such as "1234:67:89:aab:b:43:210:dead:beef". I thought about writing some custom parsing code, but realized that it's probably better to do it using functions in the `socket` module:

```
>>> addr = "1234:67:89:aabb:43:210:dead:beef"
>>> import socket
>>> a = socket.inet_pton(socket.AF_INET6, addr)
>>> a
b'\x124\x00g\x00\x89\xaa\xbb\x00C\x02\x10\xde\xad\xbe\xef'
>>>
```

Yes, this "parsed" the IPv6 address, but it returned it as a 16-character byte-string representation of the 128-bit integer value. This is not quite what I had hoped for, so how was I going to turn such a string into a large integer value? It turns out it's trivial. Just use `int.from_bytes()` like this:

```
>>> int.from_bytes(a, 'big')
24196111521439464807328179944418033391
>>>
```

The second argument to `from_bytes()` is the byte order. Similarly, if you have a large integer value, you can go the other direction like this:

```
>>> x = 123456789012345678901234567890
>>> x.to_bytes(16, 'little')
b'\xd2\n?N\xee\xe0s\xc3\xf6\xf9\xe9\x8e\x01\x00\x00\x00'
>>> x.to_bytes(20, 'little')
b'\xd2\n?N\xee\xe0s\xc3\xf6\xf9\xe9\x8e\x01\x00\x00\x00\x00\x00\x00\x00'
>>> x.to_bytes(20, 'big')
b'\x00\x00\x00\x00\x00\x00\x00\x01\x8e\xe9\xf6\xc3s\xe0\xeeN?\n\xd2'
>>>
```

Manipulating Network Addresses

On the subject of manipulating network addresses, it became a whole lot easier in Python 3.3 with the addition of a new `ipaddress` library. Here's a short example of representing an IPv4 network and printing a list of all of the hosts contained within it:

```
>>> import ipaddress
>>> net = ipaddress.IPv4Network('192.168.2.0/29')
>>> net.netmask
IPv4Address('255.255.255.248')
>>> for n in net:
...     print(n)
...
192.168.2.0
192.168.2.1
192.168.2.2
192.168.2.3
192.168.2.4
192.168.2.5
192.168.2.6
192.168.2.7
>>> a = ipaddress.IPv4Address('192.168.2.14')
>>> a in net
False
>>> str(a)
'192.168.2.14'
>>> int(a)
3232236046
>>>
```

Calculating with Key Functions

At some point, most Python programmers encounter a problem where they need to sort some data. For example, suppose you had some stock data:

```
stocks = [ # (name, shares, price)
          ('AA', 100, 32.20),
          ('IBM', 50, 91.10),
          ('CAT', 150, 83.44),
          ('MSFT', 200, 51.23),
          ('GE', 95, 40.37),
          ('MSFT', 50, 65.10),
          ('IBM', 100, 70.44)
        ]
```

To sort the data, you can use the `sorted()` function; however, it only sorts according to the first tuple field (the name), producing this:

```
>>> sorted(stocks)
[('AA', 100, 32.2), ('CAT', 150, 83.44), ('GE', 95, 40.37), ('IBM', 50,
91.1),
 ('IBM', 100, 70.44), ('MSFT', 50, 65.1), ('MSFT', 200, 51.23)]
>>>
```

To change the sort, you can supply an optional “key” to `sorted()` like this:

```
>>> # sort by shares
>>> sorted(stocks, key=lambda s: s[1])
[('IBM', 50, 91.1), ('MSFT', 50, 65.1), ('GE', 95, 40.37), ('AA', 100,
32.2),
 ('IBM', 100, 70.44), ('CAT', 150, 83.44), ('MSFT', 200, 51.23)]

>>> # sort by price
>>> sorted(stocks, key=lambda s: s[2])
[('AA', 100, 32.2), ('GE', 95, 40.37), ('MSFT', 200, 51.23), ('MSFT',
50, 65.1),
 ('IBM', 100, 70.44), ('CAT', 150, 83.44), ('IBM', 50, 91.1)]
>>>
```

The key function is expected to take an element and return a value that's actually used to drive the sorting operation. In this example, the function is picking out the value of a specific column.

It's not as widely known, but the special key function can be given to a variety of other data-related functions. For example:

```
>>> # Find lowest price
>>> min(stocks, key=lambda s: s[2])
('AA', 100, 32.2)
```

Some Easily Overlooked But Useful Python Features

```
>>> # Find maximum number of shares
>>> max(stocks, key=lambda s: s[1])
('MSFT', 200, 51.23)

>>> # Find 3 lowest prices
>>> import heapq
>>> heapq.nsmallest(3, stocks, key=lambda s:s[2])
[('AA', 100, 32.2), ('GE', 95, 40.37), ('MSFT', 200, 51.23)]
>>>
```

Final Words

That's about it for now. In the next issue, I'll plan to give a recap of highlights from the PyCon 2013 conference (held in March).

Professors, Campus Staff, and Students— do you have a USENIX Representative on your campus? If not, USENIX is interested in having one!

The USENIX Campus Rep Program is a network of representatives at campuses around the world who provide Association information to students, and encourage student involvement in USENIX. This is a volunteer program, for which USENIX is always looking for academics to participate. The program is designed for faculty who directly interact with students. We fund one representative from a campus at a time. In return for service as a campus representative, we offer a complimentary membership and other benefits.

A campus rep's responsibilities include:

- Maintaining a library (online and in print) of USENIX publications at your university for student use
- Providing students who wish to join USENIX with information and applications
- Distributing calls for papers and upcoming event brochures, and re-distributing informational emails from USENIX
- Helping students to submit research papers to relevant USENIX conferences
- Encouraging students to apply for travel grants to conferences
- Providing USENIX with feedback and suggestions on how the organization can better serve students

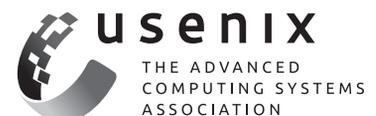
In return for being our "eyes and ears" on campus, representatives receive a complimentary membership in USENIX with all membership benefits (except voting rights), and a free conference registration once a year (after one full year of service as a campus rep).

To qualify as a campus representative, you must:

- Be full-time faculty or staff at a four year accredited university
- Have been a dues-paying member of USENIX for at least one full year in the past

For more information about our Student Programs, contact Julie Miller, Marketing Communications Manager, julie@usenix.org

www.usenix.org/students



iVoyeur

Nagios XI (fin)

DAVE JOSEPHSEN



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

We have in our community certain long-running esthetic and philosophical disagreements to which we sometimes refer as “religious battles.” The ancient vi vs Emacs battle probably exemplifies this practice, there being both a “Cult of vi” [1] as well as a “Church of Emacs” [2] (and even an alt.religion.emacs mailing list).

Being a pedantic sort of fellow, I sometimes wonder how we, as a community, struck upon that particular adjective and how the theists among us feel about our use of it in a pejorative sense. It may just be me, but when we describe these battles as “religious” it seems to imply some arbitrary and thoughtless or perhaps involuntary choosing of sides. A heritage pushed on us, or the side of the river where we are born; an idea we don't understand but for which we are forever doomed to be an apologist, or an assertion we doubt and yet must defend with our lives.

But in truth this is rarely the case with the actual battles. Everyone I know who has chosen either vi or Emacs has done so deliberately, not only fully aware of the specific reasons why, but unmindful of this global struggle for text editor domination (except for the MIT graduates among us). There are, after all, real and practical reasons for choosing one or the other.

I suppose those things are sometimes fair to say about religion among the general public, but in my experience it's not true of the theists in our community, who are likely to have chosen their beliefs with the same unbiased thoughtfulness they used in choosing their text editor. I don't say that out of ignorance or political correctness. Some of the best conversations I've had at LISA have been late night semi-drunken arguments about the nature of the universe with atheists, agnostics, and theists of various description. These usually start in the deep-end and continue into vast and unknown waters where we usually encounter Descartes, Rutherford, Aquinas, Darwin, Locke, and many other gentlemen of their ilk.

It might be that, having chosen a side, it's difficult for us to understand why other people would choose something different, and therefore those people over there who chose differently look strange and silly. And by strange I mean dancing around with snakes, healing each other in big white tents, holding literary critiques with questionable conclusions at bible camp [3] strange. *Do those people have ANY idea what they're doing?* That seems a more apt description of the Emacs vs vi battle, but it's not fair to say about religion.

The problem is: all of that strange stuff I mentioned—that bible camp snake healing—none of that really has anything to do with theism. It's just people being weird and silly in a god-themed sort of way. If I surgically removed religion from their minds tomorrow, they'd just channel all of that manic, scatterbrained silliness into weird practices in the name of science, and find entirely new reasons to knock on your door, lobby for weird laws, and write questionable things in textbooks.

Insomuch as this is what we mean when we point at something and call it a religious debate, I'm beginning to think we've made an unfortunate choice in that particular adjective. At least we've done a poor job of assessing root cause. Beliefs don't hold people, it's the other way around. If people are acting silly in the name of religion, it's unfair of us to blame religion.

Isn't the fact that we have these sorts of debates in the nerdosphere—to the extent that we've literally formed tongue-in-cheek churches around them—proof of something decidedly not religious? Some agnostic quirk of human nature that inclines us to irrationally take sides against each other over trivialities? Isn't "The Church of Emacs" just nerds being silly in a god-themed way?

We do this an awful lot actually, from Maxwell's demon [4] to "magic smoke" [5] to "The Cathedral and the Bazaar" [6], we nerds have surrounded ourselves with supernatural imagery since the beginning. That's quite natural I think, given the quantity of time we spend with one foot in another world, designing, building, and planning in non-physical, supernatural space.

Nagios, which happens to be a recursive acronym for "Nagios Ain't Gonna Insist On Sainthood," is no stranger to the phenomenon we describe as a religious debate. In my first article on Nagios XI I alluded to some of the complaints voiced by adherents of, or dare I say converts to, commercial monitoring systems. In that article I introduced Nagios XI and described its architecture—all the ways that it was and was not Nagios. In my second article on XI, I covered the new auto-discovery functionality and configuration wizards, both of which put an end to manual Nagios configuration, a favorite topic of the anti-Nagios ... heathens. In this, my third and final Nagios XI article, I'm going to quell the flames of that debate a little more by directly addressing the rest of the complaints that seem to be driving admins away from Nagios and toward commercial monitoring solutions.

Nagios Is Ugly Because It Doesn't Have Pretty Graphs

With Nagios core, the admin has to enable performance data processing in the `nagios.cfg` and then grab some external software to parse the performance data output from Nagios, and shove it all into RRDs. The two most popular choices are NagiosGraph [7] and PNP4Nagios [8], both of which I've written about in the past.

In XI, however, PNP4Nagios is integrated out of the box, and definitions exist for all included plugins. This means that without any additional configuration whatsoever you get time series data for every service you configure. The RRDtool graphs are so well integrated into the new XI user interface that the uninitiated user would never guess PNP or RRDtool were community-sourced add-ons, so you get a snazzy UI without losing any of the power and flexibility that these community-driven development efforts provide.

In addition to the RRDtool graphs, small bar-graph visualizations for metrics collected by the Nagios Core daemon, as well as remote execution tools such as NRPE, are sprinkled throughout

the interface. These do a great job of conveying capacity planning info at a glance, as well as giving the UI a very polished look.

NagVis, a tool for overlaying status data from Nagios over graphics (e.g., network diagrams or geographical maps), is installed and available in the "Maps" section of the "Home" view, and setting up your own NagVis diagrams couldn't be easier.

But wait, that's not all; rounding out the time series visualization is a Graph Explorer tool, which allows you to draw among other things, ad hoc time series and "stacked" time series graphs. The Graph Explorer uses a commercial JavaScript library from Highcharts.com and looks quite elegant. The data comes from the RRDs resident on the Nagios server via `rrdtool` fetch, and is provided to the end-users' browser to compute the graph locally. This saves the server's CPU and provides a snappy, feature-rich data visualization, allowing you to scale the graph by dragging to select a range, and providing pop-up numerical values when you mouse over any data areas. The "stacked" time series graphs include time-shifted historical data, so you can easily compare today's data to that of yesterday etc.

The Nagios UI Sucks Because You Can't Extend It

The Nagios Core UI is implemented as a series of CGIs written in C, and hasn't had a major overhaul in years. It is often maligned for its "outdated" design, and was one of the central points of contention in the Nagios community that gave birth to the fork that became Icinga. Several free replacement UIs are available from the community, the most popular of which are MK_Multisite [9] and `thruk` [10].

XI, by comparison, comes with a wholly new and redesigned PHP-based user interface. The UI as a whole is highly modular, incorporating add-on components to implement extra features. This enables the XI developers to react quickly to the needs of the user community by adding features to the UI as needed or even custom developing features for larger end-users with special needs. A notable example is the "Operations Screen," which is intended to be displayed on a dedicated monitor in a network operations center. In addition to this and other single-page summaries, custom views can be configured to rotate between pages with more detailed information on timed intervals.

Another component that implements a feature for which the core community has been begging for years is the "Mass Acknowledgment Component." This allows an admin to schedule downtime and acknowledge problems for groups of hosts and services. I know more than one sysadmin who would purchase XI for this feature alone.

Nagios-Generated Reports Look Boring

Nagios Core is capable of generating usage and availability reports by way of the "Reporting" section off the left nav. These

reports are sort of difficult to build, export, and link to, and they're basically just spreadsheets. Fine for you and me, but not something that would impress the suits.

The "Reporting" tab in XI by comparison shows some interesting data visualization techniques from the Neoformix data-visualization field. Components that implement heatmaps, force directed graphs, and stream graphs have been added to the classic reporting options. Several shiny new implementations of the core reports are also provided, each of which I find generally cleaner than their legacy counterparts and more likely to impress the wearers of neckties and high-heels in our lives. The new reports may be exported in CSV and PDF formats with the click of a button. The button, which links to a predictable URL, makes it possible for us shorts and t-shirt wearers to grab the reports automatically with tools like curl and wget.

Nagios Can Only Model Hosts and Services

Nagios Core tracks hosts and the services running on those hosts. Any larger sort of entity composed of groups of hosts, or services from groups of hosts is more difficult to model, though there are add-ons like check_mk that make this sort of thing possible.

Nagios XI, on the other hand, contains wrapper logic for grouping individual services together into higher level entities called business processes. The intent here is to implement what The Gardiner Group calls "BAM," or "Business Application Monitoring." BAM attempts to provide real-time status for critical business entities like a sales catalog Web site, or corporate email. Nagios XI implements BAM by breaking a high-level concept like "corporate email" into its requisite pieces— mail transfer agents (MTAs), mail exchangers (MXs), groupware systems, and databases—and then quantifying the relative importance of each of the services that make up those pieces as well as describing dependency relationships between them.

XI business process groups contain services that are said to be "essential" or "non-essential." A database service in our example might be considered essential, while the SMTP port on a single mail exchanger might be "non-essential" (because they are usually redundant, and even if they go down, the mail will queue somewhere else). When any essential service or the combination of all non-essential services goes critical, the XI business process logic registers this as a "problem."

Each business process group contains critical and warning thresholds that depend on the number of problems that are occurring in the group. In our example, we might imagine two

business process groups, one for SMTP-speakers (MXs and MTAs) and one for SQL-speakers (groupware systems and DBs). If the latter group registers a single problem, because a database is down, that might throw the whole group into a warning state.

Business process groups can contain other nested business process groups and so on. Our top-level entity, "corporate email," is therefore just a business process group that contains the two groups described above. It is configured just like the other two groups such that a single "problem" in any of the nested groups causes it to go into a warning state. Finally, notification commands can be assigned on each business process group in the same way they are assigned to individual host and service events. Additionally, visualization widgets exist for the top-level groups. These can be added to any dashboard or view and allow the user to drill down into the groups to see what services or sub-groups they contain.

I hope you've enjoyed this series on Nagios XI as much as I've enjoyed writing it. If you're in a corporate environment, or find yourself in want of a turn-key commercial systems monitoring solution, I can highly recommend Nagios XI.

Take it easy.

References

- [1] The Cult of vi: <http://6thstreetradio.org/~davek/vi.html>.
- [2] The Church of Emacs: <http://www.emacswiki.org/emacs/ChurchOfEmacs>.
- [3] Harry Potter is the Devil: <http://www.youtube.com/watch?v=RSwZJ55g80Q>.
- [4] Maxwell's demon of thermodynamics: http://en.wikipedia.org/wiki/Maxwell%27s_demon.
- [5] Magic smoke: http://www.outpost9.com/reference/jargon/jargon_28.html#TAG1095.
- [6] ESR's "The Cathedral and the Bazaar": <http://www.catb.org/esr/writings/homesteading/>.
- [7] NagiosGraph: <http://nagiosgraph.sourceforge.net>.
- [8] PNP4Nagios: <http://www.pnp4nagios.org>.
- [9] MK_Multisite: http://mathias-kettner.de/checkmk_multisite.html.
- [10] Thruk: <http://www.thruk.org>.

ROBERT G. FERRELL



Robert G. Ferrell is a fourth-generation Texan, literary techno-geek, and finalist for the 2011 Robert Benchley Society Humor Writing Award. rgferrell@gmail.com

There was a recent news item in my local paper, um, online news journal, about a young motorcyclist who ran off the road and was, tragically, killed. In his left hand the authorities found a smartphone, and theorized that he had been texting.

On a motorcycle, at highway speeds.

Texting.

If this were an isolated incident, a statistical freak, I could set it aside as just another sad and pointless traffic fatality. It is no such thing, however. The fact is that as the perimillennial children grow up, their pathological insistence on permanent attachment to the umbilical cord of connectivity is going to be the cause of more and more like catastrophes. I've tried hard not to be that old curmudgeon who sits in a rocking chair on the front porch and decries the upcoming generation and their unwholesome habits, but I'm finding it increasingly difficult as the always-connected disease progresses. Even I am wearing two cell phones as I write this.

Privacy issues aside, what will happen to the concept of "alone time" in the future? Are we heading for a "Borg"-type hive mind society? Will people who disconnect themselves be considered insane or social pariahs? Are you going to eat that last high-fiber bran muffin with currants? I like them slathered in artificially-flavored whipped margarine-like substance, myself.

I cherish my time totally unplugged from humanity: long walks on my property, driving with the radio off, or even simply sitting in my library with a book, swatting wasps that somehow got in under the eaves and can't find their way out. That's where I do my best thinking, if that verb can be said to apply to my often haphazard neural activity. I can't imagine how I would cope with the constant connectivity that seems to be imperative to a significant slice of today's youth, who at least are usually too preoccupied playing Words With Friends to mess around on my lawn. (Incidentally: OMG, LMAO, BBL, and BFF are not actually words.)

All right, that's enough old-guy whining. Let's move on to something else.

Somewhere deep in one of the human hippocampi (I've never been sure why I would be carrying around a pair of academic institutions for large water-loving mammals lodged in my limbic system, but neuroscientists insist and I'm too busy to argue) is a little-known nodule about the size of a guitar-picking blister on a "Dukes of Hazzard" action figure that controls our response to people who bring us bad news.

I don't know if it's a product of nature or nurture or both, but in far too many folks that li'l blister seems to be totally disconnected from the part of the brain that processes cause and effect. Maybe the axons got re-routed (or just withered away altogether) from too much "Wheel of Fortune" and "Family Feud" as a child, I don't know. What I do know is that if I walk into most institutions and find a serious flaw in their network security, the overwhelming tendency of management will be to blame me for the flaw itself—as though by observing it I somehow brought it into existence. While this might be a workable assertion in the quantum world, at the macro scale of Newtonian physics it is a load of festering armadillo kidneys.

This curious tendency to abandon basic principles of logic repeats itself on a regular basis throughout the connected world. I could write it off as a relatively harmless quirk of human psychology if the perpetrators of this logical inconsistency didn't often carry it to the next (also wholly illogical) step and prosecute the discoverer, whom they should rightfully be thanking. I've heard all the rationalizations for this action, including overstepping authority, releasing information so that the "bad guys" can make use of it, and a dozen others. I call horse hockey on all of them. Allow me to put this in simplified perspective for you.

I (hypothetically) live in a society where people have all of their private information encoded in little colored marbles. Whenever I open a bank account, register for a college class, have a medical procedure done—in short, engage in any of the usual activities that require positive identification and secure storage of personal data—I put a certified copy of this marble in a little box on the appropriate desk.

One day I'm in a college registrar's office, signing up for a continuing education class on Marble Assurance, when I notice that the registrar has left her desk to go to the bathroom. Curious about the security of my marble, which I have just witnessed her place in the box, I reach over and discover that although the box is properly locked, the lock is faulty and can easily be defeated by sliding the lid to one side. Within I see several dozen marbles, any of which I could easily abscond with and use to steal the

identity of the owner. I take only my own marble—my sovereign property—from the box and re-present it to the registrar upon her return, just to illustrate the security flaw. She notifies her chain of command about the breach.

That evening I warn some friends of mine on the social media site "MyFace" that they should be careful when entrusting their marbles to any institution that uses that particular model of box because the lock is easily circumvented. The next morning I am summarily arrested and charged under the tortuous Marble Fraud and Abuse Act in connection with the incident. I am forbidden to discuss the flaw and the MyFace post mysteriously disappears. I serve two years in prison and a further five years on probation for my heinous act.

Later in my now ruined life I run across one of those same Secure Marble Storage boxes for sale at an office supply store and purchase it with my salary from the corner convenience store, the only place that will hire a convicted felon. It is the latest and greatest version, brimming with new features and accompanied by a great deal of marketing hype, including glowing tributes from established Marble Security experts. I lock it securely and slide the lid to one side. The box falls open in my lap.

Moral: You can lead a crippled horse to slaughter, but that won't fix the hole he stumbled in.

SAVE THE DATE!

JUNE 26-27, 2013 • SAN JOSE, CA

WiAC '13

2nd USENIX Women in Advanced Computing Summit

The 2nd USENIX Women in Advanced Computing Summit brings technical women together to discuss the challenges women face in the professional computing world. Beyond mere discussion, attendees are encouraged to share ideas, best practices, and knowledge to move women forward in their professional capacity as technical people.

WiAC '13 is part of 2013 USENIX Federated Conferences Week.

www.usenix.org/conference/wiac13



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 reports of sessions at USENIX conferences.

Access to *login* online from October 1997 to the current month:
www.usenix.org/publications/login/

Discounts on registration fees for all USENIX conferences.

Special discounts on a variety of products, books, software, and periodicals:
www.usenix.org/member-services/discounts

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-services or contact office@usenix.org.
Phone: 510-528-8649



Do you know about the USENIX Open Access Policy?

USENIX is the first computing association to offer free and open access to all of our conferences proceedings and videos. We stand by our mission to foster excellence and innovation while supporting research with a practical bias. Your membership fees play a major role in making this endeavor successful.



Please help us support open access.
Renew your USENIX membership
and ask your colleagues to join or renew today!

www.usenix.org/membership

Book Reviews

ELIZABETH ZWICKY, WITH MARK LAMOURINE AND RIK FARROW

Naked Statistics: Stripping the Dread from the Data

Charles Wheelan

W. W. Norton and Company, 2013. 260 pp.

ISBN 978-0-393-07195-5

Reviewed by Elizabeth Zwicky

I am a little bit obsessed with introductory statistics books, in my attempt to convince the rest of the world that the most minimal grasp of mathematics will in fact combine with a tiny amount of statistics to produce all sorts of impressive-looking abilities. This includes not only the ever-popular ability to figure out how the news is lying to you, but also all kinds of everyday magic in trying to design, build, debug, or maintain computer systems.

The author of *Naked Statistics* shares this opinion, and so has written a book which aims at explaining some reasonably complex and subtle statistical concepts without requiring you to do any noticeable amount of math. It does include the math, should you want to follow along, but it hides it at the end of the chapters for easier skipping by people who are not enthusiastic about numbers.

If you have been resisting statistics because it seems to be made up of complex and incomprehensible mathematical formulae with the occasional curve, this book may change your mind. You will probably also enjoy it if you vaguely remember a statistics course in college but aren't quite sure how it relates to things you might care about. It's not directly related to computers, and if you find yourself needing to calculate your own statistics, you might end up wanting a more number-heavy introduction. (Or not. It's surprising how often the numbers are less important than the ability to say, for instance, "Shouldn't we be comparing these to some population?" or "Are you sure this correlation is relevant?")

Algorithms in a Nutshell

George T. Heineman, Gary Pollice, Stanley Selkow

O'Reilly, 2009. 335 pp.

ISBN 978-0-596-51624-6

Reviewed by Elizabeth Zwicky

Algorithms in a Nutshell also attempts to bring a difficult and useful domain to non-specialists. It doesn't do as good a job at avoiding looking and sounding like a mathematics text, although it does try. I'm still looking for an unthreatening algorithms book.

On the other hand, take their word for it and mine: at some point you are going to need to write code that does searches, or sorts, or looks through a problem space. Or you are going to need to deal with somebody else's code for these things that is dying horribly. At that point, knowing the best and worst cases for different algorithms is going to be important. This is as practical and friendly an introduction as you're going to get, heavy on the examples.

It's not going to substitute for a good computer science education, but it will remind you of the classes you've forgotten or bail you out when you run up against situations where your code—or somebody else's—mystifyingly fails to obey your expectations. And they include examples of how you evaluate algorithms, too.

Don't skip the ending chapters, which include some useful general principles, several of which are more elegant versions of my favorites; don't write your own implementation of any basic algorithm if you can help it, and when faced with a problem best solved with graph theory, immediately turn it into a simpler problem that no longer involves graphs.

The Art of Readable Code

Dustin Boswell and Trevor Foucher

O'Reilly, 2011. 184 pp.

ISBN 978-0-596-80229-5

Reviewed by Elizabeth Zwicky

I am imagining my college roommate at the moment, a guy who was a professional programmer and scornful of the idea of a computer science degree (it was a rental, not a dorm). He eventually learned, at the cost of a weekend of misery, that we were not kidding about sort algorithms mattering, but he would have rolled on the floor in hysterics at the idea of reading a book that has two entire chapters on naming variables and two more on how to write comments. That statistics stuff, it makes some kind of sense, but really, nobody with any sense worries about petty things like how to name variables.

We were young then, and stupid, and many years have come and gone. In that time, I've learned a lot about readable and unreadable code, and have many of my own rules of thumb (including one the authors missed: "Everybody is still basically a small child. Avoid variable names that are suggestive or naughty-sounding."). Nonetheless, I found myself saying, "Ah, yes, that makes sense—I never thought about that" from time to time, and "Oh, that explains some intuitions I never quite figured out" quite frequently.

Take my word for it; it matters how you name your variables. You can find this out from a book, or you can spend a lot of extra time debugging. The book is cheaper. Don't worry; it's not like That Guy who insists there is one true way to name variables and one true way to indent things. It's sensible, flexible advice, with examples to show you why you care.

On the other hand, if you were hoping somebody would just tell you the answer and then you could crank out readable code, this book will disappoint you, and it has to. Readability, in anything, is difficult and requires careful thought about readers. This is good advice that will help you do that thinking, but you're still going to have to work.

Living with Complexity

Donald A. Norman

MIT Press, 2011. 265 pp.

ISBN 978-0-262-01486-1

Reviewed by Elizabeth Zwicky

Living with Complexity is a book about our attitudes toward complexity and technology, which are much maligned and yet central to our day-to-day experiences.

Norman argues that our immediate opinions are wrong. He offers some strategies for both individuals and designers to deal with complexity, and he suggests new ways of thinking about issues around complexity.

There are several Donald Norman books that I love very much. This one I think is nice enough, but sadly, not earthshaking. It makes several good points (complexity is not inherently bad; social factors drive much complexity; services are often overlooked; and how is it that we can still be this bad at power strips and projectors?).

Ultimately, though, it fails to come together into a coherent whole. I prefer *The Design of Everyday Things*, but if you can't lay hands on it and haven't encountered Donald Norman before, *Living with Complexity* would certainly be a worthwhile read. It might seem more captivating to me if I hadn't been familiar with his earlier work.

MapReduce Design Patterns

Donald Miner and Adam Shook

O'Reilly, 2012. 227 pp.

ISBN 978-1-449-32717-0

Reviewed by Elizabeth Zwicky

MapReduce is the underlying technology for most companies dealing with big data. If you want to throw around serious amounts of data—you don't just want to use the cloud, you want to be the cloud—you are going to end up using MapReduce. In

many situations, you're going to use it with some sort of insulation, using a language like Pig designed to hide what's going on underneath. But there's a limit to how far that will take you, and programming for MapReduce requires some twists in how you may be used to working.

MapReduce Design Patterns is a good place to start if you need to work directly in MapReduce for some reason. If MapReduce is further below you, you may not be interested until you start pushing the envelope, at which point you need to think more carefully about what's going on. And, quite possibly, to drop into straight MapReduce.

I use Pig a lot, and pure MapReduce occasionally, so I was familiar with the basic patterns, but I still found some new and interesting ideas.

Getting Started with Raspberry Pi

Matt Richardson and Shawn Wallace

O'Reilly Media, 2013. 176 pp.

ISBN 978-1-449-34421-4

Reviewed by Mark Lamourine

When I describe a Raspberry Pi to most people, the first question I usually get is, "But what does it do?" *Getting Started with Raspberry Pi*, O'Reilly's introduction to learning computing with the Raspberry Pi, is one attempt to answer that question.

The Raspberry Pi has become the hot small hobbyist computer in the last year. It's meant to be a tool for people who want to learn and experiment with programming, network services, and robotics. The previous leader in this space was the Arduino, and people have asked me why the world needs another hobbyist computer, but there's really no comparison. The Pi is a fully outfitted general purpose computer, while the Arduino is a programmable microcontroller. This means that the Pi is suited to different kinds of projects than the Arduino. In fact, there are a number of projects that use the Pi to program the Arduino. The two are really complimentary.

The O'Reilly "Make" books are aimed at people who mean to get their hands dirty. While they are often directed at beginners, they don't hand-hold or subject readers to long lectures on fundamentals. Instead they focus on small projects with achievable goals and then leave the reader with tips for further reading.

The first chapter walks the reader through setting up their Pi. The following chapters gradually introduce Linux, Python, and then move on to hardware projects using the Raspberry Pi GPIO pins directly, or adding an Arduino. Each chapter finishes with a "Going Further" section that includes references (and in the eBook I read, links) to additional resources on the topic.

The format of the book follows O'Reilly's "Make" imprint style. This is the signature red and blue logo, big simple bold text on a crisp white background, and hand-drawn graphics. It's easy to read and very comfortable and welcoming.

This book is very well suited to the adventurous beginner. The chapters are clear and complete. It is a good idea to keep a Web browser and search engine handy. The range of topics means that there might even be something new for an experienced server administrator who might not have had a chance to play with sensors or robotics.

The low cost of the Raspberry Pi has meant that they are being purchased by (and being given as gifts to) people who have only ever had the slightest exposure to computers outside the canned Windows or Mac OS experience. Will it be the Mountains of Robotics, the Seaside of Media, the Caves of Programming, or something else entirely? *Getting Started with Raspberry Pi* is the signpost at the crossroads.

Practical Vim: Edit Text at the Speed of Thought

Drew Neil

The Pragmatic Bookshelf, 2012. 311 pp.

ISBN 978-1-93435-698-2

Reviewed by Rik Farrow

Elsewhere in the issue, Dave Josephsen refers to the argument of whether to use vi or Emacs as a religious one. For myself, choosing to use vi was more a pragmatic decision: vi was found on all of the many systems I was using at the time. I learned vi, and continued to learn new tricks, until I thought I had mastered vi.

That was 25 years ago. Over the last five years, I started noticing differences in how vi worked—for example, if I happened to pass a directory instead of a file to open. I thought the new behavior was better, displaying the directory's contents and then getting to choose the file I wanted to edit. Then I noticed other things, such as previously unmapped keys were now mapped, and *strange* things, like the screen was split.

What had happened was that vi, written by Bill Joy in the early '80s, had been replaced by vim. And when I noticed this book, I decided it was time to learn about the new tool.

I got away without knowing about vim for years because it works pretty much like vi—it's just that vim does a whole lot more. *Practical Vim* is organized as a series of "Tips" within each chapter, but starts out with several chapters of very basic vim. The author encourages more advanced users (ahem) to skip around, which I immediately started doing. I learned that vim has a special register that allows me to solve simple equations and insert the results inline. I found out how to *intentionally* split the screen, and what visual block is all about.

The book works well: the instructions are clear, examples easy to follow, and everything I tried worked. My only complaint is about vim, not the book. Now I have another set of keystrokes to memorize so I can learn all the new features. *Practical Vim* does not teach you about scripting, another facet of vim, although it does use scripting in several examples. There is much to learn.

If you have been a vi user, and have noticed something different, I encourage you to learn about vim. Or buy this book, as it can make the experience less painful and more fun.

Thanks to our USENIX and LISA Corporate Supporters

USENIX Patrons

Google
Infosys
Microsoft Research
NetApp
VMware

USENIX Partners

Meraki

USENIX Benefactors

EMC
Hewlett-Packard
Linux Journal
Linux Pro Magazine
Oracle

USENIX and LISA Partners

Cambridge Computer
Google

Conference Reports

REPORTS

In this issue:

48 LISA '12: 26th Large Installation System Administration Conference

Summarized by Aileen Alba, Nick Anderson, Yakira Dixon, Daniel-Elia Feist-Alexandrov, Rik Farrow, Nick Felt, Dybra Grande, Jessica Hilt, Andrew Hume, David Klann, Cory Lueninghoener, Tim Nelson, Mario Obeja, Barry Peddycord III, Josh Simon, Lin Sun, Aleksey Tsalolikhin, and Steve VanDevender

71 LISA '12: Advanced Topics Workshop

Summarized by Josh Simon

75 LISA '12: Real World Configuration Management Workshop

Summarized by Nick Anderson and Aleksey Tsalolikhin

LISA '12: 26th Large Installation System Administration Conference

San Diego, CA

December 9-14, 2012

Opening Remarks and Awards

Summarized by Rik Farrow (rik@usenix.org)

Carolyn Rowland, chair of LISA 2012 and appearing as energetic as ever, began the conference by saying that more than 1000 people attended LISA. Twenty-two papers were accepted for the papers track, with awards going to the Practice and Experience paper Lessons Learned When Building a Greenfield High Performance Computing Ecosystem by Andrew Keen et al., the best student paper going to Theia: Visual Signatures for Problem Diagnosis in Large Hadoop Clusters by Elmer Garduno et al., and best paper to Preventing the Revealing of Online Passwords to Inappropriate Websites with LoginInspector by Chuan Yue.

Next, John Mashey, appearing in a video, accepted the Lifetime Achievement award. Mashey is known for his work on the Programmer's Workbench (PWB) in the late '70s, contributing to the SPEC benchmark, the design of the MIPS RISC processor, and Silicon Graphics supercomputers. Arthur David Olson received the Software Tools Group award for his work on the Timezone DB. The LISA Outstanding Achievement award went to the developers of PowerShell: Jeffrey Snover, Bruce Payette, and James Truher. Finally, Phil Kizer, President of LOPSA, presented the Chucks Yerkes Award to David Lang for his work on the Linux kernel, rsyslog, and other projects.

Keynote Address: The Internet of Things and Sensors and Actuators!

Vint Cerf, VP and Chief Internet Evangelist, Google

Vint Cerf began by saying that as an Internet evangelist, he still has much work to do: the Internet has not yet reached everyone. Using domain names as a metric, there are 908.5 million machines visible on the Net, and 2.405 billion users. Only 1.5 billion of these are PC users, with much of the rest being users of mobile phone and devices.

IPv6 support got a lot better after the flag day: June 6, 2012. Today, about 25% of sites are visible via IPv6. With IPv4 addresses almost completely exhausted, IPv6 adoption must grow beyond the use of network address translation devices, which are fragile and don't do the job when cascaded.

ICANN (Cerf had been on the ICANN board for years) spent a lot of time and energy on getting support for Unicode for internationalized names. More recently, ICANN has collected more than \$350 million in fees for non-generic top level domains, something Cerf said he is still skeptical about. (After all, how many people do much typing, especially of long domain names?) Cerf pointed out that DNS still has vulnerabilities and weaknesses, and that DNSSEC with its digital signatures will help. Cerf also mentioned using Digitally-Signed Address Registration (RPKI) to protect Internet routing from a serious vulnerability in BGP4, which has been around for decades.

Cerf commented that back in the early days (1980s), people joked about Internet-connected toasters. Today, we have Internet-connected picture frames and even light bulbs with IPv6 addresses. Cerf described how he monitors his wine cellar for temperature using a device that sends him text messages if the temperature goes over 60. He once received a message every five minutes for five days while he was traveling. Cerf also pointed out that his next steps would be adding RFID to each bottle, so that removed bottles get noted (he has teenagers!), and later planned to add sensors to the corks to monitor changes in wine caused by loss of temperature control.

As people and businesses add more sensors, Cerf told us that we need to be considering issues of authentication, authorization, security, along with ease-of-use. If you consider large environments, like a factory, it's not trivial to configure and manage a large network of devices (he mentioned Arch Rock's mesh networks). And what about devices in the home? If you allow auto-registration, what's to stop your neighbors from registering your devices? Who will you allow to monitor your devices, perhaps to add (not subtract from) your security? If these devices are wireless, which is much simpler, each needs its own address.

Grouping devices by a controller (Arch Rock) seems like a good model, similar to the way we use ASNs today. Cerf included set-top boxes as other devices also in need of configuration.

Cerf is also a member of the Smart Grid Interoperability Panel (SGIP). While many US citizens consider having smart meters that can both monitor electric usage and (eventually) disable high current devices distasteful, Cerf pointed out that we use peak power only 2% of the time, but we pay to build out our generating capacity to support this tiny fraction of usage at great cost.

Cerf discussed the recent attempts to change how the Internet is governed. Certain countries had attempted to use the International Telecommunication Union as a forum to wrest control from nation-independent entities, such as the IETF, and to a lesser extent ICANN, so they can create new standards. Not that these standards will be “real,” as their real purpose is control, and the level of control desired already exists. You don’t need a standard for doing deep packet inspection other than the existing standards that allow the Internet to work. Other policy challenges that exist today include the meaning of digital certificates, intellectual property, and preservation of data and software (Digital Vellum).

Cerf brought up other challenges to the Internet, some having to do with the future of routing (OpenFlow and BGP), rethinking the use of certificates and authorities, the role of trusted computing (TCM and the requirement to sign operating systems digitally), and inter-cloud protocols. He finished on a high note, by discussing the InterPlaNetary Internet (his capitalization). Because of the huge amounts of data acquired by remote sensors, such as Mars rovers or Cassini, and the large amount of time it takes for light to travel from the outer planets to Earth, new techniques are required. One thing that has worked so far is to store-and-forward messages, repurposing existing satellites—for example, in Mars orbits—to collect messages from surface-bound rovers, then send them using the more powerful radios. I found myself thinking, “What a great way to take advantage of bufferbloat!” and the reality is not that far off. The delays inherent in interplanetary TCP/IP really require different protocols, such as Custodial File Delivery Protocol.

Cerf only had time for a single question at the end, partially because he was urged to keep on speaking. When Patrick Cable came up to the mike, Cerf walked off the stage so he could watch Patrick ask his question, as Cerf said he has trouble hearing. Patrick asked Cerf about his thoughts on regulation in general, and are there regulations that make sense. Cerf responded that there are areas where international regulations do make sense. We can’t do much about spam or Internet-based crime without the support of international law. We need international cooperation for many things. Then there are times when informal

cooperation works best, like the organizations that worked together to track the Conficker botnet.

After his enthusiastic stump speech, Vint Cerf received a standing ovation from his equally enthusiastic audience. You can view the video or download the audio of this and the other presentations at www.usenix.org/conference/lisa12/tech-schedule/technical-sessions.

Papers and Reports: Storage and Data

Summarized by Lin Sun (sunlin530@gmail.com)

HSS: A Simple File Storage System for Web Applications

Daniel Pollack, AOL Inc.

Daniel Pollack explained that all Web applications need some sort of durable storage system to hold the content and, in some cases, the code that runs the Web application. At AOL, they looked at a variety of existing solutions, including cluster file systems, scalable NAS, and parallel file systems before deciding to build their own solutions. Their first attempt was iBrix, but it had both performance issues and required client-side support. Their second attempt was to build an object store using commodity hardware and open source software. Based on these experiences, they came up with a list of requirements, including scalable metadata, separate metadata, and data system components, both multi-site and multi-tenant capable.

The storage system presented seeks to improve on the availability and operational characteristics of the storage systems. A minimal set of operations are provided and they rely on external components for any additional functionality that an application may need. Additionally, several mechanisms are built into the system that provide data durability and recovery—for example, being aware of the physical makeup of the system for both reliability and hotspot reduction.

HSS uses MySQL for metadata storage, and stores content as objects. Each object is replicated, and the location of objects is updated in the MySQL database. A simple RESTful external API is presented to clients, and HSS fulfills requests.

A list of future planned improvements could be container files to address file management and performance concerns, lazy deletes to disconnect housekeeping operations from online operations, improved geographic awareness to improve access latency, and a policy engine to manage file placement and priority in the system.

IDO: Intelligent Data Outsourcing with Improved RAID Reconstruction Performance in Large-Scale Data Centers

Suzhen Wu, Xiamen University and University of Nebraska-Lincoln; Hong Jiang and Bo Mao, University of Nebraska-Lincoln

Bo Mao began by saying that there is much more disk failure in the real world than we used to imagine. Generally speaking, the complete disk failure rate is 2% to 4% on average, and after one disk fails, another disk failure will likely occur soon.

Due to these challenges, RAID reconstruction tends to be much more important to system reliability. There are two challenges for RAID reconstruction: real-time user performance and window of vulnerability. Diverting many user I/O requests from the degraded RAID directly affects the reconstruction performance.

The existing reconstruction approaches can be categorized into two types. The first type of reconstruction optimization improves the reconstruction performance by optimizing the reconstruction workflow, such as DOR, live-block recovery, and PRO. The second type improves reconstruction performance by reshaping the user I/O requests, such as MICRO, Work Out, and VDF. Based on new observations, they found that these optimizations are ineffective.

IDO (Intelligent Data Outsourcing), a proactive and zone-based optimization, can address this problem and significantly improve online RAID-reconstruction performance. The main idea of IDO is to divide the entire RAID storage space into zones and identify the popularity of these zones in the normal operational state, in anticipation of data reconstruction and migration. Upon a disk failure, IDO reconstructs the lost data blocks belonging to the hot zones prior to those belonging to the cold zones and, at the same time, migrates fetched hot data to a surrogate RAID set.

IDO is an ongoing research project. They are working on the recovery algorithms in large-scale storage systems where the network bandwidth, the storage nodes, and the workloads are more complicated than the pure RAID-based storage systems.

Theia: Visual Signatures for Problem Diagnosis in Large Hadoop Clusters

Elmer Garduno, Soila P. Kavulya, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan, Carnegie Mellon University

Awarded Best Student Paper!

Soila Kavulya explained that problem diagnosis when using Hadoop is compounded by the overwhelming volume of monitoring data and complex component interactions that obscure root causes. Usually users want to distinguish between problems inherent in their job and problems due to infrastructure faults. Theia is a tool for visualizing anomalies in Hadoop clusters, targeting hardware failures, software bugs, and data skew. Its key requirements are an interactive interface that supports data exploration in which users drill-down from cluster- to job-level displays, a compact representation for scalability, and the ability to support clusters with thousands of nodes.

Theia's types of visualizations include anomaly heatmaps, job execution streams, and job execution details. The "anomaly heatmap" provides a high-density overview of cluster performance and summarizes job performance across nodes. It uses color variations to visualize anomalies. The "job execution stream" helps to visualize per-job performance across nodes

and a scrollable stream of jobs sorted by start time. It displays performance of Map and Reduce phases and shows job execution traces in context: job name, duration, and status in addition to failed and killed task ratios and task duration anomalies. The "job execution detail" provides a detailed view of task execution but is less compact than the job execution stream. It displays job progress and volume of I/O; it is best-suited for detecting application problems, software bugs, and data skew.

Kavulya concluded that Theia visualizations for Hadoop are compact, interactive visualizations of job behavior. Theia distinguishes hardware failures, software bugs, and data skew in addition to evaluating real incidents in Hadoop clusters. Evaluating the effectiveness of a UI for diagnosis could be a further step taken by users.

Someone asked if they include tools for automatic problem classification. Kavulya said they are planning on adding those features. Someone else wondered how well they expect this to scale. Kavulya replied that they use Perl and batch processes, so this should scale. Marc Chiarini asked about the graphic display that uses sizes to indicate disparities across multiple jobs. Kavulya said that currently they just use visual clues to pick this out. Chiarini then suggested using a mouse-over script to provide more details on the node.

Invited Talks

OpenStack: Leading the Open Source Cloud Revolution

Vish Ishaya, Nebula, Inc.

Summarized by Andrew Hume (andrew@research.att.com)

Vish Ishaya started with an extended justification for clusters and clouds and the skunkworks-like genesis of OpenStack within NASA in April 2010. Things moved quickly: the first public cloud launched in October 2010 and Rackspace switched to OpenStack in August 2012.

He then installed OpenStack on his Mac laptop and started it running, logging into the console of a newly started VM. For many people, this was an amazing part of Ishaya's presentation.

So what is OpenStack? It is the APIs that let you manipulate Compute, Network, and Storage inside a cluster. OpenStack now has seven core components: compute, object storage, block storage, networking, (machine) image, identity, and dashboard. Vish gave brief overviews of each of these, and then touched on the management issues (550 developers) that led to the creation of the OpenStack Foundation.

He then described some of the projects in incubation, including heat (work on orchestrating groups of servers/VMs) and using bare-metal servers (and not just VMs).

Analysis of an Internet-Wide Stealth Scan from a Botnet

Alberto Dainotti, Cooperative Association for Internet Data Analysis
Summarized by Daniel-Elia Feist-Alexandrov (*d.feistalexandrov@gmail.com*)

Botnets are one of the most potent arrows in a cyber-criminal's quiver: not only are they responsible for large scale DDoS attacks, they can also be used to detect and exploit vulnerable machines on a massive scale. Alberto Dainotti presented the cooperative's analysis of a 12-day scan conducted by the Sality botnet against the SIP-calling infrastructure around the world. The scan Dainotti and his colleagues analyzed is exceptional not only because of its unprecedented size, but also because of its stealthy stratagem, which made it extremely hard to detect, despite covering the entire IPv4 address space.

The main tool Dainotti et al. used to identify the scan was the UC San Diego Network Telescope "darknet," a block of IPv4 addresses that are not assigned to actual hosts. Using a lot of "investigative" analysis and the fact that, by definition, every packet that arrives at an address in this block is unsolicited, they identified a unique payload fingerprint and the UDP port 5060 as a common denominator of the scan. They determined that the 3 million IP addresses they registered were indeed unique machines by correlating their own data with that of the DShield project (an aggregator of dark and honeynet data) and data from a trans-Pacific link monitored by the MAWI/WIDE project. Another helpful fact was that all the bots that were geolocated to Egypt dropped out of the attack while the government suspended Internet connectivity during the Tahrir Square uprising.

Using a Hilbert curve and other visualization techniques to map the IPv4 realm to a two-dimensional space, Dainotti and his colleagues found that the scan's exceptional stealth was due to all bots choosing their next destination by incrementing target addresses in reverse-byte order. This meant that a generic /24 network would typically receive a total of 256 packets over 12 days from 256 different source addresses, thereby making it very hard to spot any connection between scans.

The first question concerned the fact that there was barely any scanning activity for several days during those 12 days. Dainotti confirmed that this was indeed due to a large number of bots halting their activity and speculates that this might have been due to sanitization efforts on behalf of law enforcement or anti-virus companies performing routine botnet breakups. Another participant asked whether there were any similarities between the few bots that contacted an address in the dark net twice. Dainotti answered that there were no similarities. A possible explanation was that this was due to different versions of the bot's binary.

Someone asked whether the authors estimated the cost of leasing such a huge botnet. Dainotti responded that they didn't and hypothesized that this might have been a factor in the scan's

intermittent flagging. The next participant commented on the possibility that the turnover in the botnet might have also been a result of the generally short lifecycle of a bot (due to eventual sanitization). Another participant speculated that the lack of bot activity in China could be caused by the "Great Firewall of China" and the government's repression of VoIP infrastructure.

Papers and Reports: Security and Systems Management

Summarized by Tim Nelson (*tn@cs.wpi.edu*)

Lessons in iOS Device Configuration Management

Tim Bell, Trinity College, University of Melbourne

Tim Bell presented iOS Configurator, a Django Web application used by students in Trinity College's foundational studies program. Foundational Studies is a one-year college-preparatory program, and each student receives an iPad. iOS Configurator allows those students to download fresh configurations for their iPads. Their original approach to configuration used a combination of manual edits and Python scripts, but that didn't scale very well. They needed the tool to be automatic and scale to several hundred students while allowing them to reconfigure their iPads at any time. Also, they needed to implement the replacement quickly with limited staff. iOS Configurator was the replacement.

Bell showed a screenshot of the login process. After a student logs in, the configurator authenticates her, then gets her group information and fetches a standard configuration file for that group. The configurator adds user-specific information, then downloads the profile. The app provides an administrator page that says who has downloaded their configuration and when.

iOS Configurator comprises 167 lines of Python (including comments) and 229 lines of settings (mostly boilerplate), and took a week to develop. The login process uses HTTPS with a commercial SSL certificate.

After completing the one-year program, students get to keep their iPad. Thus, Trinity did not want to restrict the students' use of their iPads. Because of this fact, Bell opted not to use mobile device management for this configuration process. Bell commented that Apple has since come out with Profile Manager in OS X Lion, which he might have used had it been available when he was creating the configurator app.

Paul Anderson asked whether students could override the settings in the downloaded configuration. Bell answered yes, and explained that that was part of their goal. Also, students can always re-download the configuration.

A Declarative Approach to Automated Configuration

John A. Hewson and Paul Anderson, University of Edinburgh; Andrew D. Gordon, Microsoft Research and University

John Hewson presented the ConfSolve tool. ConfSolve converts configuration goals into concrete configurations. Existing

declarative configuration-management tools let you specify what you want, rather than how to accomplish it. ConfSolve builds off of the CM tools by inferring valid configurations from goals that are only partially specified.

ConfSolve uses a CSP (constraint satisfaction problem) engine as its workhorse. The tool has its own object-oriented language that it compiles into a CSP, and the solution that the solver provides is then translated into a concrete configuration.

Hewson showed an example of ConfSolve working on a virtual-machine specification, which showcased the tool's ability to handle constraints (e.g., "When assigning VMs to physical machines, don't exceed the physical machines' resources") and optimization ("Use our data center as much as possible before using the cloud").

ConfSolve scales fairly well; it produced a configuration for a thousand virtual machines in around 200 seconds, and there is still room for improvement.

John then commented that ConfSolve is not a replacement for mainstream declarative configuration-management tools. Instead, he would like to see those tools incorporate configuration inference. Tim Bell asked about the complexity of the problem, and Hewson replied that the general problem is NP-Hard. Tim Nelson speculated that some kinds of configuration inference may fall into a less difficult class.

Preventing the Revealing of Online Passwords to Inappropriate Websites with LoginInspector

Chuan Yue, University of Colorado at Colorado Springs

Awarded Best Paper!

Chuan Yue presented the LoginInspector tool. Passwords are still the dominant method of authentication on the Internet, yet they are vulnerable to phishing, reuse, and more. Detection of phishing sites relies on either a blacklist or heuristics, and can thus miss zero-day sites. Moreover, users who have forgotten their password can expose other passwords accidentally by trying their "usual passwords" in sequence. Yue's user-study information showed that users really do engage in this risky behavior.

LoginInspector keeps a database of which passwords have been used when logging in to which sites. (For security, it keeps only hashes in the database, not full passwords.) If a user tries to log in somewhere that he has logged in before, but with a different password, LoginInspector intercepts the login and shows a warning message to the user. If the user has not logged into the site before (i.e., it is a potential phishing site) a similar warning message is shown. The tool is implemented in JavaScript as a Firefox addon, using the SQLite database.

Yue showed that LoginInspector has low overhead, taking the longest when inserting new records into its database. He evaluated it on 30 real sites, 30 phishing sites, and one new phishing

site. LoginInspector correctly gave warning messages on all phishing sites, where Firefox's phishing detection failed to catch seven. Chrome's failed to catch eight.

Yue commented that the effectiveness of the tool depends on users' ability to understand and heed the warnings; he intends to perform user studies to evaluate that next. Someone asked, if the site has multiple domains, will that result in multiple records in the database? Yue answered that it would. Mario Obejas asked when the tool would be available. Yue replied that it should be available in January 2013.

Invited Talks

Database Server Safety Nets: Options for Predictive Server Analytics

Joe Conway, credativ USA; Jeff Hamann, Forest Informatics, Inc.

Summarized by Cory Lueninghoener (cluening@gmail.com)

Joe Conway and Jeff Hamann started their session with a simple statement of their goal: to perform Postgres server monitoring using predictive analytics; however, they also noted that this project is really a wrapper around the underlying topic of using Postgres and R to do analysis of big data. With the stage set, they dove in to a technical description of how they are predicting congestion events on their database servers.

Joe began with a description of the tools they are using to perform their analysis: Postgres, a modern database server; R, a popular analytics engine; and PL/R, a module that runs R procedures inside of a Postgres process. He then listed the wide range of Postgres metrics they collect to perform their analysis: active and total Postgres sessions, blocks fetched and blocks hit, cache hit fraction, lock waits, free and cached memory, free swap space, I/O wait and CPU idle, blocks read and written per second, number of blocks read and written, and capture time.

After describing the tools and metrics they are interested in, Joe began a technical description of how the metrics are collected. He included several slides of PostgreSQL and R code examples that showed how the data is collected from the Postgres process, how it is automatically inserted into the database, and how R is used in this process. Joe also noted that the metrics gathering is triggered by a simple cron job, a decision they made to make the process simple, reliable, and transparent.

Following the technical dive, Joe described their method for testing their analysis process. This involved using pgbench, a tool that comes with the Postgres distribution, to simulate steady-state load and transient events on their servers. With the metrics collection pieces in place and a method of simulating events ready, the team was ready to start doing predictive analytics.

Jeff took over at this point to describe their methods. He started by stating the problem: can we do preemptive analytics work to

sense when a server is going to experience congestion? By looking for causal factors, correlations, and leading indicators of system congestion, Jeff hoped that they could do just that.

After introducing their methodology with an example matrix plot and a series of plots showing correlation and time series data, Jeff showed a real example of their work using two basic metrics: swap and the number of active and total Postgres client sessions. He started by showing several graphs of this data, and then described how they built an initial model for this data using R. After comparing this initial model with the real data, he then described how they improved the model to get a better fit.

Once the model was complete, Jeff showed how it could be used to make predictions using principal component analysis and K-means clustering. This included a description of the built-in R functions that make this easy and several graphs that demonstrated its use.

Finally, Jeff gave a brief description of statistical process control and how it relates to predictive server analytics. He described R's statistical process control package, *qcc*, and how it can be used to glean more information from collected server metrics.

After describing their future work plans involving harvesting more data from the Postgres server, doing pattern recognition, and polling multiple servers, Joe and Jeff took questions. One attendee asked whether they were familiar with Baron Schwartz's work to collect data from a failure automatically. The speakers were not familiar with it, but thought it sounded interesting. Another attendee asked whether the source code for their project was available. Joe replied that it was not yet posted, but it would appear on joeconway.com after the conference. [Editor's note: Both code and slides were present on January 8, 2013]

Ceph: Managing a Distributed Storage System at Scale

Sage Weil, Inktank

Summarized by David Klann (dklann@linux.com)

Sage Weil wrote the Ceph distributed storage system and described it in this invited talk. Sage presented an articulate overview of Ceph and answered questions as if he wrote the software (see previous sentence).

Weil began his talk with a very brief historical roundup of storage systems he called “the old reality”: directly attached storage, raw disks, RAID, network storage, and SAN. He quickly moved on to discuss new user demands including “cloud” storage and “big data” requirements. Requirements that include diverse use cases such as object storage, block device access, shared file systems, and structured data requirements. Scalability is also on the requirements list, including scale to exabytes on heterogeneous hardware with reliability and fault tolerance, and a “mish mash” of all the above technologies. And with all this comes a

cost. Cost in terms of both time and dollars. Weil proceeded to describe these costs and then to describe Ceph itself.

Ceph is a unified storage system that incorporates object, block, and file storage. On the Ceph architecture slide, Weil showed the distributed object store base he calls RADOS, for Reliable Autonomic Distributed Object Store. Above RADOS live the API libraries and other interfaces to the object store: LIBRADOS (with the expected array of language support); RADOSGW, a REST interface compatible with Amazon's S3 and OpenStack's Swift; RBD (RADOS block device), the distributed block device; and Ceph FS, a POSIX-compliant distributed file system with Linux, a kernel client as well as a user-space file system (with FUSE). Weil emphasized the distributed nature of the Ceph system noting that Ceph scales from a few to tens of thousands of machines and to exabytes of storage. Weil noted that Ceph is also fault tolerant, self-managing, and self-healing. He pointed out that the collection of Ceph tools is an “evolution of the UNIX philosophy” in that each tool (control command and daemon) is designed to perform one task and to do it well.

Weil moved on to describe Ceph cluster deployment and management. He noted that the Ceph developers are working closely with the major Linux distributions to package the tool set for easy deployment. Ceph supports clusters with mixed versions of the code by checking program version numbers in regular inter-node communication. This facilitates rolling upgrades of individual cluster participants. The protocol also includes “feature bits,” which enable integration of bleeding edge cluster nodes for the purpose of testing new functionality.

The Ceph configuration philosophy is to minimize local configuration. Options may be specified in configuration files and on the command line of the various tools.

System Log Analysis Using BigQuery Cloud Computing

Gustavo Franco, Google Inc.

Summarized by Nick Felt (nfelt1@sccs.swarthmore.edu)

Gustavo Franco presented on Google's BigQuery service and how system administrators can apply it to speed up log analysis. This makes it easier to use logs not just for troubleshooting but also to drive product enhancements. Google has used BigQuery internally for a few years, and they've only recently opened it up to external use as an official product, so it's not yet widely known. Gustavo pointed out that traditional log analysis is tiresome because one writes analysis code and has to wait a few hours to get a response (in the case of large systems with lots of logs), which also makes fixing bugs in this code a day-long process. Using BigQuery, this process takes a matter of seconds, even for petabytes of raw data.

In comparison, Gustavo noted that other approaches to log analysis do not scale as well. Just using *grep* alone will not suffice

once the setup involves several servers. Past that point, one can get by for a while sending log data to a MySQL database, but eventually the influx of data becomes so large that writes start to interfere with reads. One might consider the MapReduce distributed computation framework as an alternative, or the Sawzall programming language, which provides a script-like way to write MapReduce code (both developed by Google); however, although MapReduce is very flexible and useful for data analysis in general, it's a heavyweight solution for log analysis. For each MapReduce execution, the master has to spin up many mappers and reducers, each of which read and write to distributed storage, resulting in a significant time delay at the start in order to spin up workers and a lot of worker I/O overall.

BigQuery improves on all of these approaches by using Dremel, an internal Google framework explicitly designed for fast data analysis. Dremel uses a separate system to handle log injection, so this process doesn't interfere with running queries. On the query execution end, the major difference between Dremel and MapReduce is architectural: Dremel trades flexibility for raw power, allowing it to speed past MapReduce for certain kinds of data sets and queries. Dremel maintains a long-lived shared serving tree with always-running nodes that do not need to be spun up and can execute many queries at the same time. Each query starts at one of many top-level Mixer 0 nodes, which then sends requests to several Mixer 1 nodes, each of which farms out its portion of the request to many leaf nodes. The leaf nodes access distributed storage containing the data in records split up by column, a trick that speeds up the query. Then the leaf nodes send results back up the tree through Mixer 1 nodes to the Mixer 0 node, which reduces the data into a single result set. All data flow outside the leaf nodes occurs via RPC message passing and does not touch the disk, cutting I/O delays substantially.

At this point, Gustavo showed a live demo of BigQuery using the Web UI to execute a number of example queries on large sample data sets of dummy Web server and system logs. BigQuery uses a SQL-like query syntax intended to look familiar to users, and has a command-line UI and an API in addition to the Web UI. One of his example queries was "SELECT COUNT(*) as rows FROM Weblogs.lisa10," which took only 3.0 seconds to execute. The same query executed on the lisa163M table (which has 163 million rows instead of 10 million) took only 0.4 seconds longer. In another query, Gustavo demonstrated the ability to group Web requests to the top hits, which processed three gigabytes of data in 20.7 seconds. He emphasized that nothing in BigQuery is cached or indexed; the data is freshly scanned for each query. Someone in the audience asked whether BigQuery supports joins, and Gustavo said that it does if you establish the relationship in your logs, but the left side of the join must be smaller. Toward the end of the demo he also mentioned that BigQuery supports regex matching and various other features.

Gustavo wrapped up the presentation by explaining how system administrators could start using BigQuery to analyze their own Web server, application, and system logs. The first step is downloading the "bq" and "gsutil" command-line tools. For ongoing use, Gustavo recommended using logrotate and sharding logs into daily tables to improve performance unless the data set is fairly small. Logs should be uploaded to Google Cloud Storage in either CSV or JSON format, optionally gzipped. There was a question about the lack of support for syslog and other log formats; Gustavo said it's a work-in-progress. Once the data is uploaded, run "bq load" with a few arguments specifying what data to use and providing information about the columns, then you're ready to query away. You can even use the Google Visualization API to generate plots of results. Pricing information for BigQuery and Google Cloud Storage is available online, and both have free tiers as of December 2012. For BigQuery, the first 100 GB of data processed per month is free, and the cost per query is based just on how much data the query touches, not on the overall data size. Gustavo directed those who want to learn more about putting BigQuery to use to consult his "homework" page (<http://goo.gl/JkhFC>).

During Q&A, someone asked for elaboration about how Dremel scales, and what is processed by the leaves versus by the mixers. Gustavo responded that the leaves are only ones touching the shared storage; they do most of the data crunching and send results back to the mixers via RPCs, with different kinds of aggregation happening at different levels. Another attendee asked for the most interesting thing Gustavo had heard of someone doing with BigQuery. Gustavo said the Ads group at Google makes heavy internal use of BigQuery, but wasn't able to elaborate beyond "for cool stuff." Nick Felt (Swarthmore College) asked whether BigQuery can return an estimated time until completion for queries. Gustavo replied that it wasn't possible to propagate this information back up the tree, but queries are usually pretty fast to complete, although with large data sets or especially complex queries they can take longer than a minute. Someone else asked what to do to upload logs from an app with an idiosyncratic log format. Gustavo replied that any format is fine as long as it can be converted to a columnar structure, and the columns are given names and data types when loaded into BigQuery. Someone asked for a comparison with Splunk, a competing tool, but Gustavo declined to comment since he hasn't used it.

Plenary Session *Education vs. Training*

Selena Deckelmann, PostgreSQL

Summarized by Jessica Hilt (Jhilt@ucsd.edu)

This talk concerned the controversial issue of formal education verses on-the-job training. Sysadmins might be divided about the topic to the extent that they talk about it at all.

Selena Deckelmann began with the continually pressing problem of scalability: we can't hire the numbers we need in the sysadmin field and we can't train people fast enough. Looking to the formal university setting, we see the ability to train larger numbers but we don't see the classes designed for sysadmins. On the job, we see the necessary skills being taught but only on a one-on-one basis. Certification programs tend to be scoffed at, and books and blogs tend to be the resources to which sysadmins turn but lack effectiveness in training large numbers.

Deckelmann explored the reason why sysadmins dismiss the university setting. Citing bad teachers, ineffective classes, and abstract theories, Deckelmann agreed that there is a lot not to like; however, Deckelmann cautioned against this mentality. Instead of labeling formal training settings as snobby or impractical, she suggested we figure out how to share with teachers and universities what we need from them in order to make a great sysadmin and to bridge the gap between the training world and the education world.

In this vein, Deckelmann outlined steps to make a systematic training program that is effective. She started with a method she learned in a one-on-one setting early in her career. Use (1) defined steps with measurable outcomes; (2) explicit instructions with immediate feedback loops; and (3) pairing and modeling.

In outlining training in such a way, Deckelmann says, we are defining what success looks like to a student.

Next, Deckelmann sought to apply this method to a larger number of students taught at once, using a case study of teaching Python to non-programmers. She established a baseline for training by defining what the student comes into the class expected to know, teaching the gaps, and then having the students demonstrate the knowledge to the rest of the students. With this method, Deckelmann explained, the class proficiency rises.

Deckelmann made one strong recommendation throughout the presentation: In order to find solutions to the education debate, you need to start a fight with sysadmins about it today. She recommends fighting about the details of education (i.e., ethics versus risk reduction, nonprofit certification versus masters programs) in order to have an argument of value. Additionally, she promoted sharing existing training material and programs so that others can learn from your success, and stressed that we can't wait for people outside the industry to solve this problem.

This talk generated numerous questions as well as comments about current training programs or certification training programs. A questioner asked whether there was a current degree program that was respected for system administration. Deckelmann pointed to the Rochester Institute of Technology as

a model. Another questioner asked whether requiring formal training would decrease diversity in the field and Deckelmann referred to the Py Ladies program as an example of increased diversity due to formal training. A questioner asked if there was a group that was discussing this for further conversation and they were referred to Carolyn Rowland who was creating a list.

Papers and Reports: Community and Teaching

Summarized by Barry Peddycord III (bwpeddyc@ncsu.edu)

A Sustainable Model for ICT Capacity Building in Developing Countries

Rudy Gevaert, Ghent University, Belgium

Rudy Gevaert discussed the efforts of his institution to improve the state of IT in the universities of developing countries. In the spirit of the other talks of the Community and Teaching track, the talk was not a technical talk, focusing instead on the human side of technology and computing.

Gevaert's university has taken part in an initiative to travel to universities in developing countries and improve their IT capabilities. Unlike many such initiatives that focus on delivering computing equipment to these universities, this initiative takes the efforts a step further by taking an active, hands-on role in training and mentoring the sysadmins of these developing institutions by teaching them how to utilize and troubleshoot the equipment they are given. As Gevaert stresses, the focus of the effort is not to build infrastructure, but to build capacity by focusing on ensuring that they are able to train the participants in these programs to become effective administrators and mentors to their peers.

One of the projects undertaken is to build an in-house email service or Web service. We take for granted the saturation of cloud applications for these purposes such as Google sites and Gmail, but in these nations, this saturation simply is not an option. Gevaert alluded to Vint Cerf's LISA keynote, where Cerf mentioned that the bandwidth of an interplanetary Internet connection might be 500 Kbps at best. In Cuba, one of the nations involved in this initiative, this is not a joke—it's reality. As bandwidth is extremely limited, efforts to conserve that resource are among the top priorities, so in-house solutions are preferred to cloud solutions, and filters to prevent extensive recreational usage of bandwidth must be put in place. These efforts help facilitate these practices and more.

Above all, the one takeaway from the talk was that any outreach effort like this absolutely must be designed with sustainability in mind. Volunteer efforts lose members for one reason or another and outreach initiatives lose government funding, meaning that the volunteers absolutely must not become a dependency. System administrators in developing countries must be trained to become self-sufficient and able to solve problems in their constrained environments. Turning them into mentors is important

so that when the intervention ends, they can train their own colleagues.

Teaching System Administration

Steve VanDevender, University of Oregon

One doesn't have to be faculty to make a difference in teaching at a university. Steve VanDevender took the initiative of leading his own course in introductory system administration and presented his practice and experience report on how he developed the course and how it worked out.

A point that VanDevender stressed was that "you can't teach everything you want to teach." It's essential not to be too ambitious, and to focus on specific and attainable learning goals that can be accomplished in the time frame of the course being taught. That being said, he still took risks, such as allowing students to work in teams, choose their own OS, and design their own final project for the course. Even though these were risky decisions, they ended up being very rewarding as the students really enjoyed being able to work on projects that they thought would be meaningful.

In his class, VanDevender also had the opportunity to do the opposite of what he disliked about classes when he was a student. He therefore focused on clear and well-defined assignments where the grading would be explicitly linked to the outcomes of the course and the objective performance of the projects. Research has shown that when assessment is objective and transparent, students are more likely to respect the instructor and take the course seriously. Despite the class being challenging, students appreciated the multi-modal learning, from reading chapters of the textbook to discussing materials in class to having the hands-on experience of working with their personal system—even when it came time for the surprise "System Failure" day.

One audience member highlighted that in university teaching, instructors don't get the guidance and support that they might expect. VanDevender's institution did not have as much oversight as he first thought they would, meaning that while he had a lot of freedom in how he led his class, he did not get much in the way of feedback and formal training. Many institutions have a Center for Faculty Development that offers workshops and mentorship to help improve teaching, which system administrators may find useful if they decide to attempt to take the initiative and do something similar at their own institutions.

Training and Professional Development in an IT Community

George William Herbert, Taos Mountain, Inc.

Professional development services are a major part of many companies. George Herbert shared the story of how his company treats professional development as a major company value and has offered such services to their consultants and contractors

throughout the years, even in the face of the recent economic downturn.

Professional development services manifest themselves in many different ways. In addition to inviting guest speakers and providing mentoring, companies can offer reimbursements for taking classes or buying books, providing Safari accounts, and subsidizing travel to professional conferences like LISA. By subsidizing such services, companies keep their employees well-rounded and up-to-date on the latest technological developments.

Many companies don't treat their professional development services as the valuable asset that they are. In a field where retaining talent is so important, the attitude toward professional development can be a differentiator. Many employees find the fact that companies offer consistent professional development opportunities to be a reason to spend more time with their company rather than seeking another or going freelance. Furthermore, the professional development services at Taos only cost the company about \$100 per employee each quarter, not nearly as much as one might think. Given the impact they have on the skills and morale of their employees, they are worth the investment.

Herbert has done some initial analysis of the data from attendance sheets, compiling how well certain events have gone over. In general, employees prefer professional development with a social element to them, such as having guest speakers or having special classes on specific topics rather than buying books and resources for self-directed study. As much of the data has yet to be compiled in a way that can be easily analyzed, most of the lessons learned are anecdotal; however, this still leads to his big takeaway: any professional development initiative should be documented and measured so that its value can be clearly represented to the management in order to sustain it over the long term. Herbert looks forward to coming back next year with empirical data to back up his hypotheses about the relationship between professional development, morale, and retention.

Invited Talks

Dude, Where's My Data? Replicating and Migrating Data Across Data Centers and Clouds

Jeff Darcy, Red Hat

Summarized by Daniel-Elia Feist-Alexandrov (d.feistalexandrov@gmail.com)

Jeff Darcy discussed the basic problems faced when distributing and migrating data around the cloud. To start off he cautioned the audience that while there is no silver bullet in data maintenance, there are optimized replication infrastructures for several usage profiles. After giving a high-level overview of consequences that come with large data and varying basic environmental parameters, he gave an introduction to a very basic UNIX

tool for synchronization, rsync, and its strengths and issues. Darcy followed up by exploring different replication topologies and strategies, wrapping up the talk with a discussion of different available distributed file systems.

The basic problem that Darcy observes is that computing cycles in the cloud move quickly, while the data that is needed to perform those isn't necessarily able to follow suit. This is because replication and migration of large data is complicated by its size, rapid turnover, and variety, which becomes especially complicated when dealing with large distances and replication across multiple domains. In such scenarios keeping the data in sync across the network nodes is a formidable challenge.

The first replication tool that Darcy explores is the simple UNIX rsync, which is used in production environments such as the back-end of the Dropbox service. Although rsync performs well with large files, its downfall comes with a sensitivity to geographical distance and synchronization across multiple domains. Its architecture also entails high divergence between node states. But we learn from this simple case that the initial sync of our nodes is the least of our worries since it only occurs once. Darcy concluded this section of the talk with some advice on how to optimize the initial synchronization (such as packing files into larger archives and transferring in parallel).

Darcy then compared different replication types. He first explored synchronous replication, which, while keeping divergence at a minimum, is extremely latency sensitive. He then explored both ordered and unordered asynchronous replication. The former continuously logs the changes and thus only transmits what was changed and lowers divergence. The latter only scans the data periodically for diffs, which results in high divergence and is thus the less preferable of the two. Darcy then explored these two basic premises of replication by logging and replication by scanning in further detail.

In the next section, Darcy went back to rsync and proposed improvements to this simple tool. He concluded that scanning is inherently inefficient and introduced some well-known distributed file systems, such as AFS and Coda. Darcy championed the less well known but powerful XtremFS file system. He concluded that all these file systems handle the challenges that come with large data volumes rather well and must be chosen according to environmental circumstances (such as bandwidth and distance) and what parameters are critical to the user.

Darcy finished the talk with a recapitulation of the lessons learned: Initial synchronization is the smallest worry in data replication, whereas staying in sync is hard. Conflict resolution is a major challenge and is best approached by segregating data by consistency requirements and choosing requirements on what is "just enough" consistency.

There were no questions.

Rolling the D2O: Choosing an Open Source HTTP Proxy Server

Leif Hedstrom, Cisco Systems

Summarized by Dybra Grande (granded@coyote.csusb.edu)

Leif Hedstrom discussed the problems systems administrators face when choosing an open source HTTP proxy server. One of the many issues administrators face is choosing the correct proxy server(s) to use from the overwhelming quantity on the market. Some of the proxy servers available are either commercial or open source, and some products offer caching, proxying, or do both. Hedstrom says, "This is where system administrators get lost in choosing the correct solution that works for them, and end up visiting social media sites to get opinions from 'reliable' sources such as other administrators who work on Netflix, Facebook, Twitter, Google+, and Usenet, where they prescribe solutions to problems which are sometimes irrelevant to implement due to the fact one is running different types of systems and applications than theirs."

Hedstrom included a crash course in his presentation before discussing his research on the different types of intermediaries available. Forward Proxy basically uses the user agent "the browser," which cooperates with the proxy server itself asking it to process a request on its behalf. This can improve performance because it allows you to cache and use this data. Reverse Proxy does the opposite of Forward Proxy because it acts on behalf of the servers. The administrator is responsible for setting the rules, such as when and where to cache. On Intercepting Proxy, the user is oblivious to the system administrator's set up. This is used by businesses and educational institutions who do not want users to visit Web site content that can be dangerous to their network or systems. It can also be used to block users from getting into their Facebook or Google+ accounts from their workplace.

Hedstrom covered several HTTP proxy servers, such as ATS, Nginx, SQUID, and Varnish, emphasizing the benefits and failures of these intermediaries. ATS offers good HTTP/1.1 support and includes SSL. Hedstrom mentioned the benefits of its ability to tune itself to the system and its excellent cache features and performance. The problems with ATS is that the load balancing is incredibly lame and difficult to set up, the developer community is small, and the code is complicated. Also, there are many configuration files, and there is still legacy code that must be replaced or removed. Nginx, on the other hand, has a code base and architecture that is easy to understand. It has an excellent Web and application server that includes commercial support available from its creators. Nginx's problems are that adding extensions implies rebuilding the binary, it does not make good attempts to tune itself to the system, and there is no support for conditional requests or protocols.

Of the bunch, Squid has by far the most HTTP features, and it is the best HTTP conformant proxy today. Squid is widely used

because of its mature features, which work pretty well out of the box. One of the negative issues with Squid is that it is based on old code and the cache is not particularly efficient. It has also been traditionally prone to instability and complex configurations. Varnish uses its own configuration language and has a clever logging mechanism, which supports several commercial entities. The problem with Varnish is that it does not support SSL, and protocol support is weak. In the end, Hedstrom reminded the audience that performance itself is rarely a key differentiator, but latency and feature correctness should be.

An attendee asked about issues with logging in Varnish. Hedstrom replied that Varnish can produce several hundred lines of logging with each request because it logs everything that happens. The attendee thought this could be a vulnerability, and Hedstrom replied that varnishlog can cause latency by hammering on the disk or virtual disk.

Advancing Women in Computing (Panel)

Moderator: Rikki Endsley, USENIX

Panelists: Jennifer Davis, Yahoo, Inc.; Elizabeth Krumbach, Ubuntu, Adele Shakal, Metacloud, Inc.; Nicole Forsgren Velasquez, Utah State University; Josephine Zhao, Prosperb Media and AsianAmerican Voters.org

Summarized by Aileen Alba (aalba@csupomona.edu)

Jennifer Davis, Elizabeth Krumbach, Adele Shakal, Nicole Forsgren Velasquez, and Josephine Zhao all came together to answer questions Rikki Endsley had about women in computing. Some of the topics ranged from mentoring, networking, recruiting, and advice for women and their colleagues. Although it was a panel of women discussing women in computing, many men attended as well to find out more about how women work, think, and even feel. Each one of the women took turns answering questions and discussed their own experience. Rikki first asked, “What makes a good mentor and what skills are good?” Nicole Forsgren Velasquez made a great point when she said we all should have different types of mentors. She went on to explain, “If we only have a cheerleader mentor we miss the holes in our work, and if we only have a skeptic mentor we are always discouraged.”

The panel continued with a discussion on women in the workplace from recruiting to advice for male colleagues. Elizabeth Krumbach pointed out that when creating the requirements for a job you must be realistic. Many women will not apply to a job if they don't have all the requirements, so employers need to make sure they clarify this in their requirements. Jennifer Davis also explained that interviewers should be aware that women communicate differently from men. When women say “we” it doesn't mean that they didn't contribute to the project, it just means that they don't take credit for all of the work. Understanding women in the workspace is hard for some men because they are not accustomed to having women in their companies. One of the best pieces of advice for men during this panel was that they should not comment on a woman's appearance unless they have an

established relationship with her (Jennifer). Some terms people should be aware of when it comes to women are “derailing,” “gas lighting,” and “imposter syndrome.” Adele made it clear that these terms will help men better understand how women feel.

Carolyn Rowland pointed out that women tend to internalize failure and externalize success. She continued by saying that women tend to give credit to everyone else even if we are the ones who lead something, but if we do something wrong we take blame for it alone. We must all be aware that women seldom brag or take credit for the work they do. Another attendee asked, “How should we help women have a more positive view of themselves?” Elizabeth said women sometimes just need a push and some positive advice. Josephine also explained that sometimes it takes a woman to change herself and also advertise for herself. Women need to be less shy about themselves; it might take time for this to happen but the more we all do this the faster we will see the change.

Carat: Collaborative Energy Debugging

Adam Oliner AMP Lab, University of California, Berkeley

Summarized by Tim Nelson (tn@cs.wpi.edu)

Adam Oliner presented Carat (carat.cs.berkeley.edu), an app that helps smartphone users improve their battery life. Carat is different from other such apps because it does not just advise people to use their smartphone less; it gives targeted advice based on statistical information gathered from many users. Carat looks at how much power each app uses, not for specific, pre-defined problems.

Carat does collaborative energy diagnosis. It collects power data from each phone on which it is installed and uploads the data to the cloud, where the data are compiled into a statistical profile of power use, broken down by app installation, OS version, and more. The collaborative approach is important for many reasons: different devices are used differently, and looking at a single device in isolation would not reveal that an app on one person's mobile is consuming more power than normal. Also, more data means a more accurate statistical profile.

Carat distinguishes between energy hogs (apps that use more power than other apps across the community) and energy bugs (apps that use more power than other instances of the same app across the community). Carat provides lists of these, along with estimated power savings if the user kills the hog or buggy app. It also gives a “J-Score”—a unified score that gives the percentile battery life relative to other users of Carat.

To use Carat, just install it and open the app about once a day, to seed data about power use. After about a week, you will start receiving reports that suggest what apps to close, whether you need to upgrade, etc. Carat is available for both iOS and Android, and is free on the Apple App Store and the Google Play Store. The app code is open source, although the analysis code is

proprietary. No jailbreaking is required. They evaluated whether Carat actually improved users' battery life. After 10 days, users saw a 10% increase on average. After 90 days, they saw a 30% improvement on average.

Their initial deployment had 100 sign-ups, 75 of which installed the app. Developers got 10,000 samples. Over two weeks, they found 35 energy bugs, including popular apps such as Facebook. They also evaluated Carat by injecting three bugs into the Wikipedia app, all of which were detected. After releasing the apps on their respective app stores, they were featured on several online news sites, and found themselves with more than 100,000 users. Now they have more than 450,000. They have detected 11,256 energy hogs and nearly half a million energy bugs, some of which were quite surprising. For instance, they found a case where turning on WiFi could improve battery life.

Cory Lueninghoener noted that some of Carat's recommendations involved updating the phone's OS version. He asked whether Carat ever recommended that users not upgrade, because the upgrade consumed more power. Oliner replied that that was something that they had discussed, but decided not to do; upgrading tends to install security patches, and so it provides an important benefit that isn't related to power use.

Alva Couch asked whether Carat was aware of application-specific settings, and whether it could make recommendations at that level. Oliner answered that Carat does not, but that is something that they want to provide via a developer API.

Soila Kavulya asked whether Carat could compare the power consumption of platforms as a whole. Is Android better than iOS, or is the reverse true? Oliner answered that it would be very tricky to tell, since the two platforms tend to use different hardware and different batteries. User behavior is another factor; some people will constantly reload news feeds, etc.

Tim Nelson asked whether they received useful negative feedback, or if it was mostly trolling. Oliner replied that yes, the negative feedback often gave them insight, even if it was not directly related to Carat.

Rik Farrow asked who paid for Carat. Oliner explained that Amazon Web Services provided a large amount of resources free of charge to his research group, and he expressed gratitude.

Plenary Session *NSA on the Cheap*

Matt Blaze, with Sandy Clark, Travis Goodspeed, Perry Metzger, Zach Wasserman, and Kevin Xu, University of Pennsylvania
Summarized by Rik Farrow (rik@usenix.org)

Matt Blaze started with a reprise of his presentation at Security 2011, but that was not the scary part. Matt began with some background behind the project into open telecommunications networks with the aim of improving security for various wireless

networks. University of Pennsylvania's focus is on two-way public safety radio. And, as this is NSF-funded and not classified, they are obligated to publish their findings.

APCO (Association of Public Safety Communication Officers) Project 25 (P25) is a standard for two-way digital radio, replacing the older analog radios. There are issues with backward capability, which is what Matt spent the next 45 minutes talking about. Because compatibility is the key to standards, multiple vendors' products have similar user interfaces as well as complying with the on-air protocols.

The P25 is used by police, fire departments, ambulances, but also the FBI, Secret Service, Treasury, postal inspectors, and even the US military. The P25's digital radio broadcasts on a narrow (12.5 KHz) channel, with each 180 ms of speech converted into 1728-bit voice frames encoded using the IMBE vocoder. Security is an option, which Matt said makes him excited because he will get to write a paper. For the most part, local emergency services don't want encryption. Federal services generally do, and this is where the problems appear.

The P25 uses symmetric encryption, and cryptokeys must be loaded into the radios in advance of being used. Matt explained that they can be loaded using a big cumbersome keyloader device or over-the-air rekeying, which allows updating of keys only if keys have previously been installed. There are no communication sessions, so the sender sets his radio to select the crypto mode and key, and the receiver must recognize the mode and have the right key loaded for this to work. Matt explained that the design errs on the side of allowing things to happen: radios play plaintext by default, and will also play any encrypted broadcasts for which they have the key. There is no authentication, so there is no protection against replay attacks or falsifying credentials (radio ID).

Matt described the P25 as an "ad hoc design," and there were some things that were done correctly. For example, the radios do not reuse initialization vectors, a common mistake in stream encryption protocols. There are mistakes in other areas: radio unit IDs are sent unencrypted even when in encrypted mode, silent radios can be made to respond (giving away their presence), and the design is very vulnerable to denial-of-service attacks.

Because there is no authentication, an attacker can replay messages, even encrypted ones. Matt joked that he got the FBI off his back by constantly replaying the message: "That Matt Blaze guy has gone to bed, so we can stop watching him." Matt later explained that Travis Goodspeed had discovered that there is a \$15 toy that contains a transceiver chip that can be reflashed so it detects when an encrypted broadcast is occurring and can jam those transmissions by overriding the first 64 of the 1728 bits.

Next, Matt talked about passive analysis, looking for patterns of who is talking to whom, even if the content is encrypted. That type of analysis can be more powerful than actual content, as traffic analysis can be automated. And the P25 has a 24-bit unit ID assigned by the US government, which identifies the agency that owns the radio, and sometimes also the office and even the squad or person who owns a radio. The standard does support encrypting the unit ID, but we've never seen this and have been able to keep track of these IDs over time. If you add a pair of phased directional antennas, you can also locate radios as they transmit. Matt reminds us that the military are using these radios as well, and pointed out that even idle radios can be tricked into replying.

The radios also suffer from usability issues. The transmit crypto switch is an obscurely marked toggle switch, and that switch's state has no effect on received audio. Received audio is played if the signal is in the clear or if the signal is encrypted and the receiver has the key. Finally, rekeying is difficult and unreliable, and many agencies use short-lived keys.

Matt explained that one of the first things they decided to do in the field was to see how often people were using P25 radios in the clear. With a handful of grad students, and several thousand dollars for equipment, they were able to find out that quite a bit of federal agency and law enforcement traffic is in the clear. They decided they would focus on the federal government, by listening to just the frequencies used by sensitive organizations (so not the Park Service, but the Secret Service). There are 2000 channels allocated to the federal government, and they could determine the sensitive ones by watching for those that normally used encryption. They used an off-the-shelf hobby scanner, the Icom R-2500, which includes a P25 option, and is legally available to anyone.

They found friends with homes in high places, installed R-2500 receivers, antennas, and PCs with some software that collects metadata from received transmissions, and uploaded this information once a day. They typically got about 30 minutes of in-the-clear sensitive transmissions per city per day. By listening to this plaintext over time, his analysts, the grad students, identified which channels are used by which agencies. They also heard names of confidential informants, wiretap subjects' activities, about a wide range of crimes, and plaintext from every agency in the federal government with the exception of one—Postal Inspection.

Matt said that the friendly people in legal at U Penn found out that there is a law that specially allows people to listen to law enforcement radio traffic as long as it is not encrypted. They have tried to help, but the usability issues have prevented radio users from successfully improving the rate of encryption (about 90%). They did learn that by being a bit more systematic about

their interception systems, they could learn a lot more. They also observed that various security folklore, such as change your passwords/keys often, actually makes security worse.

Mark Staveley pointed out that you don't have to jam every packet, but Matt said that just jamming every 100th packet would introduce a little stutter in the transmission. Because you only need to jam 64 bits out of 1728, and those 64 bits always follow a synchronizing frame, it is still just a tiny fraction of the energy needed compared to jamming the entire frame. Mark then asked about the cipher (a streaming cipher) and suggested the super-secret agency Matt wondered about what would be inside the Postal Inspection office. Someone asked whether they found any evidence that the vulnerabilities they discovered were actually used by black hats. Matt did hear a couple of times about "targets being sophisticated and taking counter-measures," but these messages were in the clear. Another person said that if the encryption algorithm was developed in the '90s, it could be decrypted. Matt pointed out that although DES could be decrypted with an exhaustive search of the keyspace, AES, which uses a 256-bit key, is certainly out of range of an exhaustive search so far. Someone asked about the postal inspectors and their rekeying habits, and Matt said they don't rekey over the air, and perhaps are not changing their keys as often. The same person wondered whether perhaps they should produce a device with some useful function but that also did some jamming on the side. Matt said, "You're evil. Let's talk some later."

Rik Farrow asked how long have they been talking about the use of the receivers, and Matt said for about a year publicly. Rik then asked if they were scanning, and Matt said, yes, they are essentially sampling. Rik said that Matt and his graduate students have now collected enough information to make them an interesting target for Advanced Persistent Threat-style actors. Matt replied that they took a fair amount of care that the machine the data is uploaded to moves the data behind a firewall quickly. The easiest attack against us would be to apply to grad school at Penn and get accepted. It would probably be easier to get your own radios, Matt suggested.

Doug Hughes wondered about the first time they talked to the FBI and said that they wanted to record your over-the-air traffic. Matt said that they didn't have that conversation. They did tell the FBI, but only after they had been doing this for a while. They did talk to their IRB (Institutional Review Board) because they were collecting personally identifying information, which must remain private. They are also identifying federal agents who are making mistakes while using their radios and could get in trouble if they were identified. So they are prohibited from sharing that information with the authorities by their IRB. Someone else asked whether they are still collecting information, and Matt said they have two more radios than when they started, and

that he always asks for a room on a high floor whenever he stays in a hotel.

They have shared their software with the government, but there is a problem: their software runs on Debian, which is not approved software. They have to “smuggle” their software in, so it can be used.

Papers and Reports: Content, Communication, and Collecting

Summarized by Dybra Grande (granded@coyote.csusb.edu)

What Your CDN Won't Tell You: Optimizing a News Website for Speed and Stability

Julian Dunn, SecondMarket Holdings, Inc.; Blake Crosby, Canadian Broadcasting Corporation

When the Canadian Broadcasting Corporation (CBC) reimplemented their content delivery network (CDN) architecture and changed it to a static delivery system, they never imagined the resulting scalability issues caused by the redesign of the configurations on their CDN and servers. To remove any possibility of downtime the writers of CBC agreed origin stability, content freshness, system complexity, and cost were significant. Originally, CBC's Web site was driven by J2EE applications that rendered news content from a relational database, which was difficult to maintain and meet business requirements. CBC's origin systems now consist of an Apache Server farm with no dynamic modules. Stories are now generated on content management systems (CMS) and converted into HTML fragments containing headlines, story body, associated links, and other user-displayed metadata. These HTML fragments are then wrapped by a “story wrapper” template using Server Side Includes (SSI) in which story variables are then injected accordingly throughout the CBC Web site.

Julian Dunn explains, the implementation parameter CBC uses to optimize CDN content freshness is based on setting almost all objects except HTML to a global site TTL of 20 seconds. In the act of achieving a high origin offload, edge servers will issue GET requests to the origin with an If-Modified-Since (IMS) header to ensure object bodies were updated and not sent unnecessarily through the system. Objects such as Site Icons, JavaScript, and CSS had a rigorous change control process in which expiration is organized through file systems with top-level directories. Also, to enable last mile acceleration and origin compression, the CDN's edge server will use gzip compression to send content to end-users without needing to recompress them. To enable HTTP-persistent connections and set appropriate timeouts, the CDN will attempt to keep a pool of connections open to avoid cache misses. These measures help ensure origin stability, content freshness, system complexity, and abolish downtime.

Dunn mentioned that SSI technology suffers criticism due its lack of incorporation of languages such as PHP and Ruby.

Although SSI does not incorporate more complex languages, it provides security and performs well under high loads. It also protects the company and its employees by providing a good audit trail. Dunn concluded his presentation with several general lessons learned: (1) keep cache rules simple; (2) keep tuning knobs at origin if you can; (3) organize and categorize content; and (4) understand what “TTL” actually means. After the presentation, Brent Chapman, the session chair, asked whether the CBC ever considered automating the turnout process? Dunn responded, “It's a bit of a judgment call. There is a way to do it one way, turn off site features, or increase TTL. In the end, we will need developers to intervene. Yes, we can automate, but it is not just one knob.”

Building a 100K log/sec Logging Infrastructure

David Lang, Intuit

David Lang discussed the need for Intuit to create a high volume logging infrastructure that can handle large batches of logs. In previous years, logs grew 60% per year, and traffic has only become more concentrated over time. Additionally, 75% of the possible logs were not being fed into the system. In 2005, vendor-neutral solutions such Arcsight, Sensage, Splunk, Nitrosecurity, and Greenplum were evaluated. The result was that none of the vendors were able to handle a 100K logs/sec load or the desired alerting/reporting functions. The architecture that Digital Insight decided to use was rsyslog for gathering and transporting logs.

Before Digital Insight decided to work with rsyslog, they tested several services such as sysklogd, syslog-ng, and rsyslog. Sysklogd daemon lost thousands of logs/sec under increasing traffic volumes. Syslog-ng hit a wall around 1K logs/sec and just dropped messages above this rate. Rsyslog handled short peaks of 30K logs/sec as it processed incoming messages on the memory queue, with the restriction of being able to write only a few thousand logs/sec. If traffic spiked and was greater than the memory could handle, however, it will would start losing lots of log messages. When it came to transporting logs due to the large number of networks, they decided to implement a set of syslog relay servers. These syslog relay servers were built in HA pairs and were set on an interface of 90 while accepting the risk of the unreliability of the UDP syslog messages being blocked by the router choke points from other networks. For delivering logs, they needed a reliable solution that could support multiple copies of logs of 100K logs/sec and a GigE wire speed of 400K logs/sec. They ended up going with Multicast MAC traffic software called CLUSTERIP using Linux. CLUSTERIP's role is to hash the connection of information and divide the resulting bases into a number of buckets, which are assigned to the local machine up the stack.

Someone asked how receptive the developers and management were. Lang replied that syslog developers were very receptive to

the changes of the core syslogs, but that it was difficult getting approval or understanding of the importance of the project from the management and finance departments. Another audience member asked whether there were any nasty surprises or disappointments. Lang replied that their biggest hurdle was dealing with proprietary software, but they were pleasantly surprised with the results. Another audience member asked whether their data center had a virtual center. Lang replied, “Our data center did not have a virtual center. We are interested with what happens with virtualization, but with everything else it is a factor. You really will have to do some testing, you will need more machines and more instances.”

Building a Protocol Validator for Business to Business Communications

Rudi van Drunen, Competa IT B.V.; Rix Groenboom, Parasoft Netherlands

Rudi van Drunen described the design and implementation process of a system that tests and validates secure communication using XML messages through the AS2 Standard. This system provides a way for XML data to enter encrypted through an authenticated receiver using S/Mime. This protocol is essential to enable the deregulation of the energy market in the Netherlands. The goal of this project is to provide a test environment that can be used to certify more than 100 market parties to adhere to the new XML definition during the migration process. During this process more than 50 applications and protocol test scenarios will be verified before they are certified to participate in the new communication infrastructure.

The HTTPS and AS2 communications are handled by an Open Source Enterprise Service Bus (UltraESB). UltraESB passes the XML payload to a product called Virtualize, which is used as a virtualization engine to test validity in XML messages. Virtualize handles responses while storing data in a MySQL database. Information stored in the database includes meta information on business partners or timestamps. When it came to authentication and encryption of XML messages on the AS2 level, a Public Key Infrastructure (PKI) was used by the Dutch energy market and maintained by the government.

Someone asked whether there were ongoing certifications for certain versions. Drunen replied that recertification is necessary when vacancies or software updates occur and that a new partner coming to a new environment would need to be recertified, but it varies with different protocols. The same person asked whether they would use the two-way two-level encryption authentication scheme again within their database. Drunen replied yes, it was important to secure their database using a two-way two-level encryption authentication scheme.

Invited Talks

Surviving the Thundering Hordes: Keeping Engadget Alive During Apple Product Announcements

Valerie Detweiler and Chris Stolfi, AOL

Summarized by Nick Felt (nfelt1@sccs.swarthmore.edu)

Valerie Detweiler and Chris Stolfi, both AOL veterans of about a decade, jointly described the experience of keeping the popular tech Web site Engadget up and serving requests during peak traffic (i.e., when Apple announces new products). Chris noted that Engadget runs on the same shared publishing platform as more than 800 other AOL sites, but it got 4.4 billion requests in two hours during the last iPhone announcement, which is more than most AOL sites get during a week. Since 2007, Engadget has run a live blog for high-profile news stories such as Apple product events, using a revamped framework that has supported an increasing number of updates per event (reaching 973 updates for the iPhone 5). Chris observed that the condensed traffic surges triggered by these events can at least be anticipated, which allows them to prepare—and this is vital because the tech blog industry hasn’t always been able to weather these events, meaning even more traffic for Engadget when the competition goes down.

The overall approach that Engadget takes to withstanding these traffic surges relies on a fairly traditional LAMP stack, with several layers of protection against high traffic. Live at the event, Engadget reporting staff submit new content to the CMS, which gets passed back to MySQL (for text) and media store (for photographs). At the same time, users’ Web requests arrive and either hit the CDN or go straight to the load balancer, which has its own three-second TTL cache. Behind the load balancer is a LAMP front-end for MySQL and Apache for the media store, plus nginx as a proxy and cache for external API calls (so that Engadget can at least still serve stale content if partners go down). Memcache protects MySQL with about three gigabytes’ worth of cache per server, which Chris said is generally more than enough. He emphasized the importance of having multiple layers of caching, which together allow them to get by with only one relatively modest machine as a MySQL server per data center. Valerie showed a chart of traffic during a peak event, explaining that the goal is to have the CDN handle most incoming requests and then serve the majority of those that pass it from the load balancer cache, thus leaving only a small fraction that actually hit the Web server.

Besides the core stack, Engadget has developed strategies for withstanding traffic surges based on lessons learned from previous events. One of these is simply to lighten the load by sending fewer bits to the client; for the iPhone 4’s event they had to serve 100 Gbps, but they actually reduced this substantially for the subsequent iPhone 5 event despite having more updates and more readers. They accomplished this by switching their live blog page to update itself incrementally instead of

requiring the user to refresh the page. This allows Engadget's servers to send back single live blog updates via JSON instead of the entire page, sustaining rates of more than 100,000 JSON calls per second because the calls are lightweight enough to be cached easily by the load balancer. Another strategy is to reduce complexity, favoring availability over performance. This means relaxing geolocation constraints and serving some users from relatively far away data centers. (Valerie recounted how for one early keynote, requests were so concentrated in California that the Mountain View data center was overwhelmed with traffic, leading to a domino effect as the traffic then hit the next-nearest data center, and then the next.) It also meant removing extraneous beacons and ads from the live blog page, because third-party infrastructures would fall over under the heavy load. With these techniques, Engadget was able to stay up successfully during the entirety of the last iPhone announcement.

Chris Reiser (Groupon) asked, given all the caching, whether it was possible to clear the cache in the case of a bug. Chris Stolfi answered that it's a non-issue for the live blog's JSON calls because they're only cached for three seconds, and for the CDN they can use a tool to clear it; Valerie noted that because objects are versioned, the preferred option is to do a new publish. Jake Richard (Yahoo) asked how they determine at least an order of magnitude scale for what they need to have to handle the traffic. Chris pointed out that until recently Engadget had never stayed up the whole time, and thus hadn't known many people had tried to visit the site. Now that they do have this benefit, he said it was pretty much just a matter of doing standard load-testing to get a unit of scale and then estimating how many people they expect to come. He noted that Engadget does get influxes of new people as other sites fail, but they can compensate for fluctuations in traffic by adjusting the live blog client's query interval, say from the three second default up to five seconds, in order to control the rate of JSON calls.

Vitess: Scaling MySQL at YouTube Using Go

Sugu Sougoumarane and Mike Solomon, YouTube

Mike Solomon and Sugu Sougoumarane of YouTube discussed their recent work building Vitess, a project in the Go language designed to improve the scalability of MySQL. They divided up the talk such that Mike covered the MySQL aspect and Sugu addressed their experience using Go. Mike began, explaining that YouTube had originally scaled MySQL up to the cluster level with a collection of homegrown scripts that could be difficult to use. Vitess was born of the desire to distill those scripts down to the simplest way to manage a sharded MySQL instance. They wanted to stick with MySQL because it's popular, easy to use, and reliable, but doing so at a large scale required overcoming obstacles: making the system relatively self-managing to reduce the time needed to manage hardware, and increasing efficiency to support greater throughput without needing

thousands of connections to the database. They decided Vitess would use external replication with eventual consistency in order to get data out in near-real time, and would provide automated reparenting of slaves so that a wide array of operations could be performed conveniently by doing them in the background on a replica and then failing over to a new master. The database would be sharded primarily by the leading edge of the primary key, and would not provide cross-shard transactions, which Mike said might seem like cheating, but in their experience was a reasonable limitation.

Given these constraints, the implementation strategy for Vitess was to make minimal changes to the MySQL codebase—just two 25-line patches—and rely instead on an external tablet manager and an external query shaper, both written in Go. The tablet manager maintains the sharding of the entire space of primary keys up into individual tablets, and lives on every box running MySQL. It stores coordination data directly in Apache Zookeeper, a highly reliable notifying file system from the Hadoop project. The query shaper provides an RPC front-end to MySQL, and has been serving all of YouTube's MySQL queries in production for more than a year. It manages a pool of database connections, and provides a number of fail-safes, including query consolidation in the case of duplicate queries, row count limits for the number of rows to return, and a SQL parser that allows it to intelligently reshape queries on the fly. Besides the tablet manager and query shaper, each tablet server also provides an update stream of primary key change notifications derived from the database binary log. Work is in progress developing a row cache to support better random access performance than MySQL's traditional page cache.

Sugu described some of the highlights and lowlights of using Go for the Vitess project. He noted that the main benefit has been in productivity: writing code went quickly because the language is much more expressive than other widely used compiled languages, falling closer to Python than C++ in that regard. Go's quick compilation (for example, the Vitess tablet server compiles in less than three seconds) and well-designed set of libraries also saved development time. He touched on his appreciation for several of Go's helpful language features, including an intuitive approach to interfaces, first class concurrency via lightweight goroutines, syntactic elegance with defers and closures, and the ability to call into C code. At the same time, he also pointed out some of Go's rough edges, including mismatch between string types, lack of agreement on how to handle errors, and deficiencies in the garbage collector and scheduler (although work continues on both components and he expects them to improve). Mike spoke on deploying Go code in production, saying it was relatively easy to debug—sending SIGABRT tells you the state of every goroutine stack—and casting it as a good experience overall. He and Sugu said Vitess recently picked up three new

committers for a total of five, and invited involvement in the project at <http://code.google.com/p/vitess/>.

Someone asked for details about the version of MySQL that Vitess uses, and Mike said they were using the community build of MySQL 5-point-something with a few small patches, rather than the Google internal build. Someone asked whether they'd looked at 5.6 and GTIDs (Global Transaction IDs). Mike said the route they've chosen is applying GTIDs to the 5.1 tree using Google's stable internal patch. Vince Clark (VMware) asked about issues in debugging code with goroutines, particularly interactively. Mike answered that although Go can be massively concurrent, it has good primitives with a clear memory model, so it's generally not a problem. Triggering a panic produces a full stack trace of every goroutine in flight, which then just needs to be examined carefully to diagnose problems. Asked as a follow-up whether the lack of visible thread IDs hampered inspection, he explained that the reduced exposure hasn't been limiting, because the specific thread only mattered when interacting with certain kinds of C code. Sugu also remarked that for their one tough deadlock issue, they still just needed to crash the server and then examine the stack trace. Finally, Kent Skaar (VMware) asked whether they had used the SSL support in Go. Sugu answered that they've tried SASL and messed with the crypto package but haven't used SSL.

Ganeti: Your Private Virtualization Cloud “the Way Google Does It”

Thomas A. Limoncelli, Google, Inc.

Summarized by Andrew Hume (andrew@research.att.com)

Tom Limoncelli started with an overview of Ganeti, a management tool for clusters of VMs (either Xen or KVM). The fundamental terminology that Tom used is node equals physical server and instance equals VM. VMs can use a SAN or local disk. When using local disk, they use RAID to mirror across two nodes so that the disk is in two places, and thus we can move the VM. Moving VMs is based on two primitives: move a VM and move virtual disk (storage).

Tom said that being able to move VMs provides real benefits if the VM needs more memory than on the node it's currently on, or in the case that a disk or node is failing. He then described various roles in Ganeti, such as the master node and some processes—for example, the node daemon and Ganeti watcher—and different sized configurations (small, medium, and huge). The scaling issues involve an administrative lock on node/instance operations (not any VM internals).

Google tends to use Debian-based Xen in para-virtualization mode, with DRDB (Distributed Replicated Block Device) and local disks (no SAN). Google operates “huge” clusters in a few data centers for self-service, and one or so medium cluster per office (“office in a box”). Tom then described a bunch of

management tools and how Google manages their clusters (e.g., clusters are generally tuned for one of a few different workloads). Tom finished with a live demonstration using Ganeti on a test cluster.

The code can be found at <http://code.google.com/p/ganeti>.

DNSSEC: What Every Sysadmin Should be Doing to Keep Things Working

Roland Van Rijswijk, SURFnet bv IPv6 and DNSSEC

Summarized by Steve VanDevender (stevev@hexadecimal.uoregon.edu)

You might already be using DNSSEC and not know it. Traditional DNS does not provide authenticity or integrity information, but with DNSSEC, domain owners can digitally sign zone data, and resolvers can check those signatures to verify authenticity and integrity of DNS data.

The EDNS0 standard provides support for DNSSEC by specifying additional flags and larger UDP replies of 4096 bytes (by default) for DNS information, and is enabled by default in modern DNS server software (such as BIND, Unbound, and Microsoft Server 2008R2 and 2012). In particular a client resolver can set the “DNSSEC OK” flag to request a DNSSEC reply, and this is also enabled by default in many recursive resolving servers. Even if a client resolver doesn't ask for DNSSEC, it may use a name server that is one of the 70% of all recursive resolvers on the Internet that do have DNSSEC enabled, and 90% of those use the 4K default maximum reply size. Typical DNSSEC replies may return more than 3K bytes of data and therefore may be broken into three or more IP fragments.

Fragmentation causes problems because some firewalls drop fragmented DNS replies, originally in response to some security attacks common in the 1990s, and such configurations still exist because of outdated recommendations from vendors and auditors to block fragmented DNS UDP replies and disallow TCP DNS replies. If a resolver makes a DNSSEC request behind such a misconfigured firewall, it never receives a complete reply, and the resolver eventually sends an ICMP fragment assembly timeout message back to the server. Monitoring these ICMP messages allowed SURFnet to estimate that 1% of resolvers contacting them had this problem. Other research suggests 2% of all Internet hosts and 2–10% of recursive resolving name servers may have this problem. Resolvers may also experience serious performance issues if DNS fragments are blocked, as they will eventually retry using TCP but can take several seconds to do so.

To avoid problems on your recursive resolving servers, van Rijswijk recommended verifying that your resolvers can receive large and fragmented UDP DNS replies. DNS-OARC provides a tool for this at <http://www.dns-oarc.net/oarc/services/replysizetest>. You should also configure firewalls not to drop fragmented DNS replies and not to block TCP DNS replies on port 53. You can also reduce your EDNS0 maximum reply size to 1472

(below the Ethernet MTU) or 1232 (below the IPv6 MTU) to reduce problems with fragments.

Another problem encountered by SURFnet after enabling DNSSEC was network amplification denial-of-service attacks from DNSSEC UDP queries with forged source IP addresses. A query of 68 bytes can return a reply of 3300 bytes, resulting in an almost 50-fold increase in bandwidth between the attacker and the attack target. One attack was observed to generate 38 Gbps of traffic toward a target, with their name servers receiving 10 Kbps and sending 50 Mbps to the target.

One way to prevent such amplification attacks is to implement IETF BCP38 <http://tools.ietf.org/html/bcp38> to filter spoofed traffic. DNSSEC server operators should also monitor for such attacks and filter them. Rate-limiting DNS can also help, but rate-limiting is not yet available in all name server software and may affect legitimate traffic if not implemented carefully.

DNSSEC Deployment in .gov: Progress and Lessons Learned

Scott Rose, National Institute of Standards and Technology (NIST)
Summarized by Steve VanDevender (stevev@hexadecimal.uoregon.edu)

The US federal government has mandated that .gov DNS zones are to be digitally signed and served with DNSSEC. This was originally motivated by Dan Kaminsky's presentation on DNS spoofing at Black Hat 2008, followed shortly by OMB-08-23, which mandated that the .gov zone was to be signed by January 2009. The rest of the federal executive branch was to be signed by December 2009, and DNSSEC was added to the FISMA standard requirements for all federal information systems.

Progress on DNSSEC deployment was not as rapid as was hoped. The .gov zone was not actually signed until February 2009, and only 30% of the subzones met the original deadlines. Furthermore, 10% of the zones that were served with DNSSEC had various errors, although only a few were noticed by operators or client resolvers. Some of these errors persisted for about two weeks until they were corrected.

A number of challenges made deployment and maintenance of DNSSEC difficult. Besides trying to meet the initial deadlines, DNS data has to be re-signed periodically even if the zone data did not change. DNSSEC also required more interactions between parent and child zones, with child zone keys needing to be uploaded to parents whenever they change. Existing DNS operators also had to learn DNSSEC and sometimes had to change their service plans or even obtain new equipment. This led to some consolidation and reorganization of existing DNS service.

To address problems with lagging deployment and failed security audits, a "DNSSEC tiger team" was formed in April 2011 by the federal CIO council and staffed by volunteers. The teams

hold monthly meetings to discuss progress and problems with deployment. They produced training material and monitoring tools and created discussion forums for other government system administrators. They also produced reports for departmental CIOs, but these were not always handled quickly and may not have been all that helpful.

The "tiger team" did produce an improvement in the number of signed and valid DNSSEC domains under .gov, although the number of "island domains" and domains with errors remained fairly consistent. Currently 70% of .gov domains are signed.

Between August 2011 and March 2012 there were frequent problems with DNSSEC errors, although the rate fluctuated significantly. The most common errors were expired signatures. Centralization of some services led to bursts of errors when subzone operators forgot to sign their zones. Many of these correlated with holidays when operators were unavailable to renew signatures. Other problems included bad key rollover, when keys were mismatched between parent and child zones, or mismatched timestamps, where a child zone appeared to have been signed before its parent; however, in the first year after the "tiger team" was formed, response to errors improved significantly, particularly with the common problems of no or expired signatures on domains, with error rates reduced to about 20% of their initial levels and problem resolution times cut in half.

Rose drew a number of lessons from the US federal government's DNSSEC deployment efforts. Monitoring to report errors was the first step to indicate the scope of problems. Getting organizations to provide current points of contact for DNS and security operations improved resolution times. Operators were encouraged to automate the error-prone aspects of DNSSEC operation, especially zone re-signing. Fostering an internal community for DNS administrators made it easier to share information and solve operational problems.

Papers and Reports: If You Build It They Will Come

Summarized by Steve VanDevender (stevev@hexadecimal.uoregon.edu)

Building the Network Infrastructure for the International Mathematics Olympiad

Rudi van Drunen, Competa IT; Karst Koymans, University of Amsterdam

The International Mathematics Olympiad is an annual event held in a different country each year, with more than 600 international high school students as competitors, more than 100 jury members, and with more than 60 different languages represented. The contest itself occupies two days, but an additional five days are involved in preparation, translation, and correction and scoring of papers. The contest held in the Netherlands was hosted at two Amsterdam hotels, an Amsterdam sports complex where the competition was held, and an Eindhoven hotel for the jury members. All of these sites were networked together for communication among the contestants and jury members,

although traffic had to be isolated between those groups for security, and with substantial flexibility to allow for considerations such as people moving between hotel rooms.

The network they developed used VLANs to isolate traffic while allowing it to be consolidated on common physical links. A VPN system based on FreeBSD OpenVPN was used to provide security and allow more flexible access. Traffic was also isolated from the general Internet using NAT with a gateway at the University of Amsterdam. A backup network using 3G was also available in case their telco connections went down, and “warm standby” host replacements were available at each site. Site setup took six people working over four days, involving lots of improvisation and thorough documentation maintained in a wiki.

van Drunen drew several lessons from their experience. Expect the unexpected in site surveys. Use a wiki for documentation. Use DNS for all host information. Label everything—cables, hosts, equipment. Use open-source tools such as FreeBSD, OpenVPN, and Wireshark. Provide hand tools at all sites for hardware fixes. Be flexible by design, such as putting all VLANs on all switches and avoiding complicated procedures and layers of management. Test everything. Allow enough time for building your network. Have multiple communication methods available. Take time to prepare and build your network.

Lessons Learned When Building a Greenfield High Performance Computing Ecosystem

Andrew R. Keen, Dr. William F. Punch, and Greg Mason, Michigan State University

Awarded Best Practice and Experience Report!

Building a high performance computing (HPC) environment involves more than just getting the most FLOPS (floating-point operations per second), but in making it an effective tool for its users. HPC is critical to research and often provides a competitive advantage, but it also requires substantial funding from university administration to create a useful resource. A first attempt to build an HPC system with the involvement of a major vendor appeared to have great benchmarks, but it underperformed in real use mainly due to inadequate I/O bandwidth for storage.

Storage for HPC needs to be fast but also safe to protect user data. For their environment the team used Lustre over Infiniband for storage with the ZFS file system. This provided for snapshots and off-site replication of data for backups. To improve responsiveness, solid-state disk (SSD) was added for caching. Later, the storage system was designed to allow for failover, and it had increased CPU capacity and less use of SSDs since they found that RAID caching was not being well used.

Their HPC system had to fit in a small machine room, with lots of power dissipation and need for cooling. Spot cooling was

used to deliver cold air to system intakes, and inexpensive heat containment was obtained by using cardboard to route airflow (later upgraded to Plexiglas). They found that using standard IPMI instead of proprietary management tools made hardware management much easier; firmware updates and configuration could be easily managed remotely, and with better consistency than manual updating. Software management was done using configuration management systems—for consistency, systems were never managed “by hand.” They also found that using a single OS across the entire cluster made management easier, and an open-source distribution like ROCKS has already solved many HPC design problems. Job queuing was request-driven and allowed for managing multiple jobs in parallel; however, queue selection was automatic for users, so they did not have to learn details of the queuing system to manage their own jobs. Systems were monitored using in-band methods to track performance, Cacti to do out-of-band monitoring, and Nagios for failure alerting.

Someone asked how to set up trust properly between systems. Keen replied that one example is allowing management hosts to ssh to managed hosts in the cluster. The assumption that hosts in the cluster should trust each other is not a good one, however.

Building a Wireless Network for a High Density of Users

David Lang, Intuit

Lang attended SCALE (Southern California Linux Expo) in 2008, and like many attendees at many conferences found that the wireless network didn’t work very well. He volunteered to help design a better wireless network in 2010, with his technical expertise including experience as an amateur radio operator, after the original wireless vendor backed out shortly before the conference started.

Wireless networks that appear to work in early testing often collapse when lots of people try to use them. Technical conference networks are especially problematic because there are lots of people—and, more importantly, gadgets—in a small area. Collapse is inevitable when fundamental limits on the amount of radio airtime available are reached, but it is possible to delay that collapse, sometimes by doing counterintuitive things.

WiFi has the same problem as radio in that only one device of any sort can be “talking” at any one time on a channel. In high-density areas there are also “hidden transmitters” where devices on one side of an AP may not be able to detect devices talking on the far side. It takes little interference to corrupt transmitted packets. Wireless devices may try to reduce transmission speed to overcome interference, but that just increases the airtime they use. Even regular housekeeping traffic may use up too much available airtime to allow devices to transmit data. Many OSes use large network buffers, and this “bufferbloat” may cause a device to transmit for long periods and retransmit more when

reception fails. Turning up transmitter power on APs usually doesn't help since it just increases interference between APs and doesn't help with receiving data from low-power mobile devices. So-called "enterprise-class" APs often don't help because they typically concentrate many radios in one place and use directional antennas that create more hidden transmitters.

There are a variety of methods for solving these radio problems. Doing a site survey of your venue helps by allowing you to determine better AP placement, especially if you measure signal strength using mobile devices and tools such as MySpy and Kismet. You can also find the wired network jacks that actually work. As much as possible use 5 GHz WiFi, which has 8–18 available channels (depending on sources of interference at a location) instead of the three available in 2.4 GHz. Use lots of APs, and set them to use lower transmitter power, especially no more power than client devices use. Use existing transmission obstacles such as walls or the presence of crowds to avoid hidden transmitter problems. Placing APs closer to the floor may also help. Advanced antennas should be used carefully to direct signals away from areas rather than toward them, and directionality can also help to avoid cross-floor interference.

Using a single SSID can allow devices to roam between APs, but have separate SSIDs for 2.4 GHz and 5 GHz if both are supported since devices may give up before finding 5 GHz and obtaining better performance. Enable wireless isolation so APs don't relay traffic between mobile devices. Reduce the "beacon interval" so less airtime is used for housekeeping. Using distinct prime number intervals across multiple APs also avoids collisions between beacon broadcasts. Disable slower speeds (i.e., 1–11 MHz). On APs, reduce kernel network buffering, disable memory-intensive connection tracking, and use short inactivity timeouts to forget about inactive devices sooner.

Invited Talks

Disruptive Tech Panel

Summarized by Barry Peddycord III (bwpeddy@ncsu.edu)

Moderator: Narayan Desai, Argonne National Laboratories

Panelists: Vish Ishaya, RackSpace; Jeff Darcy, Red Hat; Adam Oliner, University of California, Berkeley; Theo Schlossnagle, OmniTI

Each panelist began by talking about his predictions for the next 10 years of system administration. Adam Oliner predicted that there is going to be a paradigm shift where system administration will take on a more substantial role in software development. Most of the development that administrators do is in the form of scripting because the APIs for the tools deployed will not necessarily be consistent from environment to environment. With the push to cloud infrastructure, the interfaces to these resources—and, by extension, the solutions developed on top of these resources—have started to converge. While scripts are appropriate when each system is wholly unique, the growing trend toward unified APIs means that solutions will be more

generalizable, and it will be more effective for system administrators to share their approaches with one another so that they can help stand on each other's shoulders. It is at this point that scripts become software projects, and system administrators begin to become developers.

Theo Schlossnagle disagreed, arguing that APIs are only meaningful when they serve as a layer of abstraction on top of a reliable resource. He asserted that the role of a system administrator is to make an unreliable infrastructure less unreliable for the benefit of their developers and users. The cloud, despite being widespread, is no more reliable than any of the other resources that make up computer infrastructure, and, in fact, sharing solutions that leverage a common API simply makes it more likely that common mistakes will be shared as well. When a script that solves a problem exists, it is very attractive even if it is inefficient or inappropriate for the usage scenario.

Whereas Oliner and Schlossnagle believed that predicting the future is easy, Vish Ishaya wasn't so sure, stating that visionaries have been making poor predictions of the future for decades. He said that rather than looking at what people are adopting, the best way to see the effect of disruptive technology is to look at what people are not doing anymore. He alluded to how C was disruptive in an era when programmers were using assembly language to accomplish tasks on their machines, as it stopped the practitioners from doing what they were used to doing in their daily jobs. He mentioned that there are many jobs that system administrators do that may be abstracted away, such as managing databases, Web services, and distributed systems. Echoing Oliner, he cited the explosion of APIs as an indicator of things to come.

Jeff Darcy changed the tone by looking at a more specific technical issue, primarily the changes to storage over time, with storage behaving more like memory and being distributed across multiple machines in networks. When storage essentially becomes permanent memory, many assumptions about the behavior of the storage can no longer be made. Although good security practices often involve clearing passwords and keys stored in memory, new practices for protecting sensitive data have to be addressed when the abstractions about where memory is located and where it is copied no longer hold. Furthermore, as more memory is distributed, the issue of desynchronization has to be considered as well. When data diverges across sibling nodes in a network or between caches held by systems, assumptions made by applications about consistent data in memory may not hold, and the decision to read invalid data or block until the data is valid can be a hard one to make in some scenarios.

In addition to predicting the changes to the field of system administration, the panelists also talked about what they were most excited about in the future of system administration, and

the responses were all over the board. Oliner was most excited about the idea that, because that technology has started hitting the limits set by the laws of physics, the next generation of administrators, developers, and academics are facing a new set of constraints that must be worked around. Because latency is the unsurpassable bottleneck of networking, the next steps are not in improving speed, but improving prediction—moving computational power to take advantage of Big Data. Schlossnagle was also excited about optimizing, with the potential for full recreation of systems from the bare metal, while Ishaya was more interested in seeing how these newly created systems would be treated as systems of their own, building abstractions. Darcy closed the discussion by saying that he was looking forward to advances in asynchronicity. As mobile devices increase in number, and latency grows due to the geographical concerns of a globally connected world, dealing with asynchronous storage and communication is the current problem that needs to be faced.

TTL of a Penetration

Branson Matheson, NASA

Summarized by Mario Obejas, Raytheon

Branson is a 24-year IT veteran who loves a “You can’t do that” challenge. He began by asking what kinds of sysadmins were in the audience, and when he asked how many security administrators were present, only a few hands went up. In a sense it’s a trick question since, as Branson asserted, security should be a component of every system administrator’s duties.

Branson then presented a series of referenced statistics to create a context for the talk by comparing the number of sysadmins supporting associates in a business to those in particular roles: 1 to 30 for sysadmin, 1 to 200 for network admin, and 1 to 1200 for security.

Branson defined black hats as individuals trying to impact an organization negatively, providing the usual list of suspects: script kiddies, bored students, hacktivists, governments, and organized crime. Branson also said that vendors, developers, and users are also often unintentional black hats. And, as always, users are the weak point in the system, subject to social engineering.

Eighty percent of US households have at least one computer. These have a plethora of operating systems, with a profusion of services and applications. The attack surface is huge. He estimated that there are 141 million workplace users in the US and 20 million of these are government users. There are more than 240 million home users, with 85 million on broadband. Given these statistics, Branson asserted that a penetration test (aka pen test) should go after users more than infrastructure.

Branson estimates there are 5 million real hackers in the US alone. Black hats have the advantage (tools, knowledge, sites,

conferences, certifications, etc.). Training for the latter includes ShmooCon, DefCon, B-Sides, LISA, etc. Branson said that a person using Metasploit can easily penetrate a network-connected WindowsXP box in <1 second. Aircrack can crack a WEP key in 6 seconds.

Unlike white hat rules, there are no black hat rules other than “Don’t get caught.” The bar for entry into the black hat world gets lower because cool tools come along every day, and existing ones get better. Survival time of an unpatched machine directly connected to the Internet is on average less than five minutes now.

Branson described five pen testing/attack steps: 1. Target reconnaissance: Pig, Maltego, Netcraft, Google are tools of the trade. Social Engineering is another standard tool: call a support line, change a password; also, use public knowledge to answer security questions. 2. Probing: where are the holes? 3. Exploit: (Metasploit, hydra, custom hacking scripts). 4. Once in, cover your tracks: clean logs, hide code, install root kits, obfuscate network traffic, disable monitoring, pivot to spread to other systems.

With knowledge of the pen tester’s list of actions, the sysadmin/victim needs to be on the lookout for unwarranted increases in support calls, spikes in Web traffic, increases in “Friend” requests, increased probe/suspicious traffic (as noticed with Snort), increased load, increases in httpd-error and EventLog (Windows) activities. After an exploit is successful, the victim may see the following symptoms: changed files on file system, changed system behavior (possibly compiler use), and network traffic changes.

Prevention starts with baselining a good system. This is in fact the primary overt message from this talk. Do the same thing your adversary would do—do reconnaissance on yourself, and know what is out there about you. Be aware of what operating systems you have, which services are available, and what levels of traffic are “normal.” You also want to baseline your ticketing trends.

You will want to use IDS (such as Snort) inside your networks, and to perform log analysis using such tools as awstats, Webalyzer, kernel/security log reduction (via Splunk, for example), log watch, and ntop. It’s very important to centralize logs and aggregate the data reduction. Tools such as OSSIM and Bo are integral and will save you time. You can also use configuration management to notice baseline changes, as well as to recover from penetrations.

Near-Disasters: A Tale in 4 Parts

Doug Hughes, D.E. Shaw Research, LLC

Summarized by Yakira Dixon (dixon@coyote.csusb.edu)

Doug Hughes began his talk with powerful slide images of disasters (a flooded Verizon data center during Hurricane Sandy, the rubble of the 1906 San Francisco earthquake, the aftermath of

Fukushima) before he discussed four unrelated near-disasters that he and his team experienced at the beginning of 2012. The first issue involved a degraded WAN. The network between their primary office and primary data center is an OC-12, an optical, leased line. If the OC-12 link went down, failover to a backup connection would cause a jump in latency. It took some investigation to figure out that mismatched fiber-optic transceivers between the partner-provided OC-12 router and the carrier-provided OC-12 equipment was causing the primary network link to go down. The carrier replaced the long range receiver with an intermediate range transceiver and the primary link was up and running again. Both transceivers were eventually replaced with short range transceivers so they could not overpower each other.

The second issue involved archive failure. A mega RAID controller for a backup storage server lost knowledge for a group of eight adjacent disks. Other disks on the controller had no issues. They tried to resolve the issue by reseating the disks and by power cycling the server. When the disks were relabeled with RAID controller logical unit (LUN) labels, the disks were visible with large integer labels instead of controller numbers, but there was output showing the label that the disk used to have. They attempted to relabel the disks with `dd` using this output but ran into namespace collisions with the new mapping. The issue was fixed by removing the eight disks, rebooting, re-inserting the disks, and restoring the label to a factory default using the Solaris command `format -e`. Then the labels were fixed one at a time. Some things that were learned: ZFS can repair disk labels wiped by `dd`. ZFS output about what disks used to be labeled can't always be trusted.

The third issue began when one of the primary application servers went offline, leaving half of their cluster machines handling NFS application requests. They began troubleshooting by running diagnostics on hardware and swapping the RAID card with a card from another machine, but the server remained stuck during power cycles. The server was able to boot normally after removing an SSD that would fail and hang the SATA bus. Doug recommended having spare RAID cards and SSDs on hand, as well as having machines available that allow for the swapping of parts.

The fourth issue was the largest and could have resulted in massive data loss—640 TB of primary storage. Doug provided some background on the storage system architecture: four Linux hosts serve GPFS and NFS to clients and communicate with two storage cabinets. The first cabinet has two storage controllers connected to disk shelves. The second cabinet has shelves that connect to their corresponding shelves in the first cabinet. If a storage controller fails, half of the paths to storage are lost. A controller failed and the vendor shipped a replacement. Shortly after the first controller was swapped out, the second controller failed, and they were told by the vendor to power cycle the two

storage shelves. When that action went awry, Doug performed an emergency shutdown of the GPFS nodes to preserve the integrity of the file system. The system was brought back up and the I/O card in the storage shelf had to be replaced. Things were stable, but broken disks needed repair. LUNs that had journals with information on how to rebuild disks were rebuilt first, and the vendor had Doug and his team perform some undocumented methods for fixing disks lost in the RAID-6 stripes.

Doug presented some meta-ponderables, things to consider based on all of the issues his team dealt with. How much information should be communicated to management during a near-disaster? Can your tape data be restored easily? Doug asked the audience how many people had tested their backups and about 12 individuals raised their hands. He added that squirrels tend to be responsible for many power outages. Doug jokingly said that squirrels were the worst natural disaster in IT history.

Closing Plenary Session *15 Years of DevOps*

Geoff Halprin, The SysAdmin Group

Geoff Halprin opened his presentation with a two-part thesis: (1) in the next decade, operations in general will look a lot like operations at Google or other major .coms, and (2) software development has changed forever, and system administration must do the same. From this point, Geoff briefly discussed the evolution of software development, starting from the waterfall model. This method of developing software was broken because it made a lot of incorrect assumptions about the development process. The response to the failings of the waterfall method was to build new methodologies, such as extreme programming, and agile development practices that embodied principles like daily collaboration between business people and developers, continuous delivery of valuable software, and using working software as a primary measure of progress.

Geoff then turned to a discussion of the incorrect assumptions made about operations. He emphasized that documents that define the waterfall software development life cycle did not mention operations at all; it was assumed that programmers dealt with operation aspects of the system. The greatest assumption made was that operations was involved with the development team in determining project requirements. DevOps is important because it makes the assumption true. It requires an operations person or team to be involved in the development process.

After a quick apology about the brevity and subjective nature of this part of the talk, Geoff talked about the history of system administration. In the past, system administrators dealt with large systems, UNIX variants and networking variants. A community of programmers, mathematicians, and scientists who found themselves doing administration work came together at LISA conferences and user groups to spread ideas about the

professions. Geoff discussed his attempt at defining the role of a system administrator, the System Administration Body of Knowledge (SA-BOK). He continued his history as he talked about the rise of the Web and the three-tier infrastructure model, the commoditization of hardware, virtualization of servers, and the emergence of the cloud, which involves infrastructure on demand, infrastructure as code and provided APIs, and automation driven by scale.

Geoff said that DevOps is to system administration what Agile is to software development; it integrates development and operations teams so they can collaborate more effectively. DevOps is a culture that encourages teams to learn from one another and make their workflow visible. Developers start writing infrastructure as code and are involved in production support. Operations contributes stories to development and uses Kanban walls to keep the team informed on what tickets are being worked on. Geoff quickly went over some DevOps tools and cloud infrastructure frameworks.

Geoff noted that DevOps isn't a complete model and that not all products or environments will fit with the model. It will take time to learn how to scale the model in traditional enterprise environments. Geoff asked whether there were any audience members who thought DevOps was a great way to run their core systems and there were no hands raised. There are a lot of problems that DevOps does not solve, such as determining service-ability criteria or how to monitor services. It doesn't look at the entire Operations life cycle or teach professionalism or ethics.

Geoff stressed that DevOps is not a new concept. While showing various slides from 1997, he talked about his past practices and tools for configuration and systems management. Later he made the point that DevOps is a continuation of a path that system administration is currently taking. This path leads to software-defined data centers where virtualization is necessary and automation is critical. Cloud standardization is changing and the next generation of cloud frameworks will be more generic. Geoff wrapped up his presentation with a statement to ponder: companies that aren't moving toward a cloud model for IT service delivery are setting themselves up to be outsourced.

Jay told Geoff that he has trouble with how DevOps changes how developers do things, and wanted to know how to keep a system safe from a developer working on systems code. Geoff said to be careful of a false dichotomy, to stop talking about the situation with a "them vs. us" (devs vs. ops) perspective, and to focus on ensuring the integrity of changes to a code repository, irrespective of who makes those changes. Someone commented that Geoff's slides about cloud-based startups were chilling and asked the audience if anyone was involved with a company that was completely cloud-based. A couple of people raised their hands in response. This person noted that cloud-based startups

were a trend in the San Francisco Bay Area. An attendee told Geoff that he worked at an Agile shop for development and was curious about resources for maintaining an Agile workflow for sysadmins. Geoff couldn't suggest any specific resources but discussed different categories of workflow and how to organize those categories using a Kanban wall or ticketing system. Garrett Wollman from MIT said that his organization doesn't produce code, and wanted to know how relevant DevOps is for his environment. Geoff recommended that people in non-enterprise environments (HPC, research, and university) look at the practices that one gets from DevOps and determine which practices apply.

LISA '12: Advanced Topics Workshop

San Diego, CA

December 11, 2012

Advanced Topics

Summarized by Josh Simon (jss@clock.org)

The Advanced Topics Workshop began on Tuesday morning; once again, Adam Moskowitz was our host, moderator, and referee. We started with our usual administrative announcements and the overview of the moderation software for the one new and several long-absent participants. Then, we went around the room and did introductions. In representation, businesses (including consultants) outnumbered universities by more than 3 to 1 (down from 4.5 to 1 last year); over the course of the day, the room included 11 LISA program chairs (past, present, and future, up from 6 last year). The Workshop is now old enough to vote (this is the 18th ATW).

For the first topic we discussed collaboration. One attendee works in a widely distributed environment and wondered how others were doing environment-wide collaboration. Some of the technical answers included a site-wide cross-team chat service (both textual and video), mailing lists, a wiki, discussion forums, Google Docs, and a centralized place to store scripts with revision control, as well as the telephone for remote users. Social answers included using Agile methodologies (especially the standing meeting where everyone quickly reports what they did yesterday, what they're working on, and what's blocked on someone or something else). Other techniques included going to lunch as a group (if you're sufficiently close-by geographically), and recognizing success so people feel appreciated for their contributions. Several found that providing these tools and services centrally with some kind of branding was helpful. Some of the challenges included getting sysadmins to work with the security and other teams due to team siloing, showing management that there's actually value in collaboration, and motivation; getting individuals to want to collaborate can be hard, and seeing collaboration as work as opposed to socializing can be tricky. Some environments require the silos, such as governmental or military sites where, by policy, the classified and non-classified sides cannot easily talk to each other. It was also noted that collaboration, especially in environments where it's new, is a cultural shift. Changes should be small and incremental, with the benefits clearly visible, to break down the perceived barriers between teams. Getting management buy-in is essential for collaboration, especially across teams or geographies, to work.

After the morning break we had our first lightning round, asking how people stay positive given the stresses of the job. Answers

included being selective about which battles to fight, biking to work or otherwise exercising, doing approved non-work things during the day (such as attending a physics lecture), eating regularly (don't skip lunch), finding things to be satisfied with or about, job hunting, keeping work at work, liking who you work with, making small changes to foster excellence, playing video games, putting things in perspective (one said he's paid "a foolish amount of money to code interesting things from home"), realizing what may be causing the problem is not within one's control or responsibility, remembering the long-term company direction, spending time on non-work activities (such as grandchildren, hobbies), talking to users about their research areas, volunteering to use one's work skills for social justice organizations, working from home, working on cool or exciting or new things, working on multiple projects, and working on projects that are rewarding. One noted that happiness is contagious; another noted that in almost all cases one's coworkers and customers aren't malicious or dumb, but that they, like you, are trying to accomplish something for the good of the company or institution.

Configuration Management

Our next topic was configuration management (CM) in general. CM is effectively a solved problem for systems, but what about for applications? One site is working on controlling complexity, describing complex setups in a human-readable, human-manageable way. Several aren't in this space yet but realize they need to be. One noted that CM is good for hardware but not necessarily for applications or services; we have to deal with computational ontologies. The game is harder when you're dealing with internal guts of applications, and managing failure states on clusters makes it an immensely hard problem. There's no good answer yet. One person noted that the current tools work for most things if you'll run those tools as root or the Administrator, but they have issues running something untrusted as the superuser. A lot of the CM tools are good skeletons but cross-component complexity of simple components is where it starts to break down (things become too complex and too fragile); you need to avoid making simple concepts complex. Unfortunately, many practical problems aren't solved in the CM space (and are barely solved in the programming language space). One noted that none of the tools track who does what where; there's no audit trail to speak of.

Someone noted that today's tools don't solve all of today's problems. If you need to solve the problem today you can use the existing tools and processes to get close enough. If you don't need to solve the problem today, you can spend more time writing the tools to move closer to the ideal. The question becomes if CM is the right model for application management. We have to achieve a balance; people need to understand they're making a choice to reach that balance. Tradeoffs are what we do all of the time; the problem is increased complexity, as we're trying to do more stuff

with the existing tools. Can we simplify? Are we doing too much? Making assumptions? Taking a step back to figure out how (and why) we do things can be useful. Revision control can be done outside the tool. “Run As” is a missing feature, and even if it exists, the communications reporting piece fails.

Several people wondered whether using CM tools to manage applications is the wrong problem. In fact, one argued that the discussion is ill-aimed and ill-conceived, conflating several topics. People want control; the tools are more for the relatively static bits of CM, and it’s being conflated with application deployment, application control, and application rollout, as well as SLA, monitoring, and health at the level above that. The bottom line is that trying to use the same tool and mind-set to control vastly different aspects of the system is not necessarily wise. We need to agree that there are three types of tools and not conflate them; the complexity comes from trying to mash this stuff all together: You don’t control complexity, you manage it.

Women in Computing

Next we discussed women in technology. One of the problems with the Women in Advanced Computing (WiAC) summit is that not many men show up. What do the men present think? Some are nervous to show up and speak in that environment because they’re worried that what they say might be misperceived despite what they try to champion and do. One question is how does someone talk about and suggest solutions to the problem without coming across as sexist (or, when relevant, racist, etc.)? A politically incorrect observation: of the female engineers one speaker has worked with, many weren’t particularly good; another had the opposite experience.

Another person noted that it varies by discipline; he’s seen women make inroads in the developer space, but engineering and chip fabrication is primarily male. Someone asked whether “diversity” instead of “women” would make it different. How do we encourage people outside the audience to come?

Several people noted they aren’t getting (even unqualified) women applicants. In 11 years, one has had all of three female candidates to interview, and another has similarly had very few female applicants. There’s a deep-seated cultural problem, especially with how girls are pushed away from math and science in general. There are conversations in progress; it makes sense and is important to observe before entering that discussion. There are institutions that reached out to train more women. Don’t just wait for women to apply, but go out and encourage them to. Recruiting for all sorts is important; diverse groups tend to come up with better, broader, and more useful ideas. If we valued it, we’d go out and get it. Someone went to a Grace Hopper conference where he was one of two or three men in a crowd of several hundred women, which was a visceral wake-up call as to how people who aren’t white men can feel every day. Respect

the norms of behavior and treat everyone professionally and courteously.

We also need to make sure everyone gets a chance to speak; not everyone is aggressive enough to fight to do so. Others agreed; one applicant left IT for a while to start a catering business, but because of the gap she’s having trouble getting back into the field.

“Make it happen” has a high risk. Some women think it’s a higher risk to be the only woman (or one of very few) in the team, group, or company. One woman present has been told that she should negotiate a higher salary because of the higher risk. There are studies all over that show—in business, engineering, grants, etc.—given identical resumes with gender-relevant or culturally identifiable names, the man’s is rated higher. Consider doing away with names in the candidate process.

Software-Defined Networks

Our next topic was software-defined networks (SDN). OpenFlow is a more-sophisticated-than-Infiniband way to take a switch and put whatever routing intelligence is on it by letting the switch talk to an off-system controller. OpenFlow gives you a way to write network control/routing software as a service on a UNIX box instead of adhering to on-board rules. This allows you to put a cheaper switch in the rack and “smear” traffic across multiple sites. For example, one person has a multiple terabit pipe in his WAN, made up of hundreds of 10 GB pipes, and he can run long-haul links for IP traffic at 95% utilization 24x7 instead of at 50% utilization. Others have begun looking into this, and find it interesting in and useful for building private cloud systems. It’s actually not new; HPC interconnects have been doing this kind of thing for ages. SDN is broader than OpenFlow; the concept is “treat the network as building blocks” with more direct and more granular control than routing protocols. The protocols influence traffic routing and shaping; SDN comprises rules, not just influence. It was noted that the standards are incomplete; Juniper and Brocade gear can do it, and it’s possible to do it with white-boxes and write-your-own firmware; QoS is in some versions but not from all vendors. For some, latency is very, very important and they need to know the latency penalty for OpenFlow and deep packet inspection (which can be hard and expensive).

Careers and Life

After the lunch break, we had another lightning round, asking about new-to-us tools we’ve discovered in the past year. Answers included Agile methodologies, Go, Google applications, having a team of minions to do one’s billing, iPython Notebook for research labs, Jenkins, logstash, Mobi battery pack for the iPhone, mosh (ssh over slow or inconsistent connections), multibeast, S3, SoundHound, tablets, using Evernote for collaboration, using git for network management, using spreadsheets for small group project management, and vagrant (to automate setting up virtual machines).

Next we discussed the aptly-named “career-type stuff.” One person mentioned his marriage failing in part due to the stress and focus of being a sysadmin and tipping his work/life balance too far onto the work end of the spectrum. He asked what do you do to leave work at work? Answers included allowing things to fail, changing one’s reporting chain (“firing the boss”), delegating tasks to others (at both home and work such as TAs and graduate students in the educational space and a cleaning company at home), empowering one’s employees to make decisions (and backing them up), having alternate work schedules (such as nine hours per day but every other Friday off), keeping Mondays and Fridays as meeting-free days, learning to say No (or at least asking how critical something is so you can level-set priorities and expectations), letting the awful processes tie management up in their own red tape, listening to audio books, meditation as part of a martial arts regimen, not logging into work email from home at all, not trying to save everyone, practicing for choir, putting recurring meetings on your calendar (named “Sleep” or “Lunch”), realizing that “good enough” often is, replying to email only if there’s a reason to do so, reserving certain times of day for family, taking public transportation that requires keeping a schedule, the discipline and focus inherent in exercise in general, using the security policies as an excuse not to VPN in, and working from home (and having a dedicated space for doing so). Whatever you do, take small steps in the process to increase the chances of success. One person noted that the vacation culture in the US is “a day at a time and stay reachable” as opposed to Europe where it’s “a couple of weeks or more and not necessarily reachable.”

Someone lets failures occur by telling the appropriate audience, “This is going to fail for these specific reasons” and then shutting up. Do it in writing so you can refer back to it after the fact. This approach does have pitfalls; if you let too much fail then you’re the incompetent one. Another reminded us that there’s a difference between “letting things fail because it’s not your job” and “letting things fail because you’re overworked.” He lets the different work-providers fight it out and decide what gets priority.

That segued to a question about career paths. One person has no promotion path other than management (which he’s hated in the past) or technical fellow (which requires him to stop being good at his current job). He asked how to deal with a pinnacle career, when all the paperwork asks, “What’s your next job?” and he has no answer for that. Advice included becoming a consultant at a different company, creating a new job to exhibit progress and using it as a goal, leaving the employer when there’s nowhere else to go, and picking things you want to do and making work provide time to build those skills.

IPv6

Our next discussion was on IPv6. One place is running out of their assigned IPv4 space. Their management doesn’t see the

issue but does see the cost of moving to IPv6 so they don’t want to do it. Tens of thousands of homegrown applications (mostly related to money) mean they don’t want to mess with it. Getting the impetus to move forward (even on small things or dual-stack) is tricky. Someone asked if anyone has experience doing this in a commercial environment. One person suspects that there’s a middle step between “working in IPv4” and “working in IPv6” since IPv6 isn’t entirely ready. Using 6-to-4 NAT may be required and it may suck for the people using it until they get around to the recoding. Someone else had a discussion at the bar last night; you can run IPv6 in parallel with your IPv4 environment, and he’s learning about IPv6 by using it at home. Another thinks of it as another Y2K problem: there will be a crisis point down the line and people will panic and deal with it then. Yet another is seeing that interest in IPv6 is finally growing beyond a small core community that’s been looking at it for a while; he can’t go to it yet due to missing vendor support. It was suggested that instead of planning a whole-company conversion, get started by just picking something and doing it, such as building a new datacenter. It was noted that this isn’t IPv6 specific; you always need a business case as to why spending the effort now is worth it.

Cloud

Clouds were up next. One person has rolled out two internal clouds. The main issue is for things that need stable naming of some kind; DNS and Kerberos are both in the cloud in his environment now. Naming and external configuration were concerns. Another is finding unexpected resistance to people using clouds; in particular, when you buy a computing resource there’s something to touch (“kick the tires”). Capital funding requires something tangible. There are serious security implications with the data entailed in the systems; you have to be careful with the appropriate controls (in the US, these include FERPA, FISMA, and HIPAA). Someone noted “cloud computing” is the latest CIO “gotta-do” buzzword, and it’s not the right tool for all jobs. For putting up public-facing Web sites, it’s great; for encumbered data, it’s not so great. It’s also expensive; it’s cheaper to build your own DC if you know what you want and need. Data locality is the single biggest issue facing some people. People are virtualizing because their configuration management systems don’t work.

One environment is using the cloud to compartmentalize to keep student personally identifiable information (PII) hidden from their corporate overlords, and spinning up infrastructure is easy. Another has been doing proof-of-concepts of new large infrastructure before buying the hardware. DevOps in the cloud is a win; do it virtual first and then do it on real hardware; however, some environments can’t put anything classified in the public cloud. Bring your own device (BYOD) and having access to

all the data they want can be problematic. Their executives are demanding the ability to do this despite security policies.

One person argued that GRID computing failed because you couldn't handle different parallel versions of the same software. Configuration isolation is a feature, not a bug. They're trying to adapt an HPC machine for a workload and are seeing little overhead and better I/O than expected. Another noted that leveraging technology is good; outsourcing and the subcontractors and so on puts data in interesting problems: once the data's out of house, it's not private regardless of the SLA and contract. Virtualization is a valid way to compartmentalize applications. Someone at a software development house noted that cloud computing lets them test at scale without paying for the hardware.

Next we had a few quick polls: Of the 28 of us in the room, only 1 does not carry a cell phone; 4 have dumb phones, 12 have Android devices, 9 have iPhones, and 5 have BlackBerrys. (Yes, some people carry more than one device.) Fourteen people in the room have some form of eating plan (and 4 blame work for the reason they go off it). Twelve of us are in new jobs since last year's workshop, about half of those to different companies as opposed to different roles within the same company. One even has a new job since the start of the session! Five people in the room are actively looking for a new job.

We had a brief meta-discussion about the workshop. Some people think some topics drag a bit (and someone jokingly suggested we bring in a gong). Some want to blacklist some specific topics. Most like the soft-versus-hard topic balance but would like to alternate between them more as opposed to a long string of only one type. It was reiterated that there's intentionally no attempt for us to accomplish something specific, and the participants agree that's still desired.

During the afternoon break, we took a quick unofficial poll: Of the 29 systems in use, there was 1 tablet, 5 PCs running either Windows or Linux, and 23 Macs.

Wishful Thinking

We kicked off the final block of the workshop with a quick topic: "If I were king, I'd wave my magic wand and...." Leaving aside that wizards or witches have magic wands and kings have scepters, answers included bringing all of Human Resources into a room and setting it on fire, explaining to the user support team that their job is to support users, firing the business unit whose profit model is effectively fraud, fixing attitudes about documentation so that getting it done is more important, fixing the research funding model so principal investigators can share resources and funding, forcing everyone to and everything into source control, getting rid of half of one's leadership chain, getting rid of legacy garbage (both things and processes), having management better shelter workers from administrative garbage, hiring more people and training them with a deep

understanding of the product, hiring quality consultants (not just the lowest bidder), killing off incompetents who refuse to learn, level-setting processes to the right amount, moving away from a forest of symlinks, putting policies and procedures together and moving away from "talk to so-and-so," rebuilding the team from the ground up, replacing all of internal IT management, staffing projects adequately, stopping others from using my team to keep their team from failing, taking more time off, tracking bugs and issues better using a good single sign-on and source-of-truth for credentials (one place has four such systems), using the inventory system correctly, and working closer to home.

BYOD

Our next topic was Bring Your Own Device (BYOD) and the security and policy issues thereof. One environment has strict policies about work versus private space and that never the twain shall meet. Executives are moving away from that; the CEO wants an iPhone not a BlackBerry, and now they want personal devices accessing the work space. They have a pilot program coming up, with stipulations that the company can zap the device and use the users' own data plans. There will be no new BlackBerrys assigned or purchased after 1/1/2013. A specific college has no security restrictions like them, so BYOD has happened forever there. They treat U-purchased devices as their own. They explain to the important people that they need more than one-size-fits-all enterprise solutions for people who don't act like they're in an enterprise. Another environment approached it organically. BlackBerry allows remote-wipe (as does Android now), but iPhone didn't. There's a guest wireless network for personal devices, and they have firewalls that can do policies for threat protection, antispy, antivirus, and so on. They're experimenting with VPNs.

One concern with BYOD is data leakage protection; client confidential information can't go on a personal device. Someone noted that Financial shied away from BYOD because of the policies, but then they moved away from that position because the risk assessment said it wasn't adding anything to avoid BYOD. Someone else noted BYOD is not keeping up with users' needs. You don't want to demotivate people to stop doing their job. Perimeters don't work; you have to secure the machines, data, services, and so on, and decide what's important to you. One environment implemented an iPad network (not on the internal network); the corporate IT people type in the password for you. Another used to have a network drop per device and hardened hosts, but they're moving away from "everything as a bastion" toward "Infrastructure as a Service." There are liability issues with BYOD (such as hardware damage). Their policy is to give people the tools they need to do the job. Users want support of company-purchased devices at home.

It was noted that it costs a horrific amount for big corporations to support phones, such that two-year support means they can give you a new phone every year or two. Some vendors support multiple personalities on smartphones (e.g., John Smith the Person versus John Smith the Employee).

“Give them what they want” is hard to do because marketing firms tell users what they want. This happens for applications as well as devices. It’s all about the data that needs to be protected and secured. Some advocate for their users, but you can’t unconditionally give them what they think they want, such as applications that aren’t licensed for commercial use. Users don’t understand what is or isn’t supported.

Incidents

Next we discussed incident management. A few days before the workshop, a major public service was down for 18 minutes due to a load balancer problem. The problem was fixed quickly, but the people fixing it didn’t communicate well with the rest of the company. This brought up the concept of incident response: how can they better organize and prepare to respond to incidents (outages, security events, and so on), and how do you test for readiness?

Someone brought up the Game Day exercises. One place had a datacenter person who was too good, so they arranged to have him arrested and taken to jail so others could have the joy of dealing with the issue. Another referenced Chaos Monkey: Do something chaotic during off-peak business hours to see what happens and how to fix it. Some places do postmortem writeups very well; they’re blameless and they focus on what happened to prevent it from recurring.

As part of this discussion we had another lightning round: if you have outages and aren’t doing anything like this, why not? Answers included being in an environment that won’t test and has no failback, doing so ad hoc, having enough real outages not to need to test fake ones, having planned outage testing for new but not legacy systems, having server-level but not datacenter-level disaster recovery planning and testing, limiting testing to nonproduction or dark datacenters, not being involved in a planning or response role, not being pressured to get back up fast and/or being sufficiently resilient for the SLA, small outages are frequently unnoticeable, and the SLA requiring only three 9s or five 8s or even three 6s of uptime. Some places do incident management postmortems, fail traffic to a secondary datacenter while they push code to the primary datacenter, or have business continuity weekends where they take down a regional datacenter. Others simply have no plans and no staff or capability to test. One person had an instance where an application died every so often and restarted itself, for four months, without user-visible notification, so it’s possible to do this kind of work too well.

Finally, we closed out again with a lightning round asking what’s coming up over the next year. Answers included allocating laptops instead of desktops, building out a home theater system, building the latest version of their supercomputer to handle 10 times the data, chairing the LISA ‘13 conference, consolidating DNS and DHCP services, coping with the next round of bigger disks, developing a standard API for an internal cloud, doing more business analytics, finding a new job, growing technical communities, implementing and migrating to a new CM system, implementing company-wide incident management, implementing more cross-silo services, implementing services for a zero-trust network, integrating the network with their new corporate parents, isolating stuff, launching new products, migrating to new versions of vendor software as old ones go to end of life, moving into the cloud, moving to a new location, relocating a lab, renovating houses, retiring at the end of 2013, revising books, selling a house, splitting a lab into two labs, and updating monitoring and visualization systems.

LISA ’12: Real World Configuration Management Workshop

San Diego, CA
December 9, 2012

Real World Configuration Management

Summarized by Nick Anderson (nick@cmdln.org) and Aleksey Tsalolikhin (aleksey@verticalsysadmin.com)

The LISA 2012 Real World Configuration Management Workshop was chaired by Narayan Desai (Argonne National Laboratories), Cory Lueninghoener (Los Alamos National Laboratory), and Kent Skaar (VMware). Thirty-seven people attended.

Narayan, Kent, and Cory opened the day with introductions. The 37 attendees introduced themselves, each sharing a pain point with the group. Introductions consumed the entire morning but there was consensus that it was beneficial and worthwhile. Culture and Secrets dominated this discussion.

Culturally, acceptance of configuration management and automation tools is still not universal. Common roadblocks include perceived time constraints (takes too long and or too difficult to use the tool), and low business value by small or isolated groups. Fear and distrust of automation was also noted as an acceptance issue. Although most people were already using configuration management tools, it is not uncommon for tools to be bypassed

intentionally. These manual changes are not always ported back into the configuration policy. Several people indicated they were not running their CM tools continuously, and for those who do, running in noop/dry-run/warn_only modes is not unusual. For configuration policy activation to be an infrequent and even manual action is relatively common.

Multiple attendees cited managing security domains as an issue. Questions were raised including how to manage “many hands” of different skill and trust levels, how to divide policy and grant access only to authorized individuals, how to manage secrets in policy, how to separate data from policy, and how to share common policy between disparate environments (typically government). To manage people, some are using ACLs (Access Control Lists) provided by a version control system, others are using approver-based gating that encourages peer review of policy. Many people are tying custom systems together or have custom tooling built around their configuration management systems. Separating data from policy varies widely. Sneakernet is still required to deal with disparate environments. Many attendees said that a unified view, or single source of information would be preferable to the many sources currently used.

Orchestration and performance concerns about CM tools and their ability to scale and manage complexity rounded out other common pain points. Orchestrating policy deployment is burdensome. Attendees voiced a desire for staged and slow controlled dispersion of policy, and there was discussion about the importance of promoting policy based on its own stability and having it roll out automatically. Integrating with other tools, namely monitoring systems, is difficult or at the least, not straightforward. It is not uncommon to perform management of monitoring and inventory systems separately from a CM tool. Mark Burgess pointed out that having CM and Monitoring separate is legacy thinking as modern “continuous maintenance” CM systems are continuously monitoring the systems on which they run. OS patch management continues to be painful. Some sites don’t patch at all, or just roll out new OS images periodically. Orchestration of larger systems brought discussion of performance overhead from configuration management tools as well as talk of how different tools themselves manage scalability (push vs pull, centralized vs decentralized).

After returning from lunch and wrapping up introductions, attendees were asked what they would fix if they could snap their fingers and have it done. The list included: complete buy-in from customers and IT counterparts, more separation of policy and data (with easier policy metadata extraction), better tooling around version control to simplify workflows, easy discovery of possibly conflicting policies, and scope of impact of a policy change. Automatic risk-aware policy deployment over a period of days with workflows that include teaching coworkers what is happening was a common desire. Mark Burgess wished for a

shift from deployment steam roller mentality to comprehensive design thinking as well as more reuse of existing parts instead of reinventing the wheel.

We kept notes on which configuration management tools attendees are using. People are primarily using one of the major frameworks (BCFG 2, CFEngine 2, CFEngine 3, Chef, Puppet), but homegrown systems are not uncommon, and new ones are still being built (e.g., Ansible, Cdist, and SaltStack).

During the final segment of the workshop, informal statistics were collected. A show of hands using the “never have I ever” (even a little bit counts) model was used to survey the attendees. Most startlingly, there were only two attendees with official QA processes for their configuration management. The weirdest things under configuration management included laser cutters (inside a 3D printer), Android phones, Raspberry Pi, routers, switches, robots, and a QNAP storage appliance.

The workshop concluded with discussion of the future of configuration management. The main concerns were enabling system orchestration (state transitions with dependencies—for example, an admin could tell systems X to transition from state A to state B, but to wait until systems Y reach a certain state first); separating development and production (Paul Krizak explained how he builds a system of dev VMs simulating production, with Jenkins used to test the change); and network configuration management (Tom Limoncelli spoke about OpenFlow, which is a way to centrally control routers).



27th Large Installation System Administration Conference (LISA '13)

Sponsored by USENIX in cooperation with LOPSA

November 3–8, 2013, Washington, D.C.

Important Dates

Extended abstracts, papers, experience reports, and proposals for talks, workshops, and tutorials due: *April 30, 2013, 11:59 p.m. PDT*
 Paper and experience report authors' review and response period: *June 3–14, 2013*

Notification to talk submitters: *June 3–14, 2013*

Notification to paper and report submitters: *June 24, 2013*

Final reports and papers due: *August 20, 2013*

Poster proposals due: *September 24, 2013*

Notification to poster submissions: *October 8, 2013*

Conference Organizers

Program Co-Chairs

Narayan Desai, *Argonne National Laboratory*
 Kent Skaar, *VMware, Inc.*

Program Committee

Patrick Cable, *MIT Lincoln Laboratory*
 Mike Ciavarella, *Coffee Bean Software Pty Ltd*
 Joe Gross, *Dropbox*
 Andrew Hume, *AT&T Labs—Research*
 Paul Krizak, *Qualcomm*
 Adam Leff, *WebMD*
 John Looney, *Google, Inc.*
 Andrew Lusk, *Amazon Inc.*
 Chris McEniry, *Sony*
 Tim Nelson, *Worcester Polytechnic Institute*
 Marco Nicosia, *Moraga Systems LLC*
 Adam Oliner, *University of California, Berkeley*
 Carolyn Rowland
 Dan Russel, *TED Talks*
 Adele Shakal, *Metacloud*
 Avleen Vig, *Etsy, Inc.*

Invited Talks Coordinators

Nicole Forsgren Velasquez, *Utah State University*
 Cory Lueninghoener, *Los Alamos National Laboratory*

Lightning Talks Coordinator

Lee Damon, *University of Washington*

Workshops Coordinator

Kyrre Begnum, *Norwegian System Architects*

Guru Is In Coordinator

Chris St. Pierre, *Wireless Generation*

Poster Session Coordinator

Marc Chiarini, *Harvard University*

USENIX Board Liaisons

David N. Blank-Edelman, *Northeastern University*
 Carolyn Rowland

Steering Committee

Paul Anderson, *University of Edinburgh*
 David N. Blank-Edelman, *Northeastern University*
 Mark Burgess, *CFEngine*
 Alva Couch, *Tufts University*
 Anne Dickison, *USENIX Association*
 Eileen Frisch, *Exponential Consulting*
 Doug Hughes, *D. E. Shaw Research, LLC*
 William LeFebvre, *CSE*
 Thomas A. Limoncelli, *Google, Inc.*
 Adam Moskowitz
 Mario Obejas, *Raytheon*
 Carolyn Rowland
 Rudi van Drunen, *Competa IT and Xlexit Technology, The Netherlands*

Education Director

Daniel V. Klein, *USENIX Association*

Overview

The annual USENIX LISA conference is the meeting place for professionals who make computing work and understand how it fails. Every year, LISA attracts system administrators, architects, site reliability engineers, software engineers, and researchers, from organizations ranging from small IT-shops to the largest computing infrastructures in the world. Attendees learn from industry experts about emerging trends and topics in software, hardware, and operations strategy and methodologies. The LISA community is very diverse; attendees have responsibilities in a range of areas, including system and site reliability, security, cloud, high performance computing (HPC), Web, networking, and storage administration.

At LISA, systems theory meets operational practice. This is an ideal environment for both researchers and practitioners to identify the key operational and system problems being faced by industry today, and to form partnerships within the community. Presentation at LISA is a path to real-world impact for leading research. Submissions in research areas such as cloud computing, software-defined networking, DevOps, large-scale computing, distributed systems, security, visualization, and management methods are encouraged. Alongside the LISA refereed papers track, posters and work-in-progress sessions provide a valuable avenue for early researcher feedback. USENIX supports open access to research via its conference publications and also awards grants to enable participation by students.

The LISA program is a rich mix of technical talks and training, panel discussions of important topics, ask-the-experts Guru sessions, and presentations by people who make things work—just like you. In addition, attendees can network with experts in a variety of fields. These relationships provide great value to organizations as they encounter subtle technical issues. The expertise gained by LISA attendees has

a long-term impact on their careers and organizations. These factors make LISA the premiere event for our community.

New in 2013! LISA Labs: New this year is a “hack space” available for informal mini-presentations by seasoned professionals, participation in live experiments, tutoring, and mentoring. This will bring a hands-on component to the conference, where attendees can investigate new technologies, apply what they have learned, and interact with other attendees in a participatory technical setting. Send ideas to lisa13labs@usenix.org.

We will provide early feedback on ideas for papers, talks, or conference activities. Beat the deadlines—email lisa13chairs@usenix.org now!

Get Involved!

We welcome participants willing to share their research and experiences. This is your conference and an opportunity to give back to the community.

First, one of the most important ways to participate in the conference is simply to attend it. Find out about the many important reasons organizations have for sending their employees to LISA at www.usenix.org/conference/lisa13/why-organizations.

The technical program also seeks submissions in the following areas:

- **Refereed Papers:** These are written papers, 8 to 18 pages long, describing work that advances the science or practice of system administration. These papers are held to high research standards and are evaluated based on their conceptual development, contribution to the field, or extension of previous work to new contexts. If accepted, the paper will be published in the proceedings and the author(s) will give a 25-minute presentation followed by a 5-minute Q&A session. These are original works which must not be submitted concurrently to another publication venue in whole or in part.
- **Practice and Experience Reports:** Bring your favorite system administration story to LISA. These can include successes as well as failures, as long as there are useful lessons imparted to the audience. Initial submissions can be in the form of a 4–10 page report or a short (5–7 minute) video submission with slides. Your proposal should include a clear description of the problem you are addressing, its relevance, the approaches and trade-offs made, and the lessons learned. Please note that we are including video submissions to make it easier to produce a PER proposal without the upfront effort of writing a report. Accepted video proposals still require a final written report for the conference. If accepted, the author(s) will give a 20-minute presentation followed by a 10-minute Q&A session.
- **Talks:** Talks are 45- or 90-minute presentations by experts on a single topic of interest to system administrators. We are seeking suggestions from people who wish to give talks or to propose topics. Talks may focus on the emerging technologies or may be retrospective, be serious or funny, cover a spectrum of related issues or dive deeply into one specific topic. We also accept proposals for panel discussions, especially when accompanied by a tentative slate of panelists. Send ideas to lisa13it@usenix.org.
- **The Guru Is In Sessions:** Q&A with an expert! Are you a guru? These sessions are a chance to share your expertise with your fellow system administrators. For the audience, these are a chance to get your questions on a specific topic or technology answered by an acknowledged expert. Submissions are in the form of a half-page description of the topic. Send ideas to lisa13guru@usenix.org.

- **Lightning Talks:** Talk about a recent success, energize people about a pressing issue, ask a question, start a conversation! Lightning talks are 5-minute opportunities to get up and talk about what’s on your mind. You can give several lightning talks if you have more than one topic.
- **Poster Session:** This is your chance to share an idea that could turn into something more formal at next year’s conference. Posters are a good way to get feedback on research that may not be ready for formal publication. Submissions are in the form of a 1-page abstract.

In addition, LISA welcomes proposals for the following:

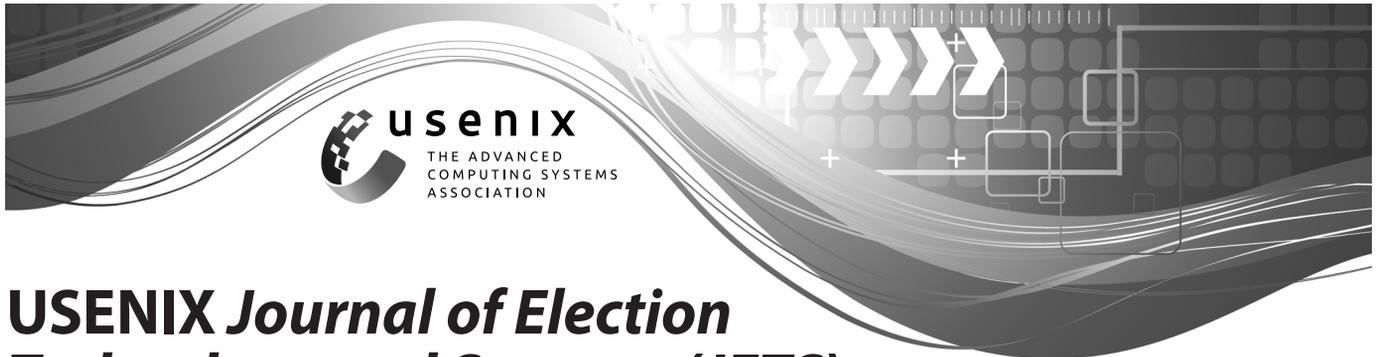
- **Workshops:** Workshops are half-day or full-day sessions for small groups (typically no more than 30 people) to share ideas and knowledge. Workshops are intended to be participatory, not instructional, and familiarity with the specific topic/area is expected of the attendees. Proposals are in the form of a 1-page description. Send ideas to lisa13workshops@usenix.org.
- **Training Program:** Tutorials are also half-day or full-day sessions but, unlike workshops, tutorials are generally intended for an instructor to share knowledge, not to be open discussions. We welcome (and encourage) suggestions or requests for new classes from anyone! Email tutorials@usenix.org with suggestions/requests or find out how to submit a proposal here.
- **Birds-of-a-Feather Sessions (BoFs):** Birds-of-a-Feather sessions are informal gatherings held in the evenings. BoF groups range from users of particular software packages or products, through those interested in discussing current problems or issues, to people interested in a particular aspect of computing. Time slots are assigned on a first-come, first-served basis before and during the conference. See the conference Web site for submitting your BoF topic and time slot.

Submission Instructions

Submissions for the Refereed Papers or Practice and Experience Reports tracks may not be simultaneously submitted to other venues. Writing must be original, not previously published online or otherwise. A major mission of the USENIX Association is to provide for the creation and dissemination of new knowledge. In order to facilitate this process, USENIX allows authors to retain ownership of the copyright to their works. See the USENIX Conference Submissions Policy at www.usenix.org/submissionpolicy for details. Questions? Contact your program chairs, lisa13chairs@usenix.org, or the USENIX office, submissionpolicy@usenix.org.

Check out the full Call for Papers at www.usenix.org/lisa13/cfp.
Join us!





USENIX *Journal of Election Technology and Systems (JETS)*

In conjunction with EVT/WOTE '13

The USENIX *Journal of Election Technology and Systems (JETS)* is a new "hybrid" journal + conference, where papers will have a journal-style reviewing process and online-only publication. Authors of accepted papers will present their work at EVT/WOTE '13, which will be co-located with the 22nd USENIX Security Symposium (USENIX Security '13). EVT/WOTE will take place August 12–13, 2013.

Important Dates

Submissions due: *April 17, 2013, 11:59 p.m. PDT*

Initial notification to authors: *Wednesday, May 22, 2013*

Second-round revisions due: *Friday, June 7, 2013*

Final notification to authors: *Saturday, June 29, 2013*

Final files due: *Tuesday, July 16, 2013*

Subsequently, the Journal will be published at regular intervals—more details available soon. Specifications about this process are addressed in the FAQ at www.usenix.org/jets/faq.

Editorial Board

JETS Editors in Chief

Walter Mebane, *University of Michigan*

Dan S. Wallach, *Rice University*

JETS Editorial Board

TBA

Overview

In a number of countries, votes are counted and transported electronically, but there are numerous practical and policy implications of introducing electronic machines into the voting process. Both voting technology and its regulations are very much in flux, with open concerns including accuracy, reliability, robustness, security, transparency, auditability, equality, privacy, usability, accessibility, cost, and regulation.

USENIX is proud to announce the creation of a new Journal of Election Technology and Systems (*JETS*), which will operate in conjunction with the ongoing USENIX Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '13). If you want your paper to appear at EVT/WOTE, you submit it to *JETS*.

JETS brings together researchers from a variety of disciplines, ranging from computer science and human-computer interaction experts through political scientists, statisticians, legal and policy experts, election administrators, and voting equipment vendors. *JETS* seeks to publish original research on important problems in all aspects of electronic voting.

JETS is an example of a new trend in academic computer science—a hybrid of a conference and a journal. All papers will have a two-round review process (longer than a conference, shorter than a journal). After the first round, authors will get anonymous feedback from the editors. Their manuscripts may be accepted without

changes, accepted with minor required changes, rejected with major changes recommended, or simply rejected. Accepted papers will have a brief window to make any necessary changes and will then be subject to an additional round of review. By having quarterly submission deadlines with rapid reviewing, *JETS* promises to offer authors a rapid and predictable process. By having online open-access dissemination, *JETS* promises timely, free access to readers worldwide.

JETS authors pay nothing to submit manuscripts and *JETS* readers pay nothing to read accepted papers. Authors of accepted *JETS* papers will be invited to present their work at the USENIX EVT/WOTE workshop. EVT/WOTE '13 will be a two-day event, Monday, August 12, and Tuesday, August 13, 2013, co-located with the 22nd USENIX Security Symposium in Washington, DC. In addition to *JETS* paper presentations, the workshop may include panel discussions and other events of interest to our attendees. With EVT/WOTE in our nation's capitol, we will have a great agenda. Attendance at the workshop will be open to the public, although talks and refereed paper presentations will be by invitation only. There will be an award for the best paper.

JETS Manuscript Topics

Papers should contain original research in any area related to electronic voting technologies, verifiable elections, and related concerns. Example topics include but are not limited to:

- In-person voting systems
- Remote/Internet voting systems
- Voter registration and authentication systems
- Procedures for ballot and election auditing
- Cryptographic (or non-cryptographic) verifiable election schemes

Example topics include but are not limited to original research on:

- Attacks on existing systems
- Designs of new systems
- Experiences deploying voting systems or conducting elections
- Experiences detecting and recovering from election problems
- Formal or informal security or requirements analysis
- Examination of usability and accessibility issues
- Research on relevant regulations, standards, or laws

Submissions will be judged on originality, relevance, correctness, and clarity.



Submission Instructions

Papers must be received by 11:59 p.m. PDT on Wednesday, April 17, 2013. This is a hard deadline—no extensions will be given. All submissions will be electronic. Submissions should be finished, complete papers (not work in progress) and must be in PDF format. Submit papers using the Web form on the *JETS* Call for Participation Web page, www.usenix.org/jets/cfp. (There will be quarterly submission deadlines for *JETS*, and EVT/WOTE will continue to be an annual workshop. See the FAQ at www.usenix.org/jets/faq for details)

Paper submissions should be at most 16 typeset pages, excluding bibliography and well-marked appendices. Submissions should be in one-column format, using 10-point Times Roman type on 12-point leading, in a text block with 1.5-inch margins on U.S.-style 8.5"x11" paper. There is no limit on the length of appendices, but reviewers are not required to read them. Once accepted, papers must fit in 20 pages—including bibliography and any appendices—in the same format. Authors' names and affiliations should not be included, per the anonymization policy that follows.

Note that you should feel no obligation to use every available page. Shorter papers are encouraged. Because *JETS* accepts manuscripts from academics across many different disciplines, we wish to be flexible about formatting requirements. Generally speaking, we want papers to follow the font size and margins listed above, but beyond that you're welcome to adopt the APA style or whatever other standard your academic area prefers.

Paper submissions must be anonymized: author names and author affiliations must be removed; acknowledgments and other clear markers of affiliation (e.g., "we used data from XXX University") should be removed or rewritten; self-citations should be rewritten to be neutral (e.g., "In previous work, Smith showed...").

Simultaneous submission of the same work to multiple venues, submission of previously published work, or plagiarism constitutes dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may take action against authors who have committed them. See the USENIX Conference Submissions Policy at www.usenix.org/conferences/submissions-policy for details.

Authors uncertain whether their submission meets USENIX's guidelines should contact the editors in chief, jets-chiefs@usenix.org, or the USENIX office, submissionpolicy@usenix.org.

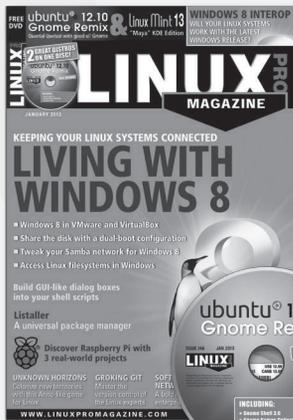
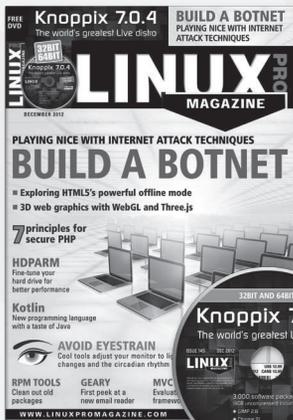
Papers accompanied by nondisclosure agreement forms will not be considered. Accepted submissions will be treated as confidential prior to publication on the *JETS* Web site; rejected submissions will be permanently treated as confidential.

Authors will be notified on the dates described above. Each accepted submission may be assigned a member of the editorial board to act as its shepherd through the preparation of the final paper. The assigned member will act as a conduit for feedback from the full editorial board to the authors.

All *JETS* papers will be disseminated online, free of charge, to all readers. If you need to embargo your final publication for some reason, please notify production@usenix.org.

Questions about submissions may be sent to the journal editors in chief at jets-chiefs@usenix.org.

RISK-FREE TRIAL!



3 issues + 3 DVDs for only \$3

PRACTICAL. PROFESSIONAL. ELEGANT.

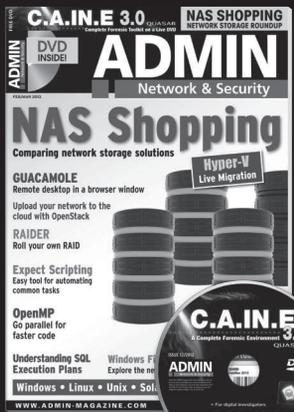
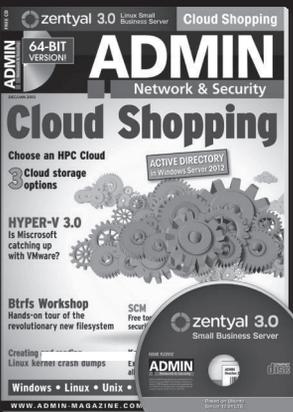
Enjoy a rich blend of tutorials, reviews, international news, and practical solutions for the technical reader.

ORDER YOUR TRIAL NOW!
shop.linuxnewmedia.com

2 ISSUES ONLY \$15⁹⁹

ALSO AVAILABLE

ADMIN: REAL SOLUTIONS FOR REAL NETWORKS



Technical solutions to the real-world problems you face every day.

Learn the latest techniques for better:

- network security
- performance tuning
- system management
- virtualization
- troubleshooting
- cloud computing

on Windows, Linux, Solaris, and popular varieties of Unix.



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

2013 USENIX Federated Conferences Week

June 24–28, 2013 • San Jose, CA

www.usenix.org/conference/fcw13

USENIX ATC '13: 2013 USENIX Annual Technical Conference

ICAC '13: 10th International Conference on Autonomic Computing

HotPar '13: 5th USENIX Workshop on Hot Topics in Parallelism

HotCloud '13: 5th USENIX Workshop on Hot Topics in Cloud Computing

HotStorage '13: 5th USENIX Workshop on Hot Topics in Storage and File Systems

WiAC '13: 2013 USENIX Women in Advanced Computing Summit

HotSWUp '13: 5th Workshop on Hot Topics in Software Upgrades

Feedback Computing '13: 8th International Workshop on Feedback Computing

ESOS '13: 2013 Workshop on Embedded Self-Organizing Systems

UCMS '13: 2013 USENIX Configuration Management Summit

AND MORE!

Registration
opens in April.

Register by the
Early Bird Deadline,
Monday, June 3,
and save.

Discounts
available!

 www.twitter.com/usenix

 www.usenix.org/youtube

 www.usenix.org/gplus

 www.usenix.org/facebook

 www.usenix.org/linkedin

 www.usenix.org/blog

