;login:
FEBRUARY 2014 VOL. 39, NO. 1
usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

File Systems and Sysadmin

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

UPCOMING EVENTS

FAST '14: 12th USENIX Conference on File and Storage Technologies

February 17–20, 2014, Santa Clara, CA, USA
www.usenix.org/fast14

2014 USENIX Research in Linux File and Storage Technologies Summit
In conjunction with FAST '14
February 20, 2014, Mountain View, CA, USA

NSDI '14: 11th USENIX Symposium on Networked Systems Design and Implementation

April 2–4, 2014, Seattle, WA, USA
www.usenix.org/nsdi14

2014 USENIX Federated Conferences Week

June 17–20, 2014, Philadelphia, PA, USA

HotCloud '14: 6th USENIX Workshop on Hot Topics in Cloud Computing
June 17–18, 2014
www.usenix.org/hotcloud14
Submissions due: March 6, 2014

HotStorage '14: 6th USENIX Workshop on Hot Topics in Storage and File Systems
June 17–18, 2014
www.usenix.org/hotstorage14
Submissions due: March 13, 2014

WiAC '14: 2014 USENIX Women in Advanced Computing Summit
June 18, 2014
www.usenix.org/wiac14

ICAC '14: 11th International Conference on Autonomic Computing
June 18–20, 2014
www.usenix.org/icac14
Paper titles and abstracts due: February 26, 2014

USENIX ATC '14: 2014 USENIX Annual Technical Conference
June 19–20, 2014
www.usenix.org/atc14

UCMS '14: 2014: USENIX Configuration Management Summit
June 19, 2014
www.usenix.org/ucms14

URES '14: 2014 USENIX Release Engineering Summit
June 20, 2014
www.usenix.org/ures14

23rd USENIX Security Symposium

August 20–22, 2014, San Diego, CA, USA
www.usenix.org/sec14
Submissions due: February 27, 2014

Workshops Co-located with USENIX Security '14

EVT/WOTE '14: 2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections

USENIX Journal of Election Technology and Systems (JETS)
Published in conjunction with EVT/WOTE
www.usenix.org/jets
Submissions for Volume 2, Issue 3, due: April 8, 2014

HotSec '14: 2014 USENIX Summit on Hot Topics in Security

FOCI '14: 4th USENIX Workshop on Free and Open Communications on the Internet

HealthTech '14: 2014 USENIX Workshop on Health Information Technologies
Safety, Security, Privacy, and Interoperability of Health Information Technologies

CSET '14: 7th Workshop on Cyber Security Experimentation and Test

WOOT '14: 8th USENIX Workshop on Offensive Technologies

OSDI '14: 11th USENIX Symposium on Operating Systems Design and Implementation

October 6–8, 2014, Broomfield, CO, USA
www.usenix.org/osdi14
Abstract registration due: April 24, 2014

Co-located with OSDI '14:

Diversity '14: 2014 Workshop on Diversity in Systems Research

LISA '14: 28th Large Installation System Administration Conference

November 9–14, 2014, Seattle, WA, USA
www.usenix.org/lisa14
Submissions due: April 14, 2014

;login:
FEBRUARY 2014 VOL. 39, NO. 1

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

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.

EDITORIAL

Musings

RIK FARROW

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

I like to attend the SNIA Storage Developer Conference [1], looking for ideas and authors for the February issue of *;login:*. Although the main focus of SDC continues to be Microsoft SMB, I get to learn about other network/distributed file systems while there. Not that SMB 3 is not amazing. If you are still using older versions of Microsoft's networked file system, you are living in the dark ages. But I was after other prey, topics of greater interest to file system and storage researchers.

I've long been interested in Parallel NFS (pNFS), the long-awaited set of standards that allow the venerable NFS to work over a collection of storage servers. So I attended the pNFS BoF one evening at SDC, only to find that I was one of only five people there. I did say this was mainly an SMB conference.

Fortunately for me, the people there, Brent Welch (Google), Matt Benjamin, Adam Emerson, and Marcus Watts (all with CohortFS) were willing to talk about distributed file systems. When I asked the CohortFS people what they were working on, the talk turned to object storage systems.

File Systems

I have a confession to make: I had no idea what a file system was by the end of the operating systems class I took in 1978. The class textbook covered IBM only, and years later, the class' professor heavily criticized my chapter on file systems for my system administration book. That was certainly ironic, as he had totally failed to cover the topic. Then again, perhaps that was why he had commented that the material was "unnecessary."

In the 1970s, file systems were a bit alien. Really. People in the operating system course labs carried cases of punch cards. These weren't their backups: they were the storage system. The ability to store files online was just beginning at the University of Maryland. So when I was tasked with writing a file system, I was stumped. I knew I had to build data structures that held lists of blocks, but beyond that, I had no idea what to do. I had never knowingly interacted with a file system.

Three years later, I had my own, home-built, microcomputer system, which ran CP/M on floppy disks, using a flat file system. My computer had built-in 5¼-inch drives, but most of my customers used 8-inch floppies (with a capacity of more than 200 kilobytes!). I had both an external 8-inch drive and a Morrow DJIO card, and because I had written the manual for the DJIO, I thought I could patch in access to the 8-inch drive. What I did was build a device driver and CP/M file system, all in two pages of C code. Things were a lot simpler then. And, fortunately for me, there were other sources of knowledge about file systems (magazine articles in *Byte*!) than the professor who had ignored this topic.

Object Stores

As you will learn when you read the object storage article in this issue, object stores have been around since work done by Garth Gibson in 1990. His initial research into network attached storage devices (NASD) grew into Panasas, and later into object storage standards. Gibson and

his cohorts had built a POSIX-compatible interface that allowed access to distributed object storage as if they were a more traditional file system. The interface hid the behind-the-scenes work of distributing data among different devices to produce fast, reliable, and scalable storage systems for super computers.

Although the Gibson work was one way of presenting an object store, another one showed up many years later, with Amazon S3. Again, the interface hides all of the behind-the-scene implementation details. But instead of a POSIX interface, Amazon chose an almost flat system of buckets and keys that refer to objects up to five terabytes in size. Each object can be represented, and fetched, via a URL that names the bucket and key. In general, S3 is clearly a long way away from the POSIX interface, with pathnames, permissions, reads, writes, and seeks.

And yet, is this really so different? I'm reminded of the impossible (it seemed at the time) task of organizing the blocks on a PDP 11/45 in my operating system class. I needed to create collections of blocks, organized using some higher layer of software, into something coherent, so I could eventually store and execute programs using my primitive OS.

Today, we take file systems for granted, and typically use whatever the OS we have installed has provided for us. Even the Hadoop Distributed File System (HDFS) uses the underlying file system for its object store, and the newer version now presents a POSIX-style (NFSv3) interface.

There certainly are places where object stores make a lot of sense. Object stores can be used for storing large amounts of data, such as photos or movies, for a database. They have worked well in supercomputing, as well as in distributed file systems such as HDFS. But what both Panasas and HDFS use is direct attached storage devices (DASD). As I was preparing to write this column, Seagate announced something quite different: Kinetic Open Storage.

With Kinetic, Seagate will provide 4 TB Enterprise drives without a SAS or SATA interface. Instead, these drives support two one-gigabyte Ethernet interfaces, and apparently have an interface similar to Amazon's S3, with PUT, DELETE, and GET for objects specified by keys—no buckets. Seagate's software will turn groups of Kinetic drives into a distributed storage system, handling sharding and some form of reliability feature, like erasure coding or replication.

I find myself feeling conflicted by Kinetic. On the one hand, moving the intelligence for managing objects right onto the drive itself appears to be a brilliant idea. Modern enterprise drives already have a lot of processing power, including multiple embedded CPUs. On the other hand, I wonder how many people will want to let go of the illusion of control over their drives. Currently, we know how to replace a drive when it fails. What happens when a Kinetic drive fails? Not that the long-favored scheme of RAID for reliability works with drives as large as four terabytes. And replication is expensive in terms of space for reliability.

You can read about other peoples' opinions about Seagate's Kinetic at the StorageMojo blog [2]. I had asked Robin Harris to write for *;login:*, and Brent Welch too. Neither actually wrote for us, but you can find both Brent's and Robin's thoughts on Seagate Kinetic in the blog entry and the comment section.

The Lineup

We begin this issue with the article about object storage systems by the CohortFS folks mentioned above, who have been working on Ceph, adding their own extensions to the project. They've done a great job of explaining the history of object storage, the standards, both official and de facto, that exist, and where Ceph and their own projects are headed.

I also met Suresh Srinivas during SDC, and he agreed to write about the new features of Hadoop 2. Working with his partner, Sanjay Radia, they outline features of Hadoop 2 that take the original version and make it a better match for more generalized types of distributed programming, including models such as MPI and streaming. They also explain how HDFS has been made more durable and much faster, along with the addition of an NFS interface for sharing access.

Jason Paree and Andy Seely approached me with the desire to share a process that they participated in within their work environment. They explain how their group implemented change management, a method for controlling changes to configuration. Whereas configuration management is a method for applying configuration, change management adds a level of control that has helped their organization reduce the growth in both the number of changes and the number of mistakes made.

David Lang continues his series on enterprise logging. David discusses reports and dashboards, where both can be much slower—actually, disk hogs—if not designed properly to use summaries.

Mark Bainter and Dave Josephsen agreed to each write about one of their most difficult experiences with debugging system problems. Although Doug Hughes has written similar articles for *;login:* [3, 4], we are hoping that articles about debugging difficult system problems can become a regular feature. Mark and Dave have written their stories as a discussion, and a bet, with me as the judge.

Klaus Haller volunteered an article about data loss prevention. Klaus thought this might be interesting, given the still-ongoing disclosures by Edward Snowden. Klaus writes based on his own experiences, and there are other systems available for data loss prevention.

Mihalis Tsoukalos also volunteered an article, this one on importing packet traces into a database. Although this has been done before, in products and tools such as Argus, I thought his work was interesting enough to include in this issue.

David Blank-Edelman joins the ranks of the cool by describing how Redis works. As with his other recent columns, David spends much of his time describing Redis itself, as the interface to Redis in Perl closely matches Redis' command-line interface. Redis certainly has some powerful features.

David Beazley takes a look at wheel, a new packaging system for Python. Although there are already common methods for creating packages, wheel adds new features and abilities that David felt were worthwhile learning about.

Dave Josephsen covers a specific aspect of monitoring. Dave looks at counters, the various types, and why they are so important.

Dan Geer and Gunnar Peterson take an interesting look at a way of measuring risk. They borrow from the financial industry to calculate a margin of safety by comparing the cost of a system to the amount spent on securing that system. Using this, you can compare how much your organization has spent on securing different systems reasonably.

Robert Ferrell writes about a new security phenomenon: cyber. Well, not so new, as Gibson's *Neuromancer* introduced the term in the '80s. But that's nothing compared to what can be done with cyber today.

Elizabeth Zwicky begins book reviews with *Perl One-Liners*, which she likes, then dives into two books on data science which offer complementary approaches to the topic, one introductory, the other Agile. With *Designing for Behavior Change*, Elizabeth considered a book for developers. She closes with a review of a basic book on Linux which in her view came up short.

Mark Lamourine has reviewed three books. He starts with an advanced college textbook on programming distributed systems, then describes a new book on DNS that includes alternatives to the venerable BIND. Finally, Mark takes a long, hard look at Mark Burgess' *In Search of Certainty*.

We have the summaries for LISA 2013 in this issue, along with the Advanced Topics summaries. Originally, I had planned on including them in the April 2014 issue, but surprised myself by getting them edited in time to make this issue. They will appear online in their entirety, including the the workshop summary for the Summit for Educators in System Administration.

As the amount of data we create continues to grow, we will need to find new ways of dealing with it. Object stores are certainly one way, and I'm sure that other methods will continue to arise over time, and through necessity.

As for my professor who taught about IBM, and I will confess I did learn some useful things, I do wish he had been more up to date. While struggling with the class assignments, I asked his assistant if there were better examples of operating systems for machines like the PDP 11. He said there were not, even though the John Lion's commentary on UNIX had been published a year earlier [5]. I would have to wait until someone shared a blurry, multi-generation photocopy of the Lion's book a couple of years later to have an example of the operating system that would provide a model for future file systems.

References

[1] SNIA Storage Developer Conference: http://www.snia.org/about/calendar/2013-storage-developer-conference.

[2] StorageMojo on Kinetic: http://storagemojo.com/2013/11/21/seagates-kinetic-open-storage-vision/.

[3] Doug Hughes and Nathan Olla, "When Disasters Collide," *;login:*, vol. 37, no. 3 (June 2012): https://www.usenix.org/publications/login/june-2012-volume-37-number-3/when-disasters-collide-many-fanged-tale-woe.

[4] Doug Hughes, "When Disasters Collide, Continued," *;login:*, vol. 37, no. 4 (August 2012): https://www.usenix.org/publications/login/august-2012-volume-37-number-4/when-disasters-collide-many-fanged-tale-woe.

[5] John Lions, *Commentary on UNIX Sixth Edition with Source Code* (Peer to peer Communications, 1977): http://www.amazon.com/Lions-Commentary-Unix-John/dp/1573980137.

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

Publish and Present Your Work at USENIX Conferences

The program committees of the following conferences are seeking submissions. CiteSeer ranks the USENIX Conference Proceedings among the the top ten highest-impact publication venues for computer science.

Get more details about each of these Calls for Papers and Participation at www.usenix.org/conferences/calls-for-papers.

ICAC '14: 11th International Conference on Autonomic Computing
June 18–20, 2014, Philadelphia, PA
Paper titles and abstracts due: February 26, 2014, 11:59 p.m. EST
ICAC brings together researchers and practitioners from disparate disciplines, application domains, and perspectives, enabling them to discover and share underlying commonalities in their approaches to making resources, applications, and systems more autonomic.

USENIX Security '14: 23rd USENIX Security Symposium
August 20–22, 2014, San Diego, CA
Submissions due: February 27, 2014, 8:59 p.m. EST
The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in the security of computer systems and networks.

HotCloud '14: 6th USENIX Workshop on Hot Topics in Cloud Computing
June 17–18, 2014, Philadelphia, PA
Submissions due: March 6, 2014, 9:00 p.m. EST
HotCloud brings together researchers and practitioners from academia and industry working on cloud computing technologies, and provides a forum for them to share their experiences, leverage each other's perspectives, and identify new/emerging "hot" trends in this important area.

HotStorage '14: 6th USENIX Workshop on Hot Topics in Storage and File Systems
June 17–18, 2014, Philadelphia, PA
Submissions due: March 13, 2014, 11:59 p.m. PDT
HotStorage '14 will showcase the latest in storage systems design, implementation, management, and evaluation, and provide a forum where researchers can exchange ideas and engage in discussions with their colleagues.

JETS* Volume 2, Number 3: *USENIX Journal of Election and Technology and Systems
Submissions due: April 8, 2014, 11:59 p.m. PDT
JETS is a new hybrid journal/conference, in which papers will have a journal-style reviewing process and online-only publication. Accepted papers for Volume 2, Numbers 1–3, will be presented at EVT/WOTE '14, which takes place August 18–19, 2014.

LISA '14: 28th Large Installation System Administration Conference
November 9–14, 2014, Seattle, WA
Submissions due: April 14, 2014, 11:59 p.m. PDT
If you're an IT operations professional, site-reliability engineer, system administrator, architect, software engineer, researcher, or otherwise involved in ensuring that IT services are effectively delivered to others—this is your conference, and we'd love to have you here.

OSDI '14: 11th USENIX Symposium on Operating Systems Design and Implementation
October 6–8, 2014, Broomfield, CO
Abstract registration due: April 24, 2014, 9:00 p.m. PDT
OSDI brings together professionals from academic and industrial backgrounds in what has become a premier forum for discussing the design, implementation, and implications of systems software.

A Saga of Smart Storage Devices

An Overview of Object Storage

MATTHEW W. BENJAMIN, CASEY BODLEY, ADAM C. EMERSON, AND MARCUS WATTS

Matt Benjamin is chief architect of CohortFS, and a founder of CohortFS, LLC. Matt is a contributor to numerous open source software packages and tools, including the NFS Ganesha and OpenAFS. Matt holds a master's degree from the University of Michigan, and a bachelor's degree (summa cum laude and Phi Beta Kappa) from the University of Missouri. matt@cohortfs.com

Casey Bodley studied computer science at Eastern Michigan University. He then worked for the Center for Information Technology Integration at the University of Michigan to develop a Windows client for NFSv4.1. He joined the CohortFS team in 2012, and has been working on parallel metadata enhancements to Ceph. casey@cohortfs.com

Adam C. Emerson studied mathematics at the University of Michigan. While at CohortFS he has worked on the Ganesha NFS server, especially improving support for NFSv4.1 and pNFS. He also collaborated in the CohortFS design process for data placement, encryption, and metadata striping. aemerson@cohortfs.com

Marcus Watts is a programmer at CohortFS. He previously worked for a large education institution where he worked with AFS and identity management. Way back when, he wrote a computer conferencing program, PicoSpan, which was the basis for the Well in California. mdw@cohortfs.com

Object storage systems have become quite popular, with implementations ranging from Amazon's S3 to backends for NFSv4.1. We describe the history of object storage, the practice and standards in use today, and work being done by groups such as the Ceph project, as well as some of our own development.

Object storage fills a gap between block storage and file systems. Simple block storage consists of fixed-size blocks as provided by traditional disk drives. Blocks can be accessed randomly but must be allocated by some scheme. Most modern file systems provide a directory hierarchy that contains variable length byte-array "files" as the leaves. Modern network applications frequently need a higher abstraction than can be provided by simple block storage, yet don't need all the complexity and limitations of a traditional file system. Object storage fills this gap by providing for larger variable-sized segments and a simple flat-naming scheme.

The most common object abstraction provides one or more collections of uniquely named objects of arbitrary size associated with some amount of metadata. Object storage tends to emphasize getting or putting entire objects, rather than reading and writing byte-ranges as is more common in file systems.

We will talk about object storage systems as two groups, which we will call the "device-like family" and the "HTTP-like family." The device-like family exposes a cluster of individual devices to clients. One example of device-like objects is the SCSI T10 object standard. The HTTP-like family presents a single interface, hiding details of how data is distributed. The best-known example of the HTTP-like family is Amazon's S3.

History and Character

Our two divisions of object storage grew up independently but have crossed over and stimulated each other.

The Device-Like Family

The modern device-like object storage paradigm traces back to work by Garth Gibson and others on the NASD (Network Attached Secure Disks) project at CMU, whose goal was the creation of a scale-out storage system. They designed intelligent drives storing variable-length objects with access being granted by a cacheable token; this allowed scale-out similar to a SAN but without clients having to be involved with actual block allocation, and so allowing disk devices to perform on-platter optimization [3]. In NASD (and its direct descendant PanFS) the file server was responsible for assigning objects to devices; NASD used a middleware to stripe virtual objects across real devices. Even in the beginning, objects were used for more than just building file systems. For example, the NASD project built a distributed streaming MPEG2 server. The SCSI T10 committee standardized the Object Storage Device command set (drawn from the NASD model), a second version has been finalized (OSDv2), and a third is currently in development.

Further research in the device-like family of object storage focused on decentralizing placement of objects on devices [4]. Other work included scaling objects to more devices as the amount of storage grew, and responding to failures and gracefully reorganizing data as devices were added or removed, as in the RUSH [5] family of algorithms. The Ceph project builds on previous work, with a cluster of OSDs cooperating to perform automatic replication, recovery, and snapshots.

CleverSafe implements its own object storage system (with a proprietary protocol) where object names are effectively hierarchical addresses within a cluster. CleverSafe makes heavy use of Cauchy Reed-Solomon erasure codes [7] for fault tolerance as well as information dispersal. Information dispersal starts with a piece of data and derives multiple chunks from it, some number of which are needed to reassemble the original. CleverSafe optionally performs encryption, integrity, and compression as part of the write operation.

The HTTP-Like Family

The most successful (and current de facto standard) representative of the HTTP-like object storage family is Amazon's S3, which, like the Elastic Compute Cloud, was launched to expose and sell access to the global infrastructure Amazon developed to run its own business. S3's operations (getting, putting, and deleting, generally of entire objects at once) fall naturally out of the common REST architecture, which structures APIs around the standard methods of HTTP [2]. This gives S3 a high-level abstraction free from many assumptions or implied structures, similar to T10 OSD. S3 provides a flat namespace of objects within "buckets," which both partition objects into flat namespaces and dictate policy.

S3 has been enormously successful, not just as a service but as an API, and it has been adopted by other cloud service providers and by software such as the CloudStack framework and the Eucalyptus cloud computing system.

OpenStack's Swift service fills a similar niche. Even though it provides storage implemented by members of a cluster to members of that cluster, all requests go through an HTTP proxy server that hides the details of distribution and abstracts away the clustered nature of access from the client.

Hybrid Models

As both these families have been developed, they've borrowed from each other. Ceph's RADOS protocol implements object pools that function much like S3's buckets, and they map directly onto buckets in the RADOS Gateway, a Web service that hides the clustered nature of Ceph behind a Web proxy.

Huawei's Universal Data Storage goes one step further, selling hardware (clusters of smart disk drives) that speaks S3 to clients while providing enterprise functionality and management.

Anonymity Networks

Anonymity networks such as Freenet and GNUnet have converged on the object-like semantics of publishing and retrieval of blobs of data in a flat namespace on a wide-scale cluster. Clients interact with the individual nodes on the peer-to-peer (or friend-to-friend) network, but may have their interactions with the ultimate endpoints obscured by layers of onion routing and cover traffic depending on their security settings. New designs (referred to in GNUnet documentation and source as "multicast") for trusted replication among peers allow HTTP-like functionality, such as resilience or distributed service of resources in high demand, while preserving anonymity and privacy.

Current Uses

Many object storage systems can be used as arbitrary key-value stores with good performance for large values. T10 OSDv2 is a notable exception as it uses 64-bit integers to name objects within a partition; it requires an index, which may be implemented in the object system, to link more interesting names to integers. Object stores are also often used as building blocks for richer systems.

Database Integration

BLOB (Binary Large OBject) fields store mostly uninterpreted data in database records and have always been awkward due to their large size, which can drive other data out of cache in the database client. The BLOBs themselves are often served more slowly than would be ideal since they have to be pulled through that database connector interface. Many database programmers address this by storing objects in files and then storing the file names in the database.

One major downside of storing BLOBs as files is that most file systems don't offer the same reliability, integrity, or replication features that cover data stored in the database. Developers are using object stores and any replication and reliability that the store in question might provide to get around this limitation. Often the object will be named with a hash of the BLOB's content to provide implicit integrity checking and deduplication. This approach has become so popular that it's starting to become integrated into database backends. OblakSoft's Cloud Storage Engine for MySQL introduces a "WEBLOB" field type that integrates storage of BLOBs using Amazon's S3 protocol directly into the database. Using HTTP-accessible objects specifically also allows Web assets to be displayed to a client by passing a URL, without proxying the data through the application.

Streaming Storage

Video streaming from object stores was prototyped with MPEG2 on NASD but hit great commercial success with Netflix's adoption of Amazon S3 to back its streaming video service. This has become successful enough that Amazon has pursued the streaming market by adding RTMP access to S3 objects [1].

Virtual Machine Images

One of Ceph's biggest current successes is the RADOS Block Device. This is a set of conventions for organizing Ceph objects into disk images that can then be booted from or mounted by virtual machines in a cloud environment. This allows the use of the object store's capabilities, such as snapshots and replication, to provide checkpointing and resilience. Snapshot layering provides a crude approximation (due to snapshots being read-only) for copy-on-write storage and deduplication.

File Systems

OSDv2 is used by Panasas in PanFS and by the free ExoFS project as the backing store for their file systems. Ceph follows this same approach, building a file system on top of the RADOS object access protocol.

The S3 FUSE utility builds a file system on top of cloud-based storage, and Tahoe LAFS's RAIC (Redundant Array of Inexpensive Clouds) plans to build a highly reliable, secure file system that straddles the object storage systems of multiple cloud providers for reliability in the event of a provider's failure.

pNFS

T10 OSDv2-based object-backed file systems have been standardized as a scale-out and reliability component of NFSv4.1 through Parallel NFS (pNFS). pNFS introduces the concept of recallable layouts to represent both permission to and details on how to access data directly at the point that it is stored [8]. Clients then access back-end storage directly without going through a front-end server. pNFS allows differing access protocols, like striping data over several NFSv4 servers, accessing data as block ranges on SCSI devices, and arranging data in recursive RAID configurations over T10 objects.

The OSD layout type lets clients be aware of, participate in, and take advantage of replication and erasure coding. Clients can read stripes from multiple devices for improved speed or perform erasure coding at the time of writing.

Ceph

As Ceph is of current interest in the storage community, and because we are basing much of our work on it, we give Ceph some special mention.

Contrast with T10

Ceph's architecture can be contrasted with that of T10 OSDv2. Object storage devices in T10 are independent of each other and under the control of some director that grants access through security tokens. Replication occurs when clients write the same data to several devices, and parity is calculated on the client and written as normal data. How data and parity blocks are distributed among devices is outside the scope of the T10 OSDv2 standard.

Ceph organizes object storage devices into a cooperative group under control of a small number of servers called monitors. In Ceph, monitors keep globally known data coordinated through Paxos. Data is distributed over devices under the control of a globally known collection of rules and data structures, which are maintained consistently by the monitors. Administrators describe the organization of storage devices, breaking them down hierarchically into zones of potential failure. Administrators also set the number of replicas and policies about where to place objects. The placement logic shared between clients and storage devices combines this policy with a monitor-maintained map of storage device status (operating, temporarily down, out of service) to calculate where individual objects are located. Clients perform writes to one object storage device, and the storage devices coordinate between themselves to perform replication and data recovery [9]. Ceph currently lacks support for erasure coding, but multiple efforts are underway to add it.

The present RADOS protocol lacks access control beyond a public key needed to communicate with the cluster at all. There are designs for access control on extremely large scale object storage systems [6].

Immediate Applications

Ceph is a large, complex system currently undergoing active development and gaining new capabilities. There are several use-cases it can address out of the box.

Ceph provides an immediately replicated file system in a single datacenter environment. A stable write to a file is considered to be complete when it has been stably recorded on all devices replicating the given block, giving resilience against drive failure. Per-directory immutable snapshots can be made by unprivileged users allowing them to version their data.

Even without the file system, Ceph can be used in the construction of private clouds that leverage the large number of applications made to work with the S3 protocol. Ceph can be dropped in as a replacement for public cloud services simply by setting up a cluster and configuring applications to use the RADOS Gateway server as the target for requests. Also, Ceph can be used as a proxy server for Swift requests in OpenStack installations, though its Swift interface is less complete than its S3 interface.

RADOS block devices (RBD) have full Linux kernel support, and can be used any way you would use any other block device, but with replication and a snapshot capability. Some people have experimented with re-exporting RADOS block devices over iSCSI to make them available to other operating systems, such as FreeBSD or Windows. RBD's biggest success has been in virtualization environments. In addition to providing an RBD device as a normal block device, RBD support has been integrated into virtualization and cloud computing systems, such as OpenStack, and any systems using libvirt (such as CloudStack and virt-manager).

CohortFS Development

Our goals at CohortFS include development in the field of object storage. Currently, we're focusing on improving the capabilities of Ceph as both an object store and a file system, and on improving NFS to take the best advantage of advanced functionality that a file system provides.

Ceph via NFS

We have added NFS access to the Ceph file system and to the Ganesha user space NFS server, and have implemented pNFS access for objects striped over Ceph OSDs. In the future, we will be developing a new layout type to better take advantage of placement strategies other than repeated striping patterns; we will also be adding support to Ceph for a recallable layout that better matches the requirements of NFS than do current Ceph capabilities.

Volumes

We are adding an implementation of volumes to Ceph, which will allow a single cluster to hold multiple independently rooted file systems or collections of objects, each with its own administrative domains of control to support delegated multi-tenancy. Our future development in this area includes automatic allocation of object-storage devices with different capabilities to fill administrator-specified quality-of-service requirements.

Erasure Coding and Client Offload

We are currently adding erasure coding support to Ceph, using the Jerasure library. In CohortFS, clients will be able to perform replicated writes and generate erasure codes rather than having to leave those to the OSDs. This frees us of the requirement to have multiple OSDs coordinate in a computation for each write, while still allowing OSDs to repair faults automatically. This is in contrast with another erasure coding project for Ceph where erasure codes are generated cooperatively by the OSDs.

Dynamically Generated Placement Functions

We take the notion of a globally known placement function to its logical conclusion by dynamically generating placement functions as fragments of executable code that are distributed throughout a Ceph cluster and to clients. This allows us to tailor data placement specifically to the requirements of the use-case. Additionally, expensive optimization of the function against the cluster description can be performed once, centrally. This gains faster placement calculation without loss of generality. Finally, this allows us to change the behavior of the system radically without having to go through the expensive and error-prone operation of a cluster-wide upgrade.

Availability

Much of the software mentioned here is available with source on the Internet. An implementation of a T10 OSD target is available from http://www.open-osd.org. They also developed an initiator and a scale-out network file system built on top of the T10 OSD protocol called ExoFS. ExoFS and the T10 initiator have been integrated into recent Linux source trees.

Ganesha is a user-space server for versions 3 and 4 of the NFS protocol and for 9P, a file-system protocol originally used for the Plan 9 operating system that is now seeing some use in high-performance computing environments. Ganesha's design is centered around a File System Abstraction Layer, allowing it to serve systems as diverse as the Linux open-by-handle interface, ZFS (through libraries), and even a proxy to other NFS servers. It is used as an NFS front-end for both free and proprietary file systems, and also functions as a platform for development of new server functionality. Ganesha is available from http://nfs-ganesha.github.com.

The Ceph distributed object store and file system is available from http://ceph.com.

OpenStack is a free Infrastructure-as-a-Service framework that implements the Swift object storage service as well as integrating well with other object storage systems, such as S3 and Ceph. OpenStack is available from http://www.openstack.org.

The Jerasure library implements many freely available erasure codes and is available from http://Web.eecs.utk.edu/~plank/plank/papers/CS-08-627.html.

Much of our work is available in the Ganesha NFS server, and some has (or will shortly be) submitted to Ceph. Other parts of CohortFS will become freely available as they are completed.

Acknowledgments

We would like to thank the National Science Foundation, which has funded our work on CohortFS through a Small Business Innovation Research grant, Peter Honeyman for his work in helping to design CohortFS, and the Ceph community for providing a flexible and open platform for development. We would also like to thank the Ganesha community for embodying everything that is good about collaborative, free software development.

References

[1] Amazon.com, "Working with RTMP Distributions," October 2013: http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/WorkingWithStreamingDistributions.html.

[2] R. T. Fielding, and R. N. Taylor, "Principled Design of the Modern Web Architecture," *ACM Transactions on Internet Technology* 2 (2002), pp. 115-150.

[3] G. A. Gibson, D. F. Nagle, W. I. Courtright, N. Lanza, P. Mazaitis, M. Unangst, and J. Zelenka, "NASD Scalable Storage Systems," *Proceedings of USENIX 1999*, Linux Workshop, Monterey CA, June 9–11, USENIX Association.

[4] R. J. Honicky, "A Fast Algorithm for Online Placement and Reorganization of Replicated Data," *Proceedings of the 17th International Parallel & Distributed Processing Symposium (IPDPS 2003).*

[5] R. J. Honicky and E. L. Miller, "Replication Under Scalable Hashing: A Family of Algorithms for Scalable Decentralized Data Distribution," *Proceedings of the 18th International Parallel & Distributed Processing Symposium (IPDPS 2004).*

[6] A. Leung and E. L.Miller, "Scalable Security for Large, High Performance Storage Systems," *Proceedings of the 2nd ACM Workshop on Storage Security and Survivability (StorageSS 2006)* (Alexandria, VA, October 2006), ACM.

[7] J. Plank, "Erasure Codes for Storage Systems: A Brief Primer," USENIX *;login:* (December 2013, Volume 38, Number 6).

[8] S. Shepler, M. Eisler, and D. Noveck, Network File System (NFS) Version 4 Minor Version 1 Protocol, RFC 5661 (Proposed Standard), January 2010.

[9] S. A. Weil, "Ceph: Reliable, Scalable, and High-Performance Distributed Storage," Ph.D. thesis, University of California at Santa Cruz, December 2007.

Buy the Box Set!

Whether you had to miss a conference, or just didn't make it to all of the sessions, here's your chance to watch (and re-watch) the videos from your favorite USENIX events. Purchase the "Box Set," a USB drive containing the high-resolution videos from the technical sessions. This is perfect for folks on the go or those without consistent Internet access.

Box Sets are available for:

- **LISA '13:** 27th Large Installation System Administration Conference
- **USENIX Security '13:** 22nd USENIX Security Symposium
- **HealthTech '13:** 2013 USENIX Workshop on Health Information Technologies
- **WOOT '13:** 7th USENIX Workshop on Offensive Technologies
- **UCMS '13:** 2013 USENIX Configuration Mangement Summit
- **HotStorage '13:** 5th USENIX Workshop on Hot Topics in Storage and File Systems
- **HotCloud '13:** 5th USENIX Workshop on Hot Topics in Cloud Computing
- **WiAC '13:** 2013 USENIX Women in Advanced Computing Summit
- **NSDI '13:** 10th USENIX Symposium on Networked Systems Design and Implementation
- **FAST '13:** 11th USENIX Conference on File and Storage Technologies
- **LISA '12:** 26th Large Installation System Administration Conference

Learn more at:
www.usenix.org/boxsets

HADOOP 2
What's New

SANJAY RADIA AND SURESH SRINIVAS

Sanjay is co-founder and architect at Hortonworks, and an Apache Hadoop committer and member of the Apache Hadoop Project Management Commitee (PMC). Prior to co-founding Hortonworks, Sanjay was the chief architect of core-Hadoop at Yahoo and part of the team that created Hadoop. In Hadoop he has focused mostly on HDFS, MapReduce schedulers, high availability, compatibility, etc. He has also held senior engineering positions at Sun Microsystems and INRIA, where he developed software for distributed systems and grid/utility computing infrastructures. Sanjay has a Ph.D. in Computer Science from the University of Waterloo in Canada.
Follow Sanjay on Twitter: @srr
sanjay@hortonworks.com

Suresh is an Apache Hadoop committer and member of the Apache Hadoop Project Management Commitee (PMC). He is a long-term active contributor to the Apache Hadoop project. Prior to co-founding Hortonworks, he served as a software architect at Yahoo! working on Apache Hadoop HDFS, where he developed features and supported some of the largest installations of Hadoop clusters. Suresh also worked for Sylantro Systems in various senior technical leadership roles and developed scalable real-time infrastructure for hosted communications services.
Follow Suresh on Twitter: @suresh_m_s
suresh@hortonworks.com

Hadoop 2 contains fundamental changes in the architecture that significantly extend the platform, taking the compute platform beyond MapReduce and introducing new application paradigms. Similarly, the storage subsystem has been generalized to support other frameworks besides HDFS. The new version significantly improves scalability and performance in both the compute and storage layers, with disk performance up to five times faster and the compute layer scaling to clusters with more than 100k concurrent tasks. Automatic failover of master servers now provides high availability. We cover all these and other key Hadoop 2 features in this article.

Quick Background

Apache Hadoop is a scalable framework for storing and processing data on a cluster of commodity hardware nodes. Hadoop is designed to scale up from a single node to thousands of nodes. Hadoop has two main components: a computing framework and Hadoop Distributed File System (HDFS). HDFS uses the commodity server nodes and JBOD (Just a Bunch Of Disks) storage drives to store the data and provide large aggregated I/O bandwidth to data. The compute framework uses the same set of server nodes for computation. The key idea is to move computation to where the data is. This enables scalable and efficient ways of storing and processing data. Storage capacity, compute capacity, and I/O bandwidth can be scaled by adding more servers.

HDFS has had a single master server for storing the file system metadata called the NameNode. The files stored on HDFS are split into one or more blocks, typically of size 128 MB. These blocks are stored on slave nodes called DataNodes. To ensure data reliability, multiple replicas of blocks are stored on a set of DataNodes. A client performs file system operations such as creating, modifying, and deleting files at the NameNode. The NameNode records these transactions in a journal, and the data for the files are written by the clients at the DataNodes. The NameNode actively monitors the DataNodes, so if a replica of a block is lost due to disk failures or node failures, new replicas are created.

Computation framework has a master server that manages the compute resources in the slave nodes. It supports parallel, distributed programming paradigms over which a vast amount of data can be processed in a reliable and fault-tolerant manner. Typically, the data is processed in parallel using multiple tasks where each task processes a subset of the data. Traditionally, the MapReduce paradigm is used to process the data in parallel. Hadoop 2.0 has a new compute framework called YARN, which supports MapReduce and other programming paradigms.

Hadoop 2 Improvements

Architectural Evolution

Hadoop 2 has made fundamental architectural changes for both the compute and storage sides of the platform.

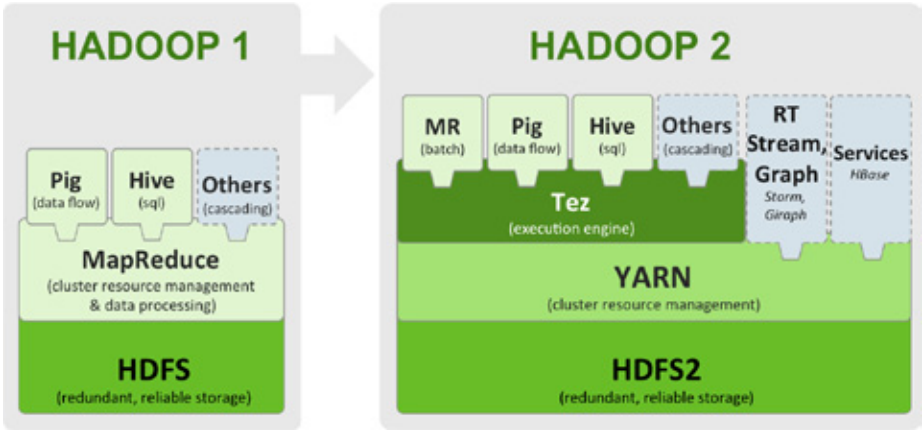


Figure 1: Comparison of Hadoop 1 and Hadoop 2 architectures

of execution. Tez allows Hive and Pig programs to be executed more naturally as a single job instead of multiple MapReduce phases, resulting in many orders of magnitude performance improvements. New breeds of applications for stream processing, such as Samza and Storm on YARN, also run as first-class applications (Figure 1). This allows a consolidation of clusters and compute resources to run heterogeneous applications, resulting in less resource fragmentation and more efficient utilization. For the IT department, this means improved hardware capitalization and simplified management.

YARN—ARCHITECTURAL EVOLUTION IN THE COMPUTE LAYER

Previously, compute resources on Hadoop were only available to MapReduce programs and forced non-MapReduce applications to be modeled as MapReduce. The YARN component generalizes the compute layer to execute not just MapReduce style but other application frameworks. As a result, YARN allows analytics queries to be executed significantly more efficiently and has allowed a new breed of applications, such as stream processing, to be supported in a first-class manner. The new architecture is more decentralized and allows Hadoop clusters to be scaled significantly to more cores and servers. Further, it promises to offer another significant future improvement: the IT department will be able to consolidate other non-Hadoop clusters (such as HPC or virtualization clusters) with the Hadoop cluster.

Decentralized Resource Management

Hadoop 1 had a single master server called a JobTracker to manage both the compute resources and the jobs that use the resources. YARN splits that function so that a Resource Manager (RM) focuses on managing the cluster resources and an Application Master (AM), one-per-running-application, manages each running application (such as a MapReduce job). The AM requests resources from the RM based on the needs and characteristics of the application being run. For example, a MapReduce application needs compute resources close to the data and has Map and Reduce functions that can be scheduled in phases. On the other hand, an MPI job may be compute-intensive and requires all resources to be scheduled together.

First-Class Support for Different Application Types

Because the AM is separate from the RM, it can be customized per application type. Hadoop 2 has a specialized AM for MapReduce and another more generalized application framework called Tez that allows generic directed-acyclic-graphs (DAGs)

Generalized Resource Modeling

Whereas Hadoop 1 modeled compute resources as Map or Reduce slots, YARN allows a more generalized notion of resources. YARN has started with the memory resource (because it was closest to Hadoop 1's "slot") but will soon be extended to support CPU, I/O, and network resources.

ARCHITECTURAL EVOLUTION IN THE STORAGE LAYER

Hadoop cluster's storage resources were previously available only to HDFS. Similar to YARN, the new storage architecture generalizes the block storage layer so that it can be used not only by HDFS but also other storage services. The first use of this feature is HDFS federation, which allows multiple instances of HDFS namespaces to share the underlying storage. In future versions of Hadoop, other storage services (such as key-value storage) will use the same storage layer.

Another fundamental storage change that is being worked on is support for heterogeneous storage. Hadoop 1 treated all storage devices (be it spinning disks or SSDs) on a DataNode as a single uniform pool; although one could store data on an SSD, one could not control which data. Heterogeneous storage will be part of the 2014 release of Hadoop 2.x, where the system will distinguish between storage types and also make the storage type information available to frameworks and applications so that they can take advantage of storage properties. Indeed, the approach is general enough to allow us to treat even memory as a storage tier for cached and temporary data.

Classic Enterprise Features

Hadoop was initially adopted by Web companies for large-scale data processing. With Hadoop crossing the chasm, different use cases and enterprises are migrating to Hadoop. With that comes the expectation of support for features that enterprise users have come to expect.

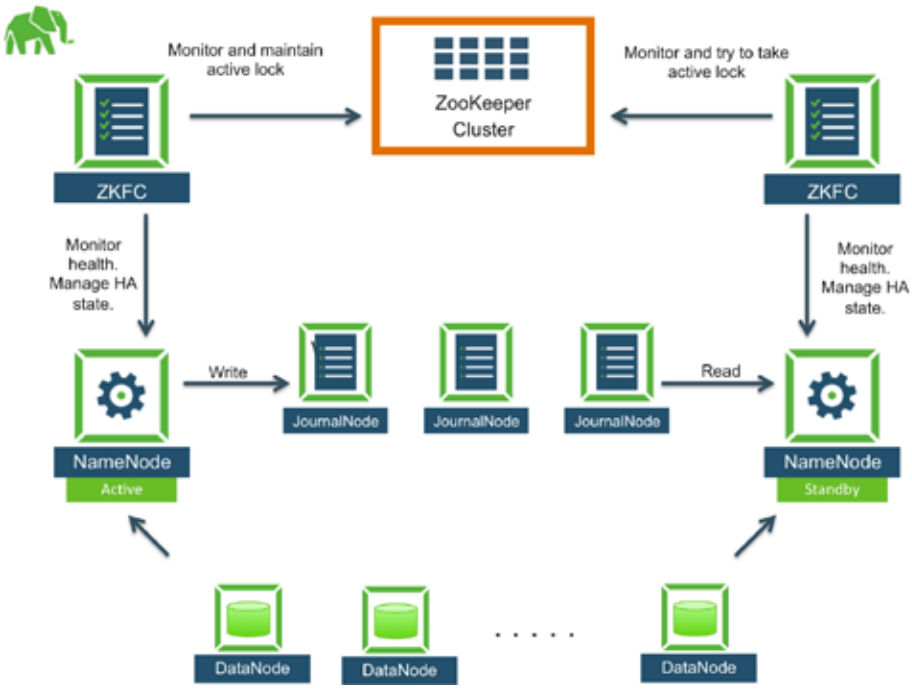


Figure 2: ZKFCs monitor NameNodes, and while the active NameNode writes to JournalNodes, the standby NameNode reads from JournalNodes to keep its state updated.

NAMENODE HIGH AVAILABILITY

While the raw storage layer in HDFS (the block storage layer) is fully distributed and fault-tolerant, the file system metadata was stored in a single master server called the NameNode. When the NameNode is brought down for planned maintenance, or on rare software or hardware failure, the cluster would be unavailable until the NameNode is restarted. Distributions such as Hortonworks Data Platform (HDP) had cold failover for the NameNode using industry standard frameworks, such as Linux HA and VMware vSphere, in their Hadoop 1 distribution. Hadoop 2 adds support for a hot standby NameNode along with a journal service. In case of failure of the active NameNode, automatic failover is triggered and the standby NameNode becomes active.

FAILOVER CONTROLLER

A new watchdog daemon called the ZKFC (ZooKeeper-based Failover Controller) manages failover of NameNodes. This daemon runs on each of the NameNodes and maintains a session with the ZooKeeper. Using ZooKeeper for coordination, one of the ZKFC becomes the leader and elects the local NameNode as active. The ZKFC performs a periodic health check of the NameNode. When the active NameNode fails health check, the local ZKFC resigns as the leader. Similarly, when the active NameNode machine fails, ZooKeeper detects the loss and removes the ZKFC from the failed node as the leader. This results in automatic failover; the ZKFC running on standby becomes the leader and makes the local standby NameNode active.

QUORUM JOURNAL MANAGER

In Hadoop 2, the file system journal no longer needs external NAS storage. The NameNode writes the journal to external daemons called Journal Nodes. The Quorum Journal Manager ensures every transaction is written to a quorum number of journal nodes using a distributed commit protocol based on Multi-Paxos. Because only one NameNode can successfully write to a quorum number of Journal Nodes, the corruption due to a split-brain condition is avoided. This ensures that the redundant and consistent copies of journal are persisted on the journal nodes. The standby NameNode reads the transactions from the journal and updates its state to stay in sync with the active NameNode.

NFS SUPPORT

Access to HDFS is usually done through the HDFS client library or over HTTP REST APIs. Lack of seamless integration with the client's file system makes it difficult for users and impossible for some applications to access HDFS. Hadoop 2 adds NFS version 3 support to make this integration easy.

NFS access is enabled using stateless NFS gateways. The NFS gateway's main functionality is translation of the NFS protocol to the HDFS native protocol. The gateway is started as a daemon on the slave nodes in a Hadoop cluster. The gateway supports three services: rpcbind (or portmap), mountd, and nfsd. HDFS is mounted on the client system, and applications can access HDFS through the local file system.

HDFS SNAPSHOTS

Hadoop 2 adds support for file system snapshots. A snapshot is a point-in-time image of the entire file system or a subtree of a file system. A snapshot has many uses:

- Protection against user errors: An admin can set up a process to take snapshots periodically. If a user accidentally deletes files, these can be restored from the snapshot that contains the files.
- Backup: If an admin wants to back up the entire file system or a subtree in the file system, the admin takes a snapshot and uses it as the starting point of a full backup. Incremental backups are then taken by copying the difference between two snapshots.
- Disaster recovery: Snapshots can be used for copying consistent point-in-time images over to a remote site for disaster recovery.

The snapshots feature supports read-only snapshots; it is implemented only in the NameNode, and no copy of data is made when the snapshot is taken. Snapshot creation is instantaneous. All the changes made to the snapshotted directory are tracked using modified persistent data structures to ensure efficient storage on the NameNode.

RPC IMPROVEMENTS AND WIRE COMPATIBILITY

Hadoop 2 has several improvements to the RPC layer shared by HDFS, YARN, and MapReduce v2. The on-the-wire protocol now uses protocol buffers and is no longer based on Java serialization. This helps in extending the protocol in the future without breaking the wire protocol compatibility. RPC also adds support for client-side retries of the operation, a key functionality for supporting highly available server implementation. These improvements help in running different versions of daemons within the cluster, paving the way for rolling upgrades.

Other HDFS Improvements

I/O IMPROVEMENTS

Improvements to HDFS speed and efficiency are added on an ongoing basis. There are many improvements to HDFS interfaces. A better short-circuit interface based on UNIX Domain Sockets allows clients to read from the local file system directly instead of inefficiently over a socket from the DataNode. This interface also now supports zero copy reads. The CRC checksum calculation done during both reads and writes is now optimized using the Intel SSE4.2 CRC32 instruction. All of these improvements have made I/O 2.5 to 5 times faster than the previous releases.

APPEND SUPPORT

Hadoop 1 required HDFS files to be immutable once they were created. Hadoop 2 allows one to append data to a previously created file.

Expanding the Community and Use Cases

YARN in Hadoop 2 expands the ecosystem beyond MapReduce and allows new kinds of applications to run on the cluster. Similarly the generalization of storage promises Hadoop storage to be used beyond HDFS.

WINDOWS SUPPORT

Hadoop was originally developed to support the UNIX family of operating systems. With Hadoop 2, the Windows operating system is natively supported. This work is simplified by the fact that Hadoop was written in Java. The dependencies on UNIX for compute and storage resource control now has been generalized to support Windows. This extends the reach of Hadoop significantly to a sizable Windows Server market.

OPENSTACK CLOUD SUPPORT

There is a growing trend to run Hadoop-on-demand and shared infrastructure. Hadoop 2 supports the OpenStack Swift file system, and it has topology improvements for virtualized environments. With OpenStack support for spinning Hadoop clusters up and down, Hadoop can now be run on virtualized hardware, both in public and private datacenter clouds.

Next Steps

Hadoop, which was originally designed around batch processing using commodity disks and servers, is changing in the face of a number of trends. The Big Data application space and Hadoop usage pattern, along with the underlying hardware technology and platform, are rapidly evolving. Further, the increasing prevalence of cloud infrastructure, both public and private, is influencing Hadoop development. Hadoop is evolving to deal with changes in how clusters are being built. HDFS and YARN architecture are growing to adapt to such changes.

Hadoop has become the de facto kernel for the Big Data platform. These exciting developments are being driven by a dedicated Apache community, all in the open. People interested in participating in this technological revolution are welcome to visit http://hadoop.apache.org.

The Evolution of Managed Change in a Complex IT Enterprise

JASON PAREE AND ANDY SEELY

Jason Paree is a Release and Deployment Manager and is responsible for controlling change in operational network environments. He received his BS in Criminal Justice Administration in 2011, but switched career goals and is currently an MBA student. This is his first published work. jasonparee@hotmail.com.

Andy Seely is the manager of an IT engineering division, customer-site Chief Engineer, and a computer science instructor for the University of Maryland University College. His wife Heather is his init process and his sons Marek's and Ivo are always on the run queue. andy@yankeetown.com.

We are members of a contract team charged with performing all aspects of the operations and maintenance of a complex and diverse enterprise network at a Department of Defense customer site. After years of rapid reaction to mission updates, management changes, and varying requirements for governance of technical change in the environment, we found ourselves managing an enterprise that was not well understood and becoming prone to unexpected failures. Over the past two years, our team developed a managed service transition process for change implementation, navigating technological complexities and influencing workplace culture to create a mature process that has delivered positive and predictable results for effective change.

Our contract team operated the enterprise for several years without strong and consistent processes for managing and implementing changes across the operational environment. Our top priorities were always to support the mission first, which sometimes resulted in IT process discipline becoming a secondary priority. Although proposed enterprise changes received formal approval, they were sometimes executed with inadequate planning and were frequently implemented suddenly and without understanding of the second- and third-order effects. This led to situations in which technicians were sometimes working without coordination, resulting in technical problems that couldn't be traced definitively and successes that could not be quantified easily.

This inconsistent change process created a working culture for our team of reacting to uncertainty as our "normal." Although we never had a catastrophic, sustained failure, we spent considerable resources in damage control after deploying changes. A second-order effect of this was that we frequently increased overall complexity by adding technology tools to "fix" our problems rather than directly focusing on our root problem of change control.

These challenges became highly visible when technicians began making preventable mistakes during change deployment, sometimes resulting in a reduction of enterprise services to key user groups for the time it took for problem isolation and resolution. Even though these events usually only lasted minutes, they gained enough negative attention from senior leadership that disciplinary actions were taken for some contract team members. Often, we were forced to stop all change to the environment and review and report any proposed change meticulously, resulting in even more instability to the network as planned critical patches and improvements became backlogged.

To address increasing leadership pressures and head off a perception of growing instability in the environment, we started having a daily change review meeting. Approvals were granted based on well-informed, technical discussion. The culture at this time placed considerable burden on a small group of select, trusted experts on our team and largely sidelined others, resulting in staff frustrations and growing attrition rates. These daily meetings didn't significantly reduce the number of failures or increase overall stability, but they did raise awareness of the number of changes happening, and they helped us gain control over awareness of change success rates and resulted in fewer surprises. This first step helped us to realize the importance of cross-discipline communication and coordination.

Understanding What's Important, and What Is Critical

As we continued to struggle through change implementation, winning small battles and making incremental improvements, pressure to dedicate resources to managing and controlling change began mounting. To address this need, the operations manager for our contract designated one technician to take on an additional duty to start a change management cycle that would start with a regular "Subject Matter Expert (SME) meeting," with the necessary and relevant technicians from each IT section required to attend and discuss change details. SMEs began to help collaborate to develop the process and meetings by staying engaged and taking the process and its purpose seriously. This was not without challenge given the fluidity of operational requirements and changes in personnel and responsibilities that slowed any real cultural change. After about four months, the instability of the group and the early process started to gel; after six months, we were starting to receive change requests from across the team and even other organizations, leading toward our process becoming the focal point for all IT change implementation across the site. Because the process introduced independent rigor, which resulted in a slowed pace of change, we received initial resistance from internal stakeholders. This required us to instill more formality into the process and to develop stronger management commitment for process deadlines, and we started focusing on building customer buy-in for the importance of process discipline.

During this evolution of building, strengthening, and enforcing process requirements, more pressure was applied to implement change with even greater reliability. This pressure resulted in a more formalized meeting structure for change review and more scrutiny on implementation procedures. Our early and loose "SME meetings" became more formal cross-divisional "Technical Review Meetings" that mandated attendance from all IT sections, including Engineering, Cyber Security, and the implementing sections. One Technical Review Meeting became two, an Initial Technical Review and a Final Technical Review meeting to provide "check and balance." Simple PowerPoint slides explaining the details of the changes evolved into Remedy ticket reports with detailed documentation showing execution plans, validation steps, and back-out plans. We developed, documented, and disseminated formal, program-level procedures clearly defining the terms, roles, responsibilities, workflows, and much more of how change is processed for implementation. As this progressive evolution was occurring, our changes were becoming more organized, predictable, and more visible. Our change implementation success rate rose steadily and dramatically.

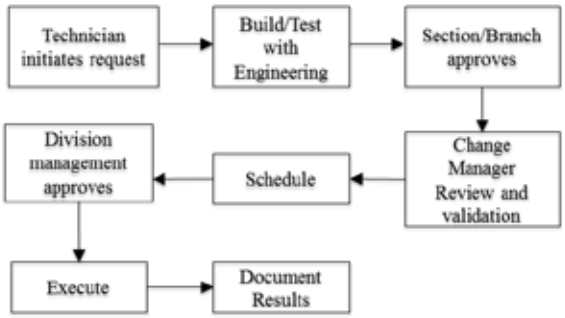


Figure 1: The initial workflow we developed for our change management process

A Formal Training Program Emerges

Although the process at this point had been documented and was becoming established, we still lacked a formal training program. Technicians were still sometimes applying changes outside of formal change control procedures. The process and procedures were there, but unless the technician understood them, the process was all but useless. Several examples became apparent where technicians applied uncoordinated changes, thinking that corrective actions and routine tasks would not require change control. These kinds of events led to increased pressure and scrutiny on changes and the processes governing them.

After another thorough review, we concluded that the process was sound but that there was no programmatic way for new employees to learn it, so it was time for formalized training and communication. Over a period of a couple of weeks, we developed a comprehensive and detailed training program, complete with workflows as shown in Figure 1, terms and definitions, requirements, roles and responsibilities, and timelines. This solidified the process and provided a road map and reference for people to use to submit change requests and to understand the timeline they should expect for planning purposes. After formal documents were disseminated and mandatory formal training provided, our change management workload increased dramatically. Over time, with an improved understanding of how the process worked, people began to better manage expectations, better plan their changes, and slow the overall rate of change. This led the different work sections to become more organized and to decrease the number of "emergency" changes that would circumvent process.

A Formal Process Is Finalized, with Room for Growth

The next phase in our implementation plan included developing a post-review process to identify, document, and disseminate lessons learned when planned change did not complete as expected. We had developed a strong process that worked well for producing success and rolling back from failure, but the process had no provision for anything other than success. This resulted in a lack

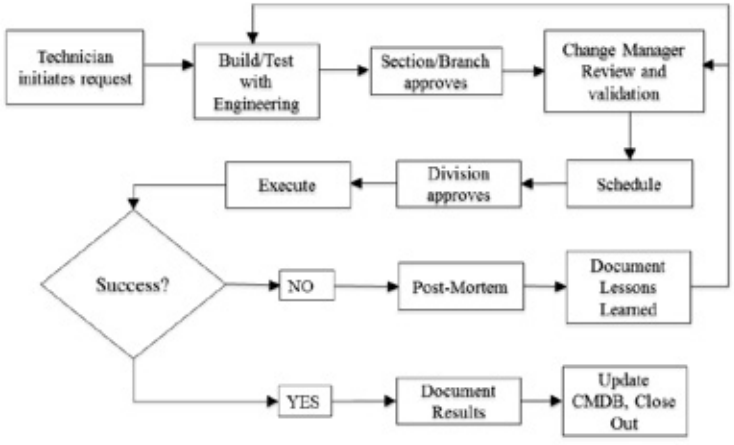


Figure 2: Our workflow after adding in feedback to deal with failed changes

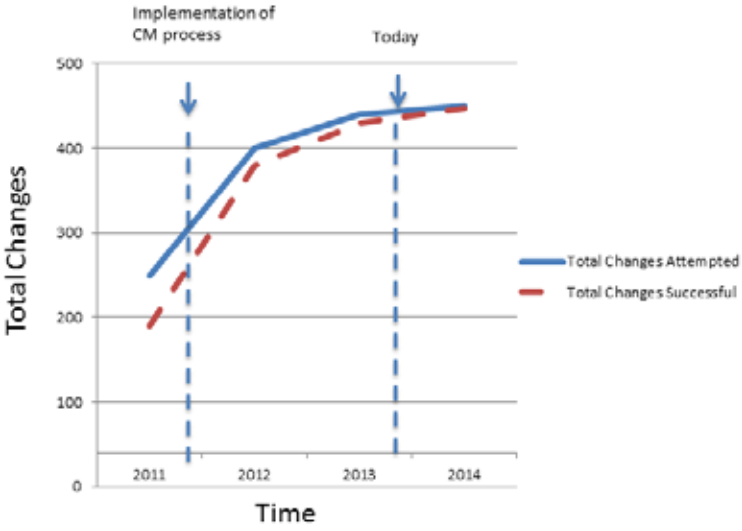


Figure 3: During the three years of this process, we have reduced the rate-of-change for new changes, while increasing our success rate.

of real understanding of failures and loss of opportunity to revise techniques and procedures based on lessons learned. We added an after-action review "post-mortem" branch to our formal process, as shown in Figure 2. When a planned change has any result besides full success, we call a post-mortem meeting with all the people who were part of planning and executing the change, as well as people responsible for system monitoring, service desk, storage, and other relevant technology families. This post-mortem meeting is organized and facilitated by the change implementation manager with senior operations and engineering leadership support, resulting in a collective approach to understanding the problem. The overall environment is explored, frequently with whiteboard diagrams. These reviews explore the need for process or procedural adjustments, the specific plan for the failed change is examined, and a play-by-play of the actual change event is discussed. This is all reviewed with a focus on discovering failure points where wrong decisions were made or where a test plan didn't reflect the operational environment. The culture of the post-mortem discussion is non-retributive: technicians are encouraged to "own your failure" so all can learn from the event.

Our Managed Change Process Today

Over the years, this natural growth of a process benefitted from some good decisions, and a little luck. The decision by senior leaders to encourage the development of the process, to prove it, formalize it, and then get it sanctioned from the bottom up rather than the top down led to broad buy-in from our contract technical staff. The early decision to keep meaningful records of change over time was essential for "proving" the value; we learned that results matter.

Over the past three years, we have made approximately 1,000 changes to our enterprise. Of those, we have had 4% completed unsuccessfully. Figure 3 shows the three-year trend of changes and failures. Noting that the presence of a strong process was an influence on reducing the rate of changes made is important. Although our actual change numbers continue to increase over time, the number of changes relative to the consistently growing complexity of the enterprise is actually decreasing. Things that would have been casually done three years ago are now scrutinized and planned meticulously. If these "casual" changes are no longer being done and the environment is more stable, concluding that we're avoiding unnecessary change, reducing the opportunity for change failure, and preserving stable systems in stable states is easy. By ensuring all change follows a rigorous review process, we gain much deeper understanding of the overall environment and better knowledge of what actually needs to change.

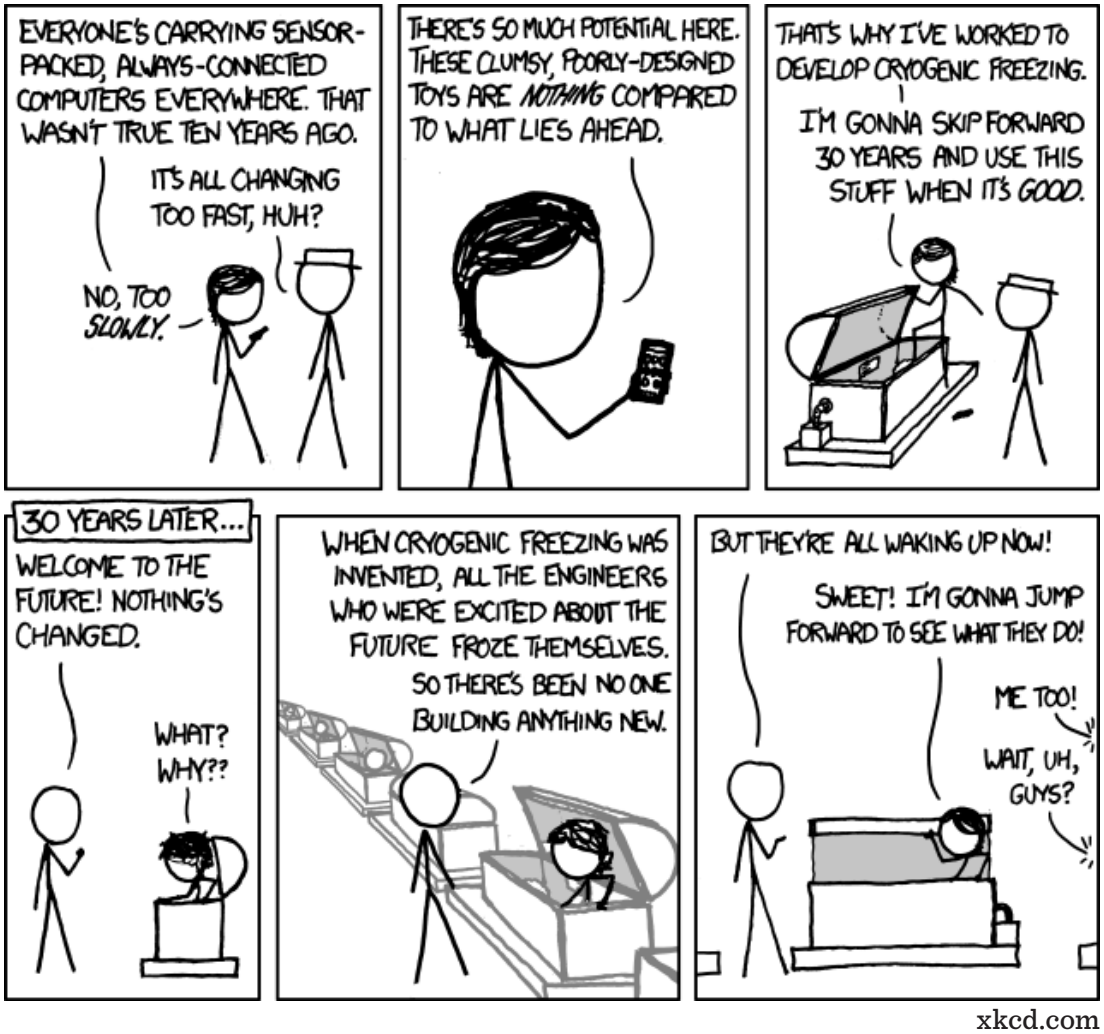
Conclusions and Future Work

Our enterprise was grown organically, driven by reactive, operational imperatives. Leadership changes over time resulted in different focus areas and new tools layered one on top of the other, creating a complex, poorly understood environment. Concepts such as the Information Technology Infrastructure Library (ITIL) were known, but over the years there was little time to formalize process. The change implementation process grew the way the enterprise did, by necessity and in reaction to challenges. Once we slowed the pace of attempted change, and after we had an opportunity to reset the technology baseline with a major overhaul of the datacenter, we were able to apply real rigor to a process that guides change implementation rather than just focusing on the change itself. Although ITIL informed and influenced our new change process, ours is not specifically an ITIL process. In the coming year, we will be growing our change process to encompass other key ITIL areas, such as release management

and configuration management, and we will tune existing steps to ensure ITIL alignment.

The future of IT change in our enterprise is becoming such that the environment is always understood, changes are always tested where possible, and change failures are always embraced as learning opportunities. Our goal is 100% change success through a living process that is embraced by all levels of our team and is easier to follow than to bypass. With improvements in the speed of change approvals and increases in overall throughput, such a process would allow an increased rate of successful IT change.

xkcd



xkcd.com

Logging Reports and Dashboards

DAVID LANG

David Lang is a Site Reliability Engineer at Google. He spent more than a decade at Intuit 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

Once you have set up a system to gather your logs, are able to filter and route the logs, and are alerted to the contents of the log messages [1], the next step is to figure out how to mine the logs for useful information to help you understand what your systems are doing and be proactive in dealing with problems. In this article, I will present strategies you can use to generate reports and dashboards from your logs as efficiently as possible.

Problems to Overcome

Before going into details on how to best generate reports, let's first examine the problems that you are going to be facing in a large environment.

High Log Volume Results in Reports that Take a Long Time to Generate

In an active network, generating anywhere from hundreds of GB to several TB worth of logs per day is easy. Doing anything with this much data is expensive, both in CPU time and, most noticeably, in the disk I/O required to read the data from disk in order to generate a report. The volume of data that you are dealing with is large enough that you are not going to have a machine with enough memory to cache all the logs for a day, let alone for reports covering a longer time frame.

Ending up with a situation in which it takes longer to produce your report than the period the report is supposed to cover is also easy. A system that requires 25 hours to produce a daily report leaves only the weekend to catch up—that is, if your weekend traffic is light enough. The stock response is that this is a "Big Data" problem; throw the data into a noSQL datastore and then query that datastore. This doesn't actually solve the problem, however, it just pushes out the wall that you will be running in to a bit. There are easier and simpler ways to deal with the volume issue.

Dashboards

A dashboard is a screen (usually in a browser) that is intended to give you an at-a-glance summary of your system, usually with graphs, dials, and other graphical elements to present the data. Dashboards frequently, but not always, have drill-down capabilities, allowing you to get more information about a particular element being displayed. This is the type of thing that managers love and put on large screens for everyone to see. Properly used, they are a wonderful tool for providing an overview of the health of your system, but improperly implemented, they can be a huge performance headache. And if the performance is bad enough, dashboards can end up misleading people working on the systems, reporting the health of your system sometime in the past.

Dashboards take a hard problem, resource issues, and make it even worse. Dashboards are best thought of as predefined reports that are run repeatedly, by several people at once.

The most common problem is that these different people are not asking for the exact same report. If an element of a dashboard is reporting how many hits your Web server has had over the past five minutes and you have 20 people viewing the dashboard, you will produce 20 different sets of results because no two people have started the report generation at exactly

the same time (one person is looking at the data from 9:20:00–9:25:00, the next is looking at 9:20:10–9:25:10, etc.). This can become a catastrophic performance problem if the data that needs to be retrieved to produce these reports (the working set) is larger than the RAM that your reporting system has available to cache the data, as each report will need to retrieve the raw data from disk separately.

The next biggest problem with dashboards is that, because they display their data graphically, putting a lot of information on a page is easy, but each item is generated independently of every other item. This means that if you have one dial that shows the total number of hits to your Web servers, and another that shows the number of dynamic pages being accessed, they will each go through all your Web server logs separately for the reporting period to get their results.

And, finally, dashboards frequently refresh faster than the length of the time on which they are reporting. So a dashboard reporting how many hits your Web servers have had over the past five minutes, but refreshing once per minute, will count each minute five times (once in each of five different refreshes until the data has aged enough not to be relevant). Because different elements may show data covering different time frames, this cannot be addressed by just changing the refresh time.

I have seen dashboards created that refresh every five minutes, have 10 dials, graphs, or tables on them, with each item covering logs for a 24-hour period and summarizing hundreds of millions of log events. Each item alone is a terrible resource hog, and when combined into a single screen and refreshed together, they can crush even large farms of servers. This is why the noSQL datastore is not the full solution; it will let you throw more hardware at the reporting problem, but inefficient algorithms can outrun Moore's Law no matter what your budget.

Ad Hoc vs. Pre-Planned Log Reporting

Ad hoc reports look for things that you did not think of ahead of time, and pre-planned reports cover what you know you are going to need, and can therefore plan for ahead of time. Most of the strategies in this article can only be applied to pre-planned reports.

Ad Hoc Reports

Ad hoc reports are the sort of thing that members of your security department are going to want to do frequently. They get a report of a problem with a given account, and then want to look at all the activity that happened on that account in the suspected time frame. They will then want to do further investigation to see what other activity happened from the IP addresses used to access that account (frequently over a larger time frame), and then are likely to want to look at activity on other accounts that those IP addresses accessed. Like tugging on a piece of yarn in a sweater, this activity can widen and unravel lots of interesting things.

Ad hoc reports also are commonly used during troubleshooting. You start off looking for all logs relevant to the place you see a problem, look for logs related to that place on other systems, and run similar reports for a time frame when you didn't have a problem to see what looks different.

Unfortunately, the only way to optimize ad hoc reports is to try to segment the logs into categories that match the likely ad hoc reports you will need to generate, partition them by time so that you don't have to look at logs outside of the required time frame, and try to make searching through the logs as efficient as possible.

The simple approach to this is to split the logs by category (so that your firewall logs are separate from your Web server logs, for example), and then rotate the log files every minute. This gives you a reasonable base to start from to grep through the logs and find things you didn't plan. Make sure you keep a copy of the logs that isn't split by category; although log events can and will get reordered a bit as they are delivered, the order they arrive in is the best approximation that you will have of the order in which they are generated, and sometimes you need to see what happened across wildly different systems.

Your archive analysis farm is a good place to do this. Log everything to one file and then have a series of filters in rsyslog match a particular type of log event, usually by program name [2]. You may want to have more sophisticated filters, especially ones that use metadata that you've added, so that your production, DR, QA, and development logs are separated from each other.

Ad hoc reports are where the Big Data approach to log storage can be a wonderful win. If you can have your logs in some sort of structured storage with full-text indexing (such as Splunk, Elasticsearch, Hadoop, etc.), you can run queries against the logs much more rapidly than you can with grep against flat files; however, these Big Data approaches tend to be very resource hungry (and, therefore, expensive), and although they are absolutely wonderful for ad hoc reports, using them for reports that you know about ahead of time is far more expensive (and can end up being significantly slower) than taking other approaches.

Pre-Planned Reports

The solution to the problems of generating pre-planned reports efficiently is easy to articulate, but much harder to implement.

The Golden Rule of Reporting: Never process a log event more than once.

This is an ideal to strive for, but you need to recognize that you will never achieve this in practice. As a result, you need to look carefully at the costs involved and work to minimize the overall expense of generating the report.

Because you only want to examine a given log message once, the Big Data approach to logs is not appropriate. If you use Splunk,

Elasticsearch, or Hadoop to produce your reports, you end up sending multiple queries for the same logs or types of logs, retrieving them, and generating one report item per query. In addition to the fact that your multiple queries all have to retrieve the same data, you also have the problem that the logs that you need to query to get one answer are going to be intermingled with other logs that have nothing to do with the report you are interested in, and the systems will need to read those logs to get the logs that they need to respond to the query.

So instead of throwing all the data in one place and then querying it, the idea is to split the data as early (and cheaply) as possible. This benefits you in a couple of ways.

1. The volume of logs is going to be large enough that no single process can keep up, so you want to be able to split the work across multiple processes to take advantage of the multiple CPU cores in modern systems and, if needed, multiple systems.
2. The analysis that you will need to do on each type of log is going to be very different, so it makes your report definitions much simpler if a given report only needs to deal with one type of log.

After you have split the logs by category, you can have a process go through each category. This process should not generate the reports themselves, but should instead summarize the logs to generate the data that the reports are based on. These summaries can be fed back into the logging system so that all of your analysis engines can benefit from one system summarizing the data.

If you do not have dashboards to support, running reports hourly or daily is practical; however, dashboards are valuable enough that it is worth complicating your hourly/daily reports to be able to support your dashboards efficiently, too. To do this, frequently create summaries of the logs you know you are going to be reporting on. For example, if you have a set of Web servers that are generating hundreds of millions of lines of logs per day, but you produce per-minute summaries of these logs, your reports only have to query and parse the summary data, not the raw data. Because the data is per-minute, dashboard reports also stop being different for different people; everyone who gets a report in a given minute will see the same results. The summary data is also much smaller, easily fitting in RAM, so you are not going to have to do much disk I/O when generating the reports.

There are two fundamental approaches to producing summary data: (1) storing the data, then summarizing it or (2) processing the data in real time and summarizing it.

STORE THE DATA, THEN SUMMARIZE IT

With this approach, you write the data someplace (as per-minute flat files or in a Big Data system), then run the summary routines against this storage.

If you use flat files, compressing them is a good idea. Using gzip, I find that it's faster to retrieve the compressed data off of disk and then uncompress it than it is to retrieve the uncompressed data off of disk. This is because (1) system RAM can hold a lot more data in its disk cache when it's compressed, so uncompressing something that the system already has in RAM is more likely than needing to retrieve the data from disk; and (2) CPU power is relatively cheap, so any system that has lots of RAM and a high performance disk subsystem usually has extra CPU power available.

Using flat files, you can have your summary routine make one pass through the data for a time frame and produce all the different stats that you are interested in for that time frame.

If you use a Big Data system, you can schedule queries to perform the various queries against the datastore to produce the results that you need. This is far more expensive because each query will be run independently from the others, requiring the data be accessed multiple times, but if you are doing this every minute against the last minute's data, the data you are querying should all be in RAM, so you at least avoid the expensive disk I/O. In any case, this is far more efficient than having each report issue independent queries against all the logs for the time frame the report is interested in.

Note that if you are using a Big Data system, you are paying (in license costs with Splunk, and in processing overhead and hardware for all systems) for the volume of all the log events, even if you end up only querying the summary data. In most cases you are probably better off summarizing external to your Big Data system and only putting the results into that system.

PROCESS DATA IN REAL-TIME, THEN SUMMARIZE IT

Instead of storing the data and then querying it, you can have rsyslog deliver the logs in real-time to your summary routines, have them parse and count the logs as they arrive, and then dump out the summary data periodically.

With this approach, the ability of rsyslog to normalize the logs with mmnormalize should be looked at carefully. This module lets you define log patterns and extract variables from those patterns. Having rsyslog dump out the data in a nicely structured and easily parsed format for your summary scripts to deal with, and might end up being far more efficient than parsing the raw formats in your summary scripts.

The best way to do this sort of summary is to have rsyslog run your program and deliver the logs directly through stdin. The rsyslog configuration for this looks like:

```
Module (load="omprog")
action(type="omprog" binary="/pathto/omprog.py
--parm1=\"value 1\" --parm2=value2" template="RSYSLOG_
TraditionalFileFormat")
```

Your program needs to exit when stdin gets closed, otherwise you will end up with a copy of it running after rsyslog restarts.

Note that if you use the old config format, you cannot have any spaces in the command line, so you will probably need to use an external script to start your program. Because you only need to do this on your analysis farms, you can be running a current version that supports the new syntax.

If you write your own summary script, you must have some method of having your script output its data on schedule. This can be as "simple" as having a cron job send it a signal and having a signal handler dump the data out and reset counters; however, you don't have to write this yourself. Simple Event Correlator (SEC) works well for this task and includes the ability to do things at specific times [3].

For example, the following SEC config file looks for Cisco ASA http log entries; creates a log entry containing total HTTP requests, number of servers accessed, and number of URLs accessed; then creates one file containing all the URLs accessed (and how many times they were accessed) and a second file for the servers accessed:

```
## On startup, zero the counters
type=Single
ptype=RegExp
pattern=(SEC_STARTUP|SEC_RESTART)
context=SEC_INTERNAL_EVENT
desc=Init counters with 0
action=eval %o %counters=()

type=Single
ptype=SubStr
pattern=%ASA-5-304001 \S Accessed URL ([^/])([^?^ ]+)
desc=gather most frequently accessed URLs
action=eval ( $counter{urls}{%1%2}++;
$counter{httpconnections}++; $counter{servers}{%1}++)

## output summary data and clear stats every minute
type=Calendar
time=* * * * *
desc=output summary data
context=!SEC_INTERNAL_EVENT
action=eval %a (scaler keys $counter{urls}); \
   eval %b (scaler keys $counter{servers}); \
  udgram /dev/log <30>summarydata: \
  CiscoLogCount=$counter{CiscoLogCount} \
  HttpConnectionCount=counter{httpconnections} \
  URLsAccessed=%a ServersAccessed=%b ;\
eval %o ( \
  open(output,">/var/log/urlcount"); \
  while (($key,$value) = each %counter{urls}) { \
    print "$key=$value\n"; \
  }; \
  close(output); \
  open(output,">/var/log/servercount"); \
  while (($key,$value) = each %counter{servers}) { \
    print "$key=$value\n"; \
  }; \
  close(output);\
  %counters=(); \
)
```

Using the Summary Data

Dashboards

If you use this summary data to drive the dials and graphs for your dashboards, you can cheaply create the dials and graphs, so when a lot of people want to look at the dashboard, it won't take your system down.

Reports

You should create reports for people using this data. Instead of creating reports structured around particular data sets, you should create reports structured around the needs of the user of that particular report, because aggregating the summary data, making calculations using that data, and inserting the data into a report is cheap.

Alerting

One obvious thing you can do is have a tool like SEC alert you if these numbers cross a given threshold. There are limits to how useful this is, however; numbers that might worry you at 2 a.m. on Sunday because they are so high that they indicate something is wrong or you are under attack, may be numbers that you would also want to be alerted to in prime-time on Monday morning because they are so low that they indicate that something is broken and you aren't serving your users. Such alerting is useful, but in practice is limited to notifying you when you are exceeding capacity.

Aberrant Behavior Detection

The round-robin database tool (RRDtool) [4] not only makes producing a wide range of time-based graphs to display the summary data you have generated easy, but it also has the interesting ability to take the historic data that you feed it, predict a range in which new data should fall, and flag when the new data is outside of this range. This uses the Holt-Winters Time Series Forecasting Algorithm to predict what the next value should

be, and allows you to calculate confidence limits from this so that you can do things like alert if the measured value is more than two standard deviations away from what is expected. This algorithm will detect repeated patterns in your data so that it not only matches your daily usage pattern variation but, once it has about 10 cycles worth of data, can detect the difference between weekdays and weekends, for example, while still accounting for a continuing increase in use over time. The details of how to do this are out of scope for this article, but there is a good writeup at http://cricket.sourceforge.net/aberrant/rrd_hw.htm.

Artificial Ignorance

In addition to counting how many times something happens, another useful report to have is an "unknown log report" of the type produced by the "artificial ignorance" approach described by Marcus Ranum [5]. This consists of deliberately filtering out log entries that you understand, then prioritizing the remaining log entries based on what shows up the most:

1. Filter any log entries that you want to report on to a process to generate the appropriate report for that type of log entry.
2. In what's left, filter out any log entries that you know are not important, but count them and report this count. If the number of times that an insignificant event happens changes drastically, this may be significant.
3. Take what's left and sort the logs based on their contents, and produce a report that shows the most common logs.

A person can then look at this report and quickly spot strange things that have happened.

Taken to the extreme, you can tune your artificial ignorance report to the point that you have no logs in it at all. At that point, anything that shows up in the report becomes significant.

Getting there is a lot of work, and you quickly reach the point of diminishing returns. Even on a large network, surprisingly few different log entries are produced. Large networks tend to have a lot of the same thing on them, so once you identify what should be done with a given log entry, you don't care whether you have two servers producing that log entry or 2000; either way, it's handled. A few days' worth of effort filtering the log messages probably can get you down to a report that shows you events that have happened fewer than a dozen times in the first couple of pages of the report. Also, the report probably will show you some errors that are happening on your network that you were not aware of and want to fix before going a lot further.

Running separate artificial ignorance reports against each category of log messages is best. Dump all the messages that don't match your reporting rules into a file for that category and then periodically run this data through a filter along the lines of:

```
cut -c 17- |sed -e s/"port [0-9]* "/"port PORT "/g \
  -e s/\[[0-9]*\]/"[PID]"/g -e s/"pid=[0-9]*"/pid=PID/g\
  |sort |uniq -c |sort -rn >other-logs.report
```

You may find that on your network, there are some other fields that are in frequent log messages that make otherwise identical messages look different, which is what the sed statement in this filter chain is addressing.

Then take a look at the results. If you have a log message that shows up a lot (or a lot of similar log messages that show up a lot), add a rule to match them. Repeat until you can scan the entire report in a short enough time that you no longer care; you don't need to drive it all the way to empty.

Summary

Producing dashboards and reports from a high volume of logs—and doing so efficiently—is possible, but if you are not careful, you easily could find yourself with a system that is orders of magnitude larger than you would need for efficient generation, and still running into performance problems.

Split up the work, and try to make it so that no log ever needs to be examined in detail more than once, and try to limit the number of times it must go through a filter.

At this point, I have covered the basics of a full enterprise logging system. In future articles I will go into the various topics in more detail, which includes covering performance tuning of different tools. If you have specific topics on which you would like me to focus, please email Rik Farrow (rik@usenix.org) or me and let us know.

References

[1] "Enterprise Logging," *;login:*, vol. 38, no. 4, August 2013: https://www.usenix.org/publications/login/august-2013-volume-38-number-4/enterprise-logging.

[2] "Log Filtering with Rsyslog," *;login:*, vol. 38, no. 5, October 2013: https://www.usenix.org/publications/login/october-2013-volume-38-number-5/log-filtering-rsyslog.

[3] "Using SEC," *;login:*, vol. 38, no. 6, December 2013: https://www.usenix.org/publications/login/december-2013-volume-38-number-6/using-sec.

[4] RRDtool: http://oss.oetiker.ch/rrdtool/.

[5] Artificial ignorance: http://www.ranum.com/security/computer_security/papers/ai/.

Loser Buys
A Troublesome Retrospective

MARK BAINTER AND DAVE JOSEPHSEN

Mark Bainter is the Director of Global Support and Operations at Message Systems Inc., where he works with a talented team of sysadmins to provide support and outsourced monitoring and management for Message Systems' cross-channel messaging software. In his spare time he currently enjoys woodworking and tinkering with Arduino hardware.
mbainter+usenix@gmail.com

David Josephsen is the sometime book-authoring developer evangelist at Librato.com. His continuing mission: to help engineers worldwide close the feedback loop. dave-usenix@skeptech.org

Dave: Rik Farrow wants some troubleshooting stories for the next *;login* issue. Loser buys?

Mark: You sure about that, son? Need to check your balance first or anything?

Dave: Just for that, I'll let you go first.

Mark: Your funeral. First, a bit of background—the company I currently work for has a tool called "Gimli" [1], which you can link against your application and then use to ensure that it remains up and functioning, similar to "supervise." Gimli also does heartbeat monitoring, however, and if it crashes or has to be killed and restarted for being unresponsive, will take a snapshot of the stack, along with various details about the process for later review.

Dave: Nice. I bet that comes in handy.

Mark: Oh, it does. It's helped us resolve all sorts of weird issues that would otherwise require you to sit and watch and wait for the problem to happen. Best of all, you don't have any of the unpleasant side effects of some alternative methods of catching intermittent issues. Not to mention increasing the resilience of your application.

Anyway, a while back, in the middle of the night, I get an alert—the /var partition on one of the servers I manage is filling up quickly. That's definitely not good. Investigating, I find the disk is already nearly full, and in moments I find the culprit to be the log file for this Gimli process. The process it's managing is wedged, and Gimli is in a loop logging the same error over and over into the log, reporting that it has received a TERM signal and is terminating. That's really odd—I've never seen this failure condition before. I kill both processes, clean the Gimli log, and restart.

Reviewing the logs subsequently offers no clues as to what happened. There's no stack trace either. Curiouser and curiouser. I don't like unexplained activity like this, but it's the middle of the night and I'm at a dead end. I turn in.

The next morning it happens again. Same time. Now I have a hint. After I restore service, I start looking through the application's activity during that window, into the system logs, cron jobs, etc. It doesn't take me long to correlate the log rotation with the time window where this is occurring.

Dave: Ah, 4 a.m., when the logrotate bugs come out to play.

Mark: Exactly. This process that Gimli is monitoring is set up with a fairly standard daily rotation, followed by compression and then a postrotate instruction to send a HUP signal to force reopening of the logs.

I spin up a VM and start doing some testing and at first I can't reproduce the problem. I run the log rotation and everything works fine. Then it hits me. Some time back I made a modification to the logrotate script! By default we were not rotating the error log for this process, because it is almost never written to. This node, however, was throwing a lot of errors which another team had been investigating, so in the interim I had set up log rotation to keep it from filling the disk.

I add the path to the error log in the logrotate script on my virtual machine, rotate the logs, and sure enough, the rotate failed, and the log was filling up. Now I have a readily reproduc-

ible problem. Of course, this still doesn't explain the why, or the why now. After resetting the test, I do some tests with GDB, which is frustrating because the heartbeat method used by this app was sending a USR1 signal which kept causing GDB to stop.

Dave: You know, you can set a nostop pass signal handler in GDB to get around that [2].

Mark: Yeah, but at the time I wasn't aware of it, and there's another favored tool to reach for that could readily report those without being interrupted—namely, strace. In short order I discover that there are actually two HUP signals being sent in short succession.

Dave: Right. Logrotate would have sent one signal for each logfile unless you set the sharedscripts option.

Mark: Yeah, well, I didn't remember that when I set up the config, but I remembered it now. The full explanation here of what was happening requires some understanding around how the Gimli process interacts with the processes it monitors, so I'm going to gloss over some of that for the sake of not boring our readers. Basically, when Gimli saw the HUP come in, it created a new version of itself to take over monitoring the process, but the second HUP came in before that execve could complete. As a result, the two copies of Gimli would become confused, and continuously try to kill each other in a vicious loop, resulting in flooding the error log with the termination messages. Since neither would honor the TERM signal fully as a protective measure for a monitoring process, the loop never ended. Thankfully, more recent versions have addressed this weakness.

Dave: Heh, if you'd named it "Claudius" instead of "Gimli," it might have been more adept at fratricide. Okay, so now I understand the why, but I'm confused why it suddenly started happening. Wouldn't it have begun delivering a double HUP as soon as you first modified logrotate? Why didn't it happen right after you made the change?

Mark: That's the real kicker isn't it? Luckily, this was the easy part to figure out. When I first implemented it, as I said, this node was throwing a lot of errors, forcing me to implement the rotation. Since then, the problem causing those errors had been fixed, unbeknownst to me.

Dave: Oh! It was a race condition. When the log files had content, the time logrotate spent copying and compressing them would've given Gimli enough time to fork, so everything was fine. It was only when the log files were empty that the HUPs would win. Nice!

Mark: Got it in one!

Dave: Damn...that's a great story, and I'm not sure I can top it, but here's my favorite troubleshooting story. I like it because there's a bit of dramatic panache at the end.

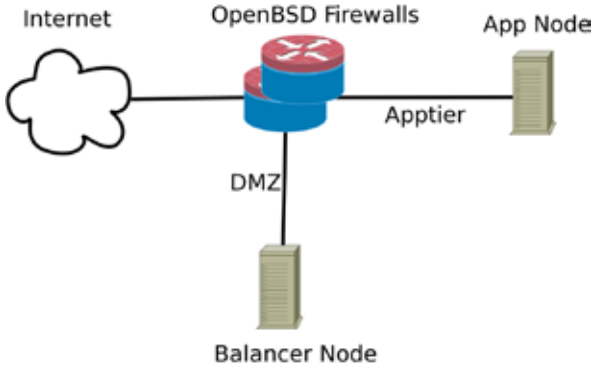


Figure 1: In Dave's troubleshooting conundrum, packets were disappearing somewhere between the load balancers and app servers. One of the pair of firewalls is a backup, using CARP to share state with the active firewall.

Anyway, we were having trouble with a Web application that we'd just put in production. The setup looked like Figure 1. The problem presented as some intermittent latency when using the app. Sometimes it worked fine, other times it was very slow, and still other times it didn't work at all. And this wasn't like, over the course of one hour it'd be slow and the next it'd work fine, this was like, one HTTP request might work fine while two others executing concurrently did not.

Mark: Sounds fun.

Dave: It wasn't. And for the first and only time in my professional career, when the developers started blaming "the network," it looked like they were actually right. Working our way down the stack we were pretty convinced that packets weren't getting to the application server. Somehow this network was eating packets, and obviously it wasn't any sort of ACL or filtering stuff because some requests were making it through just fine.

So here's the background you need to understand: the internal core routers were two OpenBSD systems running CARP (Common Address Redundancy Protocol) with pfsync. It's pfsync's job to replicate the firewall state table between the master and failover nodes, such that, if they ever fail over, the failover router will know about all the active and authorized connections. Without pfsync, the backup router would drop all the existing network connections and force them to re-handshake. We have a physical network port on each router configured specifically just for pfsync, and the two routers are directly connected to each other via a crossover cable on this port.

CARP creates a magical virtual network device whose status is shared between the two routers. CARP is what actually enables the backup router to detect and take over for a failed master router without the MAC or IP address changing for other network entities.

The balancers meanwhile operated using a multicast MAC address...

Mark: Was there IGMP?

Dave: No, and that's important. As you know, in the absence of IGMP the default behavior of a Cisco router is to broadcast the multicast packets to all live switchports in the same VLAN.

Mark: I think I see where this is going.

Dave: Shut up. Anyway, there I was, stumped. I could tcpdump the traffic as it came from the Internet to the balancers. That looked okay. But only a subset of the traffic was making it to the application tier nodes. It was maddening because I couldn't seem to narrow it down to any one device. Watching the firewalls carefully I could see that they weren't failing over. Watching the packet traces in the apptier, it looked like traffic worked fine for a while and then every so often, a good connection would just freeze. Eventually, when this happened the apptier node would send an RST and stuff would start working again. The balancers seemed to be getting traffic okay, but they were also freezing and RSTing every so often.

I had a bit of an ah-hah moment when I started looking at the packet traces on the failover firewall. It appeared to be getting a copy of all the multicast traffic that was destined for the balancers. This was odd because in CARP backup mode, the failover router isn't answering ARP on its CARP virtual devices, and should therefore not receive any of the traffic for those shared IPs.

Mark: Even if the failover was getting traffic, it shouldn't be routing it.

Dave: Exactly my thinking. Evidently the traffic was appearing on the backup firewall because in the absence of IGMP, the Cisco 3750 was broadcasting that traffic to all active switchports in the VLAN, including those of the failover router. But that traffic should be harmless anyway since the failover router would just drop it all on the floor. I was back to square one.

Mark: Or were you?

Dave: Or was . . . shut up. I stared at the rack a few minutes, trying to imagine every possible path a packet might take through this rather simple little network, and something interesting occurred to me when I imagined what might happen to me if I were one of those multicast packets that had been duplicated to the failover firewall. The interesting thing was that I would wind up in the inbound packet buffer on the firewall's DMZ port while the firewall checked its state table and ACLs. Our assumption that the traffic wouldn't be forwarded is based on the fact that the backup firewall wouldn't have a state table entry for the connection in question.

Mark: Right, the failover firewall would compare the source and destination addresses of the packet to its internal list of existing states, and then, not finding one, it would drop the packet.

Dave: Except OpenBSD's pfsync service replicates that state table between the master and failover CARP nodes. The failover router has every active state that the master does, and therefore DOES in fact have a state table entry that matches the packet. So there's an interoperability bug between Cisco and OpenBSD pfsync . . .

Mark: OpenBSD assumes the Cisco won't give it a packet it doesn't ARP for . . .

Dave: Yes, exactly, and Cisco assumes OpenBSD isn't going to forward a broadcast packet because it won't exist in its state table.

Mark: So why isn't there a broadcast loop that affects the master firewall node? Wouldn't the master also receive a copy of the multicast packet?

Dave: No, because the master firewall is the default gateway device for the network, so it's the switchport that originated the traffic, and will therefore not receive a copy of the broadcast.

Mark: Man, that's hairy. What happens when the failover node tries to route that packet?

Dave: I don't know exactly. It's undefined, but in that network, intermittent latency and lots of RST ensued.

Anyway, here's the best part. I have this big eureka moment, and jump up out of my chair excitedly describing my hypothesis to the other engineer who is working the problem with me. He thinks I might have it solved, but wonders out loud how I'm going to test whether I'm right or not. Not answering, I walk over to the rack and with as much showmanship as I can muster, ceremoniously rip out the pfsync cable connecting the two routers. TA-DAAA problem solved!

Mark: Nice, but what about clean failover?

Dave: It's not really an issue for us, because we usually use Pix's on the edge, but you could manually configure the switch ARP-table so they didn't broadcast, or you could use IGMP if possible, or, yeah, just run the firewalls without pfsync, which might bite you later on, but not very much. The network would "hiccup" whenever they failed over, and the users who did get an error could hit the reload button and everything would be fine.

Dave: Well, I think you probably won that one. I mean you had interprocess warfare and GDB!

Mark: Really? I kind of liked yours because I might one day try to run PF and mod_proxy_balancer with Cisco switches, and you probably just saved me a headache.

Dave: Well, Rik, we'll leave it to you. Who's buying?

Rik: Dave, you're buying. While both stories are good, Mark did a better job of explaining exactly what had gone wrong, as well as having more twists and turns. Your solution, breaking the con-

nection between firewalls, fixes the problem without telling us exactly what was going wrong. Not that figuring that out would be easy, as it likely lies in the IP stack of OpenBSD somewhere.

References

[1] https://github.com/MessageSystems/gimli.

[2] http://sourceware.org/gdb/onlinedocs/gdb/Signals.html.

When Data Is a Risk

Data Loss Prevention Tools and Their Role within IT Departments

KLAUS HALLER

Klaus Haller's work focus is on IT risk, compliance testing, and test organizations. He has concrete working experience with Symantec's Data Loss Prevention tool and brings an infrastructure and operations as well as a business analysis perspective. He has been with Swisscom IT Services since 2006 and worked as a consultant with various customers, mainly in the banking industry. He is a frequent conference speaker and publishes in various magazines. klaus.haller@swisscom.com

Snowden is a reversal point for IT security and risk. Before him, many saw IT security as equivalent to a medieval town wall: keeping outside hackers and malicious code away from the company. Firewalls, virus scanners, and application security testing (e.g., to find SQL injections) fit the town wall approach. But Snowden was different. He was from the inside of the organization. He collected large amounts of sensitive data. Then, he got the data out of a highly secured IT organization, which had to learn from the press about the case. In this article, I will explain such data-related risks in IT departments and how data loss prevention (DLP) tools help to manage them.

Understanding the Business Risks

Computer professionals think in terms of technical components: operating systems, applications, and databases. In contrast, data-related risks require a business view. First, there is the risk of not adhering to regulations. Second, there is the risk of losing competitive advantage due to data leaks. Third, as a side effect of the two previous risks, security incidents might harm an organization's reputation.

A data leak means that sensitive data, such as customer lists or cost calculations, leave the company. Other examples are engineering drawings stored in CAD systems, research data in pharmaceutical companies, or source code in the software industry. If companies lose such data to competitors, this threatens their position in the market.

The focus of data-related regulatory risks is customer data. The risks correlate especially with a worldwide customer base, outsourcing, or global work distribution. There are standards such as the Payment Card Industry Data Security Standard (PCI-DSS) or the Health Insurance Portability and Accountability Act (HIPAA). There are European or Swiss data protection laws and the EU-US safe harbor agreement. They impact whether data can be transferred to a subsidiary or to sourcing partners in the same or in a different jurisdiction. Violating any of the regulations can harm the reputation and result in interventions of regulatory bodies and fines. When employees violate laws, even if instructed to do so by their superiors, there is also a direct personal risk for them.

Risks in Development, Test, and Production Environments

Even if systems are engineered and operated securely, and IT and business enforce the need-to-know principle with roles and a strict user management, the data-related risks remain. Their root cause is normal users using their normal access rights, just not as intended. Table 1 matches abstract business risks with IT security incidents, for which concrete solutions can be defined.

The first business risk is that sensitive data leave the company. This can happen by mistake. For example, a user sends an email to a wrong person or attaches a wrong file. There is also the risk of transferring data outside of the company as part of industrial espionage, e.g., by sending data to a personal account or copying it to a USB stick. Business users working in production are the source of such risks as are engineers in development and test. The latter can

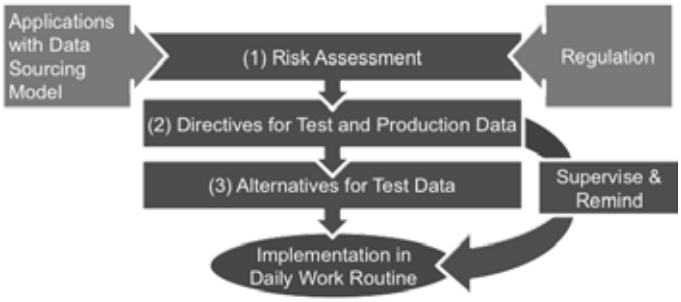


Figure 1: The three preparation steps for effective data-related risk mitigation

be even a higher risk. In production, the need-to-know principle is often enforced strictly. This reduces the number of persons who can misuse (large amounts of) sensitive data. Also, direct database access is limited to the small group of admins. In development and test environments, by contrast, engineers often have access to applications without any authentication. To make things worse, they can connect to the database directly and submit SQL queries. If a test database contains production data, engineers can extract, for example, a complete customer list with one single query. Still, there is a reason why many test environments contain production data: engineers need appropriate test data and database copies are a convenient solution.

The risk of violating regulations is obvious in production environments. A server with customer data must be placed only in datacenters in an appropriate jurisdiction. Centralized server provisioning reduces the error risk in production. The same risk exists in development and test environments, however, for which outsourcing and offshoring is much more common. Here, decisions are often made in a decentralized way. This increases the risk that sensitive production data gets into offshored or outsourced development and test environments via a database copy, file transfers, or manually entered data.

The regulatory risks increase if a company's business spreads across various countries. Consolidated datacenters and centers of excellence for certain areas (e.g., payroll processing) reduce costs; however, the more production data are transferred around the globe, the higher the risk of violating regulations.

Aimless Activism vs. Effective Risk Mitigation

When companies and risk managers understand the data-related risks, some managers might think about writing an email including the following three policies:

- Customer data must remain within our country!
- Sensitive data must not be copied into test environments!
- Intellectual property and customer data must not leave the company!

Such an email may increase awareness, but most of all, it causes confusion. These policy statements are ambiguous. On one hand, it is unclear what exactly is prohibited. "Sensitive data" is a broad term. On the other hand, developers and testers do not know what they should do instead. They rely on adequate data for their work. Thus, before sending such emails, managers should go through three preparation steps (see Figure 1). In the first step, the legal and the IT risk departments together assess the risk. Which data are sensitive from a business and a regulatory point of view? The outcome is a list of the risks with the severity of a potential incident and the probability of an occurrence.

The second step is to elaborate a directive for production and test data. The management must decide which risks it accepts. The directive must provide a data classification scheme, which explains in detail which data is defined to be sensitive. It must identify suitable datacenter locations and state whether outsourcing is possible and to which partners and jurisdictions. The directive should also define for development and test environments which data can be transferred to whom, in which jurisdiction testing is allowed, and details regarding test data anonymization (if applicable). What must be anonymized? Is it sufficient to delete the customer names only? Are customer addresses sensitive as well? What about booking texts or contracts with suppliers?

Such a production and test data directive restricts the work of developers and testers. Thus, the third step is about providing alternatives. Synthetically generated test data or (very good!) anonymized data from production environments could replace the complete database copies from production to development and test environments (see [1] for details). These alternatives might require new tools, can change the organization and its teams and processes, and, thus, can tie up resources and take time.

Business Risk	Production	Development and Test
Competitive advantage	Data loss by mistake	Data loss by mistake
	Data loss due to criminal act	Date loss to criminal act
Regulatory	Datacenter / production servers placed in inappropriate jurisdiction	Data transferred to / typed in an inappropriate jurisdiction or sourcing partner

Table 1: Risk types and environments

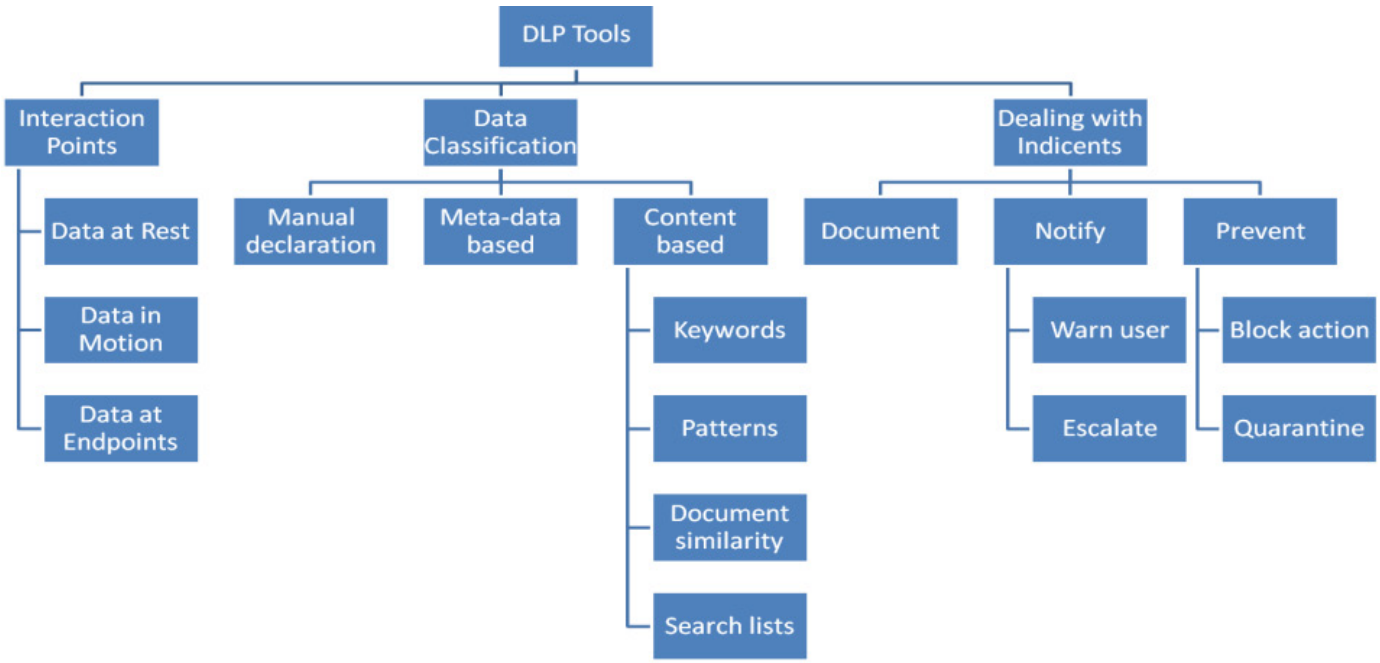


Figure 2: Characterizing DLP tools

When all three steps are completed, the management communicates the directive together with the alternatives for testers and developers. From this moment on, users in production as well as testers and developers must follow the directive; however, IT environments are similar to teenagers' rooms. Forcing them to clean up once does not ensure that everything is perfect for the next weeks. Regular checks for sensitive data are required, for which data loss prevention (DLP) tools can help.

Helpful DLP Tools

Various vendors offer data loss prevention tools. The market is dynamic and features vary. There are comprehensive solutions from the big players, such as McAfee and Symantec, or from smaller vendors, such as myDLP. Others focus on niches (e.g., Proofpoint or Microsoft Exchange). Three questions help to characterize a product or a concrete installation (Figure 2):

1. What do DLP tools look at (interaction points)?
2. How do they identify sensitive data?
3. What options are provided to react to incidents?

Three options exist for the interaction points between the DLP tool and the IT infrastructure (Figure 3):

- Data at Rest. The DLP tool searches for sensitive data in files, SharePoint servers, databases, or other kinds of repositories. The idea is to find sensitive data at places that nobody is aware of. Certainly, enterprise resource planning systems store sensitive data, e.g., customers, costs, and profits. But in many companies, critical data exist in many files as well—e.g., Excel spreadsheets.
- Data in Motion. Data are transferred within the company and to outside recipients via the network, e.g., by emails, FTP, or social media. The DLP solution can monitor the zone-internal network traffic for sensitive data as well as the traffic to other zones or the Web.
- Data at Endpoints. Here, laptops, PCs, and mobile devices are the focus. They can get lost with data on them or data can be copied from them to removable devices such as USB sticks. So it is desirable to monitor data downloads as well as data-related activities on endpoints.

When DLP tools detect sensitive data in an email or in a user's spreadsheet, they must react. Standard options are:

- Log and document the security incident in the DLP tool or in a central information security management system (ISMS).
- Notify users when they try to perform noncompliant actions, or escalate such incidents to their line manager or the HR department.
- Prevent wrongdoing by blocking actions (e.g., emails for data in motion, or downloads for data at endpoints) or quarantine files by moving them to a secure folder (data at rest).

Blocking wrongdoing seems to be the best idea, but it is not always true. If the DLP tool blocks half of the employees' emails "to be on the safe side," the DLP tool will be switched off within minutes. Thus, a first phase is always about improving the rules for data classification. But even afterwards, notifying the user or writing logs and evaluating them periodically remains the option-of-choice for less severe incidents.

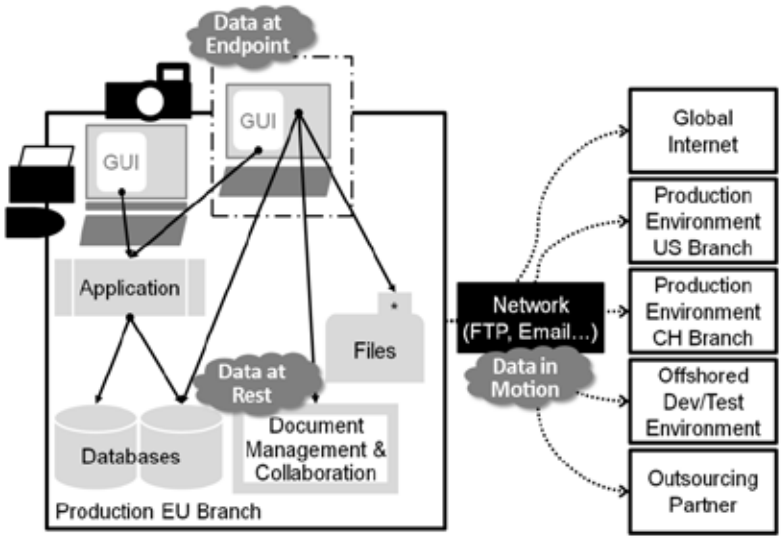


Figure 3: Interaction points in a multinational company with various network zones

The biggest challenge is data classification. The DLP tool must decide whether data, emails, files, etc. are sensitive. Options are:

- Manual declaration. A risk manager tags a folder or files as sensitive. From this moment on, the DLP tool prevents screen dumps, for example, when a sensitive file is shown on the screen.
- Content-based classification. The DLP tool looks inside files or emails. The main techniques are (1) keyword search, (2) document similarity, (3) patterns, and (4) search lists. Keywords are strings whose appearance signal sensitivity to the DLP tool, e.g., a term "strictly confidential." Document similarity means that the DLP tool has a collection of sensitive files, for example, templates for offers or contracts. Similar files are assumed to be sensitive. So the DLP could be triggered if a user tries to send out hundreds of contracts. Identifiers such as credit card numbers or social security numbers often have a specific format, e.g., four digits, a space, four digits. Patterns allow searching for such identifiers in emails or files. Finally, search lists provide a list of sensitive data items, such as all customer email addresses or customer credit cards. If one item appears in a file or email, for example, the DLP tool raises an incident.
- Metadata-based classification (e.g., names or IP address ranges) helps when deciding about sensitiveness. They can be combined with content-based strategies. Then, emails with sensitive data can be sent within the company, but the DLP tool prevents such emails from being sent out.

The big challenge is to configure the DLP tool such that it finds "real" incidents without raising many false alarms. Database/SQL developers might help more than security consultants with a background in firewalls and virus scanning.

DLP Features and Risk Reduction

The DLP tool can reduce the risk of criminal or accidental disclosure of sensitive data with its data-in-motion and data-at-endpoint features (see Table 2). They identify and block such data transfers via the network or to mobile devices and USB sticks. The data-in-motion features monitor data transfers by email or file to other jurisdictions, to sourcing partners, or to development environments. Database transfers can be tricky for DLP tools. In this case, the IT department's database copy process must prevent inappropriate data transfers. Still, DLP tools can help in assessing whether a database contains sensitive data. As stated in the example with the teenager's room, periodic checks are needed, especially in test and development environments.

The data-at-rest features can reduce the overall exposure to data-related risks. Periodic sanity checks of file systems or SharePoint servers can find unofficial data collections. Cleaning them up means less sensitive data will be floating around. This

Concrete risk	How DLP helps	Data at/in...		
		Rest	Motion	Endpoints
Data loss by mistake Data loss due to criminal act	Inappropriate data transfers (e.g., email, FTP, mobile devices, or USB sticks) identified and blocked		X	X
Datacenter/production servers placed in inappropriate jurisdiction	n/a			
Data transferred to/held in inappropriate jurisdiction	Entry check during file/database transfers	(X)	X	X
	Periodic sanity check of files/databases	X		
(Spread of sensitive data)	Periodic sanity check of files	X		

Table 2: How DLP helps reduce risk

lessens the risk that such data get lost or transferred to a wrong jurisdiction, environment, or outsourcing partner.

Limitations and Risks

DLP tools are 100% reliable when searching for a 30-chars long, alphanumeric string such as UAW-47594W48406DE488242O34333W. Searching for all emails or documents about a customer or patient is much more difficult. If the name is "Peter James Miller," how might the contacted person react to reading the salutation "Dear Mr. Miller"? Or to "Peter Miller" or "Dear James Miller"? And how do you ensure that you do not confuse "Peter James Miller" with a non-sensitive name "Peter Max Miller" or "Peter Miller"? Similar to any information retrieval system, the DLP tool must balance the risk of not finding certain incidents and the risk of raising too many false incidents. Important terms are recall (if there are 100 incidents that should be found, how many will you find?) and precision (out of 100 incidents raised, how many are true incidents?). It means balancing the risk of leaking important data and violating laws against having too many incidents, which cannot be handled. No company can afford to have an IT security officer read every second email, not to mention the impact on the work environment. One sub-problem is format issues. So what happens if a social security number "123-45-6789" is written as "123456789" or "123 45 6789"? Companies can enforce standards for the data in databases, but this is nearly impossible for emails and Excel or Word documents.

Besides the limitations, there are several risks: laziness, non-adequacy, circumvention, and data loss of the DLP tool. Laziness reflects that users start relying on the DLP tool instead of thinking for themselves about what is allowed. This is dangerous because DLP tools cannot find all critical events. Non-adequacy means using a DLP tool to clean up files and data. DLP tools are good at detecting a broad variety of violations, but when DLP tools are used to clean up exactly the data items the DLP tool finds, there is a nearly 100% probability that sensitive data remains, which the DLP tool did not and will never find. Only a root cause analysis of the incidents leads to an understanding where and why sensitive data shows up at the wrong places. Circumvention means that users, especially with criminal intentions, search for ways to fool the DLP tool when they learn how the DLP tool works in detail.

Finally, the biggest risk can be the DLP tool itself. If it stores customer lists or sensitive documents to find copies or similar documents, losing data from the DLP tool becomes the worst case scenario. This includes, first, the direct loss of unencrypted, sensitive data such as customer or credit card lists. Second, there is the risk of telephone book attacks. For example, the DLP tool might be directed to a large list of potential client names, creating an incident for each real client name. So the set of incidents is the full client list. Third, even if lists and documents are encrypted or hashed, they must be highly protected and must never end up on mobile devices. If the DLP tool is not open source, it is never clear how strong the encryption is and whether attackers related to governmental agencies have back doors to break the encryption.

In conclusion, data loss prevention tools enable companies to detect and prevent inappropriate data handling. This allows companies to address regulatory risks and risks related to the loss of intellectual property.

References

[1] K. Haller, "Test Data Management in Practice: Problems, Concepts, and the Swisscom Test Data Organizer," Software Quality Days 2013, Vienna, Austria.

Using a Database to Store Data Captured with tcpdump

MIHALIS TSOUKALOS

Mihalis Tsoukalos is a UNIX system administrator who also knows programming, databases, and mathematics. Follow Mihalis on Twitter: @mactsouk
www.mtsoukalos.eu

In this article, I show you how to store network data in a MySQL database and how to take advantage of the SQL language to query stored data. I know that this has been done before, but I thought that it would be a good exercise to create my own solution.

Although I selected the MySQL [1] DBMS, you can use PostgreSQL, Oracle, or even a NoSQL database such as MongoDB [2]. For the network data capturing, I will use the trusty tcpdump [3] utility.

Advantages

Storing your network data into a database has many advantages, including the ability to query your data offline, being quicker than pure disk I/O because databases store their data optimized, embedding intelligence in your data by using a database, distributing your network data in many databases, using the reporting tools that support your database, easily querying your network data if you already know SQL, and giving a remote user access to your data using network access to the database.

Storing your network data in a database has some disadvantages, as well, including the fact that if you have a busy network, the database storage you will need will be big. Also, you will need a person to administer the database, if you do not already have one.

The Solution

Before being able to use any network data, you must first collect network data using the tcpdump utility. You can also use WireShark [4] or tshark [5] to capture network data, but tcpdump is more popular for network capture.

A Perl script will be used for selecting and inserting the data into the MySQL database. The DBI and DBD::mysql Perl modules must be pre-installed on your UNIX system for the Perl script to work. The script also uses the Data::Validate::IP Perl module for catching erroneous IP addresses.

The Perl script implies that the network data is already captured with tcpdump. If you want to tell tcpdump to capture 2000 network packets and exit, the following command will do the trick:

```
# tcpdump -i eth0 -c 2000 -w login.tcpdump
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture
   size 65535 bytes
2000 packets captured
2004 packets received by filter
0 packets dropped by kernel
```

You can alter the captured packets by adding more command-line arguments to the tcpdump command. Please note that you usually need root privileges to capture network traffic from a network interface.

The Perl script uses the following tshark [9] command to convert the network data into a more readable format:

```
$tshark -r login.tcpdump -T fields -e frame.number -e
 frame.time_relative -e ip.src -e ip.dst -e
 frame.protocols -e frame.len -E header=y -E quote=n
 -E occurrence=f
```

The format of the exported text file will look like the following:

frame.number	frame.time_relative	ip.src	ip.dst	frame.protocols	frame.len
1	0.000000000	109.74.193.253	2.86.13.236	eth:ip:tcp:ssh	194
2	0.011654000	174.138.175.116	109.74.193.253	eth:ip:udp:dns	97
3	0.011733000	109.74.193.253	174.138.175.116	eth:ip:icmp:ip:udp:dns	125
4	0.048020000	208.67.217.17	109.74.193.253	eth:ip:udp:dns	106
5	0.048107000	109.74.193.253	208.67.217.17	eth:ip:icmp:ip:udp:dns	134
6	0.055475000	2.86.13.236	109.74.193.253	eth:ip:tcp	66
7	0.081615000	83.145.248.129	109.74.193.253	eth:ip:udp:dns	75
8	0.081692000	109.74.193.253	83.145.248.129	eth:ip:icmp:ip:udp:dns	103

As the `frame.protocols` column may contain many values, we will use the last one.

The Implementation

The table that holds the network data contains a field called "id" that auto increments and acts as the primary key for the table. The "packetNumber" field is the packet index for a given network traffic capture, whereas the "dt" field is the time that the packet appeared since the first packet of the given network capture. The "sourceIP" and "destIP" fields contain the source and destination IP values, respectively. The "protocol" fields holds the protocol name of the network packet. Last, the "length" field holds the packet size in bytes.

You can choose to include additional fields depending on your network and the problem(s) you want to examine.

Importing the data into the MySQL data is also performed by the Perl script. If you have a fast computer and a network without extremely high traffic, you can even store your network data in (near) real-time.

The Perl script is called netData.pl and takes a single argument that is the name of the file that holds the tcpdump network data.

Querying the Database

The Perl script executes all the following queries and displays their results.

Find the Connections per Protocol:

```
mysql> select count(protocol), protocol FROM NetData GROUP BY protocol;
+-------------------+-----------+
| count(protocol) | protocol |
+-------------------+-----------+
|            1084 | DNS      |
|             794 | ICMP     |
|               1 | ICMPv6   |
|               1 | SSH      |
|              70 | SSHv2    |
|              50 | TCP      |
+-------------------+-----------+
6 rows in set (0.04 sec)
```

Find the average packet length per protocol. Also print the number of packets:

```
mysql> select COUNT(*), protocol, avg(length) from NetData
GROUP BY protocol;
+----------+------------+--------------+
| COUNT(*) | PROTOCOL | avg(length) |
+----------+------------+--------------+
|    1084 | DNS      |     95.8127 |
|     794 | ICMP     |    123.0743 |
|       1 | ICMPv6   |    118.0000 |
|       1 | SSH      |    194.0000 |
|      70 | SSHv2    |    356.3571 |
|      50 | TCP      |     66.6400 |
+----------+------------+--------------+
6 rows in set (0.04 sec)
```

Find the number of different Destination Hosts used in the destIP column:

```
mysql> select count(distinct(destIP)) from NetData;
+--------------------------+
|count(distinct(destIP))|
+--------------------------+
|                  279 |
+--------------------------+
1 row in set (0.01 sec)
```

Find the Top-10 Source IPs:

```
  mysql> select count(*) as TOTAL, SourceIP
  from NetData
  GROUP BY SourceIP
  ORDER BY TOTAL DESC
  LIMIT 10;
+--------+-----------------+
| TOTAL | SourceIP      |
+--------+-----------------+
|   841 | 109.74.193.253 |
|    76 | 2.86.13.236    |
|    38 | 204.74.106.104 |
|    25 | 175.41.186.83  |
|    24 | 89.149.6.76    |
|    22 | 90.155.53.34   |
|    20 | 218.248.241.3  |
|    20 | 114.134.15.205 |
|    19 | 200.29.243.21  |
|    17 | 14.139.5.22    |
+--------+-----------------+
10 rows in set (0.03 sec)
```

Here is the output of the Perl script for the network data I captured:

```
$ ./netData.pl login.tcpdump
Erroneous IP!!
   count(protocol)  protocol
   3756             DNS
   142              SSH
   100              TCP

   COUNT(*)         protocol       avg(length)
   3756             DNS            107.3387
   142              SSH            354.0704
   100              TCP            66.6400
   count(distinct(destIP))
   278

   TOTAL  SourceIP
   1682   109.74.193.253
   152    2.86.13.236
   76     204.74.106.104
   50     175.41.186.83
   48     89.149.6.76
   44     90.155.53.34
   40     218.248.241.3
   40     114.134.15.205
   38     200.29.243.21
   34     14.139.5.22
   Number of rows inserted: 999
```

If you connect to MySQL using the command line shell or a GUI application, you can execute whatever SQL query you wish. Only your imagination and SQL can limit the way you can utilize the data.

If you are using a NoSQL database, such as MongoDB, which is my favorite NoSQL database, you may need to learn how to use the MapReduce [6] technique to query the database. The results will be the same, but the implementation will be a little different. MongoDB is more focused on storing semi-structured data.

In conclusion, determining your own requirements and creating the queries that fit your needs is the first thing to do before inserting any data in a database. A statistical package such as R [7, 8] can also be used for visualizing and exploring your data. Download the netData.pl script from the USENIX site [10].

References

[1] MySQL site: http://www.mysql.com/.

[2] Kristina Chodorow, *MongoDB: The Definitive Guide, 2nd Edition* (O'Reilly Media, 2013).

[3] tcpdump: http://www.tcpdump.org.

[4] WireShark: http://www.wireshark.org/.

[5] tshark: http://www.wireshark.org/docs/man-pages/tshark.html.

[6] MapReduce: http://docs.mongodb.org/manual/reference/method/db.collection.mapReduce/.

[7] R Project: http://www.r-project.org.

[8] Mihalis Tsoukalos, "Using the R Advanced Statistical Package," *Linux Journal*, August 2013.

[9] Using WireShark Command Line Tools & Scripting: http://www.youtube.com/watch?v=CWOCqGmu1aI.

[10] netData.pl: https://www.usenix.org/publications/login/february-2014-volume-39-number-1

Practical Perl Tools

Redis Meet Perl

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 26+ 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 2005 conference and one of the LISA 2006 Invited Talks co-chairs. David is honored to be the recipient of the 2009 SAGE Outstanding Achievement award and to serve on the USENIX Board of Directors.
dnb@ccs.neu.edu

One tool that you may have heard all of the cool kids(tm) are using today is Redis (http://redis.io). If you haven't heard of Redis, this column may introduce you to a lovely tool you can add to your repertoire. First we'll talk a little bit about what Redis is and why people like it. Once we do that, we can get into the Perl side of things. Just a quick warning for those of you who have seen this happen before in this column: the Perl stuff we're going to look at is a pretty straightforward layer on top of the basic Redis functionality. I'll consider an "Oh, is that all there is to it" reaction to be a good sign. But I just wanted to warn you lest you were hoping for gnarly Perl to impress your friends with at parties.

What Be Redis?

Redis is one of those packages that gets lumped into the NoSQL gang. These are software packages designed to help with certain scaling problems because they provide really simple but really fast storage and retrieval of data often with a little bit of "make it easy to distribute the data over several servers" thrown in. What they trade off in complexity around the storage and retrieval of data (ACID compliance, full query languages) is sheer performance and ease of use by other applications.

In many cases these software packages act like a "key-value" storage mechanism (i.e., like a Perl hash, you can store a value under an associated key for retrieval by that key later). An example of another well-known key-value store is the memcached package, something we may visit in a future column. Redis is a bit spiffier than a number of these packages because it actually understands more data types natively than most of the other players in the space.

In addition to this functionality, I believe Redis is well-liked by the people at those parties I mentioned before because it is fast, performant, stable, well-engineered, and quite easy to get started with right out of the box, even if you've never touched the thing. This may sound like what you hope every tool would be, but in my experience finding one isn't as easy as one would hope.

Redis Basics

Okay, so let's get into the fundamentals of using Redis. Redis gets packaged pretty easily so that you can usually find a way to install it quickly on the operating system of your choice. A second ago I grabbed the latest version to my laptop with "brew redis," but your copy could be almost an apt-get, yum, or wget/curl away (the source looks pretty easy to compile, although I have not done it). Given all of this, watch me hand wave about the installation (waves hands) so we can move right on to using the software. Redis comes with a massively commented sample config file (redis.conf), so feel free to bend it to your will. For this column, we're just going to assume you brought a Redis server up in its stock configuration (i.e., redis-server /path/to/redis.conf).

Redis comes with a command-line client (redis-cli), so we are going to play with it a bit before we show the Perl equivalent. This is similar to the approach we took when talking about RRDtool many moons ago. Let's use that client to do what every other key-value store can do first:

```
127.0.0.1:6379> SET usenix:address "2560 Ninth Street, Suite 215"
OK
127.0.0.1:6379> SET usenix:state "CA"
OK
127.0.0.1:6379> SET usenix:zip "94710"
OK
127.0.0.1:6379> GET usenix:address
"2560 Ninth Street, Suite 215"
```

Nothing exciting, right? If I tossed a ton of clients or millions of records and it did that, that would be cool but probably not all that exciting. A half a notch more exciting would be something like this:

```
127.0.0.1:6379> SET usenix:members 10
OK
127.0.0.1:6379> INCR usenix:members
(integer) 11
```

So why is that more exciting? Surely I could just do a GET and then a SET of the number of members + 1 (not the real number of members, by the way) instead of using a special increment operator. I could do that, but what if my typing or my script is kinda slow and some other person attempts to do the same operation? It is conceivable there will be a race condition in which I'll wind up incrementing a number that isn't the current one. INCR performs the operation in an "atomic" fashion, which means that you can have many separate clients incrementing the value and you don't have to worry about them stepping all over each other. There are a number of other fun things we can do to simple strings (append to them, get substrings, treat them like bit vectors, etc.). But let's get beyond strings...

More Redis Data Types

Although constructing all sorts of data structures in your application with just plain strings is possible, Redis makes it even easier for the programmer by internally supporting some of the more popular ones. For example, Redis handles lists for you trivially:

```
127.0.0.1:6379> LPUSH usenix:conferences LISA
(integer) 1
127.0.0.1:6379> LPUSH usenix:conferences OSDI
(integer) 2

127.0.0.1:6379> LRANGE usenix:conferences 0 -1
1) "OSDI"
2) "LISA"

127.0.0.1:6379> RPUSH usenix:conferences Security
(integer) 3
127.0.0.1:6379> RPUSH usenix:conferences FAST
(integer) 4

127.0.0.1:6379> LRANGE usenix:conferences 0 -1
1) "OSDI"
2) "LISA"
3) "Security"
4) "FAST"
```

The LPUSH command adds items to the left of the list (the front); RPUSH adds them to the right (the end). The LRANGE operator can be used to return parts of the list (using 0 and -1 means start at the first element and go to the end). There is a whole host of other list-related commands, including:

```
127.0.0.1:6379> RPOP usenix:conferences
"FAST"

127.0.0.1:6379> LRANGE usenix:conferences 0 -1
1) "OSDI"
2) "LISA"
3) "Security"
```

Here we've treated the list like a stack and popped the last element off the end using RPOP.

Let's Take a Perl Break

There are more data structures we should look at, but let's take a break to look at some Perl code. Here's a translation of the initial redis-cli example to Perl. In this column, we're going to be using the most popular Redis module (called Redis), although there are number of other choices available on CPAN:

```
use Redis;

my $redis = Redis->new;

# using usenix-from-perl as part of the key name just so it
# is clear on the server that the data is coming from this
# script and not the redis-cli command lines.
$redis->set( 'usenix-from-perl:address' =>
            '2560 Ninth Street, Suite 215' );
$redis->set( 'usenix-from-perl:state'   => 'CA' );
$redis->set( 'usenix-from-perl:zip'     => '94710' );

print $redis->get('usenix-from-perl:address'), "\n";
```

This prints out '2560 Ninth Street, Suite 215' as you'd expect. Perl code that uses Redis is an easy leap from the command-line example, no? Let's go back to the Redis data structures because we're going to run into a few of the even cooler Redis features shortly.

Two More Data Structures

I would be remiss if I didn't mention the two remaining supported data structures. The first, very familiar to Perl folks and mentioned early in this column, is the hash. It isn't immediately apparent what a hash type might mean when it comes to talking about a key-value store (which sounds like a hash already). Redis hashes are probably most equivalent to the Perl hash-of-hashes data structure. Each key in Redis is connected to a value that has associated fields (each with its own values). For example, let's make a hash that lists the cities where the LISA conference will be held in upcoming years:

```
127.0.0.1:6379> HSET usenix:lisa-conference 2014 "Seattle"
(integer) 1
127.0.0.1:6379> HSET usenix:lisa-conference 2015 "D.C."
(integer) 1
127.0.0.1:6379> HMSET usenix:lisa-conference 2016 "Boston" 2017
"San Francisco"
OK
```

Here we've used two different commands to populate the usenix:lisa-conference hash (which contains the years as fields). The first, HSET, sets a single field at a time. The second,

HMSET, lets us set multiple fields at a time. Retrieving the info in each field can be done with (I bet you are seeing the pattern in names) HGET:

```
127.0.0.1:6379> HGET usenix:lisa-conference 2017
"San Francisco"
```

If we want to retrieve multiple fields, we can use HMGET with a list of the fields we want to retrieve:

```
127.0.0.1:6379> HMGET usenix:lisa-conference 2014 1016
1) "Seattle"
2) (nil)

127.0.0.1:6379> HMGET usenix:lisa-conference 2014 2016
1) "Seattle"
2) "Boston"
```

In the example above, I left in my typo so you can see what gets returned if you ask for a field that hasn't been set previously. (I don't actually know where LISA was in the year 1016; I only started attending in the 1900s.)

You can probably guess this, but just to make it explicit, the Perl equivalents of the commands above are direct translations:

```
$redis->hset( 'usenix-from-perl:lisa-conference',
'2014' => 'Seattle' );
$redis->hmset( 'usenix-from-perl:lisa-conference',
'2016' => 'Boston', '2017' => 'San Francisco' );
my $location = $redis->hget( 'usenix-from-perl:lisa-conference',
'2017' );
```

One last data structure type and then I want to show you two more magical things Redis can do with its data storage. The last data type is one that doesn't really have a direct analog to Perl's built-in data types: sets. Redis implements sets in two flavors: standard/unordered and sorted. The standard set is basically an unordered collection of elements that can be added to, subtracted from, tested for membership, and so on. And just like your junior high school days, you can perform operations between sets, such as finding their union or intersection. Let's use a set to keep track of the current USENIX board members. First we'll add them to the set:

```
127.0.0.1:6379> SADD usenix:board margo
(integer) 1
127.0.0.1:6379> SADD usenix:board john
(integer) 1
127.0.0.1:6379> SADD usenix:board carolyn
(integer) 1
127.0.0.1:6379> SADD usenix:board brian
(integer) 1
127.0.0.1:6379> SADD usenix:board david
(integer) 1
127.0.0.1:6379> SADD usenix:board niels
(integer) 1
127.0.0.1:6379> SADD usenix:board sasha
(integer) 1
127.0.0.1:6379> SADD usenix:board dan
(integer) 1
```

Now, if we show the members of the set, you'll see that they come back in a different order than they were added to the set (in this way, the lack of preserved order does resemble keys in a Perl hash):

```
127.0.0.1:6379> SMEMBERS usenix:board
1) "dan"
2) "john"
3) "carolyn"
4) "david"
5) "margo"
6) "niels"
7) "brian"
8) "sasha"
```

Once we have constructed our set, we can query to see whether an element is in it:

```
127.0.0.1:6379> SISMEMBER usenix:board niels
(integer) 1
127.0.0.1:6379> SISMEMBER usenix:board santa
(integer) 0
```

This test is fast, even with large sets (O(1) for you CS geeks). If we had defined multiple sets in this example, we could have determined how they differ, their intersections, and other fun things with a single Redis command.

Redis has a variation on sets called sorted sets. Sorted sets are like standard sets, except each member of the set has an associated "score." The score should be, according to the doc, "the string representation of a numeric value, and accepts double precision floating point numbers." The members of the set are kept in a sorted order based on this score. So, for example, if you wanted to model a leaderboard, you could create a sorted set using each member's score. As you rewrite each person in the set back to Redis with a new score, the members of the set rearrange themselves to stay sorted. This type of functionality can make parts of your application trivial to write. Let's do a toy example. If we had a bunch of different registration systems reporting back to a Redis instance the number of signups for each conference like so (yes, very fake numbers), we might have each system submit one of these commands:

```
127.0.0.1:6379> ZADD usenix:attendees 100 LISA
(integer) 1
127.0.0.1:6379> ZADD usenix:attendees 50 OSDI
(integer) 1
127.0.0.1:6379> ZADD usenix:attendees 30 FAST
(integer) 1
127.0.0.1:6379> ZADD usenix:attendees 75 Security
(integer) 1
127.0.0.1:6379> ZADD usenix:attendees 60 NSDI
(integer) 1
```

Now we can see the top three conferences listed in order of their attendance:

```
127.0.0.1:6379> ZRANGE usenix:attendees 0 2
1) "FAST"
2) "OSDI"
3) "NSDI"
```

Oh, wait, that's not right. That is indeed the first three conferences in the list in ascending order. We actually wanted to see the list sorted in descending order:

```
127.0.0.1:6379> ZREVRANGE usenix:attendees 0 2
1) "LISA"
2) "Security"
3) "NSDI"
```

Ah, much better. Sorted sets also make it super easy and super fast to find out where a particular member lives in the set:

```
127.0.0.1:6379> ZRANK usenix:attendees NSDI
(integer) 3
```

This says that NSDI can be found in the third place in the ranked order. Like the other data types, there are a whole slew of sorted set commands available. Instead of dwelling on them, let's finish the column by looking at two kinds of behind-the-scenes magic we can invoke.

Caching and Sub-ing

The first functionality I want to mention shows up in other key-value stores, but I still think it is kind of magic. Redis lets you set a time-to-live on any key. You can either set that value as the number of seconds a key should stick around via EXPIRE (which you can refresh using another EXPIRE), or provide a specific time for the expiration using EXPIREAT. Congratulations, you have self-maintaining cache.

The second thing Redis does that might make you squeal in delight is provide a special pub-sub mode. Pub-sub (i.e., publish-subscribe) is described in Wikipedia as

> "a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers. Instead, published messages are characterized into classes, without knowledge of what, if any, subscribers there may be. Similarly, subscribers express interest in one or more classes, and only receive messages that are of interest, without knowledge of what, if any, publishers there are."

In Redis (and in other contexts) the classes are called channels. A client will connect to the server and SUBSCRIBE to a set of channels either directly by name (e.g., SUBSCRIBE usenix) or via a globbing pattern like PSUBSCRIBE *usenix*' (for all channels with usenix in their name). If other clients use the PUBLISH usenix 'some message' command, the subscribed clients to that channel will get this message.

Because we're getting close to the end of the column, a Perl column, let's see a demonstration of this mode via a Perl sample. The one thing that makes this sample a little more complex than the previous translations is that pub-sub is coded up using callbacks. Callbacks are little snippets of code that are called when a message is received (vs. having some function you call that returns a value). As the Perl Redis module documentation states:

> "All Pub/Sub commands receive a callback as the last parameter. This callback receives three arguments:

- The published message.
- The topic over which the message was sent.
- The subscribed topic that matched the topic for the message. With 'subscribe' these last two are the same, always. But with 'psubscribe', this parameter tells you the pattern that matched."

The first thing we might write is a script that subscribes to some channels. It is as simple as this:

```
use Redis;
my $redis = Redis->new;

# subscribe to the LISA and Security channels
$redis->subscribe( 'LISA', 'Security',
    sub my ( $message, $channel, $subscribedchannel ) = @;
      print STDERR "Someone said something interesting:
           $message\n"; }, );
# loop, waiting for callbacks, or for 60 seconds to pass
$redis->wait_for_messages(60) while 1;
```

The code above connects to the server and subscribes to the LISA and Security channels. When we subscribe, we specify a subroutine that runs when a message comes in (it receives the arguments we just quoted from the doc) and prints out what it receives. Because this is a callback, if we just did a SUBSCRIBE, the script would subscribe to the channel and exit; it never would have the pleasure of smelling a freshly received message. We must tell it to wait around for messages and process callbacks, hence the last line that loops forever waiting for messages.

The script that actually publishes a message is trivial:

```
use Redis;

my $redis = Redis->new;
$redis->publish('LISA', 'Perl is great!');
```

If we launch the client and then run this script, it predictably prints out:

```
Someone said something interesting: Perl is great!
```

So with that example, I want to wrap up the column. We've only looked at a small subset of the commands Redis offers (for example, there are indeed commands for deleting info), and we haven't even mentioned some of the support for high-performance use, such as its pipelining feature. I'm hoping this brief look, along with a sense of how easy it is work with Redis from Perl once you know the commands, inspires you to go digging for more.

Take care and I'll see you next time.

The Wheels Keep on Spinning

DAVID BEAZLEY

David Beazley is an open source developer and author of the *Python Essential Reference* (4th Edition, Addison-Wesley, 2009) and *Python Cookbook* (3rd Edition, O'Reilly Media, 2013). He is also known as the creator of Swig (http://www.swig.org) and Python Lex-Yacc (http://www.dabeaz.com/ply.html). Beazley is based in Chicago, where he also teaches a variety of Python courses. http://www.dabeaz.com, dave@dabeaz.com

If you've ever had the pleasure of installing third-party Python packages, you already know that it can be a bit of a mess. There are a variety of different tools, file formats, and other packaging complications. Frankly, it's enough to make your head spin.

Over the past year, a new Python packaging format has emerged in the form of a "wheel" file—so named because a wheel is a common packaging form factor for cheese, as in the big wheel of cheese that you might find at a cheese shop. Naturally, this is a reference to a certain cheese shop in an obscure Monty Python sketch, but that should have been obvious. I digress.

When the new wheel format emerged, I'll admit that I mostly ignored it. Python packaging is not my favorite topic, and the thought of having to think about yet another file format was relatively low on my list of day-to-day priorities; however, in recent months there has been a concerted effort to have package maintainers support the new wheel format. For example, the Web site http://pythonwheels.com/ currently shows the wheel status for the most popular Python extensions. As the author of one such extension, I was starting to get questions about wheels and was ashamed to admit my ignorance.

So, what in the heck is a wheel, you ask? In this installment, we'll take a look at wheels, Python packaging, and related topics. As we'll see, there are some rather interesting aspects to wheels—especially for anyone who needs to maintain, test, or deploy complex Python applications.

A Quick Review of Python Packaging

Before jumping into the subject of wheels, a quick refresher on Python packaging is probably in order. First, the Python Package Index (PyPI, at http://pypi.python.org) is the definitive site for locating and downloading third-party packages. If you go here, you'll find virtually all available packages listed, along with links to downloads, documentation, and more.

If you want to, you can download a package directly from PyPI and install it manually on your machine. Typically you would download the source and look for an enclosed setup.py file. You would then run `python setup.py install` on that file to perform an installation. For example, if you wanted to download the pytz extension for handling time zones, here are the steps that you might perform. In this example, replace the `curl` command with anything that simply downloads the source from PyPI. Also, depending on how Python has been installed, the final step might need to be performed as root using sudo.

```
bash % curl -O https://pypi.python.org/packages/source/p/pytz/pytz-2013.8.tar.gz
bash % tar -xf pytz-2013.8.tar.gz
bash % cd pytz-2013.8
bash % python setup.py install
```

Instead of manually installing a package in this manner, an alternative approach is to use the optional setuptools package. setuptools gives you the `easy_install` command, which automatically contacts PyPI, downloads the most recent version, and installs it for you. For example, instead of typing the above commands, you could simply type the following statement (again, you may need to use sudo depending on your Python installation):

```
bash % easy_install pytz
```

setuptools saves you the trouble of downloading, unpacking, and running the setup.py file yourself; however, it does quite a bit more than that because it will also download and install any dependencies. This can be useful if you're installing something much more complicated. For example, if you wanted to install the Python data analysis library pandas (http://pandas.pydata.org/), typing `easy_install pandas` will not only install pandas but also all of its dependencies, including numpy, python-dateutil, six, and pytz.

Although easy_install is commonly described in tutorials and documentation, the pip command (http://www.pip-installer.org) is a bit more modern, performs a similar function, and seems to be coming the preferred way to install packages. pip operates in a manner similar to easy_install. For example, to install a package, type a command like this:

```
bash % pip install pandas
```

Under the covers, pip actually requires the use of setuptools, so if you're using it, you'll actually have both easy_install and pip installed. This obviously begs the question: What is the major difference between the two? That's a big question, but pip makes a number of subtle changes to the installation process. For example, pip downloads all of the dependencies and builds them completely before attempting any kind of install. As a result, the install will either succeed in its entirety or not at all. On the other hand, easy_install might end up performing a partial install if some part of the installation process fails midway through. pip also provides some additional commands, such as the ability to uninstall a package.

Perhaps the most notable feature of pip is its ability to "freeze" and recreate your exact installation configuration. For example, suppose you had spent a lot of time making a custom Python setup. You can type the following command to freeze it into a requirements file:

```
bash % pip freeze >requirements.txt
```

This creates a file requirements.txt that looks like this:

```
Django==1.6
SQLAlchemy==0.8.3
numpy==1.8.0
pandas==0.12.0
ply==3.4
python-dateutil==2.2
pytz==2013.8
requests==2.0.1
six==1.4.1
virtualenv==1.10.1
wsgiref==0.1.2
```

Now, suppose you were setting up a new Python installation or performing a deployment to a new machine. If you wanted to recreate your environment, you could simply type the following:

```
bash % pip install -r requirements.txt
```

This will download, build, and install everything in requirements.txt for you—very nice.

Virtual Environments and Deployments

Once you've mastered the basics of installing packages, you might think that it's the end of the story. After all, how many times are you actually going to sit around installing packages? As it turns out, it might be a lot more often than you think.

One of the more popular extensions to Python is the virtualenv tool (https://pypi.python.org/pypi/virtualenv). virtualenv allows you to make entirely new Python environments for working on new versions of code, experimentation, and testing things out. To make a new virtual environment, simply type the following:

```
bash % virtualenv spam
```

This creates a new directory spam/ in which you will find a new Python installation. This installation is actually a "blank slate" of sorts. The directory spam/bin includes Python as well as easy_install, pip, and virtualenv. No other third-party extensions are included. But that's the whole idea—with a virtual environment you get to start over.

Not to worry! Remember the requirements.py file you just created with pip? Let's recreate our setup in the new virtual environment with a single command:

```
bash % spam/bin/pip install -r requirements.txt
```

This will churn away for a while, but when it's done, you'll have a brand new Python environment with everything installed. Because it's an isolated environment, you can continue to experiment with the setup without breaking the default Python installation on your machine.

Naturally, a similar process can be used if you're deploying a Python application to new machines or to workers out on the cloud. Simply make sure you distribute the requirements.txt file and use it to recreate the environment when you need it.

A Performance Headache

If you've made it this far, you will have made recreating your Python environment easy; it all works fine except for one huge headache, which is the performance of it all. When you type the command `pip install -r requirements.txt`, all of the required packages will be downloaded, compiled, and installed from source. Although there are ways to cache the source locally and avoid the download step, the compilation and installation process can take a substantial amount of time. For example, installing a new environment from the requirements.txt file shown here takes a little more than nine minutes on my machine. Much of that time is spent running the C compiler for the numpy and pandas extensions.

Although that might not seem like much time, it can add up quickly if you find yourself making many virtual environments or recreating the Python environment as part of a deployment script. Surely there should be some way to perform a binary installation from pre-built packages instead. Wouldn't that be much faster? Yes, it would.

Enter Wheels

The newly introduced "wheel" standard is an effort to solve this problem. In a nutshell, a wheel is simply a pre-built Python package. Because it's pre-built, none of the usual source compilation steps are necessary. Instead, all of its contents can simply be copied into place.

To see how wheel works, you first must install the separate wheel package. Just use pip:

```
bash % pip install wheel
```

Next, let's make a special directory for our wheels:

```
bash % mkdir /tmp/wheels
```

Once you're done with that, type the following command using the requirements.txt file from earlier:

```
bash % pip wheel --wheel-dir=/tmp/wheels -r requirements.txt
```

This command will churn away for a while, but when it's done, the /tmp/wheels directory will contain a collection of .whl files like this:

```
bash % ls /tmp/wheels
Django-1.6-py2.py3-none-any.whl
SQLAlchemy-0.8.3-cp27-none-macosx_10_4_x86_64.whl
numpy-1.8.0-cp27-none-macosx_10_4_x86_64.whl
pandas-0.12.0-cp27-none-macosx_10_4_x86_64.whl
ply-3.4-py27-none-any.whl
python_dateutil-2.2-py27-none-any.whl
pytz-2013.8-py27-none-any.whl
requests-2.0.1-py27-none-any.whl
six-1.4.1-py27-none-any.whl
virtualenv-1.10.1-py27-none-any.whl
wsgiref-0.1.2-py27-none-any.whl
```

Okay, that's interesting, but what's the point, you ask? The real benefit comes when you later want to recreate your Python environment. Using the wheels directory as a kind of cache, type the following commands to make a new virtual environment:

```
bash % virtualenv spam2
bash % spam2/bin/pip install --use-wheel --no-index --find-links=/tmp/wheels -r requirements.txt
```

This will completely recreate your Python environment exactly as before; however, instead of taking nine more minutes, it now only takes four seconds. That's a speedup of about 13,500%, in case you were keeping track. Needless to say, this ability to almost instantly recreate your Python environment on a moment's notice is interesting.

Under the Covers

The gory details of what's going on inside a wheel file can be found in PEP 427 (http://www.python.org/dev/peps/pep-0427). To be honest, this document mostly made my head hurt, and it did not fully illuminate the big idea at work. Thus, here are the big picture details that you might care to know. A .whl file is simply a .zip file containing the compiled/built package. Everything needed to make the package work is contained inside. For example:

```
bash % unzip -l ply-3.4-py27-none-any.whl
Archive: ply-3.4-py27-none-any.whl
  Length     Date    Time    Name
 ---------  -----   -----   -----
       82  11-25-13  12:02   ply/__init__.py
    33040  11-25-13  12:02   ply/cpp.py
     3170  11-25-13  12:02   ply/ctokens.py
    40739  11-25-13  12:02   ply/lex.py
   128492  11-25-13  12:02   ply/yacc.py
      518  11-25-13  12:08   ply-3.4.dist-info/DESCRIPTION.rst
      439  11-25-13  12:08   ply-3.4.dist-info/pydist.json
        4  11-25-13  12:08   ply-3.4.dist-info/top_level.txt
       93  11-25-13  12:08   ply-3.4.dist-info/WHEEL
      808  11-25-13  12:08   ply-3.4.dist-info/METADATA
      804  11-25-13  12:08   ply-3.4.dist-info/RECORD
 ---------                   -------
   208189                   11 files
```

When a wheel is installed, the files are moved into place in the normal site-packages directory. None of the usual "build" or "compile" steps take place. Needless to say, this is the reason why installing a wheel is super fast.

The other important detail is that the file name for wheels encodes platform-specific details as appropriate. For example, many packages include some component of C code. For these packages, the name will encode information about the underlying architecture. For example, the wheel created for pandas has the following name on my system:

```
pandas-0.12.0-cp27-none-macosx_10_4_x86_64.whl
```

This becomes useful if you're supporting different kinds of machines (Linux and Mac OS X) or different architectures (32-bit vs. 64-bit). Essentially, you can build a wheel cache with the different versions, and the appropriate version will be used during installation.

Big Picture: Using Wheels in Practice

Currently there is an effort to have the authors of commonly used Python packages upload their code to the Python Package Index as wheels in addition to their normal source distributions. As more users start relying on the wheel format, this will make installation faster; however, I also think that this is the least interesting aspect of wheels. This is because you can still reap their benefits regardless of whether or not a wheel is actually uploaded by a package author.

Imagine that you are the maintainer of Python at your company. Internally, you might have an officially approved version of the interpreter as well as a standard set of third-party libraries that you use in all of your applications. As the Python maintainer, you've probably already written some scripts or instructions for setting up the officially approved Python environment on a new machine. All of this works, but it's also a bit slow.

With wheels, you basically can create your own custom package repository of pre-built extensions. If you point users and installation scripts at that repository, you can turn that laborious installation process into an operation that only takes a few seconds, which is pretty neat. Also there are interesting implications for scripts that push code out to clusters and other large system installations.

Alternatively, imagine that you're the maintainer of an application in your organization and you need to give it out to users. These users may have Python on their machines, but perhaps not all of the compilation tools needed to build complex extensions such as those involve C code. Not to worry—you could create a little install script that points them at your custom wheel repository. When they install your application, they'll get all of the pre-built bits that you've made for them. Again, that can solve a lot of problems.

More Information

Official information about wheels can be found in PEP 427 (http://www.python.org/dev/peps/pep-0427). Tutorials and documentation can be found at http://wheel.readthedocs.org/en/latest/. There, you will find even more advanced material, such as how to attach digital signatures to wheels. Although wheels are new, they have been blessed officially as the Python standard. You're sure to be seeing more of them in the future.

iVoyeur
Counters

DAVE JOSEPHSEN

Dave Josephsen is the sometime book-authoring developer evangelist at Librato.com. His continuing mission: to help engineers worldwide close the feedback loop. dave-usenix@skeptech.org

My friend Chris counts people. Well, sometimes he counts people, and although I can tell you where, I couldn't tell you why. I only found out by accident, and the act of discovery felt enough like an unwelcome intrusion that asking *why* seemed out of the question.

It happened when a group of us went over to his apartment at lunch. He lived across the street from the building in which we all worked at the time, and it wasn't uncommon for us to grab a sandwich or whatever, and head over to his apartment to play some "DOA3" or "God Hand." On this particular day, Chris had left on his kitchen table a small metallic four-digit "hand tally," like the ones you see in the hands of parking attendants and people who take your ticket as you come through the door.

My friend Kelly, who, like myself, has a penchant for sometimes accidentally asking embarrassing questions, picked it up and, glancing at the number (37 IIRC), asked what it was. Chris replied, in so many words, that the number represented people that he talked to at a party the night before, and didn't appear eager to discuss it much further.

We said nothing more about it. But I often wish that Kelly had had the lack of decorum to pursue it, or that I had, because I don't know about you, but thousands of questions occur to me. Important, nagging questions that, having gone unanswered, keep this story in my memory year after year, even though nearly a decade has passed. When I die, I will die with these questions somewhere in what remains of my dementia-riddled mind. Was this a recurring polling interval, or was it a non-periodic metric? Was he storing it as an incrementing counter that rolled over at 9999+1, or was he manually rolling it over daily and storing it as a gauge? Had he plotted the distribution? Was it uniform? What was the mean and sigma? Had he done any predictive analysis?

I was tempted to begin this article with something like, "Counters are a foundational concept in systems monitoring," although it would have been beyond condescending of me. You don't need me to tell you that counting things is a foundational concept. Counters are about as simple a concept as exists. Yet, in the context of this story about Chris, it strikes me that counter metrics are also sort of subtle and sometimes misunderstood.

Indeed the questions I have about Chris's quantification fetish that I cannot answer have mostly to do with the implementation details of the counter data. I can, for example, imagine myriad reasons why he might want to count the people he talks to—that he has a personal goal to talk to 10k unique people, or is trying to improve his interpersonal skills by repetition. I've often fallen down this or that "life hacking" rabbit hole and can easily relate. But the details surrounding how his counter metric is stored imply real and interesting systems constraints.

For example, Chris, like me, cut his teeth on MRTG and RRDtool, so for him to store his metric as type "counter" in a round-robin database would be natural. RRDtool's concept of a counter grew out of SNMP counters. Network devices, such as switches and routers, store metrics, such as byte-counters, on each interface using a 32-bit number. RRDtool's counter type is intended to store the current value polled from a router byte-counter directly and automatically compute and display the derivative, or rate of change, of that value. When you say "store it as a counter" to Ops people, this is the notion you have conjured in their minds—that of an ever-increasing 32- or 64-bit value, which, if plotted directly, would always look like a diagonal line increasing from left to right.

In the mind of a programmer, a counter is also usually a 32-bit value, although a good programmer would probably disown that, muttering something about it being architecture dependent, and something else about uint32. The programmer's counter usually has a short name (like "c") and the programmer usually has handy incrementors and decrementors (like "++" and "--") built into his or her favored language that enable simple and terse manipulation of the counter value.

If my friend Chris tired of having a manual clicky thing in his pocket that he had to interact with, and then later, manually copy the value from, we can imagine that he might replace his hand-tally with something like an Arduino board that could uniquely identify the voices of the people he talked to. If he did this, internally, he would probably use a 32-bit integer with a small name (like "p") in his code to track his "people" metric. This would be good, but he would still want to store the metric externally, so he could see graphs without manually copying values from the device. The Arduino could use WiFi (or whatever) to broadcast his metric every five or ten minutes, which would enable him to see not only how many people he spoke to but also when he did a lot of talking over the course of the evening.

He could store the value externally in Graphite [1], a metrics collection system with a super-simple API. Some monitoring systems, such as Graphite, only store data as a value that may increase or decrease (a "gauge," in RRDtool parlance). If you want a useful graph of a counter stored as a gauge in Graphite, you wrap your counter in the "derive()" function when you graph it. So if Chris wants to store his metric in Graphite, he'll need to write his code such that the total value of p is sent to Graphite every so often, like RRDtool with the SNMP byte-counters.

Incrementor operators such as "++" are ubiquitous, easy to understand, and make a lot of sense in contexts like counting people. If, in Chris's code, instead of sending the value of p every time, it would be nice if he could create some sort of object, or type, which he could simply increment each time he had a new conversation with a unique person. Graphite's socket API is nice, but because it doesn't have a counter type at all, it doesn't provide a simple means for Chris to, in his code, say "p++". Chris would either have to track the current total value himself and then write a wrapper for himself to increment and push it, or he'd have to query Graphite for the current total value, increment it, and then push it back.

Either way, Chris has scalability details to consider. If he tried to track the total value himself, he would limit concurrency. If, for example, he eventually attempted to replace his Arduino device with a series of room-based devices, each of which reported more generic stats on all conversations in their respective room, each device would have a different total value of p for Chris, and would therefore step on each other when they tried to report up to Graphite. If Chris tried to query the current value for p for every measurement, he'd have a race condition, limit the polling interval, and introduce a dependency between metrics collection and storage, which is an unwise design choice.

Instead of communicating values, if Chris's code could send an Incrementor signal to the external storage system would be ideal. That way, he wouldn't need to store or acquire the current value of p to increment it, and his code could be distributed and concurrent. Scalability concerns would be pushed up from the metrics collection tier to the metrics storage tier, and the number of bits on the wire would be smaller and more predictable. This, in a nutshell, is why Statsd [2] was created. Statsd was designed to be a middle layer between Graphite and your metrics sources. It can listen for Graphite protocol values as well as special purpose instructions, such as incrementor signals. Statsd can also "roll up" multiple broadcasts from metrics collectors into single updates, taking some of the strain off Graphite itself.

In case I haven't made the subject of counters convoluted enough, to run Statsd on every server, or centrally, or both is common practice. We can imagine, for example, Chris's multi-room conversation-counter system, tracking per-room metrics, in which case each room would have its own value of p that would be tracked as a separate metric. If we wanted a total value, we could sum() them at visualization time in Graphite. Alternatively, we could run Statsd at a central location, and point every room at it. We could still track per-room metrics this way, as well as allowing every server to increment a global total value counter. Finally, we could run Statsd on every server, which could give us a high localized polling interval, and then point each server Statsd instance at a centralized Statsd, to roll everything up and maintain a global counter.

In my opinion, the programmatic notion of a counter as "a value we use to count things," is more intuitive than RRDtool's "ever-increasing, sometimes rolling-over, value we assume you're going to want to plot the derivative of at some point." More importantly, the two definitions are not compatible with one

another in a data-storage sense. We cannot store the former in a box designed for the latter.

For example, Chris might eventually run into some situations in which he wants to decrement p, especially if every conversation is automatically being counted for him. "Conversations" beginning "Where is the bathroom?" and "Do you have change for a $20?" might not seem to him to be valid data, and although there's no reason he couldn't decrement p using Graphite and Statsd, he'd be stuck kludging around with RRDtool. Counters like those in Coda Hale's Metrics [3] library are often used to track things such as cache entities and live threads, and for these purposes decrementors are an absolute necessity. In a more general sense, this implies that when we undertake to decide on a metrics storage system, we can no longer assume we already fully understand the primitives based on their names. We need to be sure we know what is meant when someone says "counter."

In the past few years, the field has grown laden with metrics collection systems that each have a thing called a "counter," the meaning of which is often expected to be intuitive (because, duh, what could be more obvious than a counter). But heads up. As I hope Chris and I have illustrated, counters are not simple, and they're only sometimes used for counting things.

Take it easy.

References

[1] Graphite: http://graphite.wikidot.com/.

[2] Statsd: https://github.com/etsy/statsd/.

[3] Coda Hale's Metrics: http://metrics.codahale.com/.

Margin of Safety or Speculation?

Measuring Security Book Value

DAN GEER AND GUNNAR PETERSON

Dan Geer is the CISO for In-Q-Tel and a security researcher with a quantitative bent. He has a long history with the USENIX Association, including officer positions, program committees, etc.
dan@geer.org

Gunnar Peterson is a Managing Principal, Arctec Group, which provides security and identity architecture consulting, development, and training across the globe. Gunnar consults and trains developers, architects, and security pros in secure coding and architecture. He is a contributing analyst at Securosis, an IANS faculty member, and blogs at http://1raindrop.typepad.com.
gunnar@arctecgroup.net

> The measure of success is not whether you have a tough problem to deal with, but whether it is the same problem you had last year.
>
> —John Foster Dulles

In the stock market, financial institutions that are considered to be well run sell at a premium: their stock price is greater than their tangible book value, the price/book ratio. What is that book value? A simple number that is easy to acquire and understand, book value is the asset's dollar value carried on your balance sheet. Applying book value to IT, what cost did you incur to develop, deploy, and operate your system? That's its book value.

Why would anyone pay more than book value for a bank's assets? Because some banks make higher quality loans and take less risk. Investors deem Wells Fargo and US Bank to be well run: Wells Fargo trades at 1.5x book value and US Bank trades at 2.0x book value. Conversely, banks that are thought to be less well run sell below book value: Citigroup has traded at near half its book value since 2008.

The stock market's premium for Wells Fargo and US Bank and its discount for Citigroup may or may not prove to be well founded, but what those price ratios tell you is the value that investors place on the quality of the assets and the risk management of those companies.

In this spirit, we propose using a Margin of Safety calculation to compare the book value of a company's IT assets (software, servers, development, and so on) to book value of the security controls and services used to defend those assets. We suggest that the difference between these two numbers assesses the level of safety for assets in your enterprise. If the assets' book value is well covered by the book value of the security controls, then you are making minimal assumptions as to the efficacy of your security systems. If the gap is wider, you may be asking for heroic efforts—too much—from your security services and team.

In investing, paying less than $1 for $1 of assets is an example of a Margin of Safety. What we seek to show here is where the line between safety and speculation occurs in information security systems.

A disclaimer: we make no attempt here to address a number of important concepts. We consider the basic book value to be a number, a number that has the personality of a brick; it does not change much, it is not subject to interpretation. To us, that's beautiful. It's also limited. In using the book value metric, we do not address the earnings power of the assets, nor do we attempt to measure the efficacy of the security controls.

The earnings power of the assets is arguably the single most important business metric; the efficacy of the security control is arguably the most crucial security metric. Why, then, do we propose to not address either? Simple—both earnings power and efficacy are highly subjective, path dependent, fraught with errors, time varying, and prone to willful misinterpretation. You throw out a subjective number in a meeting and then defend that subjectivity to anyone who disagrees. This is metrics at their worst, people marinating in their own biases rather than letting the numbers do the talking.

Book value, on the other hand, is appealing for all its brick-like qualities. For one thing, it is hard to argue with. You either paid $10M for SAP or you didn't. It's there in black and white, and, better yet, the accounting department will back you up on it.

Okay, so it's a good number, but what can you use it for? To quantify assumptions, illuminate priorities, and identify opportunities for improvement.

To illustrate how book value quantifies assumptions, let's consider a company that runs a customer Web site and a customer mobile app.

The customer Web site cost $5M to develop and deploy. The company spent $250k on security software and services for that Web site.

The mobile app cost $1M to develop and deploy. The company spent $25k on security software and services for that app.

Now we use the Asset/Security ratio to compare the cost of the project versus the cost of the security services in Table 1.

The Margin of Safety shows that the mobile project has, on a relative basis, invested half as much in security as the Web app. Margin of Safety is a coverage metric. Coverage metrics are useful precisely because of the assumptions they do not make. Applications and systems are built to do something functional. Functionality can be measured in the present. Risk lies in the future. The Margin of Safety cover shows what's invested to absorb unfavorable future events. The Web app team invested 5% of its budget in failure mitigation.

	Web	Mobile
Asset	$5M	$1M
Security	$250k	$25k
Asset/Security Ratio	20	40
Margin of Safety	5%	2.5%

Table 1: Asset/Security ratio for our example Web site and mobile app

Does this mean the Web app is secure and mobile is doomed? Hardly. What it does mean is that company management is assuming one of the following: the mobile security team is twice as effective as the Web team or the mobile threat environment is half as dangerous as the Web threat environment.

Calculating the Margin of Safety for Web versus mobile shows a simple way to compare across projects, that is, the Margin of Safety imposes an ordinal scale across those projects and ordinal scales are decision support tools, per se. The metric is a means to an end, not an end in and of itself. Are the managers who allocated half the security budget to mobile assuming the team can execute? Are they assuming that they have tools to close the gap? Are they assuming "no one hacks mobile"? Looking at book value does not answer these questions, but it gives a framework to ask these questions, and have rational conversations about how to move forward.

Book value can be used for security architecture, not just projects. Consider what your organization spends on network, host, application, and data security, then compare its book value to the book value of the non-security spent to develop, deploy, and operate those assets.

To illustrate, we fabricate Table 2. In it, what we see is that IT assets like applications and data are underinvested. Assuming the developers and DBAs are highly skilled, care deeply about their work, are trained in secure coding and configuration, and have built their own tools, this could be a non-problem, but absent assumptions like those, the Margin of Safety points to a yawning gap in security coverage of the organization's most valuable IT assets.

In short, security spending should be treated like a bank's assets—a purchased good that is on the books. Using that book value as the starting point lets you cleanly separate the objective measure (book value) from the assumptions you are making (leverage ratio). The amount you extend beyond your security spending is your company's leverage. Note that, just as with banks, leverage itself is risky and amplifies any risk that you already have on your books.

	Security	IT Dev & Ops	Margin of
	$amt/year	$amt/year	Safety
Network	1,000,000	2,000,000	50%
Host	750,000	3,000,000	25%
Application	350,000	5,000,000	7%
Data	50,000	2,000,000	2.5%

Table 2: Some values fabricated to illustrate differences in the Margin of Safety

Margin of Safety benefits:

1. Almost anyone at any level in the organization can calculate this for any of their projects in ~20 minutes.
2. The Margin of Safety can be compared across projects.
3. Gives you a way to see where you are more exposed and some idea where to allocate resources.
4. Uncontroversial and simple to understand metric.

Margin of Safety limitations:

1. Silent on the quality of either earnings power or efficacy.
2. Silent on threats and deployment—so manually you would need to adjust for what is appropriate "internal" leverage vs. DMZ leverage.
3. Like most everything, datasets are likely not available to the general public to test.

As to the benefits, try it out and report back. As to the limitations, we prefer silence to rank speculation.

/dev/random

ROBERT 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

CYBERWORDS

CYBER, CYBER, burning bright:
In the headlines, left and right;
What misplaced idolatry
Has brought thee thus to primacy?

Cyber- a combining form representing computer (cybertalk; cyberart; cybercafé) and by extension meaning "very modern" (cyberfashion). [http://www.thefreedictionary.com/]

Based on this introduction and my little attempt at poetic mockery up there, you probably think I'm going to launch into some tedious diatribe against those who make up terrible new words or overuse a basically meaningless existing one, but as usual you do me an injustice. I am in fact newly fascinated by the versatility of the adjectival prefix cyber. It is not only versatile, it apparently possesses much arcane power—as evidenced by the sudden and dramatic increase of any budget in which it appears. It evinces a great deal of bluster linguistically, as well. Take any ol' word that's been around since your grandmother was in analog diapers, tack cyber onto the front end, and bippity-boo! A new threat/trend/industry/social sensation leaps fully formed from the forehead of iniquity. Allow me to illustrate:

Terrorism becomes CYBERTERRORISM!

Crime becomes CYBERCRIME!

Criminals become CYBERCRIMINALS!

Power becomes CYBERPOWER!

Bullying becomes CYBERBULLYING!

Punks become CYBERPUNKS!

Patriot becomes CYBERPATRIOT!

Spy becomes CYBERSPY!

Espionage becomes CYBERESPIONAGE!

Employment becomes CYBEREMPLOYMENT!

Security becomes CYBERSECURITY!

Space becomes CYBERSPACE!

And, of course: Armageddon becomes CYBERARMAGEDDON!

ad nauseum (see Free Dictionary entry, above, for an even more horrifying expansion of meaning).

I employ the imperative voice here because CYBERWORDS are larger than life and must be shouted when used. I had to wear reading glasses while typing this, not so much because my eyes are bad as to counteract the necessity for scooting further back from the monitor to keep the CYBERWORDS from damaging them with their forcefulness. On the surface, CYBERWORDS are little more than etymological kitsch, but a deeper investigation reveals their subtle psychological potency. Since the prefix cyber conveys no specific information—only a vague suggestion that computers are somehow involved and the technical aspects are therefore beyond the average person's grasp—it can be employed in a wide variety of situations where no other single prefix would work. More importantly, it automatically lends the user a certain degree of "hacker cred," since only a hacker would use words with cyber in them, right? It is well known that in private communications, we . . . I mean they . . . use cyber as many times as possible. It's cyberiffic! Invoking cyber in conjunction with employment is a sure-fire winner. You can spice up a ho-hum résumé in no time flat by converting all your boring normal experience and positions into cyber gold. I have been saddled with résumé-reading duty on several occasions at my own cyberjob and I never fail to be stricken by the number of people with significant cyber-professional lifestyles. After a few dozen exposures to this pathology, I realized that virtually any job in existence could be cyberized. Take this seemingly innocuous narrative, for example:

> I stocked shelves in the electronics department, including CDs, DVDs, mice, printer inks and paper, cables, software, and assorted hardware such as Ethernet hubs, USB hubs, external hard drives, and thumb drives. I also swept and mopped the floors at night after we closed.

Let us run this tiresome, unmarketable mishmash through the miracle process of cyberization and cybervoilà! A polished, employment-assuring chronicle of high-tech cyberwizardry:

> Coordinated cyberlogistics by ensuring the continued supply of cybermedia, cybermateriel for written cyberoutput, and cyberdatacommunications equipment. Also responsible for cyberdatastorage continuity and maintaining a cyberclean cyberenvironment.

Now, that's one young technophile on target to get the cyberjob they've always dreamed of. (Yes, I know that "they" does not belong in a sentence referring to a single individual, but as a columnist you have to be careful about these things. Originally I had "she's," but then I thought someone might believe I was implying that women were only capable of menial jobs. When I tried "he's," I could hear someone complaining to USENIX that one of their columnists said that only men were smart enough to have computer-related jobs. I settled, therefore, on "they've" because English has no third person singular gender-neutral pronoun except "it"—but to use that in reference to a person is to imply that person is not truly human. Besides, according to Wikipedia "they" is now accepted in that grammatical role, and we all know how authoritative Wikipedia is. Capisce?)

As with any overused word, phrase, prefix, suffix, or hyperbole, it's going to get more and more difficult to understand precisely what a word beginning with cyber- actually means. I propose that we begin work now on a translation algorithm that we can modify as the insanity progresses. Perhaps it would be best to adopt the antivirus program model and create signature files we can update regularly as new atrocities are spotted in the wild. You can download the latest ones once a month and run all of your cybergarble through them to get a rough translation.

At its most fundamental, cyber simply denotes "computer" or "digital." For example, cybersecurity is now being used as a replacement for "computer security" or the more ambitious "information security." For the most part you can take any job that formerly contained elements derived from its association with the computing realm and slap cyber on it with impunity. Computer geek? No, cybergeek. Printer repair technician? Uh uh, cyberprint specialist. Computer-based training? How droll: cybereducation, my good man.

This current pandemic of cybermania appears, according to my research, primarily to have cascaded from the adoption by the U.S. government of cyber words in Requests for Proposals and their ilk. In order to appear more in tune with the requirements and therefore more likely to score those lucrative contracts, firms began to couch themselves in cyberosity. The more times they worked cyber into the proposal, the more warm fuzzies they and their stockholders got. Naturally, whatever products they create under the aegis of that contract will be liberally sprinkled with cybers, as well.

I must confess here and now that I was guilty of contributing to this disease early in its incubation period. In 2000 I wrote a feature article for *Security Focus* called "Calling the CyberCops: Law Enforcement and Incident Handling." Had I known where cyber was taking us, or rather where we were taking it, I would have lopped it off and left it next to my workstation for the nice lady who cleans my office to carry away.

There will come a time, probably in the very near future, when cybers are so plentiful that they will overtax the linguistic environment and begin to force native words into extinction. At that point we will have no choice, if we want to preserve some semblance of comprehensibility in the language, but to declare hunting season on them. I will be right there on opening day, armed to the cyberteeth.

Book Reviews

ELIZABETH ZWICKY, WITH MARK LAMOURINE

Perl One-Liners

Peteris Krumins
No Starch Press, 2013. 138 pages.
ISBN 978-1-59327-520-4
Reviewed by Elizabeth Zwicky

Perl One-Liners is what anything named a cookbook should be. This book is not an attempt to cover every Perl one-liner ever, or even every useful Perl one-liner ever; it's not an attempt to cover Perl from end to end. Instead, it's a tour through the landscape of interesting and/or useful Perl one-liners. You can learn a lot about Perl from it, and if you spend much time throwing files around on UNIX, you'll also learn a few interesting tricks. Plus this book is just plain fun.

The introductory chapter is the weakest because it reads like it was added afterwards to provide some background. If you know something about Perl, this chapter isn't necessary. If you don't know much about Perl, this chapter probably isn't sufficient. Skip ahead to the good stuff, and if you get stuck, refer to a better introduction to Perl source.

As a crufty old UNIX type, I would have liked more of the one-liners to mention the existing commands they duplicate. Some of them do, but others don't. This book does not explain that "nl" exists and is happy to number lines for you, or that "grep" knows how to give you context lines.

And a final cranky complaint: for goodness' sake, eight random letters does not a password make. No, not even if you add numbers. And not really if you go to 14 characters. Take your one liner to generate eight random letters and call it a cat-naming scheme or something, but do something less pitiful for passwords. Punctuation. Uppercase characters. Start with 14 characters, not as a possible extension. Otherwise, you are living in the past century, and even then, not at the cutting edge.

Agile Data Science

Russell Jurney
O'Reilly, October 2013. 158 pages.
ISBN 978-1-449-32626-5

Doing Data Science

Cathy O'Neil and Rachel Schutt
O'Reilly, October 2013. 360 pages.
ISBN 978-1-449-35865-5
Reviewed by Elizabeth Zwicky

These two books are both interesting, somewhat eccentric takes on data science, and they make a nice complementary pair.

Agile Data Science walks the reader through one agile approach to data science, end-to-end, covering everything but the actual data science—the tools, the processes, the agile mindset. If I were doing data science for a startup, I would devour this book, and then almost certainly do something else. But that something else would be easier and more intelligent because I'd read the book. And I would have at least some idea about the tools in every piece of the chain.

If you have an existing agile organization and want to add in data science, this book will probably help. If you have an existing data science organization and want to become agile, look elsewhere. This book offers a perfectly nice description of one possible solution, but it's not a discussion of converting to agile.

Doing Data Science is a book version of a course about data science. It does a minimal job of covering tools, and concentrates on algorithms and the idea of data science. The book is a fascinating read and covers a lot of territory. I enjoyed the classroom feel and the introspection, for the most part, but I felt that other parts didn't work as well. The data visualization section, for instance, raised interesting questions, but didn't provide the sorts of insight I found in the algorithm discussions. In the end, I felt like the intriguing data art could have been left out without any great loss.

Doing Data Science provides a good introduction for people getting interested in data science.

Designing for Behavior Change

Stephen Wendel
O'Reilly, October 2013. 346 pages.
ISBN 976-1-449-36762-6
Reviewed by Elizabeth Zwicky

I'm always interested in behavior change, because it's one of the big problems in security: How do you convince people to be more secure? This is a book about building products that change behavior, and now I'm kind of sad that I'm not in a position to try to build a product that would convince people to be more secure. That never actually seemed like a vaguely plausible idea to me before, but now I have read a whole book dedicated to the idea that you could actually build software to change behavior, without even engaging in unfounded optimism.

The author talks believably about both behavior change and product development. In fact, if you're going to build a product that's not about behavior change, you could do a lot worse than reading this and just ignoring the parts about behavior; you'd be left with good advice on defining, developing, and evaluating a product. He advocates an approach based in science, right down to actually figuring out whether you're getting anywhere. He's

open about the difficulties involved, and totally practical about issues like statistical knowledge. How low can you go on statistics and still have reason to believe your numbers? What should you do if you're pushing that boundary and need to know how many test subjects are enough? If a professor would be really handy, how would you get one involved?

Linux Utilities Cookbook

James Kent Lewis
Packt Publishing, 2013. 198 pages.
ISBN 978-1-78216-300-8
Reviewed by Elizabeth Zwicky

You can't cover all of Linux in 198 pages. In order to do something useful in that little space, you need to define an audience and scope tightly. Sadly, this book does not do that. Instead, it veers back and forth between system administration topics and introductory topics, resulting in a chapter on file systems that talks about what "ls" and "cd" do, but starts with a discussion of inodes and superblocks.

Worse yet, that discussion is neither technically nor grammatically accurate. "Things that are not available in the inode are the full path and name of the file itself. This is stored in the /proc filesystem under the PID (process ID) of the process that owns the file." This is stunningly incorrect. (Hint: files still have names when the system isn't running. The names are in the file system.)

I could go on at length, but the bottom line is that this is not an adequate resource for new Linux users or new Linux administrators. People interested in being power-users would be better off with something like *The Linux Command Line* (William Shotts, No Starch), and new system administrators should tackle the intimidating bulk of *UNIX and LINUX System Administration Handbook* (Evi Nemeth et al., Prentice Hall) or *Essential System Administration* (Aeleen Frisch, O'Reilly).

Programming Distributed Computing Systems: A Foundational Approach

Carlos A. Varela
MIT Press, 2013. 271 pages.
ISBN: 978-0-262-01898-2
Reviewed by Mark Lamourine

I'm not sure I'm qualified to review this book, but I'm going to do it anyway. *Programming Distributed Computer Systems* is the first real computer science book I've read in a long time.

Programming Distributed Computer Systems is a teaching textbook aimed at graduate or high-level undergraduate students in computer science. The author's goal is to teach four recent formal models of distributed programming with both theoretical and practical examples.

Varela divides the book generally into two sections. In the first half of the book he treats the models and the formal languages that have been created to express them. In the second half he presents a set of classic programming problems and demonstrates their solutions in languages that support the new models.

In the introductory chapters, Varela first introduces and details the lambda calculus, the basis for sequential programming theory (and implemented as functional programming). I actually have a little experience with functional programming and I found this an excellent refresher course. The next three chapters introduce newer formal languages which add features or concepts to express different aspects of the newer models.

Each model and the corresponding formal language has a name that doesn't necessarily have any additional meaning. They are known as the pi calculus, actors (an extension of the lambda calculus), the join calculus, and the ambient calculus. The programming chapters use Pict, SALSA, and JoCaml to demonstrate the pi calculus, the actor model, and the join calculus, respectively. (There's currently no language that implements ambients.)

Varela explicitly states in the introduction that he intends to present all of the theory before beginning any of the programming examples. He claims that he's found that to be the most effective way to consume and digest the concepts. I'm not sure I agree. In the early chapters, he details the syntax and semantics of each new calculus, but the "meaning" that he offers is almost purely in terms of the formal language itself. He teaches the rules, but not really what they mean and why they're useful or make sense. It's not until he's done with all of the theory that he gets to showing how the formal model applies to real problems. I tried reading each chapter in sequence as he recommends.

Although I wouldn't want code interleaved at the presentation of each new concept, I found myself slogging and struggling to retain the rules because they didn't yet have a meaning for me. To his credit, Varela offers an alternate map for the book. Instead of reading the chapters sequentially, he indicates that you can follow each theory chapter in the first half of the book immediately with the matching programming chapter in the second half. I'd recommend that path, especially for the solo reader.

This book is meant to be taught. Although Varela's writing is clear and not dense or turgid, the topic is abstract (at least to a professional system administrator). A couple of hours of lecture and discussion a week would certainly help throw light in the shadows and fix the ideas in my memory. A set of well-crafted programming assignments would help even more. A solo reader will need a lot of discipline to get everything he or she should from this book.

I'm still trying to decide whether I would recommend this book to anyone I know. It is extremely well written, but I don't know

that many people who would be interested in this topic presented in such an academic way. Formal languages are used (mostly) to reason about programming rather than to produce working applications. The languages that are used aren't mainstream enough for large-scale applications. If I were teaching the analysis of distributed programming, I'd definitely use this book. I enjoyed reading it, and I'm still working my way through the examples again to see what I missed the first time. I guess you'll know best if that appeals to you, too.

Alternative DNS Servers

Jan-Piet Mens
UIT Cambridge , 2009. 695 pages.
ISBN: 978-0-9544529-9-5.
Reviewed by Mark Lamourine

Alternative DNS Servers gives several good first impressions, starting with its heft and with the quality of the paper and printing. Then Jan-Piet Mens had my cranky grammarian heart when he properly constructed the title of the introductory chapter of his book: "Introduction to the DNS." My happiness grew as I read the rest of the book.

This is a book I wish I'd known about when it was published almost five years ago. It's a fantastic addition to the canonical O'Reilly book by Cricket Liu and Paul Albitz [1]. Unlike Liu and Albitz, Mens discusses a wide range of alternatives to ISC BIND, eight in all. He also addresses something I've wondered about for years: how to create a database-backed DNS service.

The DNS is the most fundamental Layer 3 protocol of the Internet. Without it, little else will work. In fact, the DNS works so well that even most system administrators don't fully understand what happens when they issue a lookup query. The first three chapters of Mens' book are an excellent introduction to the workings of the DNS, from the process of resolution to the contents and meaning of the resource records as returned by `dig`. The third chapter itemizes the considerations when planning a DNS service deployment.

This is where Mens gets to the "alternate" in the title. The next 11 chapters are a whirlwind of DNS servers. Some are small, suitable for self-contained labs, whereas others are backed with enterprise-quality replicated databases. In each case Mens highlights the features that would make each one the right choice for an environment. He does include BIND backed by both relational and LDAP databases, but it's treated as just another possibility. This section should be an eye opener even to people who've been running DNS services for a long time.

The DNS servers range from tinydns and dnsmasq, which are suitable for small work groups and labs, to PowerDNS, Microsoft DNS, and BIND backed by MySQL or OpenLDAP, which are commonly used to provide enterprise-level services. This section also has chapters on configuration variations, such as dedicated caching nameservers, private DNS roots, and split DNS for NAT networks.

Mens doesn't leave the reader with a whirlwind tour of DNS servers. The final section addresses operational issues and discusses how they'll affect the choice of server and deployment options. There are chapters devoted to database update procedures, internationalization, DNSSEC, and relative performance (with *real* measurements and details of the testing environment).

The appendices provide details that would have been diversions if they had been placed in line. Many books provide a primer for setting up an LDAP server. Mens provides a collection of tips and tools for problems he's found through experience. His toolkit includes things such as a pattern to avoid updating a zone file without also incrementing the SOA serial number, tuning the database with MySQL-defined functions, and small DNS servers in Perl.

Now comes something I didn't expect. Within a year after publication with UIT, Mens made the complete text of *Alternative DNS Servers* available free in PDF form at the link [2] below.

The DNS is an overlooked element of general system administration. It doesn't need to be the domain of a priesthood or the cranky steampunk machine you don't touch out of fear that it will fail in mystical ways. *Alternative DNS Servers* makes it possible for ordinary system administrators to deploy and manage production-quality DNS services sized properly for any environment.

[1] *DNS and BIND*, 5th ed., ISBN 978-0-596-10057-5.

[2] Also available free as a PDF at http://jpmens.net/2010/10/29/alternative-dns-servers-the-book-as-pdf/.

In Search of Certainty

Mark Burgess
XtAxis Press, 2013. 445 pages.
ISBN: 978-14923891-6-3
Reviewed by Mark Lamourine

I could write a whole article about my thoughts on this book, but that wouldn't be a review; that would be an analysis. I had some pretty strong responses both to the writing style and the content. For the first time since I've been reviewing books, I actually sought out the author so that I could be sure I understood his intent before I responded based only on my assumptions.

Mark Burgess is well known in the sysadmin and configuration management (CM) communities. He has some fairly strong and somewhat controversial opinions about the state and direction of development of CM philosophy and tools. He's also one of only a

handful of people I know of who are even trying to study system administration and CM as proper academic disciplines. Lots of people are doing CM (such as it is), but not many are *thinking* about it in what I consider a rigorous way.

So, let's talk about the book.

In Search of Certainty is presented in three sections. The first two challenge our assumptions about how computers and computation work. In the third section, Burgess describes a way to reason about computer systems and applications that (he claims) has the ability to resolve the issues of stability and uncertainty that are inherent in any large complex distributed system.

In the first section Burgess highlights the flaws in our assumption that our computer systems are inherently stable and that we are in complete control. He begins by discussing the concepts of scale and emergent phenomena. The next sections examine the fact that measurements are necessarily discrete and only approximate the actual state of the observed system, even setting aside experimental error and communication uncertainty (more on those later). He touches on the idea that modern distributed computer systems are so complex (in the colloquial sense) that their behavior has become complex (in the mathematical sense). That is, their behavior has become non-linear and unpredictable in the same way that weather systems and financial market behavior are. We can't control them at a macro scale because we can't even understand them. They fail in spectacular and unexpected ways because they are both unpredictable and contain much more energy in temporary equilibrium than the system can withstand when that energy is released suddenly. The section completes with a chapter on the idea of stability as the process of finding the (local?) minimum energy state of a system (the "zero state") and the idea of engineering to make sure that the desired state of a system is its zero state.

In the second section Mark turns his attention to our view of the state of the systems we create and manage and the uncertainty inherent in any communication over distance. Here he explains the limits to what we can know about the state of a system. Based in information theory as first described by Claude Shannon in 1948, his argument is that even when we get an answer to a query about a system, even when you account for measurement error, there is still the possibility that the message was garbled in transmission or just plain misunderstood by the receiver. (These are not the same thing!) In the same way that quantum uncertainty is a fundamental principle rather than something we can overcome with finer measurements, information uncertainty is a fundamental feature of communication. In fact there is a sense in which they are actually facets of the same properties of physics.

The last section is where Burgess finally starts to restore our hope that we can ever build real, useful systems and have any chance that they will perform as we expect. Given the triple problems of complexity, entropy, and uncertainty, Burgess argues, it is foolish to expect to build and control large, complex, stable and robust information systems by imposing structure and state rigidly from outside.

His response is also based in information theory. Rather than trying to model large systems from above, Burgess proposes modeling them from the bottom. Over several decades, he's formulated something he calls promise theory. A promise is a purely local expression of some desired state. It is more than a simple assertion, which can only have two states, true or false. A promise can be fulfilled or not, but it can also indicate whether the state required repair at the last state check. Relationships between parts of a computer system are expressed as mutual promises between the parts. Promises can be grouped into collections that express more complex structures and behaviors and can also express how the parts will respond if some promise is not (or not yet) fulfilled.

Burgess claims that promise theory and its expression in the software system provides a means to describe the large systems we are deploying in a way that will allow us to avoid the catastrophic failures that are inevitable using traditional configuration methods. He's developed and evolved a software service called CFEngine to embody the theory.

Back to my impressions of the book.

I think I came to *In Search of Certainty* with expectations that Burgess didn't ever plan to meet. I was hoping for a tight technical exposition leading to a conclusion. What he wrote was a twisty personal journey through elements of classical and quantum physics, information theory, and quite a few personal anecdotes and inspirations. For me, the really interesting material doesn't begin until the end of the final section.

I was left hungry to learn more of the underpinnings of promise theory. He spends a lot of time explaining how our current top down software and service creation methods are doomed to ultimate failure but doesn't give much time to explaining how promise theory can be used for engineering.

I found some of the analogies between physics and computation a bit thin. Gas pressure and Boltzmann's constant don't really have that much to do with the tendency of hardware and software to fail. Hardware failure aside, general system configurations don't degrade over time without humans making uncontrolled changes. Although the statistical similarities between one specific computer system behavior and some properties of quantum mechanical systems may be interesting, the differences limit any comparison to a curious coincidence.

Although I agree with his assessment of the current state and trends in creating computational systems and find the mecha-

nism of promise theory interesting, I find Burgess's arguments that promise theory is the way to solve the problems a bit weak. It's not that I think he's wrong, just that his arguments don't make the case for him. In fact, I think there are elements that really deserve more attention than they have been given up to now.

I really want to see more formal research into how the use of more or less purely declarative languages (of which CFEngine is only one) can be used to describe complex systems in a way that is significantly different from any other method. I want to see real demonstrations that the stability and uncertainty issues inherent in large complex distributed information systems are actually mitigated in practice when compared to other methods. I want to see research into how to apply promise theory (or anything else) to the problem of engineering emergent behaviors from localized concrete state definitions.

I came to this book hoping for some of those answers. What Burgess provided was a meandering travelogue of how we've reached where we are now as seen through the lens of his own personal journey. That's not without merit. I think the science and mathematics could have been expressed much more briefly and clearly, but in the end, I think I got the message. I will certainly read whatever comes next, because I want to know, too.

NOTES

USENIX Member Benefits

Members of the USENIX Association receive the following benefits:

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

Access to *;login:* online from December 1997 to this month: www.usenix.org/publications/login/

Access to videos from USENIX events in the first six months after the event: www.usenix.org/publications/multimedia/

Discounts on registration fees for all USENIX conferences.

Special discounts on a variety of products, books, software, and periodicals: www.usenix.org/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/ or contact office@usenix.org. Phone: 510-528-8649

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, *National Institute of Standards and Technology*
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

EXECUTIVE DIRECTOR
Casey Henderson
casey@usenix.org

Transition of USENIX Leadership

by Anne Dickison and Casey Henderson

Two years ago, USENIX took an unusual step in appointing Co-Executive Directors to handle a challenging transition. It was exactly the right move for the organization and for us personally. However, the organization is largely now past that transition.

Thus Anne has decided that it's time for her to take on a new challenge. She concluded her tenure at USENIX in January. Casey has since assumed the role of sole Executive Director of USENIX.

We have enjoyed working with the USENIX communities in our Co-Executive Director roles. It's been a pleasure to further develop relationships that we'd begun during our previous positions at USENIX and to establish so many new connections. Anne has thoroughly enjoyed her time at USENIX and will miss working with such a tight-knit and exciting community. Casey looks forward to continued collaboration with you all in her new role. It's an exciting time for USENIX; everyone in the office appreciates your support as we move forward. Please contact execdir@usenix.org with any questions.

Rikki Endsley recently interviewed Anne and Casey for the USENIX Blog. Find out more about Anne and Casey's shared leadership and what's next for USENIX at usenix.org/blog.

2014 Election for USENIX Board of Directors

Nominating Committee Report

Every two years, according to our Bylaws, the USENIX Association holds elections for all eight members of its Board of Directors. The board consists of four officers: President, Vice-President, Secretary, and Treasurer, and four at-large members. Also, according to our Bylaws, the board appoints a nominating committee whose job it is to ensure that we have a strong slate of candidates to run for election. In addition to the nominating committee, any five Association members may together nominate additional candidates to run for any position.

The USENIX Nominating Committee is pleased to nominate the following individuals for the 2014 Board of Directors election:

PRESIDENT
Brian Noble, University of Michigan

VICE-PRESIDENT
John Arrasjid, VMware

SECRETARY
Carolyn Rowland, National Institute of Standards and Technology (NIST)

TREASURER
Kurt Opsahl, Electronic Frontier Foundation

AT LARGE

Cat Allman, Google

David N. Blank-Edelman, Northeastern University

Fabian Monrose, University of North Carolina, Chapel Hill

Christopher Small, Tivli

Dan Wallach, Rice University

Hakim Weatherspoon, Cornell University

We are excited to present this slate of enthusiastic, talented individuals who are interested in serving the USENIX Association.

Margo Seltzer and Niels Provos
2014 USENIX Board Nominating Committee

Ballots will be mailed to all paid-up members in early February 2014. Ballots must be received in the USENIX offices by March 17, 2014. The results of the election will be announced on the USENIX Web site by March 26 and will be published in the June issue of *;login:*.

Thanks, Rikki Endsley

by Casey Henderson

;login: has seen many changes over the past couple of years as it continues to evolve to respond to the needs of its audience. Rikki Endsley, wearing her *;login:* Managing Editor hat, has contributed greatly to that progress. Rikki led the effort to redesign the magazine to maximize its content and present the information in more usable fashion; she collaborated in the creation of the *;login: logout* electronic-only supplement, which she named, and has sought out many of its contributing writers; and she created consistency in the experience of the magazine in such ways as standardizing each issue's length. We're now moving on to the next phase of *;login:*, which involves streamlining its management and production staffing to assist with the USENIX budget. Thus this issue will be the final for which Rikki serves as Managing Editor, a role that *;login:* will no longer maintain.

While we will miss Rikki's enthusiastic participation in the regular management of the magazine, we're excited that she'll continue to be part of the *;login:* team wearing her USENIX Community Manager hat. (Rikki has so many hats that she totally rocks, by the way; you should ask her about them as well as her collections of shoes and records sometime.) Her dedication to *;login:* as a service to the community, as an outreach tool, and as simply an excellent publication is an asset to USENIX that we're thrilled to be able to channel in a different direction to similar ends.

Please be assured that *;login:* production and management will continue to be as seamless to you, the reader, as always. With Rik Farrow as Editor, Michele Nelson as Production Manager, and Arnold Gatilao and myself on the production team, we will work to ensure that *;login:* continues to meet—and hopefully exceed—your expectations.

Questions? Contact login@usenix.org.

Conference Reports

In this issue:

27th Large Installation System Administration Conference

Washington, D.C.
November 3-8, 2013

Wednesday, November 6, 2013

Opening Remarks and Awards

Summarized by Rik Farrow (rik@usenix.org)

The LISA co-chairs, Narayan Desai, Argonne National Laboratory, and Kent Skaar, VMware, opened the conference by telling us that there had been 76 invited talks proposals, of which 33 were accepted. To fit these in, the co-chairs changed the Invited Talks format so that most sessions were composed of two talks. I was skeptical, as were some others, but the talks actually turned out well, with an excellent set of IT tracks.

There were 24 paper and reports submissions, and 13 were accepted for presentation. The Best Paper award went to Thomas Knauth and Christof Fetzer, Technishce Universität Dresden, for "dsync: Efficient Block-Wise Synchronization of Multi-Gigabyte Binary Data." The Best Student Paper award went to Cristiano Giuffrida, Călin Iorgulescu, Anton Kuijsten, and Andrew S. Tanenbaum, Vrije Universiteit, Amsterdam, for "Back to the Future: Fault-Tolerant Live Update with Time-Traveling State Transfer." (See the open access papers and videos on our Web site at www.usenix.org/publications/proceedings/lisa13.)

John Arrasjid presented the 2013 LISA Award for Outstanding System Administrator to Brendan Gregg (Joyent) for his groundbreaking work in systems performance analysis methodologies, including the Utilization, Saturation, Errors (USE) methodology. Brendan received the award and made two presentations later in the conference, standing in for a plenary speaker who was taken ill at the last minute.

Next, John presented the Software Tools Users Group (STUG) award to Puppet. Luke Kanies was not present to receive the award, but another member of Puppet Labs stepped up in his place. John mentioned that the award was not just to Puppet Labs, but for Puppet in general.

Dan Rich, the current LOPSA president, presented the Chuck Yerkes award to Lawrence Chen, a senior system administrator at Kansas State University. Rich said that although Chen maintains the university email systems, DNS, and load balancers, he is also active both on IRC and many mailing lists, such as lopsa-discus, bind-users, and several FreeBSD lists. Chen often provides not only answers to questions, but also frequently includes examples based on his own experience.

Keynote

Summarized by Thomas Knauth (thomas.knauth@tu-dresden.de)

Modern Infrastructure: The Convergence of Network, Compute, and Data

Jason Hoffman, Founder, Joyent

Jason started the talk by naming the three main producers of information: nature, humans, and machines. The information produced by nature far outweighs that of either machines (e.g., servers, sensors, cars) or humans (e.g., pictures, photos, emails). For example, to sample a single person's genome will generate 20 TB of data.

The infrastructures built to handle the ever increasing data volume have evolved over the years. First, we saw the convergence of network and storage, leading to networked storage architectures. Second, we saw the convergence of compute and data, exemplified by Hadoop. With Hadoop, instead of moving the data over the network to where you want to perform the computation, you start the computation on the node that has the data. This, of course, requires that you actually can compute on your storage nodes.

Next, Jason talked about the technologies used by Joyent to build an infrastructure that can handle today's computing needs efficiently. Joyent has been working with and contributing to a lot of open source technology. For example, SmartOS is a combination of technologies from Open Solaris with QEMU/KVM. Notable technologies from the Solaris world included with SmartOS are ZFS, DTrace, and Zones. Jason also talked about the virtues of implementing system-level software—e.g., HTTP, DHCP, DNS servers—in a dynamic, C-like language environment such as JavaScript/node.js.

In Jason's view, there are two kinds of compute jobs: ephemeral and persistent. For ephemeral compute jobs, Joyent's solution is a system called Manta. Users don't have to worry about spinning up instances to crunch their data, but submit jobs via a Web API run directly on the object storage nodes. Persistent compute jobs follow the established paradigm of provisioning VM instances and manipulating data via a file-system abstraction.

One question from the audience was about the size of the "small" 200 MB SmartOS bootloader used at Joyent. Jason jokingly

responded, "Well, it's definitely smaller than a typical Android update." Another question was from a Google guy who wanted to know how to deprecate SSH as the universal fix-it-all tool for server troubleshooting. Although Jason had no direct answer to the question, he said that at Joyent updates are applied to the core OS image, and services are rebooted periodically to roll out updates (instead of SSHing into each box and applying patches/updates individually).

Invited Talks 1

Summarized by David Klann (dklann@linux.com)

SysAdmins Unleashed! Building Autonomous Systems Teams at Walt Disney Animation Studios

Jonathan Geibel and Ronald Johnson, Walt Disney Animation Studios

Jonathan Geibel and Ronald Johnson talked about how they engineered a restructuring of the information technology group in order to increase its effectiveness and improve its responsiveness to their customers.

Both of the presenters have 21 years' experience at all levels of IT from engineering to management. Both are currently in management positions at Disney. They started the talk with an overview of the physical Animation Studios facilities and a trailer of the movie *Frozen* (commenting that rendering ice and snow is one of the more difficult tasks for computer animation).

The IT organization looked similar to most organizations before the changes: top-down, hierarchical, with technicians reporting to managers reporting to directors. They encountered all the problems associated with this structure, including resource contention, pigeonholing, indirect communication, silos, and more. They endeavored to remove the silos and improve the experience for everyone.

Geibel and Johnson established small, autonomous teams of two to six people. Each team acts as a small startup, and is responsible for research and development, engineering, and support of their specific service. Physical space is a key part of this restructuring. The team members are located in close proximity and have their own "war rooms" in order to avoid formal standing meetings.

Team roles include primary and secondary members, and a team lead. The team lead has technical expertise in the discipline and is responsible for tactical and strategic leadership. This person has no HR duties, simply results-oriented duties. The team lead works with and communicates directly with stakeholders.

Primary members are dedicated to a team. Each person is a primary member on one of the teams, but people can play additional (secondary) roles on other teams. There is no time-slicing: a person's primary role takes precedence over their secondary roles. Secondary members add technical and soft skills to a team.

After describing the new structure, they explored the roles of the managers in this scenario. Managers form their own team, but individual managers are not responsible for functional areas. The management team is responsible for career and team coaching, and team investment strategies. One interesting aspect of this new structure is that people can choose their managers regardless of the kind of work they do. They can also change managers based on preferences unrelated to the functional areas. Each team can (and does) have members that report to different managers. Managers operate more like venture capitalists dealing with issues like maximizing investments (financial and human), constantly evolving the teams, providing resources for the teams, and helping the team leads define goals.

The current organizational structure is completely flat. There is no hierarchy, and the team leads are at the top of the "food chain." The presenters looked closely at whether they simply created a bunch of "little silos," and as best as they tell they haven't.

Geibel and Johnson wrapped up claiming that the restructuring has been successful, and that their next experiment literally is going to break down the walls between groups.

Someone from Qualcomm asked how revolutionary this was. He noted that because of the Pixar merger, Ed Catmull loves autonomy and a culture that lets things happen based on results. This seems to have been an influence on the change. The presenters responded by saying that it certainly happens. They make sure to have one-on-one talks frequently with individuals and need to make sure things are flowing correctly. Managers walk around and keep their eyes on things, and make small tweaks. They noted that the only way to move up is to be the best at things, the team lead is the best person, and team leads can't let themselves get stale in their area of expertise.

Mario Obejas from Raytheon remarked on a similar effort there. Their teams are 25% collaborative, 75% isolated. His experience is that a noisy environment destroys the isolated environment. This is an experiment at Raytheon; one team initiated the change. They have a similar style with team members, so they think it'll work. There is a quiet space available, but they're about to implement a major building restructure, and the quiet space is going away in three months. It's been a team-by-team change, and it's not for everyone.

Becoming a Gamemaster: Designing IT Emergency Operations and Drills

Adele Shakal, Director, Project & Knowledge Management, Metacloud, Inc.; Formerly Technical Project Manager at USC ITS, ITS Great Shakeout 2011, IT Emergency Operations, and Drill Designer

Adele Shakal presented a lively and creative talk on emergencies. She suggested treating emergency preparedness as a (serious) game, and encouraged attendees to be the gamemasters at their organizations.

Shakal started with a couple of questions for the audience: "How many of you have coordinated a drill in your organization?" and "How many of you have been the central point of communication during a zombie apocalypse?" Quite a few people raised their

hands in response to the first question, only a couple did for the second.

Shakal stressed Point One with respect to the context of emergencies and emergency drills: even though we are IT people and our focus is on recovering IT assets after an emergency, people's safety and well-being take top priority. She differentiated the goals and activities of Emergency Response (immediate activities after an emergency) and Emergency Operations (long-term activities after an emergency). Shakal highlighted the positive effect that amateur radio organizations have on contributing to disaster recovery.

Business continuity planning and resiliency planning, Shakal said, are areas that need attention in all organizations. Gamemasters need to identify critical business functions (CBF), including business impact analysis. They need to assess risks and likelihoods of various situations, and identify recovery objectives. Recovery objectives include recovery point objectives (RPO) and recovery time objectives (RTO) for the identified CBF. Of course, these are all brought to light in meetings the gamemasters hold to bring organization representatives together. Shakal reminded attendees to bring treats to those meetings: they keep folks happy and mouths full. Shakal asked how many in the audience have designed disaster recovery for some IT services or for all services. She was impressed when many participants responded affirmatively to the latter.

Moving on to emergency planning and drills, Shakal emphasized the importance of keeping plans current, available, and relevant. Her three-point planning advice was to (1) hope for the best, (2) plan and drill for the most likely, and (3) cope with the worst if it happens. Using examples from other industries, Shakal listed several options for modeling emergency operations centers (EOC) or incident headquarters (IHQ). She listed entities that offer example plans, including the National Incident Management System (NIMS), the National Emergency Management Association (NEMA), the International Association of Emergency Managers (IAEM), Citizen Corps, and the Community Emergency Response Teams (CERT). She talked about the importance of showcasing your organization's EOC/IHQ, and again stressed the benefits of providing food and drink to participants.

Shakal continued with the goals of drills. Life safety drill goals focus on responding to immediate concerns and situations, including evacuations, first aid, safe refuge locations, and information collection and communication. She noted that focusing on life safety will show others that you understand that people are your organization's most important assets. Basic IT emergency operations drill goals include assessment, reporting, and recovery, as well as communication with customers and other outside entities. One important facet of assessment is determining whether people have the necessary places to work during recovery. Shakal further noted that this exercise is fundamentally different from creating an actual outage; the focus must be on communication. She pointed out that simulating a service outage will be much more chaotic if you do not have control of the communication aspect.

Moving on to "mapping unknown terrain," Shakal discussed the need to have current and updated lists of key personnel contact information, a publicized list of top CBF, and a mapping of IT services and infrastructure that contribute to those CBF along with people who can provide updates about the recovery of those services. Regarding the terrain, Shakal stressed the point that people should map only terrain that they need. You shouldn't try to create a comprehensive service catalog for drills if you don't already have one. Don't try to solve all of your organization's problems, simply identify critical business functions and recovery objectives as well as relevant infrastructure and services to support those functions and objectives.

Shakal used the analogy of secret notes for designing the theoretical IT emergency. The notes are essentially the scripts that guide personnel during the drills. Prepare the notes ahead of time, and hand them out with timing instructions to participants. Use the timing and efficacy of the notes to compare with actual performance when you evaluate the drills. Shakal reminded the audience to allow for time to introduce the drill structure beforehand and time to discuss the drill afterward. She presented an example of a secret note chart showing times and events and stressed the importance of respecting people's needs by ending the drill on time.

During an advanced drill (after getting basic drills under your belt), Shakal recommended performing simulated activities such as injuries. She advised against simulating death situations, and instead referred to this as "unavailability." Shakal noted that advanced drills can be intense and that organizations should time them carefully: not too often, but often enough to be effective.

Shakal concluded her discussion with suggestions of guru-level drills. These advanced drills include actual interaction with outside entities (media, government agencies, disaster response agencies, etc.), intentionally conflicting status updates, variable delays in status updates, and randomized simulation of personnel unavailability. Speaking from experience, she remarked on the need to ensure the technical accuracy of secret notes and drills. They need to "make sense" to the participants in order to be accurate simulations. She again emphasized the importance of respecting the time constraints of participants and the benefits of keeping the exercises lively and creative.

Wrapping up, Shakal handed out ten-sided dice to attendees, again focusing on keeping drills fresh and engaging for all participants.

Mario Obejas asked if Shakal had experienced interactions with the FBI's InfraGard program. Shakal has not, but acknowledged

its benefits, and suggested an exercise akin to an "FBI tabletop" interaction.

Paul from Qualcomm asked about recommendations for emergency planning and drills with smaller teams and whether top management is not supportive. Shakal replied that in her experience the top management said "Go forth and do this." Further, that mid-level managers or team leads can't really do this without the larger organization being involved. She suggested holding a showcase as a brown-bag session for other managers: shop around to other managers and directors, use the metrics to sell it to the larger organization. Getting the names of other supportive managers and directors is a good start and doesn't need a "drill" to get going.

Steven Okay suggested that in addition to cookies, bring tech: folks geek out over the cool stuff. He described the Cisco vehicle that is a self-contained mobile network disaster unit. Okay also suggested bringing WiFi access points so participants can stay connected if they need to. He also pointed out that universities often have ham radio clubs (Shakal pointed out that they wear black vests during drills), and critical resources on campus know who the hams are and use them.

Invited Talks 2

Summarized by Andrew Hoblitz (ahoblitz@cs.iupui.edu)

A Working Theory-of-Monitoring

Caskey L. Dickson, Google

Caskey Dickson, a Site Reliability Engineer at Google, talked about common solutions to some of the common problems he has seen involving monitoring systems. He noted that although there are good tools for some problems, monitoring still seems to be stuck in the 1990s. Caskey discussed Google's own homegrown monitoring systems, but to move beyond the hit-and-miss approach to monitoring, they have developed a formal model for such systems. Caskey defined these metrics as "the assignment of numerals to things so as to represent facts and conventions about them," quoting S. S. Stevens. Caskey noted that some of the many reasons for monitoring include operation health, quality assurance, capacity planning, and product management, and said that large quantities of high resolution data need to be monitored reliably.

Caskey said they created metrics at some minimum level of abstraction, with raw counters plus some attributes, and the placing of time series into an accessible format with aggregated and post-computation metrics. Metrics should be brought together in one place and need to remain useful after aggregation to allow for the extraction of meaning from the raw data. Caskey said that when anomalies are detected, something has to communicate this in a useful and meaningful way. He then went through a set of tools—top, sar, *trace, MRTG, Nagios, Ganglia, Cacti, Sensu, Logstash, OpenTSDB, D3.js, Graphite, and Shinken—and recommended a mixed-stack approach; he said that internally, Google uses a combination of Nagios, Graphite, Sensu, Logstash, and Ganglia.

Caskey was asked about information collection and noted that there are many solutions to this problem and that he was only trying to provide representative solutions to solve common problems at different levels of the stack. Caskey then received a question about whether he has used any alternatives to Nagios. He said he had used many alternatives to Nagios, and that he would be open to using a variety of them. The final question was about random input and output. Caskey said that although Google has solutions available, the problem is ongoing for them.

Effective Configuration Management

N. J. Thomas, Amplify Education

N. J. Thomas, a UNIX system administrator for Amplify Education who evangelizes the benefits of installing configuration management tools, started off by noting that he had been using configuration management for ten years and had seen it grow, while acknowledging that it is still in its infancy. N. J. gave nine major principles for effective configuration management, namely, the use of: (1) configuration management (e.g., CFEngine, Puppet, Chef) regardless of team size; (2) version control (e.g., Git, Subversion, Mercurial), (3) a validation testing suite (to test changes as they are made); (4) a code review tool (such as GitHub, Gerrit, Trac); (5) your CM's templating facilities; (6) your CM for provisioning; (7) your CM for automated monitoring; (8) a CMDB; and (9) your CM for DevOps. N. J. defined DevOps as "a software development method that stresses communication, collaboration, and integration between software developers and information technology (IT) professionals."

N. J. pointed out that the point of his talk was to describe the current best practices when installing a configuration management system, to be agnostic as to the choice of particular tools and CM systems, and to provide lessons learned that would be useful when building or maintaining an effective infrastructure that is orchestrated by configuration management. N. J. said that these tools are a time machine that allows system administrators to see what their thoughts were at any given time, as well as a way to go back and look at what previous system administration gurus had done in the past before they left the organization.

One questioner wondered about situations with extreme separation of duties, and whether it would be all right for different teams to handle different parts of the solution set that N. J. had presented as best practices. N. J. said it isn't important how many teams work on this approach, but only that the team is getting the approach right. Someone asked about referencing ticketing system numbers and version control systems. N. J. answered that it is good to integrate ticketing systems with CM tools and that it is also good to use proper VCS practices. Someone wondered which continuous integration tool he uses. He noted that Cruise Control, Hudson, and Jenkins are all potential solutions for continuous integration.

Invited Talks 3

Summarized by Scott Murphy (scott.murphy@arrow-eye.com)

Our Jobs Are Evolving: Can We Keep Up?

Mandi Walls, Senior Consultant, Opscode Inc.

Mandi Walls, technical practice manager at Opscode, gave an enlightening talk on the changes happening to our profession in modern times. The talk covered the evolution of system administration, external influences that have affected the evolution of the field, the state of the field, the effects of the digital economy and globalization on our field, personal development, and where to go from here.

Mandi rolled the clock back by covering the events that shaped the field, our evolution. Back in the day (legacy times), our roles were to protect the company's investment. Machines were expensive, resources were scarce, and sysadmins were supposed to run things and fix stuff. This was all influenced by an investment mentality that computers cost money—boxes, networks, power, and cooling. The monetary cost impacted attitude and behavior and created a "No!" culture. Sysadmins got rewarded for protecting investments, and this fostered a culture that hoarded information and provided fertile ground for the rise of the "Bastard Operator from Hell" (BOFH) archetype.

Mandi then examined the current state of the profession, which is not as advanced as we would like. We need to build on previous work, not reinvent the wheel. More succinctly, progress in a field doesn't happen when everyone must start from scratch. The prior generation prevents this and we end up trolling the next generation with the mantra of "I had to figure it out, so should you." This wastes enormous amounts of time as we reinvent everything. Looking at industry, car manufacturers do not do this—the automotive industry has advanced. Other professions have moved on. Medicine has specializations, yet they share information. Why are many of us still in the dark ages organizationally?

The field is constantly changing, and things get more complex. This creates specialist roles such as networking, storage, datacenter operations, Web operations, IT, etc. We need to evolve the field. According to the paper "Prospects for an Engineering Discipline of Software," written by Mary Shaw at CMU back in the late '90s, there are three stages to the evolution of a field: craft, commercial, and engineering. We want to be in the engineering stage with our profession.

There are still challenges to moving the profession forward. Unskilled workers hold organizations back. Investing in training and professional development must be considered a higher priority. Junior people need experience and how do they get that? Learning from the senior people in an apprenticeship and journeyman role has not been embraced by the profession as a whole. This doesn't take into account new tech, updates, and kernel patches. Constant change is endemic to the system. We need to make sure that "stuff" coming from the product team will be well supported and well maintained and will operate as expected.

Mandi continued with the digital economy and globalization; it is here and is pervasive. As time passes, more of our day-to-day life is lived or augmented online. In many countries, we are nearly constantly connected. This requires systems that are increasingly complex and interconnected to provide the many services in which we are investing ourselves. The barrier to technology adoption is getting lower all the time and expertise is no longer required to use the services, which is not a bad thing, but we need to remember that the users of these services are not "dumb" or "muggles" or some other disparaging description. They are our customers and they no longer need a technical background in order to make use of the technology. We need to evolve from the BOFH syndrome and change the way we relate to others. We are the enablers of this new economy. Protectionism and lack of engagement with our users limits our growth and our ability to adapt to the changes around us. Today, escaping technology takes extreme effort because it is everywhere around us and growing. This creates new opportunities for more people.

Mandi went on to consider how we enable ourselves to be an integral and better part of this new world. We need to develop new skills by borrowing practices from software engineering, learning to touch type, using the tools we have available such as version control, learning to code (or code better), and embracing the cloud. We need to have better organizational awareness. Document everything. Share. Use a wiki or similar tool and make sure people have access. Perform testing and code reviews, check each other's work. This will build trust in your processes. Through this whole process, we need to be proactive—say "Yes!" instead of "No!" and if you can't say yes, ask questions and provide reasons why you are saying "No."

Mandi moved toward a conclusion by asking, "Where do we go from here?" The cost of systems has been mitigated by the "cloud" and other services, so gatekeeping is mostly dead. Reject the BOFH syndrome as that mentality has no place in the now or in the future. Work on getting to "Yes," and help enable your organization. Share knowledge, as protectionism and information hoarding keep us from building on previous work. Find our value proposition when we aren't the guardians of large expensive systems; we are the facilitators of large amazing ideas.

And finally "Why?" Because someday we are going to be running the O2 systems on starships.

Invited Talks 1

Summarized by Thomas Knauth (thomas.knauth@tu-dresden.de)

User Space

Noah Zoschke, Sr. Platform Engineer, Heroku

Noah Zoschke, senior platform engineer at Heroku, talked about the many layers of abstraction sitting between a user's application and the bare metal hardware. In a typical platform there are at least five layers: starting from the language-level virtual machine, e.g., Ruby, over the OS-level container, e.g., LXC, then the Linux guest OS, the hypervisor, before finally executing on the physical CPU.

In the second part of the talk, Noah proposed future directions where kernel and user space should be heading. Besides more expressive APIs and less clutter in general, that lower layers expose information to the upper layers is important.

Questions raised at the end of the talk reconfirmed the main points of the talk: clean abstractions and APIs are important for the application developer. They draw the boundaries of what the developer can and cannot do on a platform service. Also, some of the layers re-implement the same functionality, e.g., caching, which may lead to unexpected and adverse effects. Noah said that Heroku's approach to dealing with failures is to fail fast. If a component behaves weirdly, it is stopped and restarted on a different node.

Observing and Understanding Behavior in Complex Systems

Theo Schlossnagle, CEO, Circonus

Theo Schlossnagle's talk focused on the need for good monitoring solutions to aid the developer/administrator in understanding complex system behavior. Events in distributed systems are especially hard to correlate because there is no global clock to perfectly order events. This makes it hard to answer even a simple question about which component is responsible for a certain part of the system's behavior.

Theo provided examples of dos and don'ts in the context of monitoring: e.g., when to use polling instead of pushing for metrics. Polling is best suited for infrequently changing values while the push mode is best for fast-changing metrics. The push model is best implemented in an event-based fashion, i.e., hook into the mechanism that leads to a counter being updated to get notified whenever the counter changes. This way, no change will get lost.

Questions at the end of the talk included whether Theo could give recommendations for data visualization tools to help diagnose problems. Theo answered that there are open source as well as commercial tools available to do the job. As for systems which do not have DTrace, e.g., Linux, what other means are available to collect in-depth metrics? System tap is an alternative to DTrace on Linux, and oprofile is also an option.

Invited Talks 2

Summarized by Cory Lueninghoener (cluening@gmail.com)

Building a Networked Appliance

John Sellens, Syonex

John Sellens started his talk with a simple statement: he wanted to build a thing. The focus of his talk was not about the thing; instead, it was about the technological path he took to get his thing from the idea stage to the product stage. Throughout his talk, John was purposely vague about the thing he created; this was for two reasons: to prevent the talk from being a marketing talk, and to show that the same process can apply to a wide range of projects.

John wanted to create an appliance that he could put on a remote network and use to collect information. He began by describing his original idea, which was large and comprehensive. After seeing other companies that had a similar idea but that were no longer in business, he concluded that he needed to narrow his scope to something more doable. This led him through a multi-year series of design iterations, each time making the design a little cheaper and a little smaller. He finally settled on the Raspberry Pi as his base, which gave him all of the hardware features he needed at a price point he liked.

After finding the perfect hardware, John next needed to find a perfect software stack. The hardware helped dictate his operating system choice, and he concluded that going with the community is best: the Debian-based Raspbian distribution. The rest of the software stack was based on as many standard tools as possible, plus some custom scripts and a custom read-only root file-system implementation.

John's final design is a simple Raspberry Pi device that runs autonomously; it calls home every 15 minutes to ask for instructions, reporting in as work is done and with an hourly summary. Security is simple and minimal, and provisioning of the hardware is highly automated. John ended by noting that this technical narrative was only a small part of the whole product creation story and included a link to more information about his product: http://www.monbox.com/.

How Netflix Embraces Failure to Improve Resilience and Maximize Availability

Ariel Tseitlin, Director, Cloud Solutions, Netflix

Ariel Tseitlin's talk focused on how Netflix increased their resilience against failure by taking advantage of failure itself. He began with an overview of how Netflix streams video to its 40 million customers, culminating in a TV test pattern graphic that he described as what keeps him up at night.

To prevent the video streams from failing, Ariel described how Netflix built the Simian Army, a suite of tools that purposely injects failures into their cloud infrastructure. By ensuring that failures occur, they have forced themselves to design around failure and make it a non-issue.

The Simian Army is a collection of tools that inject different types of failure into the production Netflix cloud: Chaos Monkey (randomly shuts off cloud nodes), Chaos Gorilla (shuts down entire availability zones), Chaos Kong (shuts down entire regions), Latency Monkey (injects arbitrary latency), Janitor Monkey (cleans up the cloud environment), plus several more. Ariel noted some important details surrounding the use of these tools: they're used during business hours, can be disabled when a real customer issue is going on, and are used as part of a larger culture that focuses on learning instead of blame.

Ariel closed by describing Netflix's strong relationship with open source software and by noting that the Simian Army is available to others as an open source project.

Invited Talks 1

Summarized by Rik Farrow (rik@usenix.org)

Storage Performance Testing in the Cloud

Jeff Darcy, Red Hat

Jeff Darcy began by showing a Venn diagram with three ovals representing the three major elements of his talk: "performance," "storage," and "cloud." Jeff described two types of performance measurement, for the network and for storage, and how bandwidth and throughput can improve with additional threads, whereas latency often deteriorates with more threads. Jeff quoted Jeff Dean's idea of "tail at scale = fail," referring to a front-end that spawns hundreds of requests, and the one system that is always slow means all responses will be slow.

When measuring cloud or cluster storage, Jeff pointed out several ways not to do it, even though these methods are popular. First, if you measure a test from beginning to end, the laggards throw off the time. Second, some people fire up streams sequentially and add up the results for a total. Jeff called the third technique "stonewalling," where the tester starts up many threads, stops timing with the first one that completes, and scales up to calculate the total for all threads. Jeff labeled this as really wrong, as it ignores all of the slowest I/O. Instead, Jeff pointed out that you actually want to see the complete distribution of each run, showing an example graph (slide 8) that begins with a staircase, drops off suddenly, then finishes with a long tail of stragglers.

Jeff turned his analysis on storage performance factors: the size of requests, types of requests, cached/buffered vs. synchronous, and read vs. write loads. For example, GlusterFS makes a lot of use of extended attributes, so performance tools that ignore this in favor of data will produce incorrect results. IOzone can do lots of different tests, but stonewalls by default. FIO does better at random distributions, and filebench attempts to simulate both I/O and the producer/consumer relation. Jeff mentioned Dbench, which is for Samba testing, and COSBench, which has a bizarre configuration. Jeff said that we need better tools.

Jeff then looked at cloud issues (the third oval). First, in a cloud environment, you can have noisy neighbors, and can't even expect to see similar results in subsequent tests. He showed results from tests within Amazon, Rackspace, and Host Virtual. Amazon had the highest performance, but also the highest variance, and Host Virtual had the lowest performance, and the smallest variance.

Jeff also mentioned the clunker problem, which is that you will sometimes get a "clunker instance." Netflix tests for this when spinning up a new instance, and discards instances that appear to be clunkers. The problem might be the host, but could just as easily be an overloaded switch. Also, cloud providers will ignore things such as synchronous write flags. He concluded by saying that massive variability is what makes testing storage performance in the cloud difficult. Characterize your workloads, test many times, across many providers, and think in terms of probabilities instead of averages.

There was time for only one question. Someone asked whether he looked at error rates. Jeff replied that he had not, but sometimes you might get an instance performance of zero. He suggested that people look at http://hekafs.org/index.php/2013/05/performance-variation-in-the-cloud/ for an example of storage performance in the cloud.

Surveillance, the NSA, and Everything

Bruce Schneier, Fellow, Berkman Center for Internet and Society

Bruce Schneier actually spoke from the cloud, successfully using Skype with two-way audio and video for his presentation. Bruce had been speaking about a similar topic at an IETF meeting, and spoke without notes or slides for 35 minutes, easily managing to engage, enlighten, and entertain the audience.

He started by saying that he thought the neatest thing about the NSA snooping was all of the codenames. For example, Musculature for tapping connections between datacenters, Quantum for packet injection, and Foxacid for serving exploits from a Windows 2008 server. He then pointed out there will be a lot we will never know, except that the NSA has turned the Internet into a giant surveillance program.

Bruce pointed out that data is a by-product of the information society. Companies and computers mediate our interactions, and this data is increasingly stored and searched. At this point, it is easier to save all data than throw it away. Surveillance is the business model of the Internet, and can be seen as a natural by-product of using computers for everything, producing a public-private partnership in which the government accesses data already held by corporations. Understanding why the NSA wanted to take advantage of this is easy.

Bruce described metadata as surveillance, and used the analogy of hiring a detective who watches where you go and with whom you meet, the metadata of your daily life. Other intelligence agencies besides the NSA do this, and, in fact, so do other well-funded nations; the US has a privileged position on the Internet, be we know the Chinese are doing more or less the same kinds of surveillance.

We do have choices. We can use encryption, which Snowden told us, in his first interview, actually meant that the NSA got ten times more data from Yahoo! than from Google, because Google used SSL. Tor is safe, but the endpoint security is so weak there are ways around Tor and other forms of encryption.

The NSA is investing in groundbreaking technologies having to do with encryption. Bruce guesses that this may have to do with breaking elliptic curves (used for public key cryptography), breaking RSA, and/or breaking RC4, a stream cipher with known weaknesses, and the Windows default until recently.

The NSA was not prepared to be exposed, and we will see some changes. US companies have found that it is good for their business to fight NSA monitoring, and international blowback has hurt US businesses with cloud-styled solutions.

Bruce talked about making eavesdropping more expensive in his IETF presentation. Right now, collecting everything is cheap. Change the economics, and we can encourage targeted instead of wholesale collection. Currently, using encryption is a red flag. But if lots of people used it, encryption would provide cover for those who need it. Also, having a handful of email providers (Google, Microsoft, Yahoo!, Apple) is less safe than having 10,000 ISPs.

We need more usable application-layer encryption. Twenty years after PGP was invented, we still don't have usable email encryption. Bruce also declared that open source projects are harder to subvert, and that we need more personal security projects. Anonymity tools, such as Tor, work, and the more the better.

Finally, we need assurance that the software we use does what it is supposed to and not anything else. The NSA can't break the laws of physics, and encryption helps limit bulk collection. This is largely a political problem: transparency, oversight, and accountability. And making the Internet stronger will mean everybody wins.

What the Snowden docs tell us is what anyone with a budget can do. The worse blowback will be countries clamoring for their own, domestic Internets, that they can use to surveil their citizens. The ITU taking over the world would be a disaster, but we do need better Internet governance.

Someone wondered about the chances of people giving up their conveniences, like iCloud. Bruce agreed that's true, and that Google's profits show that people care little about their data. But regulation can help here, just like it helped with child labor or organ donor markets. Someone else wondered how we are going to achieve a secure Internet. Bruce responded that the tools must be baked in, whether AV or encryption. Someday, data may reside totally in the cloud, and companies like Google want people to feel that this is secure. Mario Obejas (Raytheon) pointed out that solutions like NoScript are not good for family members. Bruce agreed, and said that the more work you can do behind the scenes, like turning on whole disk encryption, can help. You can raise the bar to prevent wholesale collection, but not targeted collection.

Someone asked Bruce to summarize the IETF meeting. Bruce said that he was surprised that the IETF wants to do what they can to harden the Internet, but there will be long, hard slogs to any standards. We've learned of an attack against the Internet, and the scale and scope of it were a surprise. Someone suggested enlightening the populace about security, and Bruce asked him if he had met the populace. Educating users is fraught with danger. John Looney (Google) had talked to the European parliament about security, and they focused on cookies, leaving Looney saddened. Bruce agreed, saying that regulations focused on one technology are irrelevant. What's needed is focus on the broader problems—privacy, not cookies.

Bruce closed by pointing out that fear has subsided in the US over the past ten years. There wouldn't even have been a debate if the NSA revelations had occurred ten years ago.

Invited Talks 2

Summarized by John Dennert (jdennert@iupui.edu)

Leveraging In-Memory Key Value Stores for Large-Scale Operations

Mike Svoboda, Staff Systems and Automation Engineer, LinkedIn; Diego Zamboni, Senior Security Advisor, CFEngine

Mike Svoboda from LinkedIn presented how they used CFEngine and Redis to solve problems in their previous administration process and deal with the transition from administrating about three hundred machines three years ago to tens of thousands of machines today. The use of both CFEngine and Redis has allowed them to move from making changes in the blind and taking several hours to spin up a new machine to being able to make informed changes and spin up an entire datacenter in an hour or even minutes. All of this while dramatically increasing the number of computers they support without having to increase their staff size.

CFEngine is an IT infrastructure automation framework that has proven to be vertically scalable and has no reported security vulnerabilities. Because it is written in C, CFEngine has the advantage that is not reliant on an underlying VM, such as Python, and is lightweight. Each machine will pull down the appropriate configuration profile and make the changes necessary. Cached policies are used if the device is offline.

Redis is an advanced in-memory key-value store that has built-in support for hashes and dictionaries. Redis is used to keep track of information about all the machines, specifically those in production, and can be queried at a local, datacenter, or global level. The most commonly requested information is cached to speed up access time. The information at a local level is spread across several servers per site, providing load balancing and failover and, for larger queries, can be accessed in parallel as each server involved in the query is asked for the same information. They found compression of the data upon insertion to be significant as it allowed them to leverage the computing power of the end-nodes and to minimize the amount of RAM needed as well as the amount of data transmitted across the network.

This combination has allowed them quickly to answer questions about production machines that LinkedIn engineers were asking, but that the IT staff hadn't been able to answer previously in an accurate or timely manner. Rather than injecting SSH into each machine multiple times a day looking for information requested by the engineers, they are now able to access an up-to-date status report of the production environment and to make changes as necessary using CFEngine.

Diego Zamboni from CFEngine came up for a few minutes at the end of Mike's talk to describe a few of the features in progress for future releases of CFEngine, such as a dashboard for the Enterprise version. He also announced that CFEngine is moving toward being more open about what they are working on for upcoming releases. CFEngine is currently working on release 3.6, and the dashboard may be included in it or possibly the following release.

Someone asked how they deal with the freshness of the data. Mike answered that the time-to-live was 60 minutes and the data is refreshed every five minutes. Additionally, he was asked whether they could track changes, and he replied that they could maintain history and that they have ZFS snapshots. Mike was also asked about how users get data out of the system. He said that they currently get it out through a command-line interface using a Python object as there is currently no GUI or Web interface. When asked about whether they had tried using Puppet for this, he responded that CFEngine provided them with automation and that it allowed them to spread out the pushing of configuration to their machines and to set the splay time. He was later asked about how Redis compared to other memory value stores. He answered that due to Redis's built-in support for dictionary and hashes, they were able to work on optimizing it rather than working on building support for dictionary and hash values. Diego was asked about the upcoming dashboard he had described and whether it would be available for both Enterprise and open source users. He said the plan was only to release it to Enterprise users, but that could change.

What We Learned at Spotify, Navigating the Clouds

Noa Resare and Ramon van Alteren, Spotify

Ramon van Alteren and Noa Resare co-presented, switching back and forth every couple of minutes or so. Ramon started by briefly describing Spotify, a streaming music service made of many smaller services designed by hundreds of engineers working in autonomous teams across multiple continents. The challenges they faced included letting developers be developers so they could build awesome tools. Each team also needed to be able to develop and test with other teams as many of the services relied on other services.

In 2009, Spotify had approximately 30 engineers and used KVM hypervisors. They tried using DNS as a database and found that to be a bad idea. The system could be accessed from anywhere and worked okay given the relatively small number of engineers working at the time. Over time the decision was made to make the switch to CloudStack. Although CloudStack was not as highly available as the previous system, they now used SSH key authentication to authenticate over HTTP and to receive the token needed for access. This was a step in the right direction, but they didn't want to have to run a full CA/PKI.

CloudStack served them well, but the speed that Spotify was growing and changing led them to continue to explore other options. CloudStack had a lot of features that weren't needed, and Spotify had trouble scaling the hardware backend to match the increasing demands as their product's popularity and their company continued to grow rapidly. This experience taught them how difficult it can be to run a cloud.

The latest change was moving to their current provider, Amazon; it took about two weeks to switch the backend over. Having Amazon engineers on-site has proven to be extremely valuable as they have continued to improve their service and deal with networking problems that have been the biggest ongoing issue. The switch to Amazon has made it easier to spend money because creating virtual machines is now easy, which led to the implementation of new policies to make sure that once a VM is no longer needed it is allowed to die. Their motto is, "Spending is fine, wasting is not." Their challenges moving forward are to continue to keep up with production and to continue the rapid improvements made in integration testing. Their final words of wisdom to others working on a cloud-based application was that the asynchronous task model is important and the earlier you start using it, the easier it will be to use.

Noa and Ramon teamed up to answer questions at the end. Someone asked about the creation of dynamically generated instances for developers. They answered that this was done by individual teams with central configuration done using Puppet. They were also asked about what the development cycle looked like. Their response was that there is no standard pipeline but rather that best practices are made available and then each team is responsible for its own testing. The final question was about how they dealt with deploying multiple services. Noa and Ramon explained that they use a central testing environment and that a team can connect to another team's testing environment as needed for testing their systems together.

Lightning Talks

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

Lee Damon introduced the session. Lightning talks aren't like rehearsed talks with slides, but are brief talks (five minutes or less), ideally inspired by other things at the conference.

Doug Hughes described developing a network switch autoconfiguration system by finding a simple Ruby DHCP server that he extended to supply boot images and configuration data to new switch devices as soon as they were plugged in.

Tom Limoncelli is tired of giving good advice, so instead he provided some items of bad advice, such as: The waterfall development model has a great aspect—you can cash your paycheck and leave before anyone knows whether the code works. You can hire more DevOps people by giving existing employees new business cards that say "DevOps." What's better than DevOps rapid deployment? Editing code directly on production servers!

Alan Robertson described how he raised his two daughters who now both have careers in IT. When one daughter was an infant,

he wrote a VIC-20 program that changed screen colors when she pressed keys. She loved it and learned she was in control of the computer. The other daughter loved math so much she would ask for math problems while riding in their car, so Alan taught her powers of 2 because they were easy for him to remember. He knew he had been successful when he had to go look for his grounding strap in his daughter's room.

Greg Mason's management wanted to develop business processes by dumping things into Google Docs spreadsheets, but instead he helped them build a better solution by finding a cloud-based help-desk system and involving them in obtaining requirements and letting them customize it to suit their needs. He no longer has to deliver sadness as a service.

Jorj Bauer talked about how open-source VoIP solutions have scaling problems with call transfers above about 15,000 users because they haven't been able to use the proprietary method in commercial products. He helped reverse-engineer the commercial method to develop an open-source implementation.

Eric Wedaa had to help plan a full datacenter power down, in particular in what order things would have to be turned back on to work again. There were still problems, such as undocumented IP addresses, someone using an obsolete printout of the spreadsheet with the power-up plan, and the datacenter doors locking secure when the power was removed and security not having any keys, but by luck someone was still in the datacenter to let others back in.

Lee Ann Goldstein made a public service announcement about issues with the .mil domain migrating to using DNSSEC and all email addresses under .mil being migrated to a .mail.mil subdomain, making it difficult for outside correspondents to reach .mil users. She advised people having problems reaching .mil domains to get in contact with .mil admins about these problems.

Brian Sebby described how taking improv classes helped him become a better sysadmin by improving his communication skills. In improv, you learn to say "Yes, and..." to what your partner says, which acknowledges their needs but lets you add your own comments, helps you build a relationship with your partner, and focuses on immediate needs without dwelling on the past.

Stew Wilson talked about how writing Dungeons & Dragons scenarios made him a better sysadmin and documenter, because they require a focus on paring descriptions down to basics without ambiguity. If you just tell people what to do, they will do it, but if you tell them why they're doing it, they come up with better ways.

Tom Yanuklis encouraged sysadmins to get involved with maker spaces for meeting and learning from other craftsmen and possibly even getting help with building things you need, such as Arduino-based environmental monitoring sensors. If you don't have a maker space in your area, consider starting one.

Ryan Anderson recounted his experiences with setting up an LDAP server for his organization, first with Red Hat Directory Server, then migrating to OpenLDAP when he needed to upgrade. His main points were to look at requirements before jumping into things and to use configuration management to implement things well and get self-documenting code.

Peter Eriksson developed a fast backup method for MySQL by using a Python script to flush database tables, taking an LVM snapshot of the file system with the database files, and then dding the snapshot image to another system. He could then verify the backup using a slave database, and scaled the backups from 50 GB to 1.1 TB.

Heather Stern talked about issues with managing cloud computing versus co-location or shared hosting. She offered three questions she uses: "Is that an it? Is that a them? Is that me?"

Stephen Okay described how more and more of the devices, or even users, he manages are robots, especially for telepresence applications, and what this means for system administration when the systems you manage are physically moving around under someone else's control.

Lee Damon gave a semi-improvised talk based on topics suggested from the audience, combining the topics of robotic dinosaurs and Google Glass (which he was wearing). He described how he had set up his phone to show text messages on his Google Glass display, and that he was happy running computers not powered by dead dinosaurs.

Brian Sebby also gave an improvised humorous talk on the audience-provided topics of managing cats with Chef and cat pictures on the Internet.

Stew Wilson also returned to talk about combining the security principle of least privilege with configuration management to give his users the ability to manage software installation, licenses, and firewall and VPN settings themselves without need to have root or involve sysadmins.

Ryan Anderson gave a second talk on what he's learned in the last three years working at a company that drastically reduced their sysadmin staff from six to two people. He used Linux, virtual machines, and configuration management to expand services and save money, which also allows his remaining staff person to cope with him being away at LISA for a week.

Nathen Harvey said that if you really want to "level up" your professional skills, quit your job. The best way to learn new skills is to work somewhere else with new people. If you aren't willing to quit your job, try interviewing for other jobs every 12–18 months to learn things about how other companies work.

Adam Powers told everyone they're doing monitoring wrong. You shouldn't care if a disk is full if your application is working, so your monitoring should only tell you when your application is broken.

The final talk by Marshal Weber was on risk management in the high-frequency trading industry, where people have been commended for things like chopping through a wall to open a door to ventilate an overheating machine room or abruptly powering off a development server to cannibalize its power feed for a critical production server that lost a redundant feed. Your personal definition of risk may be different from your company's definition of risk.

Thursday, November 7, 2013

Plenary Session

Summarized by John Dennert (jdennert@iupui.edu)

Blazing Performance with Flame Graphs

Brendan Gregg, Joyent

Brendan Gregg's talk about Flame graphs was for everyone from beginners to experts. He shared how and why he had developed Flame graphs and showed different examples of how it had helped him answer questions and sort through data in a way that he couldn't with other tools. He has the code and examples posted on GitHub (https://github.com/brendangregg/FlameGraph).

The first example he showed was from when he was trying to determine why a production MySQL database was having such poor performance. The condensed output was over 500,000 lines of text. After converting this information into a Flame graph he was able to view the same information in a visual and interactive manner on a single Web page and find the source of the problem. Reading a Flame graph from top to bottom allows you to see the ancestry of a particular function, whereas reading from bottom to top allows you to follow the code path. The top edge of the graph is on-CPU directly and going down the stack shows who called the function that is on-CPU.

At first Brendan tried using the stack depth as the y-axis and time as the x-axis but this proved difficult to extract useful information out of so he switched to having the x-axis display the function calls alphabetically, which allowed him to merge adjacent functions better and greatly improved the readability of the graphs. The length is proportional and allows for quick visual comparison between different functions. This layout also makes it easy to spot code path branches and makes it easier to understand what is happening over a period of time.

Now that profiling is easier and faster, it can be used in more places and to examine different performance data. The hash function allows for use of a consistent coloring scheme to make it easier to examine data gathered from different systems. Flame graphs can also be used to analyze data from different profilers so long as the profiler provides full and not truncated data. By converting the profile data to single lines, it can easily be grepped to focus on specific areas.

By taking the Flame graph output and displaying it using SVG and JavaScript, a common standard can be used and the information can be viewed interactively in any browser. For instance, even rectangles too small to display text in can be moused over to display their information. It also provides a tool that can be used for different tasks, from teaching students about kernel internals to being used to analyze production environments. Examining different statistics such as memory allocation, logical and physical I/O, and on- and off-CPU data allows different classes of problems to be examined using the same tool.

There was only time for one question at the end—why use 97 or 99 Hz.? Brendan's answer was that he'd chosen that so as not to interfere with timed events and that he had tried sampling at 100 or 1000 Hz and had seen the effects due to the increased precision of kernel timing.

Women in Advanced Computing Panel

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

Moderator: Rikki Endsley, USENIX Association

Panelists: Amy Rich, Mozilla Corporation; Deanna McNeil, Learning Tree International; Amy Forinash, NASA/GSFC; Deirdre Straughan, Joyent

Endsley described the panel as part of an initiative started last year to provide practical solutions and tips to recruit women for careers in advanced computing. She was also glad to see a mixture of people in the audience since everyone needs to be part of the solution. She then had the other panelists introduce themselves.

Endsley's first question to the panel was how they recruited women for computing jobs. Straughan said, "You have to go where the women are," suggesting other conferences such as the Grace Hopper Celebration of Women in Computing and AdaCamp. Rich said there is also an "old girls' network," and that having a culture of respect in the workplace is important, as well as avoiding job descriptions aimed at more aggressive personalities. Forinash recommended supporting internship programs, which is how she found her job at NASA. McNeil said that managers should ask themselves whether they want to have a different voice on their team.

Endsley then asked the panel about generational differences in the workplace, particularly from a management perspective. Rich said she is in a workplace where most of her coworkers are younger than she is, and that it's important to build good relationships and listen to the younger and less experienced people. McNeil thinks of herself as a mentor, not a manager, and tries to support others on her team. Straughan sees her job as being about providing training rather than management, and since the person who reports to her is older than she is, she won't tell him how to do his job. Endsley remarked that as a "Gen-Xer," she finds people of different ages have different expectations about the workplace.

Endsley asked the panel how they get people to recognize their accomplishments and successes without seeming pushy or whiny. Forinash said she has done things like give her boss a list of things she's done or has told her boss when she's very busy on something. McNeil said that telling other people what

she's doing gets them excited and inspires questions. Straughan described giving her new managers a checklist of what she put into a training course to show them how much went into it. Rich talked about "impostor syndrome" and how she builds bridges between IT and other groups by complimenting good work by them to their managers, so other groups compliment good work by IT to its managers.

Endsley expanded on impostor syndrome and how she sees it as common in academics, and more common with women than men, which makes women feeling unqualified reluctant to submit conference talk and paper proposals. She asked the panel for their advice to people who feel they are unqualified to contribute. McNeil uses private encouragement and reviews their work with them. Forinash recommended writing down what you do, and participating in Toastmasters to practice speaking and presentation skills. Rich said everyone feels uncomfortable about public speaking at first, and has done things like co-present with others so they can fill in each other's lapses. Straughan helps encourage people she works with and promotes their work to conference organizers, who come to her for help in finding presenters; she also finds that technical people tend to underestimate their skills because they compare themselves to the giants in their field. Endsley suggested smaller events, and Straughan local meetups, as good places to start.

An audience member asked the panel how to make your voice heard when other people don't seem to value you. Straughan said sometimes it's easier to find recognition outside your company. Rich said it helps to have advocates, and being unafraid to express her opinions, also uses that to help others be heard. Forinash said it can take time to establish a reputation so others realize your abilities. McNeil finds it important to recognize the different perceptions of others, and Rich suggested reframing what you're saying to the level of your intended audience. Endsley added that once you recognize you have impostor syndrome, you can address it, and recommended the book *Confessions of a Public Speaker* by Scott Berkun.

Endsley said that people writing their own bios may want to avoid looking "braggy," but that's what a bio is for. She then asked the panel if they thought men were told not to be too braggy. Straughan has helped write bios for others because it's easier to brag about your friends than yourself. Rich and Forinash both update their résumés annually to help remind them of their accomplishments. McNeil said getting feedback from others about what they see in you helps you get out of your own head. Endsley thinks it's good to float things past other people; once she was asked to proofread a friend's résumé and rewrote it based on what she knew of the friend's accomplishments.

An audience member described as being another "Gen-Xer" on her seventh career, got an English degree after the dot-com collapse and is now reentering the technical workforce. She asked the panel how they deal with career change. Forinash and Rich both said being able to say when you don't know something, but are eager to figure it out, helps others trust your skills. Rich also recommended the LOPSA mentorship program as a way to connect with experienced people for advice. McNeil said that if you've had other careers, you're probably good at a lot of things and taking on new ones. Straughan observed that the fastest way to learn something is to teach it, and to not be afraid to ask for help. Endsley finds value in networking via mailing lists, user groups, and conferences.

Jessica from UCSD said she enjoys being involved in technical outreach at her campus but no women come to their meetups; she asked the panel how to encourage women to be involved in these social events. Straughan said sometimes individual women feel intimidated by the preponderance of men at such events, but finding a group of women to go together can overcome that. McNeil suggested having meetups with coffee instead of beer. Rich said women need incentives to attend social events, such as opportunities to find new jobs.

Another audience member asked the panel for their advice to men as coworkers and managers. Straughan said she doesn't see it as men versus women, but promoting techniques to help all people be heard, even ones who feel reluctant to contribute. Forinash said the best way to encourage women is to simply treat them like human beings. McNeil suggested paying attention to what others say and being respectful of different communication styles. Endsley said men can use these techniques to help each other, and should invite female colleagues to participate. Rich observed that part of her job as a manager is to point out things that people are good at.

Chris St. Pierre (Amplify) recommended the wiki at geekfeminism.org as a resource. He noted that most people on the panel aren't managers, and asked them to comment on being females working in "hard" tech. Forinash said she doesn't want to be a manager and would rather be a nerd. McNeil agreed that she feels technical skills are her strength. Rich said that because she couldn't stand things being done badly or not at all, she became involved in management when others weren't handling it, and feels she is good at relating to people and motivating them, so management is what she needed to do. Straughan, on the other hand, has never thought of herself as a sysadmin, and has seen a "respect gap" when female coders were promoted to management and stopped coding.

An audience member who described herself as both a sysadmin and a manager noted that all the people she manages are men. She sees women as being focused on changing themselves when men don't give that a second thought, and that being a sysadmin is a tough job for women. She also sees women as having strengths for leadership in things such as big-picture thinking and communication skills. Endsley said she also sees men who are willing to change their behavior, and asked, "How are people raising their sons?" Straughan also said being a sysadmin wasn't just

a "guy thing," and both men and women may have trouble with interaction skills, but people can help each other to become more self-aware. Endsley and Rich both said they had been told they were aggressive or intimidating. When Rich was asked by men how they should behave around women, she answered, "How about like an adult?" and that even some men don't appreciate "locker-room humor." McNeil said being a sysadmin is still a tough business, but people stay in the profession because they enjoy the challenge.

An audience member described how her father raised her to be good at technical skills but other girls were growing up to be "pretty" instead of technical-minded. Some males don't want to recognize her technical skills, so she asked the panel how they operate in a male-dominated field. Endsley said she took her daughter to conferences and recommended fathers should also. Rich said boys should be raised to have respect for women, and she became good at technical skills because her father was bad at them. She also said that interviews with successful young women in technical jobs showed they learned not to care what other people thought.

An audience member asked what they should look for in company culture when considering job offers that seemed otherwise equal. There was general agreement on quality of benefits (such as maternity/paternity leave), opportunities for training and advancement, flexible hours, and the ability to work from home. Rich also said talking with other employees may reveal whether the company has a "glass ceiling." Straughan said to look for overall diversity and not just the male-to-female ratio. Rich also said that any kind of "brogrammer" culture is a red flag, and doesn't like the expectation you'll live at the company.

An audience member commented that they got HR to remove names and genders from applications during evaluation to reduce gender bias, but HR still wrote ads that got more men to apply. Rich asked whether it was pointed out to HR that they were doing that, and to give them information on how to write more gender-neutral job descriptions. Endsley said job ads like those didn't just discourage women. Both Forinash and Rich felt that to encourage more women in technical fields it was important to them personally to be visible examples.

Another audience member said they saw a cultural disrespect for management in technical fields, and asked how to get past that disrespect when women are promoted to management. Forinash admires her manager for retaining his technical skills after becoming a manager, but not all managers do. Rich said she tries to maintain respect by having good interpersonal relationships with the people she works with, showing them she's interested in helping make sure things are done right and that they're recognized for their work, and making sure they know she's engaged in their careers.

An audience member asked whether the panel thought the "BOFH" mentality would disappear in the next few years. Forinash succinctly observed, "Assholes won't disappear." McNeil thinks it's easier for women to work in IT today than it ever has been, but society still doesn't encourage it, and she sees the number of women entering the field decreasing. Rich confessed she secretly loves the BOFH stories, but they are from another age, and the best way to get things done is not to be controversial, saying, "I like movies where people get murdered, but I don't go out and murder people."

An audience member asked, "How do you be a role model instead of a figurehead?" Forinash said she felt this was up to managers, and that peer awards reflected respect from peers rather than giving people awards for being in a minority. The audience member continued her question by observing that in academia, she was asked to do outreach because she was a woman and not because she was doing serious technical work. Forinash said to be good at what you do regardless of expectations. Straughan recommended talking about technical work rather than about being a woman. McNeil summed it up as "You're a woman in engineering, rock it!"

Invited Talks 2

Summarized by Cory Lueninghoener (cluening@gmail.com)

Hyperscale Computing with ARM Servers

Jon Masters, Red Hat

Jon Masters, chief ARM architect at Red Hat, spoke on how advances in ARM processor technology are making it possible to fit 1,000 to 10,000 nodes in a single datacenter rack. He began with a description of what the ARM architecture is, noting that it is a design that can be licensed by anybody, is not tied to a specific vendor, and can create low-power systems. After showing a picture of the current ARM celebrity, the Raspberry Pi, he went on to describe server-class systems that also use the same processor design.

Jon told the audience that the ARM architecture makes it possible to fit thousands of server nodes into a single rack that uses only a couple of hundred watts, which increases server density and reduces datacenter PUE. He showed a picture of an existing rack drawer with many nodes within it and described how the architecture encourages a fail-in-place operational model, where failed nodes are replaced in bulk (if at all) instead of on a case-by-case basis. He also touched on the newly released 64-bit ARM architecture, noting that 64-bit systems are currently much more expensive than 32-bit systems.

For the next section of his talk, Jon described the software stack that Red Hat has been working on for the newest generation of ARM servers. He focused on the 64-bit ARM architecture, describing the work needed to get Fedora 19 ready in advance of the 64-bit release. He spent some time on how the new 64-bit architecture uses standards such as ACPI and UEFI to be more consistent across multiple vendors, and how Red Hat has used this standardization to speed up the Fedora port. After describing details on the porting process and the package build process,

Jon noted that there is now a Fedora 19 remix that boots natively on 64-bit ARM hardware and contains more than 12,000 packages. Before taking questions, Jon pointed attendees to #fedora-arm on Freenode IRC for more information.

One audience member asked whether there are low-cost 64-bit ARM systems that hackers can play with. Jon answered that $5,000, the current cost for a system, is indeed a lot to spend on the hardware and that he is working with companies to push that below $500 in the relatively near future and cheaper after that. Another audience member asked about commercial adoption of the new 64-bit systems and whether Red Hat will support them with their RHEL product. Jon carefully chose his words when answering, noting that if the market responds there will be RHEL support, but that he has nothing to announce right now. Finally, an audience member asked how 64-bit ARM will handle peripherals. Jon told us that the new architecture is moving toward ACPI and enumerable buses.

As a grand finale, Jon asked the audience "Who wants a Beaglebone ARM board?" When a large number of hands were raised, he threw a small Beaglebone box out into the audience and an enthusiastic attendee caught it.

LEAN Operations: Applying 100 Years of Manufacturing Knowledge to Modern IT

Ben Rockwood, Joyent

Ben Rockwood began his talk with a video showing President Obama addressing the country because of Web site problems with the new Affordable Care Act. He used this as an example of how the work that system administrators do has changed a lot in the past ten years: we are now at a point in which a Web site like Twitter or Facebook being down is treated like a national or world catastrophe. With this basis, he went on to speak about how to use already-existing manufacturing knowledge to improve modern IT practices.

Ben next talked about several models of system management. He described the DevOps flow, showing a diagram that demonstrated the flow from developers to operators to customers and back again. He encouraged a holistic view of systems design, noting that value in systems comes from the sum of the behaviors of the parts.

After spending some time describing how manufacturing has become increasingly automated and efficient in the past 100 years, Ben described 11 practices that easily can be translated from manufacturing to system administration. These included building trust relationships, sharing vision, standardizing work, eliminating waste, process improvement, and spending time where the work happens, among others. In each case he showed how manufacturing has used the practices to make their processes more efficient and increase productivity, and described how the same practices can be used by system engineers to improve their own workflows. Ben closed with words of wisdom: know who you are, know where you are, and know where you want to be.

An audience member noted to Ben that eliminating waste needs to tie into global optimizations, to which Ben replied that he was correct and that doing anything just for the sake of doing it is wrong. Another person wondered whether the way the talk tied manufacturing to software systems was a good fit, because software is a lot easier to change than a factory. Ben suggested that the questioner should spend more time in a factory, as they are a lot more malleable than he thinks. He then noted that the principles behind the manufacturing practices are actually the important parts.

Invited Talks 1

Summarized by Cory Lueninghoener (cluening@gmail.com)

Systems Performance

Brendan Gregg, Joyent

Brendan Gregg was originally scheduled to speak about Flame graphs, but that talk was moved up to a plenary session due to a cancellation. In the talk's place, Brendan spoke during this session about how systems performance monitoring has changed since the 1990s. He started by defining systems performance: the analysis of applications down to bare metal. He noted that this is an activity for everyone, from casual to full-time system analyzers.

With that definition introduced, Brendan presented a series of comparisons between how performance analysis has changed in the past 20 years. First he looked at systems performance as a whole: in the 1990s, text-based tools were used with inference and experimentation to gain insight into systems; in the 2010s, we have a lot more insight into the systems and can use standard methodologies to instrument them. In the 1990s, operating system internals were studied in textbooks; in the 2010s, operating system internals are studied via dynamic tracing. Systems in the 1990s had murky observability, with parts of the system hidden; systems in the 2010s are incredibly transparent. The 1990s had text-based or simple line and bar graph performance visualizations; the 2010s has heat maps and Flame graphs that display more information more easily. Finally, in the 1990s, we used ad hoc checklists and the tools we had at hand as performance methodologies, while in the 2010s we have workload characterization, drill-down, and other methodologies at our disposal.

Hacking Your Mind and Emotions

Branson Matheson, SGT

Branson Matheson gave a highly engaging talk about social engineering, covering both how to use the practice to your advantage and how to prevent yourself from being taken advantage of. This wide-ranging talk started out with some keys to being a good social engineer (confidence, ability to think on your feet, knowing your limits, theatrics) and some indications that you may not be so good at social engineering (cracking a smile when trying to pull a prank, freezing up when confronted, not being comfort-

able with play acting). He then talked about some of the basics of social engineering: making what you're doing look normal, keeping your target goal in mind, adapting quickly to change, and establishing a false trust relationship.

Focusing on the concept of trust, Branson described ways social engineers can build that false relationship. These involved such tactics as making yourself look like your target would expect, making yourself or your techniques look official, and exploiting emotions. He then described how some day-to-day social engineering takes place: advertising, businesses, car salesmen, and the police. From there, Branson went through a quick, but thorough, description of how to pull off a social engineering caper.

Branson finished the talk by showing how to avoid being the victim of social engineering. He described defensive actions to use on both the phone and in person, focusing on how to avoid disclosing too much information about yourself or your company. This included a number of concrete suggestions, including how to recognize leading questions and how to still get the perks of giving away information without actually giving away information.

Invited Talks 2

Summarized by Tim Nelson (tbnelson@gmail.com)

The Efficacy of Cybersecurity Regulation: Examining the Impact of Law on Security Practices

David Thaw, Visiting Assistant Professor of Law, University of Connecticut; Affiliated Fellow, Information Society Project, Yale Law School

Thaw is both an attorney and a computer scientist. His research investigates how cybersecurity legislation affects what security admins do in practice. Surveys can be annoying and often don't ask the right questions, so Thaw's group interviewed chief information security officers at several large companies. Privacy concerns prevented him from explicitly naming the companies involved.

His interviews found that most were unaffected by laws such as Sarbanes-Oxley, but that others, especially the new breach-notification laws, had a significant impact. The impact was not always positive: because the laws mandated reporting a breach only if the lost data was unencrypted, they resulted in a flurry of investment solely in encryption. In some cases, this disrupted other aspects of security practice by diverting time and money into encryption, regardless of whether they might have been better spent elsewhere.

Thaw's group went beyond interviews and quantitatively measured the effect of these new laws. To do this, they examined online breach reports from 2000 through 2010. Although using breach-reports as a metric is not perfect, at least it addresses whether an entity is focusing on security. He found that the number of breaches reported increased in August 2004 (when the laws took effect) but began to subside by September 2008. The decrease in reports was much sharper for previously unregulated entities, which suggests that the breach-reporting laws did have a positive effect on security practice.

Thaw presented his results to Congress and helped prevent the passage of a bad data-breach reporting law. The study was performed more than two years ago. Because the world changes rapidly, Thaw would like to do a follow-up study with more sysadmins. He opened the floor for discussion early. Someone asked how badly these laws were written and whether they might allow people to encrypt poorly and ineffectually. Thaw replied that the laws varied by state, but that in many cases the wording was quite weak. For instance, "renders unreadable" may be satisfied by ROT13. Further, nothing prevents a law from hard-coding specific encryption algorithms, rather than references to current best practice.

Eric Smith noted that Thaw's research largely looked at consumers of security products, and asked whether he had investigated producers as well. Thaw answered that he would like to in his next study. Rik Farrow asked about the distinction between prescriptive and standards-based legislation, noting that HIPAA was much more flexible than the breach-reporting laws. Thaw favors standards-based legislation because it forces companies to think about security and come up with a plan, rather than following specific narrow requirements. He went on to say that security is only one-third technical, because the administrative and physical aspects are also important.

Someone else asked whether in a small organization a flexible law would boil down to a directive. Thaw answered that HIPAA, and similar legislation, takes size and capability into account. When asked about the process of reporting breaches, Thaw mentioned that very few breach laws actually have a centralized repository for reporting, and that the Open Security Foundation's data has become difficult to obtain, making his research more difficult.

The Intersection of Cyber Security, Critical Infrastructure, and Control Systems

Sandra Bittner, CISSP, Arizona Public Service, Palo Verde Nuclear Generating Station

Bittner began her talk stressing that safety is the first concern at a nuclear power facility. The US nuclear industry is required to apply lessons learned in one facility (called operating experience) to all US commercial facilities to prevent similar situations from occurring in other US facilities. The Nuclear Regulatory Commission (NRC) requires, in the area of cybersecurity, that all US commercial facilities protect their facilities against nuclear sabotage and the release of radiological material. The facility is free to choose the cybersecurity techniques, methods, and implementations they use but are subject to rigorous inspections and approval by the NRC. Her facilities are some of the largest generators of electricity in the US, and are meant to have a lifetime of 60 years.

She spoke about the facilities, but had to remove some pictures and material from her presentation upon being told that her talk would be recorded. She summarized some challenges inherent in securing control systems such as power plants, wastewater treatment plants, etc. These systems often contain both analog and digital components, a plethora of sensors, and electronics that run nonstandard protocols. Because certification is required, and recertification may take as much as a year, sometimes critical infrastructure operators must decide to delay software patches, or even not to patch at all. They are therefore heavily dependent on physical and network isolation. Stuxnet presented something of a wakeup call to the industry, because the cyberattack used on an Iranian nuclear enrichment facility made use of portable media to carry the infection past the "air gap."

In IT an outage is often the worst thing that can occur, but in their situation, an outage is far less worrisome than losing control of the facility. Critical infrastructure is also revisiting the topic of responsible vulnerability disclosure rules to exercise caution in areas that could incite public panic. Another topic of interest was forensics challenges for critical infrastructure because facilities are not easily shut down without public risk of loss of life. Bittner also noted that there are many cybersecurity "dialects" spoken, especially at a large facility. There can often be a terminology gap between the IT staff and the Operations staff (OT), and it is vital to send IT staff to plant training so that they understand the entire system.

Rik Farrow asked about frequency of software updates, mentioning that later versions of Windows have improved significantly. Bittner answered that they have a time-to-live on all of their equipment, and they can often only replace and install equipment during an outage, which may occur years apart. Thus, the last good version of software installed may be past its formal end of life by the time it is replaced. Farrow also asked about certifications necessary to work in a nuclear facility, and whether there was concern about validity. Bittner replied that there are required certifications and that in the area of cybersecurity she expressed concerns that established public certification programs should work hard to discourage the dilution of their credentials, and pull credentials when necessary.

Mark Bartlet noted that some recent compromises of control systems were done remotely, and asked what rationale could possibly lead to these systems being connected to the public Internet. Bittner replied that in those cases typically there was a lack of understanding about cybersecurity for the control system and a drive for business efficiency that conflicted with maintaining isolation barriers. Overall business will always want quick access to production control system information, but this should be weighed against the cybersecurity risks.

Invited Talks 1

Summarized by Tim Nelson (tbnelson@gmail.com)

A Guide to SDN: Building DevOps for Networks

Rob Sherwood, Big Switch

Sherwood began his talk by observing that network configuration has changed very little since the mid-'90s. In that time, we've made incredible strides in managing our server systems, but the networks themselves lag behind. Companies build vertically integrated networking appliances, and we lack the equivalent of Chef or Puppet for our routers. Software-Defined Networking (SDN) is a step in this direction, and is largely enabled by the OpenFlow protocol—SDN is to OpenFlow as the Web is to HTTP. Using OpenFlow, programs running on a centralized controller machine instruct the switches, making true network-programming possible. The need for automation and custom development lead to the DevOps trend, and Sherwood compares SDN to DevOps for networking.

Sherwood was careful to explain that SDN controllers are only logically centralized; they may actually reside on a cluster for fault-tolerance and load-balancing. Thus the controller does not actually represent a single point of failure.

As a motivating example, Sherwood brought up backup management. Large-scale network backups require reservation of bandwidth, ports, and devices, as well as a keen understanding of future bandwidth requirements. Configuring this consistently can be difficult. In contrast, an SDN-enabled application can (in a sense) talk to the network and schedule an appropriate time for the backup operation. This is analogous to scheduling in an operating system.

Switch hardware is already fairly flexible, and so we need only to leverage it intelligently to implement SDNs. Current work involves bringing OpenFlow up to the pre-SDN level of features and performance. OpenFlow-enabled switches are inexpensive, and, more importantly, one can Google to find prices, because the switches need not be proprietary hardware. There are also numerous software switch programs available, such as Open vSwitch. Openflow has also enabled or inspired many new tools, including the network simulator Mininet; benchmarking tools for SDNs like cbench and oflops; and Onie, which allows OpenFlow switches to net-boot and even dual-boot.

Someone asked why there wasn't an "OpenFlow for generalists" akin to Chef or Puppet. Sherwood replied that OpenFlow isn't the right level of abstraction for that, but that languages are already being built on top of OpenFlow. Another person asked how well OpenFlow works with legacy hardware. Sherwood answered that while switches need to be OpenFlow-enabled to use OpenFlow, such switches usually have a hybrid mode that enables bridging between OpenFlow and traditional networks. When asked what happens if the switches lose their connection to the controller, Sherwood replied that since the controller pushes down rules that persist on the switches, the network

won't simply stop. Instead, how well the network works will depend on how good the rules themselves are.

OSv: A New Open Source Operating System Designed for the Cloud

Nadav Har'El, Cloudius Systems Ltd.

Nadav Har'El spoke on OSv, a new virtual operating system. OSv is intended for guest processes in the cloud, and Cloudius hopes that it will become the default operating system for virtual machines. Har'El began by showing the virtualization stack with hardware at the base up through the host operating system, hypervisor, guest OS, JVM, and finally the running application at the top. He pointed out that many of these components duplicate effort. For instance, the OS, hypervisor, and the JVM all provide protection and abstraction in the interest of virtualization. OSv removes this duplication from the OS.

Mainstream operating systems were written with many assumptions that no longer fit the most common uses of virtual machines. Today, a VM often runs only one process (like memcached), has no users, and requires few services. Even the file system is not always needed. OSv was written from scratch to take advantage of this.

In OSv, while multithreading is allowed, there is only ever one process running at a time. It therefore has no separate kernel address space, and system calls are just function calls. It also provides a ZFS file system, an efficient (netchannels) TCP/IP implementation, and provides enough Linux emulation to run binaries. It can be run in KVM/XEN now, and VMware compatibility is expected soon.

Narayan Desai asked how one would debug applications in OSv if there is only ever one active process. Har'El replied that, for the moment, they attach debuggers to the guest process itself on the development machine. He also pointed out that OSv is still young, and they are still adding features to support debugging.

It was also mentioned that OSv's one-process architecture looks remarkably similar to Novell NetWare from 15 years ago. Har'El replied that OSv is about more than just the single process. For instance OSv is a virtual OS, and NetWare was not.

Rob Sherwood asked what a smarter hypervisor might do if it could assume that the guest processes were all running OSv, and whether we would just end up with containers at that point. Har'El answered that he does not think that the hypervisor will be going away anytime soon, but if OSv becomes predominant, he envisions hypervisors taking advantage of the guarantees OSv provides.

Invited Talks 2

Summarized by Jonathon Anderson (anderbubble@gmail.com)

Enterprise Architecture Beyond the Perimeter

Harald Wagener, Systems Engineer, Site Reliability Engineering; Jan Monsch, Security Engineer, Google Security Team, Google

Jan Monsch and Harald Wagener presented an Internet-based network security model that serves as an alternative to a more traditional perimeter-based model.

They described a baseline enterprise network security model as a system of "moats and castles." Security in this system is traditionally based on a trusted, privileged intranet zone that is logically separated and protected from other intranets and the public Internet. In this model, access to privileged services is granted based on access to the trusted network that is protected by "high walls" and "strong gates." Remote access is granted to trusted users by granting access to the protected network, often by way of a virtual private network connection. Attackers have evolved to address this specific security model, requiring them to exploit only the weakest point in the security perimeter in order to gain access to the network and its private services.

Google has addressed the problem of internal network security by looking "Beyond Corp." In this network security model, no privilege is associated with having access to the internal network. The perimeter itself is replaced with client certificate authentication that can be coupled with existing user authentication systems. This has improved their user experience by allowing their users to work from anywhere on the public Internet, with the only limits to internal service based on configurable policy. This has also allowed more user workflows to be moved to cloud services, as physical and network location is no longer a factor in authentication and authorization.

The Overcast architecture blueprint combines an authentication certificate database with a meta-inventory of system state. This allows service authorization to be based on the metadata associated with the client system. Does the system use full-disk encryption? Is it a desktop or a laptop? What software is installed?

The workflow patterns supported by this new security model have presented a few challenges. Access to network-attached storage is less viable on the now pervasive wide area network, and users should be moved to asynchronous local storage wherever possible. Certain legacy workloads have proven difficult to port, necessitating some traditional VPN access; but these networks can be reduced in scope around only these legacy services.

Further implementation details and examples, as well as the discussion of code availability, was deferred to more specific talks the following day. For more information, see "Managing Macs at Google Scale" and "OS X Hardening: Securing a Large Global Mac Fleet," below.

Drifting into Fragility

Matt Provost, Weta Digital

Summarized by Jonathon Anderson (anderbubble@gmail.com)

Matt Provost used themes drawn from the books *Drift into Failure* by Sidney Dekker and *Antifragile* by Nassim Taleb to reframe and redirect efforts to improve stability in complex systems.

As an example, he opened with a story of a seemingly simple deployment of an NFS cache. In keeping with best practices, the service was tested and deployed in stages to progressively larger sets of the production environment. Yet, within two hours of deploying the seemingly good service to the entire facility, all production operations had ceased: the deployed cache had been compiled as a 64-bit system, but the application using that cache was compiled as a 32-bit binary. Despite testing, the problem only became apparent when the cache delivered an inode with a number larger than 32 bits.

Systems are often fragile, denoted by disruption in the face of variability. One response to such fragility is to make the system "robust," often by introducing redundancy; but redundancy often impairs the ability to detect systemic problems during testing by hiding problems within individual components.

Public root-cause postmortem reports provide further insight into the difficulty of troubleshooting complex systems. Examples from Amazon, GitHub, and CloudFlare all showed ample preparation and past experience that seemed to guarantee success yet failed to anticipate the interaction of components in the system.

Matt advocated the creation of "antifragile" systems that improve in response to variability rather than deteriorate. One approach to creating antifragility is to generate artificial variability in the system, making this variability routine rather than exception. As one example, he referenced the Netflix "Simian Army" (see "How Netflix Embraces Failure to Improve Resilience and Maximize Availability," above) and provided his own simplified "Cron Monkey" implementation: by rebooting development servers automatically and regularly, he prevented these servers from drifting into production use. Also, single-points of human failure can be detected by simply going away from the system for some period of time.

He also advocated the simplification of systems wherever possible, even going so far as to disable file-system snapshots to ensure that backup services are reliable. Legacy systems should be actively decommissioned, though, in Q&A, he admitted that variety in user culture can make it very difficult to remove a service once it has been deployed.

Papers and Reports

Summarized by Carolyn Rowland (unpixie@gmail.com)

Live Upgrading Thousands of Servers from an Ancient Red Hat Distribution to 10 Year Newer Debian Based One

Marc Merlin, Google, Inc.

Many of us in the community keep an eye on Google because they're doing what we're doing but at scale. Often the lessons from Google show us what is possible when you must automate because it is too expensive not to do so. This presentation is no exception. Marc presented Google's solution to updating and maintaining thousands of Linux servers with minimal effort.

He started by talking about their in-house Linux distribution: ProdNG, self-hosting and entirely rebuilt from source. All packages are stripped of unnecessary dependencies and libraries making the image quicker to sync, re-install, and fsck. Google chose Debian because it had a better base of packages than Red Hat. With previous distributions, they spent too much time packing software that wasn't included in the distribution. They considered Ubuntu, used on their workstations, but it wasn't ideal for the servers.

The first step was to check the golden image into revision control. When they were ready, every server started booting the same image at the same time. They also found that they could do a side-by-side diff of images as needed for testing. Ultimately, there was no way to guarantee that they could seamlessly switch distributions from Red Hat to Debian. It was hard to find beta testers or to convince their internal users to run production services on a very different Linux distribution.

They had to find all packages that inherited from RH 7.1 that were never needed (X server, fonts, font server for headless machines without X local or remote, etc.). It's amazing how many packages come with a distro that you don't really need on servers. They converted the remaining Debian packages (debs) to rpms, keeping both distributions libc/binary compatible.

Sometimes it's the little things that bite you.

The coreutils upgrade was scary due to amount of breakage created upstream. The day they removed /etc/redhat-release is the day they broke a bunch of Java that parsed this file to do custom things with fonts. One rule on the team was whoever touched something last was also responsible for fixing the breakage, revert the change, fix the bug, try again later.

Then they started the painstaking upgrade of around 150 packages carefully stripped and built from scratch in a hermetic chroot environment. The process was to upgrade a few at a time by building them in ProdNG and putting them in both ProdNG (not used yet) and the current image.

After upgrading the packages, they would throw a regression tester at the new image, run all of the daemons, make sure they come up, test crash reboots, and test reverting to the old image for completeness.

There were some lessons learned along the way:

- Maintaining your own sub Linux distro in-house gives you more control (if you have in-house expertise).
- At large scale, forcing server users to use an API and not to write on the root FS definitely helps with maintenance.
- File-level syncing recovers from any state and is more reliable than most other methods.
- You'll be able to do crazy things like switch distributions.
- Don't blindingly trust and install upstream updates, because they could conflict with your config or even be trojaned.
- If you can remove services/libs you don't need, that leaves you with fewer things to update and fewer security bugs to worry about in the future.
- Running the last Fedora core or Ubuntu from five months ago is more trouble than it's worth. Full updates of servers every three years is reasonable, so Debian stable or RHEL are the way to go for servers.
- Depending on your server setup, partial upgrades may be better for you. Choose a distribution that allows this, such as Debian.

David Ehle, Argonne Laboratory, asked why they built everything from source. Marc responded that when you take binaries, you only have signatures, and you're trusting Red Hat. You're not sure there's not a trojan in the binary. If you build from source, you can have your own optimizations, and you can update with some stack protections. You can use the compiler options you like. You have more luxury (control). We do modify some daemons (e.g., better command logging).

Managing Smartphone Testbeds with SmartLab

Georgios Larkou, Constantinos Costa, Panayiotis G. Andreou, Andreas Konstantinidis, and Demetrios Zeinalipour-Yazti, University of Cyprus

The number of mobile devices in use today is now exceeding the number of laptops. There is the tablet, Raspberry Pi, smart glasses, smartbook phones, and 53% of smartphones in 2016 will be running Android. People are also not changing their phone OS when a newer version is released because manufacturers are not supporting the older phones. This means a lot of old phones running old operating systems are still in use. How can a smartphone developer cope with this software/hardware fragmentation?

Georgios Larkou presented SmartLab (http://smartlab.cs.ucy.ac.cy/), a comprehensive architecture for managing mobile and virtual smartphones through a Web browser. You can manage devices, issue shell commands, transfer files, see the debug information, and do sensor mock-ups as well. You can even go to SmartLab and rent a smartphone. Imagine you are a developer and you want to test your app on 50 different smartphones. You can do that with SmartLab. It also gives you a platform to manage all of our gadgets in the future. You might need to handle a fleet of devices installed on 1000 buses. SmartLab will support that next year. The SmartLab developers are talking with a local ISP and will have smartphones running on buses in the future. They will monitor all of them to get data for other experiments. Additionally, people tend to change smartphones more often than they change computers. It would be advantageous if SmartLab could use all of these smartphones to build Beowulf-like clusters.

This architecture has been evolving since 2009. SmartLab provides fine-grained control over ARDs (Android Real Devices) and AVDs (Android Virtual Devices). There is remote file management, and a home directory and /share directory, and access to all SD cards from the devices. If a user drag-and-drops a file from one device to another, it will copy through the SmartLab architecture. If the person drops it in the /share directory, then it will be distributed to all Android devices. The same thing happens with the remote shell; the user can send commands to multiple devices.

SmartLab builds upon standard open tools for its architecture. Benefits of the SmartLab open architecture include experimental repeatability, urban-scale deployment, federation issues (similar to PlanetLab) so others can contribute, and personal gadget management.

David Lynn (Vanderbilt University) asked whether the cell radios in the actual devices are active. Georgios said no, because they don't have SIM cards on the devices. They are talking with an ISP to provide mobile data. Marc Merlin wondered how the virtual devices compare to simulators that have good native speed. Georgios answered that the real devices are not good for I/O-intensive experiments, as it takes too much time to push and install an app on multiple devices; on the other hand, it takes much longer to install the app on the simulators. Marc then pointed out that if you're running out of memory because you have a virtual disk, then you are fighting for a disk head. This is still useful for some applications but for others you might be better off running simulators. Georgios responded that they want everyone to be able to have access to real devices.

YinzCam: Experiences with In-Venue Mobile Video and Replays

Nathan D. Mickulicz, Priya Narasimhan, and Rajeev Gandhi, YinzCam, Inc., and Carnegie Mellon University

The YinzCam team started with wireless capabilities in the football stadium in Pittsburg. It was a distraction-filled work environment, and inefficient because they had to have someone on-site at the venue every game. How would this have scaled if they continued to do on-site operations for multiple teams in multiple cities?

They took on four new installations in 2011 and had to fly across the country to San Francisco for every one of their home games. The costs would have continued to increase, especially the travel costs for the on-site person.

The team thought they could centralize their operations in a single remote location. This would eliminate the distractions from the on-site environment and allow efficient communication between staff. Plus, they could cover for each other. It also

meant a reduction in travel costs because everyone would be together at the remote site.

They would need to provide reliable access to remote resources. Using a VPN would provide secure connectivity between all sites, allowing the team to monitor content remotely. They are only allowed to publish content inside of stadiums because of licensing restrictions, but they can play that content over the VPN without breaking those restrictions.

They got to the point of having 18 in-venue installations, 83 source video streams, and 98 physical and virtual machines.

They leveraged Amazon Web Services (AWS), connecting to the nearest AWS region (east or west), then established a connection back to the other AWS region. Now they could have a remote ops workstation through a remote management gateway into the servers inside of the stadium. They reduced costs in 2012 and 2013 because they only had to go to the actual site for a one-time installation', and that was it for the rest of the year.

Another challenge was how to quickly and accurately generate replays of game footage. How it would work: a fan goes into the application and pulls up the highlights list. Then s/he selects the replay after choosing the camera angle. Originally this was a manual cutting workflow that worked like this: an interesting event occurs during a game, a remote YinzCam operator clicks a button at the end of the event, a tool searches back in time for the start of the event, the system produces a video file, the operator annotates this file with metadata, and, finally, the system publishes the video in the mobile app.

This method was time-consuming, error prone, and repetitive, and the manual metadata generation took time. It required full operator attention, and there was no pipelining (e.g., there was only one replay at a time).

The team resolved this by using resources for what they do best: machines are well-suited to repetitive tasks such as approximating replay boundaries and encoding video. Humans perform well on visual tasks such as visual verification of the instant replay window. They also needed to allow remote override of automated decisions because these decisions might be incorrect. Humans would verify replays.

The automated replay cutting architecture included two data feeds: one was video, the second included the statistics about the game. Luckily timing rules allowed the team to estimate the start and end times for every replay in the game. Video streams could also be easily duplicated in cache. The team set up a proxy server that split the content to all existing clients. Segments of video were served over HTTP while standard proxies were used to cache video data. Public clouds have ample bandwidth; you pay per GB, but the bandwidth is practically infinite. The requirement was for five hours of uptime on game day only with rapid scale-up and scale-down on demand.

Once the stream was in the cloud, it was migrated to a live streaming server. The team produced a copy of content to the cloud so it could be served from the cloud using the bandwidth available there. A copy of the replays and livestreams were sent to the in-venue WiFi network which operated just as it did before.

Some of the lessons the YinzCam team learned from this effort include:

- Manage operations remotely
  - extract ops from field environment
  - ensure easy, reliable access to remote systems
- Invest in system automation
  - automate error-prone manual processes
  - allow overriding of automated processes
- Take advantage of the cloud
  - short-term CPU-intensive processes
  - bandwidth-intensive processes and services

David Lynn (Vanderbilt University) pointed out that the Tennessee Titans are about three miles from Vanderbilt, and cellular coverage at the LP field sucks. Sprint is being used for wireless, apparently. Nathan replied that in Sprint's case, DAS (distributed antenna system) is often done on carrier-specific lines. When a DAS is installed, it is often installed by a single carrier, and not all of the major carriers will be on it at once. Sprint may not even be on the DAS but some of the other carriers may be. DAS handles a lot more traffic than a single cell tower would outside of the station.

Andrew Myers (Lean Logistics) asked whether they used blackmagic or matrix cards, and how much COTS was used. Nathan replied that they used blackmagic cards and software encoders like ffmpeg. Andrew also asked if they used commercial streaming software or their own, and Nathan responded that they wrote a little bit of software that takes a TCP stream and copies it to a different endpoint. Someone asked how support is handled for the deployed equipment. Nathan said that clients purchase and maintain the hardware.

Friday, November 8, 2013

Invited Talks 1

Rethinking Dogma: Musings on the Future of Security

Dan Kaminsky, Chief Scientist, White Ops
Summarized by Jonathon Anderson (anderbubble@gmail.com)

Dan Kaminsky, perhaps best known for his work pointing out a fundamental design flaw in DNS, offered his thoughts on the security landscape predicted for 2014.

He started by framing the current state of security as that of a hacked-in afterthought: design the system, then make it secure against exploits. In today's globally Internet-worked world, security has become an engineering requirement of its own, but we don't know how to deliver security at scale; we don't know how to authenticate users; we don't know how to write secure code;

and we don't know how to punish the "bad guys." Instead, when system security fails, blame falls on the victim.

In the absence of any existing solution, Dan says that we have no room for religious arguments about how to do security. We need new science with new data and new instrumentation.

Dan likened existing security models to static puzzles. Any puzzle of arbitrary complexity can be solved given sufficient incentive and time; and with no consequence for failed attempts, any security system can eventually be broken as well. Instead, security should be approached as a game: an interactive challenge where the security professional has the advantage of superior information. In essence, the defending player "doesn't have to play fair." Security professionals have a wealth of data signals to mine, including system logs, network activity, and honeypots; and the time it takes an attacker to probe an unknown system for vulnerabilities provides ample time to respond with a consequence. The remaining challenge is in managing the signal-to-noise ratio, reducing false positives as much as possible.

Dan touched on a series of recent trends and events in the security community, including the increasing difficulty of securing virtualized systems; reports of "badBIOS" exploits; the Microsoft and Facebook Internet Bug Bounty (IBB) program; improvements in sandbox technology (particularly asm.js); and controversies arisen from recent revelations regarding the United States National Security Agency (NSA).

The larger issue in securing systems, however, is usability, both for the end-user and the administrator. Key management is always more difficult than the cryptography it uses, and security is more easy to circumvent when it is more difficult to implement correctly.

Dan fielded several questions from the audience. He defended his preference for Windows 8, calling all modern operating systems "essentially single-user machines." He responded to recent research into application sandboxing, calling efforts to extend the Capsicum project into hardware "one of the most interesting things [he's] heard." When asked about Bitcoin, Dan marveled both at the currency's existence and at the fact that it has remained unbroken despite the intrinsic financial incentive to defeat it. When asked about the upcoming termination of support for Windows XP he responded that he's more concerned that Web browsers for the OS might stop receiving security updates, calling the browsers the "gateway to the world." He closed with anecdotes from Google's experience running honeypots, providing recommendations for deploying them as part of a site's overall security strategy.

Invited Talks 2

Summarized by Scott Murphy (scott5@ovsage.org)

ZFS for Everyone

George Wilson, Delphix

George Wilson of Delphix started with a poll of the room checking to see how many are using ZFS. It seemed that pretty much all common platforms are in use and represented by the audience. He then gave a short bio of his involvement from the first introduction of ZFS to Solaris 10 up to his current OpenZFS activities and then gave a quick overview of the ZFS timeline.

He went on in a little more detail regarding OpenZFS. It was launched in 2013 as the truly open source successor for ZFS. When ZFS was being developed, most of the work was done by Sun/Oracle. Now the contributions are coming from across the spectrum. Basically, OpenZFS is an umbrella project founded by Open Source ZFS developers representing multiple operating systems. The stated goals are to raise awareness of the quality, utility, and availability of OpenZFS; to encourage open communication about efforts to improve OpenZFS; and to ensure consistent reliability, functionality, and performance of OpenZFS. This is all coordinated through the project Web site: http://open-zfs.org.

He then displayed a slide showing the involvement of companies with OpenZFS, stating that OpenZFS is becoming a big driving factor for adoption by companies for their products.

Moving on to the management of OpenZFS, he explained how they deal with features across platforms by using "feature flags." He then gave an overview of the feature flags available in OpenZFS. These include the Pool feature, LZ4 compression, compressed L2ARC, enhanced write throttle, per I/O device queues, and imbalanced LUNs enhancements.

Someone asked why the per I/O queue was processed in the order described. George answered that the queues are processed in priority order and checked to see if any of the queues have not gotten their minimum number into the active I/O pipeline. If a minimum has been met for any higher priority queue and there is a queue without anything in the I/O pipeline, then the next highest queue will be processed and the next until they are empty or have reached their minimum. This effectively guarantees that all queues get processed and blocking is reduced.

George went on to say that upcoming features would include enhancements for pool fragmentation and zero block compression. There are many more features in the pipeline.

George concluded with an appeal to get involved with the project. If you are making a product, let them know, if you are a sysadmin, spread the word and contribute to the wiki.

Someone asked, "Is there a feature that would allow you to remove a fragmented imbalanced LUN and redistribute the data across any new LUNs?" George mentioned that there is a block pointer rewrite project driven out of Oracle, but they don't

want to implement it in ZFS; it was a grandiose idea and would ultimately have killed development by making it difficult to add features later. He added that there is an idea to introduce a device removal logic that would allow you to effectively remove the LUN and to recreate a new virtual device that looks like the LUN but then gets data redistributed to it. There is a writeup on device removal that is being tweaked to include both redistribution and device removal which would cover this case.

Someone else asked about the new combined development process and how it feeds back into the various things we actually use. George said that the idea behind the dev process is to take all of the platform-specific layers and tease them apart to create a common ZFS repository in order to allow any platform to take that wholesale. They have also created a testing facility, which is available today in the open source community, that will make it easy to test regardless of the platform. You test on any platform with the same test structure, ensuring it will be available on all supported platforms.

Someone else asked, "Do you have any advice for those still on Solaris who may want to move to this platform?" George answered that for those who are doing migrations, it will be difficult. The easiest way might be to use a ZFS send, but Oracle may have added a feature which will make this incompatible if you are in a newer revision. If you are running Solaris 10 it should be easy. Someone asked a related question, whether there is a version number to avoid exceeding. George responded that if you are running anything less than version 28, you can just import the pool. He also said that they will be putting some migration notes on the Web site and you should check there for more details.

The final questioner asked whether Oracle is interested in the OpenZFS efforts or were they doing their own thing. George said that he was not sure about interest, but they are definitely doing their own thing. The OpenZFS folks have talked to developers about adding feature flags so there would be compatibility, but Oracle has its own agenda and business needs. The community is interested in what Oracle is doing, such as larger block support. Good ideas can come from everywhere.

Manta Storage System Internals

Mark Cavage, Joyent

Mark Cavage, software engineer at Joyent, gave a talk on the Manta Storage System, starting with a short history lesson.

Back in the '80s as part of a magazine article series, Jon Bentley asked Don Knuth to demonstrate " literate programming" using a classic problem, word frequency analysis. He also asked Doug McIlroy (inventor of UNIX pipes) to review the solution. Knuth's solution was done in ten pages using a custom algorithm in WEB, his own literate programming language, with a from-scratch implementation of everything. McIlroy did a one-liner with standard UNIX commands and went on to make several points to illustrate why his solution was acceptable, including the following two: not everyone has a Don Knuth on staff and not all problems are the same, but they may be similar.

This illustrates the UNIX philosophy, which is specifically creating small programs that do one thing and do it well. This also influences the approach to building programs.

Moving the time machine forward to today, the aggregate term "Big Data" provides the same class of problem, illustrated by Google's MapReduce paper. Mark wanted to apply the UNIX philosophy to it. To that end, you need a few items: parallel execution capability, the compute interface to be a raw OS with the standard OS tools, cloud vendors to provide multi-tenancy, and the ability to store an arbitrary amount of data.

Beginning with storage, you have three choices: block, file, and object. Object storage is similar to file, no partial updates, no exposed volumes, and a universal protocol. The challenge is how to make UNIX work with an object store efficiently. Next, you virtualize the OS—in this case with zones. This provides one kernel on bare metal and many virtual containers with their own root file systems. This is much more efficient than hardware-based virtualization. There is also a rich interface between the global zone and the individual tenant zones.

Putting it all together, Manta is a scalable, durable HTTP object store with a namespace that looks like a POSIX file system with in situ compute as a first-class operation.

Mark then did a successful demo, the Manta version of "Hello World," demonstrating that the file system was available on his laptop talking to the Manta cloud. The demo showed a prototype UNIX environment running on an object store.

He went on to describe the Manta design parameters. They chose to be strongly consistent vs. eventually consistent. This is based on the CAP theorem (Brewer, 2000) which can be paraphrased as "on any storage system in the face of network failures, you can choose for the system to be consistent or available, but not both." They chose strongly consistent because in an eventually consistent system, nothing up-stack can ever be consistent again. If you build strongly consistent, you can always put an eventually consistent layer up top. An object store needs to be strongly consistent for maximum functionality. In HTTP terms, system up returns 200s, system down returns 500s. Given these parameters, their system is built to be highly available. They can tolerate everything up to a single datacenter failure—every object in Manta must be a contiguous file.

He then showed a diagram that resembled the classic three-tier architecture, in this case front-end, metadata, and raw storage. The front-end consists of Stud (SSL terminator), HAProxy (HTTP terminator/Load Balancer), a Web API (restify on Node.js), and Redis (authentication cache).

The metadata tier consists of Moray (custom node.js key/value interface), metadata copies on 2+ Postgres databases, replica-

tion topology managed via ZooKeeper, and a consistent hash on directory names.

Raw storage is a custom bare metal system called the Manta Shrimp, which consists of 73 TiB in a 4U 256 GB DRAM RAIDZ2 box, SmartOS (ZFS, Zones), storage interface (nginx), and the ability to support compute jobs.

The final part of storage is how they implement group membership, which is through a custom DNS server in Node.js. Participants write an "ephemeral node" in ZooKeeper on startup. This "mostly" works, as the name service caching daemon (NSCD) and ZooKeeper have issues.

Mark continued with a discussion of the compute flow. Users submit jobs which specify pipelines to run either on each input separately (Map) or all inputs together (Reduce). Inputs are objects accessed as regular files, outputs are saved as objects. User programs run inside transient zones managed by the service. Resource usage is capped but makes allowance for bursty activity. Input is taken via objects mapped in as read-only files (for Map) and redirected as stdin. Upon completion, ZFS rollback is performed and the zone is rebooted.

Mark revisited Bentley's challenge from earlier showing the Manta variant of McIlroy's solution to the challenge with parallelism and did a demo using a word find on all the works of Shakespeare. It was fast.

Customer usage tracking (aka metering and reports) uses Manta. Every *N* minutes, they upload database snapshots. Once the snapshots are uploaded, a job is kicked off to aggregate usage by customer. There is nothing special here, just MapReduce.

Garbage collection is an interesting problem—links in Manta are copy-on-write UNIX hard links. On every delete, the record's state is recorded and then Manta crunches the delete logs and all live records by hourly sending all records about an object ID to the same reducer. The reducer can then determine whether the object is truly garbage; all garbage objects are emitted to Manta as a list per backend server. The backend server then polls Manta for new objects to delete.

Mark concluded with a series of implementation lessons. The modules node.js, bunyan, restify, fast native-dns—and, particularly, MDB, DTrace and Flame graphs—were handy for debugging; Stud is very buggy. If you are going to use nginx, make sure you have the version with the fsync patches. nginx resource requirements will need tweaking. They use ZooKeeper for now, will hopefully find a replacement soon. Postgres is running with lots of churn with a 24/7 duty cycle, so there will be a lot of tuning, vacuuming, analyzing, and table fragmentation.

There were no questions.

Invited Talks 2

Summarized by Georgios Larkou (glarkou@cs.ucy.ac.cy)

Apache Hadoop for System Administrators

Allen Wittenauer, LinkedIn, Inc.

Allen started the presentation by asking about Hadoop adoption among the attendees, continued with a brief MapReduce and Hadoop introduction, and provided some real examples of Map and Reduce functions in Perl and Python. Next, Allen provided some background information about Hadoop genesis and how Hadoop evolved over time, from the Google's MapReduce Framework and file system to the open source version we all use today.

Hadoop binds the MapReduce programming layer and the HDFS file system together in order to provide support for file-system-agnostic behavior since MapReduce does not care if you are running HDFS on Amazon's S3, for example, or GlusterFS, or any other file system (e.g., ext4 or ext3). HDFS uses the NameNode to provide a dynamic metadata store for each file, similar to an iNode server. Additionally, Hadoop implements a scheduling framework and uses a JobTracker to schedule tasks on a distributed cluster. The bad news is that Hadoop is built by developers for developers, thus it is not designed for system administrators and/or support staff.

Allen pointed out some of the Hadoop problems (e.g., single point of failure) and provided some suggestions for (junior) administrators, including not to always trust the logs since most of the time, tailing the logs will not tell you the whole story. Allen asked the audience to remember just one character, a "%" (percentage), since that character is the key to making Hadoop work. In order to better explain why, he provided an example from nature, an ant colony. He said that if the colony loses an ant, that is not a big deal, the colony will survive without any problems; but if the colony loses the ant queen, it will probably face a variety of problems and might not be able to recover. Consequently, he asked the audience to think of the master node in the Hadoop framework as if it were an ant queen.

The solution that Allen suggested to the audience was to use a monitoring system to monitor the master nodes, and he presented a custom build of Nagios that they use at LinkedIn. He also suggested performing health checks on the NameNode and the JobTrackerNode every "x" amount of time. Additionally, he suggested extensively testing everything before running the framework, and noted that even if you introduce simple tests such as moving files to HDFS, performing simple MapReduce tasks can save a large amount of time since you will be able to detect most of the errors before you turn on the framework for production. Finally, he suggested monitoring resource usage and mentioned that developer teams or individual developers will not understand any Java heap problems, but they should always write their code as if they had limited resources. For this, Allen suggested the introduction of a quota system.

In terms of Hadoop security, Allen suggested always being aware of any open ports and noted that since it is difficult to revoke a user-level certificate, administrators should use Kerberos. Kerberos supports the use of Active Directory certificates and provides mechanisms for easily revoking user certificates. As a result, the administrators will have more time for building "cool infrastructures" than doing boring user management.

Lastly, Allen presented the "White Elephant" open source project contributed by the LinkedIn team. White Elephant provides tools for monitoring an individual user's usage by analyzing the JobTracker logs for every job submitted to the system. The tool allows the Hadoop administrators at LinkedIn to ensure that every developer obeys the fair usage policy.

Someone asked whether putting a quota on everything might introduce big hits on metadata operations. Allen responded that they did not observe any kind of metadata problems since Hadoop actually kind of cheats; Hadoop actually knows the block size when you ask for file storage and thus can calculate the quota you may potentially use, based upon the default block size and the number of blocks the user asks for.

Someone else asked for advice regarding the naming scheme of the file system or machines, especially if you are going to have more than one cluster. Allen responded that they use generic names for everything. He also provided an example of how LinkedIn had chosen the naming scheme for their machines and some best practices based on their experiences.

Jennifer from Yahoo!, noted that Allen is one of the active Hadoop committers but not one of the project management committee (PMC) members. Jennifer expressed her concern that Hadoop has a lot of operability issues and asked whether there are any people from the PMC who are operational-side focused and, if not, how can Allen get on the PMC and how can she vote for him. Allen responded that he does not believe that anyone from the PMC is operational focused. Regarding how he can get elected to the PMC, Allen said that it is a much more difficult question which he cannot publicly answer.

An attendee asked Allen whether he knew any framework dedicated to automatic health checks for Hadoop. Allen responded that he is not aware of an existing framework but it would be advantageous if anyone could provide directions on how to build one or contribute to developing an open source framework for health checks.

The last question concerned security and how an administrator can actually secure a Hadoop cluster. Allen responded by providing some examples from their configuration at LinkedIn. 'LinkedIn uses iptables for firewalling their clusters, and when they finish configuring the whole cluster they usually export the resulting iptables and provide them to the network administrators to implement in the network stack. Allen also expressed concern regarding the fact that switches might not have enough memory to implement the whole iptables configuration.

Optimizing VM Images for OpenStack with KVM/QEMU Fall 2013

Chet Burgess, Senior Director, Engineering, and Brian Wellman, Director, Operations, Metacloud, Inc.

Brian presented Metacloud's experiences running production clouds for their clients and, as a first step, he presented all software versions of the tools covered during the presentation. He continued by presenting the most common disk formats, RAW and QCOW2, and a VM container format, AMI. RAW disk format provides a direct representation of a disk structure; it can be sparse, is widely supported by hypervisors, and can be treated as a block device. On the other hand, QCOW2 (QEMU Copy on Write version 2) format is part of the QEMU project and supports pre-allocation as well as on-demand allocation of blocks. QCOW2 supports a wide range of features such as read-only backing files, snapshots, compression, and encryption. Additionally, Brian presented a short comparison between the two formats and some suggestions based on their experiences. Finally, Brian presented the AMI (Amazon Machine Image) container format, which consists of three different files: the AMI raw disk, the Amazon Kernel Image (AKI), and the Amazon Ramdisk Image (ARI).

During the second part of the presentation, Brian illustrated how a user requests and launches an instance by selecting the flavor of an instance (e.g., quantity of CPUs, RAM, disk sizes). All users' requests are scheduled to the nova-compute node which is responsible for making the image available locally. Additionally, Brian provided some information regarding the procedures that nova follows in order to allocate a disk file of either RAW or the QCOW2 format where the default disk format is nova.

During the third part of the presentation, Brian presented best practices for preparing an image OS for a cloud environment. He started by presenting a bunch of essential tools such as QEMU, cloud-init, etc. Additionally, he showed how these tools can be easily installed and configured on Ubuntu and CentOS.

Brian presented best practices on how an administrator should prepare a cloud image regarding authentication, networking, and hotplug support. He suggested that, as a first step, administrators should disable SSH password-based logins and disallow remote root logins via SSH. Administrators should create a single user and add this user to the administrators group and allow root login without a password from the virtual console. At this point, he noted that there is no actual security benefit if you introduce passwords, and it will introduce a lot of overhead in case you would like to change the administrator password on all pre-distributed machines. Moreover, Brian suggested that administrators should allow MAC addresses to be generated dynamically each time an instance is spawned and eliminate MAC address binding. Additionally, he recommended configuring DHCP clients for persistence. The reason behind this is that a VM should never give up trying to obtain a DHCP lease, because otherwise the machine might fall off the network and require administrative intervention. Finally, he demonstrated

how an administrator should configure a machine for hotplug support in order to prevent rebooting while attaching or detaching a hard disk drive.

Brian concluded his presentation by summarizing the contents of his presentation and providing his contact details for the audience.

An attendee from the University of Maryland noted that since Metacloud provides everything as a service, they have access to their customers' data and asked if they have any policies or auditing mechanisms. Brian answered that Metacloud is actually building and operating the cloud using their clients' datacenters, and thus they are not using their own infrastructure or datacenters. Everything resides inside their customers' firewalls, so from that point of view they follow the same guidelines that their customers follow regarding data protection and auditing.

Invited Talks 3

Summarized by Carolyn Rowland and Rik Farrow

Managing Macs at Google Scale

Clay Caviness and Edward Eigerman, Google Inc.

If you work at Google and you want to use a personal platform other than Mac, you have to make a business case for it. Google has made a big push away from Windows, to using Macs for desktops. Google has 64 offices around the world (http://www.google.com/about/company/facts/locations/) and 42,162 permanent employees as of the third quarter of 2013, as well as lots of contractors, temps, and vendors. As of the most recent month where they have login statistics, they are managing 43,207 Macs.

Google has a MacOps team charged with keeping those machines secure and operational. They use none of Apple's management tools, as Apple hasn't had a major release of their management tool since 2006. And since the release of the iPhone, they've lost their attention as far as enterprise management tools. Instead, they use open source tools and a few commercial products.

Google is very security and privacy conscious and has a very strong bias toward open source. If they don't find an open source tool, they have a very strong bias toward building their own tool. They are trying very hard to move away from the corporate network. They believe that all of their tools should work on the Internet, not only on the Google network. They support everyone from software engineers and SREs to masseuses and chefs.

They are also moving away from the corporate network to the Bag of Infinite Holding. They encourage their users to keep their data in the cloud and do everything they can in the cloud.

Google uses a single monolithic image for all users. They can do this because their package and config management are so good that they can produce a single image and customize it through other management tools. And Google is open sourcing the projects we use to do this.

Google uses Puppet for configuration management. They only do enough Ruby to get confused about it, and have clients pull configuration with the master. They use a sibling product called format that gathers information and sends it up to the server. You define the state of the machine as you want it to be and then Puppet makes it happen.

crankd is a Python script that registers system config events, Cocoa events. They use it every time the network state changes, to see if, for example, you're on the corporate network, a VPN, behind a captive portal, etc. They can also track application launches and quits, logging it for usage which is good for licensing and security.

Users do terrible things. They once had a user describe Puppet as a virus to a team mailing list including a list of ways to disable it and work around it. Occasionally the Google team does terrible things. So they needed Plan B.

Plan B is a management watchdog, and is just a shell script because they've seen cases where the system, Python or Ruby, breaks in interesting ways. They figure if the shell is broken then nothing is working. Plan B runs periodically and reports whether the management agents are running. If not, Plan B tries to download and re-start them.

Google uses three open source tools to manage their software life cycle from beginning to end. Munki is an open source project out of Disney animation. The client end is made to look like an Apple software update, and it manages software update for users. Munki can push Apple software updates or their own packages. They can prepackage things with optional software, and users will get a package that Google has okayed for them to install. The backend of Munki just looks for an HTTP server. They built the backend on Simian and that gives them fine-grained controls on how packages are pushed out. They manage hundreds and hundreds of packages without issues.

For security and monitoring, they built a tool called Cauliflower Vest (an anagram of "filevault escrow"). Cauliflower Vest is open sourced, allows for key escrow for full disk encryption (FileVault 2), and uses App Engine internally to Google. All of the access to the client's key is logged and audited. The user can get the recovery key or incident response, or someone on the MacOps team can get access to the locked hard drive.

They owned up to not having a great story on inventory management and have another open source for it. They get most data from factor (the tool that works with Puppet) and a few other sources. Most of the data they get from factor (related to Puppet) and a few other sources. This data gets reported to App Engine and sent up to other inventory management systems at Google.

To summarize security, encrypt all things, log everything, and log everything centrally. Make sure everything is in source control, make sure anything you do can be done again exactly the same. You don't want to do something 40k times.

Someone pointed out that some of their users have a negative opinion of Puppet managing their Mac for them. Assuming the existence of independent hacker types who want to manage their own Macs, how do they balance that? They try to go with a carrot-stick approach. The access to internal network resources is dependent on a machine certificate. In order to get that certificate, certain internal checks have to pass, demonstrating the machine is up to date. If it isn't, they can revoke the certificate. They also give users flexibility to opt-out, but not a simple way to opt-out. There are always some edge cases, so they try to create as much freedom as they can.

Another person wondered whether Google doesn't have the ability to ask Apple "Can you just make that work?" The team replied that they ask Apple lots of things. Joking aside, Apple takes a lot of their suggestions. Google saw Lion 10.7 was great and asked for key management, and a year later they got some sort of key management. They're easily one of Apple's biggest enterprise customers. But Apple easily sell 3x as many Macs at the Palo Alto Apple store than they sell to Google, so Apple knows which side their bread is buttered on. With the original FileVault 2 in 10.7, you could give the security question and key encryption and send it up to Apple. But there's no way for that person to get it back, and Google needed a way for the enterprise to get the data back.

Mario Obejas (Raytheon) asked about Pymacadmin (https://code.google.com/p/pymacadmin/) and they replied that Pymacadmin is mostly crankd. Mario continued by pointing out that many of us have all kinds of edge cases. Those of us with lots of engineers have the same problem. What does Google do for those repeat offenders where they don't have a business case, but who are repeatedly defeating the controls Google has in place. Does Google have escalations? A divergence report? Contact their manager? The team responded that they try to fix things technically up to a point. If that doesn't work, it's an HR problem...sort of. If there's a reason they're doing this constantly then maybe there's a workaround they can give them or a knob they can tweak. It doesn't happen very often. Hypothetically, they might have a developer who was installing the latest OS before they approved it, but that's also rare. Mario wondered what they did with the guy who actively posted on the mailing list about how to defeat controls, and they answered that they shamed him and his manager. He was good about getting feedback.

Another person asked how they manage user preferences (bookmarks, local prefs) and the team answered that they trust the users to back up. Their users use Chrome which backs up its preferences to the cloud. They encourage people to use iDrive. They discourage users from keeping data locally. They don't do backups at this level.

Someone asked since they let users keep their own backups, how do they make sure they don't just keep an unencrypted backup that may contain something valuable to the corporation. Their answer was that it is hard to keep users from doing bad things if they really want to. They provide guidelines to using Google tools that they trust. They try to keep an eye on tools running on the machines, like Apple's Time Machine which they have not been able to encrypt. They'll contact them and encourage them to use something else. You can't stop users from plugging a USB drive in and copying stuff. Instead, they have guidance that says "don't be an idiot."

Finally, someone asked why they use Mac OS when they have Chrome OS, and the team answered that Chrome OS is young. They have a lot of engineers, some need to run MS Office. They have needs that go beyond what Chrome OS can do right now.

OS X Hardening: Securing a Large Global Mac Fleet

Greg Castle, Security Engineer, Google Inc.

Greg explained that Google hardens Mac OS X using Apple built-ins and add-ons. To start with, full disk (FileVault 2) encryption is critical for an enterprise fleet of laptops. They use Cauliflower Vest for key escrow, and the user gets notified if anyone requests a key to unlock the files on their disk.

The next step is to keep badness off the platform. Apple provides Gatekeeper, a tool that taints downloads via Web browsers. Anything that is not signed is blocked, and you can added your own keys for your enterprise developed apps. If you call malware from cmdline or POSIX APIs or shellcode, then Gatekeeper doesn't protect against those cases.

XProtect (File Quarantine) is similar to Gatekeeper, except that it works more like AV. There are currently only 40 signatures and the update frequency is pretty low—seven updates in six months. Apple has introduced minimum version requirements that allows you to stop people from running older versions of software (e.g., Java). The format of the signatures is so simple, it seems that Google could extend the signature list themselves. Greg mentioned that you must keep the "Report to Apple" checkbox ticked, or there is no logging when XProtect blocks something. Also, it would be useful if Google could have a separate plist (configuration file), or get Apple to include signatures that Google creates.

Application sandboxing is used by Chrome, Adobe Reader, and all app store apps. All major software vendors are using sandboxes. Apple has made this simple for developers with Xcode, and Google has been working to sandbox their own management daemons and tools. The most simple case is just to trace execution and log, then you can run sandbox-simplify to dedup the information and make it more usable.

Sandboxing is not perfect, and Greg mentioned Meder's ruxcon 2013 presentation.

Greg then said that Java is no longer installed by default. Also, XProtect enforces a minimum version of Java. A Java bug was exploited by the Flashback attack in April 2013. The nssecurity plugin wrapper (also works on Linux and other plugins) Java now has includes whitelisting of versions (this old version of

Java with this Web site is okay, but no others), and JRE expiration dates (if you try to call the plugin after the expiration, you can't).

Greg said that the other elephant in the room is Flash. Given that most of Google users' Macs have Chrome installed, and Chrome includes Google's own version of Flash, one that is sandboxed, why not try uninstalling Flash? They started with a small group, moved to a larger group (which he called the canaries), and Google eventually was able to get 94% of Mac users to uninstall Flash. The rest presented use cases for keeping Flash (separate from that in Chrome).

The Google philosophy is no corporate network, and for this to work, they need to be able to apply patches, tweak configurations, and collect logs. The configuration and patching were discussed in the previous talk. Google collects one billion lines of log info per day for its Mac fleet using an internal tool. They also use a tool called GRR (Google Rapid Response, http://code.google.com/p/grr/) so they can handle incidents anywhere.

They also use the auditing that comes with Macs. OpenBSM (auditd) is an implementation of Sun's Basic Security Module written by McAfee Research under contract to Apple. OpenBSB is very different from the auditing in Linux in that it audits classes of events. Greg called the auditing information invaluable for detection and incident response.

David Lynn (Vanderbilt) asked since they are accepting log info from any machine on the Net, how do they keep nation states from blowing out their log server? Greg said they use machine certs and SSL. Someone else asked whether they are trusting Puppet certs to be their end-all of security. Greg replied no and went on to say that exposing your Puppet server is not something he would recommend. They have a shim that sits in front of Puppet that uses a cert they issue themselves, and they proxy through an SSL connection to their Puppet server. Clay and Ed in the previous talk discussed where Google is using distributed Puppet to send the manifests down to the client; in that case they don't need these sorts of proxies.

Someone asked about the OS X firewall. Greg responded that they haven't replaced it, but doesn't believe the security value of firewalls is enormous.

Rik Farrow (USENIX) asked whether they watched for anything in particular, such as when Apple changes the underlying system tools operation (moving events to launchd).

Greg said that they use the OpenBSM framework to monitor that sort of thing. Rik also asked whether they had fixed the problem where Firewire could be used to access memory, even in a screenlocked system, and Greg responded that Apple had fixed that with their 10.7 release (a firmware update).

Invited Talks 1

Summarized by Georgios Larkou (glarkou@cs.ucy.ac.cy)

Cloud/IaaS Platforms: I/O Virtualization and Scheduling

Dave Cohen, Office of the CTO, EMC

Dave started the presentation with some background information regarding the explosion of data growth and how we can use I/O virtualization in order to address it. Dave noted that the transition from static terminals to mobile computing introduced the need of data services and anywhere/anytime connectivity. He discussed the notion of economies-of-scale, an economy where factors cause the average cost of producing something to fail as the volume of its output increases. As a result, datacenter operators have to bring the operational-cost-per-megawatt-of-power down via sharing and automation. Additionally, the explosive growth in the size and diversity of data-under-management has forced datacenter operators to look for ways to decrease the data redundancy rate.

In order to reduce the data redundancy rate below 2x, multi-site erasure coding is being adopted by datacenter operators. The adoption of multi-site erasure coding is predicated on scale and caching. Dave provided an example of distributed storage with dual-site mirroring, which improves reliability and availability, with redundancy roughly equivalent to RAID5, and another one with distributed storage with multi-site erasure codes and improved reliability and availability of dual-site while reducing redundancy below 2x.

During the second section of the talk, Dave presented I/O forwarding, which bridges the gap between compute and storage in horizontally scaled environments and introduces an intermediary tier that provides impedance matching between compute and storage. In addition, Dave presented some existing libraries that the audience can potentially use in order to build a complete I/O forwarding engine. He presented a complete architecture of a I/O forwarding engine and briefly discussed most of the modules.

Dave then discussed IP Internetworking and how network virtualization can be used as the means for supporting I/O forwarding. He provided an example of how a datacenter operator can evolve the datacenter network to support scalability and mentioned that almost every cloud provider he is aware of is currently transitioning to a more scalable datacenter network architecture. Dave commented on the idea behind programmable switches supporting this transition and the explosive (almost infinite) amount of bandwidth needed to support the newly introduced architecture and noted that this bandwidth can be treated as a schedulable resource. In order to justify his suggestions, Dave presented four illustrative deployment scenarios, Google's Data Center to Data Center Connectivity, Microsoft's BGP-Only/SDN-Controlled Data Center Network, Open Network Lab's BGP-Only/SDN-Controlled Internet Exchange Point, and BGP-Only/SDN-Controlled Network Architecture recently posted by Ivan Pepelnjak. Additionally, he provided a survey of

Network Virtualization Controllers and highlighted the Open Network Operating System (ONOS) developed by the Open Network Lab.

Dave ended by presenting Autovias, a network-partitioning scheme for I/O forwarding. Autovias is not a product but it is a technology that allows a user/service manager to schedule network bandwidth for their service. The Autovias concept was conceived in 2012 and derives from Google's B4 Data Center to Data Center solution as presented at the Open Networking Summit 2012. Additionally, Autovias employs a Network Virtualization Controller such as VMware's NSX controller. Autovias was first demonstrated and discussed publicly at EMC World in May 2013. In conclusion, Dave presented a top-down view of a network, that we couldn't have before, which allows the administrator to be able to monitor, model, and modify over time from the controller.

An attendee asked how an administrator can manage the bandwidth between multiple sites/groups and whether it is based on VMs or source-destination IP addresses. Dave responded that it is based on subnets that ideally provide IP failover and use standard BGP techniques for moving things around.

Cluster Management at Google

John Wilkes, Google

John started the presentation with a map of Google's datacenters and commented that the location of each datacenter is as near as possible to their customers. Additionally, he showed one of Google's datacenters in Finland. He also defined the terms cluster, a set of machines connected with a high speed network, and cell, a set of machines managed as one entity and went on to explain why cluster management is important. A cell runs jobs, made up of one to thousands of tasks for internal users, and a manager, one per cell, is responsible for allocating these jobs and assigning them to machines.

John presented the two kinds of jobs Google receives: batch jobs which are things you submit that produce a result and, if they succeed, exit. Batch jobs regularly run and produce a result in a few minutes. Service jobs, the second type of job, start and stay up until there's a software update. Google Docs and Gmail are examples of service jobs. Additionally, John presented job interarrival time, small batch jobs that continuously arrive at the cell and services which remain and consume resources for a larger amount of time. Afterwards, he presented some graphs derived from a cell's logs and illustrated that the jobs that arrived in each of the three cells did not actually consume all the available RAM or CPU cycles. As a result, a significant amount of resources were just idling. He provided a public cell workload trace (search for "john wilkes google cluster data") consisting of a 29-day cell workload trace from May 2011 and asked the audience to download the traces and try to write a better schedule manager in order to achieve better resource utilization. Of course, John mentioned that better resource utilization is not the only goal of a good scheduler since there is a variety of other SLI/SLOs that the scheduler has to satisfy: for example, availability (e.g., max outage duration), performance (e.g., time to schedule and start a job), and evictions (e.g., percentage of jobs with too high an eviction rate).

Google's current cluster management system was built in 2003-4 and it works pretty well. But it is difficult to extend it and add extra features, and it is difficult to add new people because of its complexity. As a result, John presented a brand new system, coined Omega, which is currently being deployed and prototyped. The main goals of the new system are to be easy to use and flexible. Additionally, John presented Omega's architecture and briefly described its components. He also briefly commented on some of the advantages introduced by Omega, such as the distinct parameterized schedulers for batch jobs and service jobs, cell reconfiguration without breaking SLO promises, a calendar that can answer questions about future events (e.g., "Can I execute this batch job on this cell without breaking SLOs?"), and sharing decisions made by a scheduler with other schedulers. Finally, John presented the results of a performance experiment (simulation study) between the old monolithic scheduler, UC Berkley's Mesos scheduler, and Omega. The results showed that Omega performed better than the two other schedulers and, more specifically, provided 3–4x speedup for MapReduce jobs.

In conclusion, John presented some open issues—for example, smart explanations. According to smart explanations the system should provide reasonable information to the user: why the job did not run and some advice on what the user can do in order to fix these problems. John predicted that cluster management configuration may be the next big challenge.

The first questioner noted that most of the existing configuration systems are declarative and asked what is the next direction in configuration management systems and whether we can learn from biological systems. John answered that we need a mix of declarative and imperative, with the introduction of some constraints for safety reasons. He also stated that he believed that machine learning will emerge in the area of building smarter configuration systems.

The second questioner wondered how good people were at giving estimates about what resources they needed. John answered that if people know about every job that requires more resources than what they have requested, they will be really good and they will provide a really good estimate of what they need. Of course, systems tend to be much better at estimating the amount of resources a job might need since they can take into consideration logs from previous executions or pre-calculated parameters, but the current systems might fail to provide a good estimate for newly submitted jobs without supervision.

Someone asked why Google is implementing its own configuration system/scheduler when some open source solutions already exist. John answered that he can hypothesize why...but he won't.

Invited Talks 3

Summarized by Ben Cotton (bcotton@funnelfiasco.com)

Managing Access Using SSH Keys

Tatu Ylönen, SSH Communications Security

Tatu Ylönen, the inventor of the widely used SSH protocol, began with a brief introduction to SSH before launching into the thesis of his talk: unmanaged SSH keys are a security problem. Most organizations don't manage or remove old SSH keys. Since servers trust clients, compromising a client can allow access to more important servers.

As an example, a global top-10 bank that Ylönen worked with had over 1.5 million authorized SSH keys. Over 2 million daily key-based logins occurred to the 2000 core production servers. In many organizations, over 90% of the possible access credentials to production servers are SSH keys.

Unmanaged SSH keys are an excellent attack vector. Organizations don't know who can access what and who is accessing what systems or information. SSH keys can bypass other access management mechanisms. They allow unattended file transfer and escalation from development to production environments, non-PCI to PCI environments, etc. Most critically, SSH keys lack effective termination of access.

Current and pending regulations address SSH key management, often indirectly. Proactive management is important to prevent regulatory issues before they happen. The first step is to establish a controlled provisioning process that documents the purpose and owner of each authorized key. Once provisioned, SSH credentials should be continuously monitored for automated and interactive access. Since some legacy systems will break if the keys are changed abruptly, careful remediation of those systems is necessary.

In Ylönen's experience, the process of remediating one million keys can be a multi-year project. Some tools are available, but they don't necessarily cover the full process; many only do discovery. SSH Communications Security offers two tools and consulting services.

Ylönen concluded by saying that most organizations with large UNIX or Linux environments have a serious compliance and security problem, the scope of which is not widely understood. He finished with time for a single question. An attendee asked whether it was possible to automatically expire SSH keys. The answer was no.

Secure Linux Containers

Dan Walsh, Red Hat

Containers are a hot topic in Linux and security, but people don't always agree on what a container is. Walsh based his presentation on his definition of containers as a user space, not a kernel-space construct.

A container has four key elements: (1) process isolation, giving processes their own private corner of the world; (2) resource management, controlling how much of a given resource a process gets; (3) security, where tools like SELinux enforce security policies to keep the containers from overstepping their authorization; and (4) a management component that allows administrators to control the container. In the Red Hat Enterprise Linux container architecture, namespaces provide the process isolation, cgroups provide resource management, SELinux provides security, and libvirt is used as the management tool.

Because containers have some similarities to virtual machines, including being managed by libvirt, Walsh examined the two concepts side-by-side. Containers provide benefits like improved startup and shutdown speed, ease of maintenance and creation, and scalability. Virtual machines, specifically KVM, offer the ability to boot different operating systems, thus providing full separation from other clients, and supports features like live migration.

Walsh also presented a container product called "Docker." Docker bundles a micro-operating system with an application. This allows developers to bundle their application without regard for what host the application will run on. Red Hat has recently announced an effort to contribute to Docker development to allow containers to work on either Docker or Red Hat's OpenShift PaaS offering.

Walsh closed with a live demonstration of building, using, and tearing down containers. The audience was briefly surprised to see he wasn't running SELinux inside the container, but that turned out to be a feature of the container. The talk ran past time, but the audience was content to watch the demonstration until it ended.

Closing Plenary

Post Ops: A Non-Surgical Personal Tale of Software, Fragility, and Reliability

Todd Underwood, Google

Summarized by Andrew Hoblitz (ahoblitz@cs.iupui.edu)

Todd started off by saying that system administration as we currently know it is over, and that the time has come to quit feeding human blood to machines. He noted that machines currently eat system administrators' times, passion, and imaginations. Todd noted that he shares the same history as many system administrators who started off at ISPs, but that it might be time to revisit common views of system administrators and their nostalgia for the past. Todd noted that common objections he has heard are that Google's solutions to problems may not apply at smaller scales, that Google has massive amounts of software engineering available to it, and that Google has already solved all the problems you are going to encounter. Todd rebutted these arguments by noting that they may solve common problems for everyone and that open source solutions will allow system administrators to focus on tasks which are more interesting to them. Todd then described a number of "war stories at scale" to

show that Google has its own share of problems which it has had to address at all.

Todd then began describing Google's approach to Site Reliability Engineering by providing a "mythical" history in which he described Google as frugal and argued in favor of costs which don't scale linearly. He said that one example of this type of solution was developers deploying their own code in production through automated processes so that they would not be responsible for repetitive and tedious tasks which machines can perform better. He also talked about the importance of being able to balance competing demands and improve availability and efficiency while helping improve the deployment times of new services. And he described systems developing emergent behaviors during their post-deployment evolution.

Todd then described DevOps at Google, which he sees as a cultural and professional movement. In a traditional infrastructure, development and operations are walled off from each other, he noted, whereas DevOps is the blending of development and operations together in to a single way of being and doing things. He said that DevOps strives to improve operations as a discipline and integrate operational concerns into business practices. Todd then quoted Adrian Cockcroft's definition of "NoOps," where software developers work directly with production and automation removes operational tasks entirely.

Todd went on to describe operations management as applied to software-based production while noting that operations research has a set of constraints while software production has its own set of constraints. Todd said that operations research may be relevant for the physical infrastructure stack (power, fiber, etc.) and for maximizing SLAs in the short run, but that it may not be as applicable towards software organizations. Todd said he would still like to maintain the ethic of caring about production and the ethic of privacy and security.

Todd closed by noting that he is advocating for a transformation from Ops, to DevOps, to NoOps/PostOps. Operations is still needed to identify new problems and prioritize development, but every other aspect of operations should be deleted. Todd said it is important for technical talent to demand that their management knows what is going on technically, because this breeds a better environment for everyone. He wants to work in an area which is constantly looking to improve, and that even though system administration is eventually going to go away, the world is moving towards distributed application development. The bathwater that will get thrown out is the repeatable work and the configuration management, while the baby that will stay is the production ethic, monitoring of systems, release engineering, and constructive cynicism.

Advanced Topics Workshop at LISA '13

Washington, D.C.
November 5, 2013
Summarized by Josh Simon

Tuesday's sessions began with the 19th annual Advanced Topics Workshop; 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 about 2 to 1 (down from 3 to 1 last year); over the course of the day, the room included five LISA program chairs (past, present, and announced future, down from 11 last year but much closer to our five- and ten-year rolling averages of 7 and 5.5, respectively).

Preventing Mistakes

For the first topic we talked about how to prevent the "I know what I'm doing" syndrome. One person had several outages over the past year caused by either development (code mistakes) or operations issues; examples in the industry include Knight losing $172M/sec (http://bit.ly/1a9Vapb) and Twilio's billing system sending out over-large bills (http://bit.ly/17egzzp). How do you make sure you (or your peers) don't cause unexpected outages?

This led to a lively discussion about processes and culture. One answer to the question was to have a backup so someone can go away (on vacation, a promotion, or departure) and the organization still has the skills. Another answer was to change the culture from "A checking up on B" to "A and B working together." If the culture is such that someone works with you to share the risk and responsibility, outages become less frequent. This requires putting in processes so nothing is ad hoc, such as "always do it in a test bed/sandbox first."

Document your changes (you do have a change management policy, right?). Include things like a change template: Do you need to update monitoring or other services? Is there a rollback plan? If the change is customer-visible, were they notified in advance? This leads to focusing on the customer experience; anything that's customer-visible should get greater scrutiny.

Also, document the policies and procedures. Consider having the policy include "Have a second set of eyes review it" before performing the task. Your processes also need to be written down and followed. Creating that documentation alone reduces the effort; even simple checklists can help prevent problems (at least as long as they're followed). This also acts as the bridge to automation; you should automate what you can.

Other comments included slowing down to do it right so it only needs to be done once, and then automate it; being willing to say No; reminding people that even simple changes can have catastrophic consequences; and remembering that some people learn from making mistakes.

Finally, if there is an outage, you need to have some form of after-the-fact discussion, but framing the questions there is key. "What did so-and-so do wrong" implies a blame culture, but "What did our system do wrong" implies a process culture. The latter is more likely to lead to positive results.

Favorite Tools

After that discussion wrapped up we took a poll: what's your favorite new-to-you tool over the past year? Answers included nine-foot tall cabinet racks, C++11, Colibri, CamScanner, dmarcian.com to analyze dmarc mail data, Docker, Emacs as a server on Mac OS, Evernote (premium subscription), Git, Graphite, a GPFS-native RAID appliance, having a good project manager, LaunchBar, logstash, a new office chair, online collaborative tools like Google Docs or Dropbox, packer, Quicksilver, S3 Glacier, and vagrant.

Fostering Communications

Just before the morning break, we discussed fostering communications and teamwork. One person had an environment where a group of developers created a product that used root for everything, including reformatting disks. He needs to cultivate a culture of looking for information instead of working in a bubble. The consensus around the room was that breaking down the silo walls between teams is important. Some suggested remediation included job shadowing; managing the developer/administrator role/skill gap; creating a moderating team with representatives from multiple groups, whose purpose is just to ask questions so that alternate perspectives are considered in the requirements phase instead of after implementation; "making rounds" like a doctor to see your customers and make sure their needs are being addressed; using other technologies such as a shared wiki and online chat; offsite social gatherings like lunch or bowling; and having the developers involve the sysadmins early in the process.

Software-Defined Networking (SDN)

After the morning break we talked about Software-Defined Networking (SDN). Someone introduced it to us at last year's workshop, and one of our cohort has thought about it for the last year and hoped vendors would come out with more hardware. One problem is that not everyone agrees about what SDN needs; to some it's running config management on the switches, to others it's running OpenFlow, and to still others it's having Perl or Python APIs to access the switch in real time. It seems to be gravitating towards defining it as the separation between the data layer (protocols) and the control layer. The hardware is being consolidated into the ASIC manufacturers like Broadcom; the chips are integrated into the switches (either on-board or with add-on cards). OpenFlow (v1.3 or later) promises better control for sampling on the fly, such as with an intrusion detection system. Someone sees its major use cases as bandwidth management (between multiple datacenters with quality of service) and multi-vendor cooperative partnerships (such as managing hundreds of thousands of virtual machines on thousands of physical machines so they need VXLAN on top of the switches to manage things). However, another of our cohort talked to his networking team to get SDN in-house, and got a lot of push-back because "it's just using Python to control the switches." He had to educate them; they don't see the bus that's coming up on them.

Manufacturing IT

That segued into our next topic, Manufacturing IT. Ben Rockwood gave a talk about looking at IT from an Ops Management perspective a few years back, and applying manufacturing's building concepts to IT: moving systems to single-feature deployments, reducing cycle time, and so on. The question was whether there is any practical use for it yet. One person used the analogy of sysadmin-as-mechanic talking to the customers-as-drivers; it's not just about fixing the car, but about making it more efficient.

One person recently had a gratifying experience; they're migrating from an old job scheduler to a new one on their cluster. The most immediate and effective way to communicate the new capabilities and kindle the users' excitement was to sit down with them, watch what they're doing, and improve the user experience. Sysadmins need to remember that not everyone understands the system in the way that we do. Another noted that asking the users direct questions was essential. We need to make sure the problem we're trying to solve is actually worth solving now; you may find that other areas need the resources focused on them instead. It was also noted that there's both the design and operations levels, and how you work at each is different. At the operations level, once you know how things should be done you should automate it as much as possible to free yourself up to think about more at the design levels.

One person noted that we're focusing on the one aspect of the DevOps concepts here. A counter-example is that we have a lot of operations automation, so what's the value-add of change control? What's the model for where the efficiencies should be? He's disappointed that we're not going beyond what we're looking at and looking at other industries. It was pointed out that Alva Couch can't get system administrators and operations researchers involved in each others' areas.

Personnel Issues

Our next topic was about personnel issues. One person, now effectively a CIO of his company, had a person whom he called "the single worst sysadmin manager ever." This person trained his staff to believe "the fastest way to get fired is to get noticed," would do the work and not explain why, and would take tasks away from the technical people. This CIO inherited this team, fired the manager in question, and tried to retrain the SAs, but unfortunately wasn't that successful. Despite providing opportunities to learn and grow, only two people of a nine-person team looked into it (and got spot-bonuses and promotions, both publicly announced, and commensurate salary increases). What could he have done better?

One person said it comes down to autonomy, mastery, and purpose, and with the previous manager all three were broken. Reinstilling one is possible, but reinstilling all three may be impossible. Starting with purpose is best; once you have purpose you can build mastery. Another noted that the sysadmins themselves need to feel responsible for something to be engaged in or with it. There's certainly a fear of blame (if something goes wrong), and a lack of responsibility leads to the "who cares?" thought. As someone else noted, one can teach a toddler to respect edges by letting them fall off a couch but not out the second story window.

Someone suggested developing a post-mortem culture. Any and every time something goes wrong, a post-mortem is required. Another suggested that for major projects they should hold a post-mortem even or especially if everything went well. Make it clear that it's not an issue of blame but of learning. Also, having a Wheel of Disaster meeting where the team is given a list of scenarios and then choose one, with someone running it (as the gamemaster) and someone else as the on-call person, and do a dry-run of the troubleshooting "on paper" in the meeting.

Another asked if the sysadmins this manager hired were the right ones to begin with, or if they hired great people and broke their spirit? The response was that it was about a 50/50 split. Money and title is not always enough motivation and reward, and for some, money can even be a demotivator. Be careful to be supportive and not to blame someone if something goes wrong. As an example, were the sysadmins given incorrect or incomplete instructions?

Someone asked if the CIO talked to the individual sysadmins one-on-one as to why they didn't step up. Sometimes people don't want to say anything publicly. Their manager did and got "I had to do this other thing instead" as the response. Another added that trying team consensus brainstorming on what the team's top priorities should be to allow them input to the decision may make them more likely to buy into it. People need to take pride in their work, not necessarily take responsibility for it (which can be an onus). For recognition and awards, one environment gives out peer-nominated awards quarterly across IT, as voted on by the managers. They don't have a post-mortem culture but a preventative action culture so you can make things better before the disaster.

Finally, classes that teach what the knobs and widgets do but not why you might set them one way versus another aren't that helpful, so several write their own Best Practices documents that start with the why before going into the how. That way, when you check to see if something is still working, you can see whether the why changed.

Career Management

Next we discussed topics under the umbrella of "career-type stuff." One person is nearing retirement age. Another works for a small company acquired by a bigger one and isn't sure what's happening with his role. Yet another is being pushed into more of a mentor role than a hands-on technical role. The general question amounts to, "What should I do?" To lend perspective we took some quick polls. Of those in the room, half were born between 1950 and 1969 and half between 1970 and 1989; two people have over 30 years of experience, nine had 21–30 years, and seven had 11–20 years of experience.

The consensus was that you should enjoy what you do; focus on the opportunities provided by change, instead of on any fear; let go of control, which can be hard; look at the big picture; make sure your boss both knows and approves of what you're doing, be it hands-on technical or mentoring or direction; and work with your manager to explore areas of personal interest that are relevant to the organization's goals.

One person has the title of Director and actually gives direction, acting as the wise elder (but not the wise guy), setting direction but not dictating the details of how to get there. Write up your own job description and poke holes in it to see what's missing. For those in acquisitions consider the opportunity as a new job. Another notes that you can't grow within your comfort zone; growth requires moving outside that zone (and eventually expanding it).

On the subject of handing off tasks (as one approaches retirement, or is promoted, or otherwise leaves one's existing role), several worried about what will get left behind and possibly dropped, because in the specific case their coworkers are just putting in their time and won't do anything extra. Getting people out of this mind-set is hard. The consensus here was that while "document it all before you go" isn't enough, you can still set up your former area to succeed, but once you're gone they can choose otherwise and it's not your fault.

Another question was whether anyone else found that they want to do what they like on their own time and just have a job to pay for it? The short answer was Yes: one person is around a lot of actors who would love to do acting full time but can't make a living at it. Another has a job mostly writing code, and meetings are there for getting the requirements to write the code. And yet another moved to a job that's more in line with his interests and does the things he hated at his old job for fun and with less politics on the side.

Whither LISA?

After the lunch break, a USENIX Board member asked us to discuss the LISA conference itself: is it where it should be, or should things change, and if so how? One person's sense is that LISA both needs to change and is changing; they wondered how we think it should change, and what purpose it serves.

From a marketing standpoint, many agree that the conference has needed better marketing for a long time. One person (involved since 1997) found out about the conference on their own, and asked how others found out about it. Answers in gen-

eral were from a boss, coworker, or friend (about half the room); exposure to other USENIX conferences (four people); higher education, such as a computer science department's posting; unrelated events local to a past conference (two people) and several of us simply couldn't remember. Only one person first heard about the conference from USENIX. There are also more events these days, both regional (Cascadia, LOPSA-East) and specialty (PyConf). Should we better integrate the joint conferences (for example, a Puppet track instead of Puppet on the side)? More marketing needs to happen through more venues; word of mouth just isn't enough.

As for the content, the consensus was that it's fine, though there's always room for improvement. One person notes that many years ago, many of the things he found out about at LISA were things he could introduce to his workplace right away, and he's seeing much less of that. The refereed papers seem much more niche-specific. It's not clear whether this is because the papers and talks have changed, or if it's because we've gotten older. As for what USENIX can do to make it more relevant to those of us in the room: over the years, people who come to the workshop are here to give back to other people and they get workshops out of the conference. Someone noted that (by design) this workshop is for the more senior people, so between the hallway track, the ATW, and giving back, we're probably not the audience to answer the question posed.

On the subject of content, one person asked if the refereed paper model is obsolete, as that may be part of the problem. USENIX tends to come from the academic side of things where papers are all, but LISA tends to come from the practical side of things, and these concepts are at best orthogonal. Several of the workshop attendees have never written a paper, with the "Nothing I do seems to be of sufficient interest to other people" rationale.

As for the tutorials, there seems to be an impression that seniors see them and think they shouldn't need to spend a whole (or half) day on that subject just to get to the one chunk they want late in the tutorial session. There is a perception, at least among those in the room, that tutorials tend to be more for the junior to intermediate administrators (more as topic introductions); some should be aimed at seniors, for deep dives into a more narrow subject. LISA seems to be marketed more to the beginning through intermediate administrators, but the senior people definitely add value. One suggestion was to partner with or at least visit and snoop at other conferences like Velocity or SCALE to see what they're doing that we're not. We took a quick poll of the PC members in the room to see how many went to a competing conference this year and more than usual had.

Another person noted that students are missing. There used to be a significant visible student presence and that's no longer true. The role LISA has played, in addition to where to learn about the new things, was to provide a common language and a framework for discussion. There are more specialists now (DevOps, only-servers, only-storage, and so on), and LISA is a place for generalists and for bridging the gaps between the specialties. Someone else concurred; one of our competition conference's attendees skew much younger than LISA. How do we bring in more (and younger) people? Are we aiming for more juniors, or just more people? The conference has multiple tracks, but it's not always clear who the tutorials are for (junior to intermediate), and who the workshops and tech sessions are for, and where the senior folks fit in. Getting coherent content can be hard.

That led to a brief digression about giving back. Someone believes that many of us are still attending LISA after many years to give back to the community. We took a straw poll and only eight of 20 in the room at the time actually think they're giving back (including the five past LISA chairs). Some of that's from being in the Hallway Track and being available for others to ask questions, and others are teaching. One notes that especially for people who haven't been attending for long, the Hallway Track can be very intimidating. We need more informal ways of meeting people other than "let 'em loose in the same room."

Other Jobs

Our next topic was what we did outside of our day job as our "other job." Answers included being in a leadership role at church, bicycling, building (doing general contractor work), contributing to public software development projects, cooking, making art, officiating hockey games, parenting and grandparenting, photographing, teaching Highland wrestling, volunteering for both technical and nontechnical organizations. That's interesting but what's the cross-over? What do we learn in the one that helps out in the other?

One of the cooks notes that from cooking he got mise en place, staging everything before the stove goes on, and he applies that to datacenter moves. The wrestler learned humility. He used to wrestle and had fun, and he did well . . . until he was matched up against someone nearly half again his size in the last heat. He translates this to the technical arena by looking at how the younger people were coming along and he can't keep up—and the humility in not keeping up was almost an epiphany. The photographer now understands how to prevent people from abusing his time. People are hitting him up for free photo shoots, not realizing what time is involved in doing stuff. That translates well to IT work: time is valuable, and "just this quick thing" is often neither quick nor appropriate for the existing project. Several think books could be written about how parenting translates into dealing with people. One now appreciates the years of couples counseling and the relationship with their spouse, especially in contentious situations. Another notes that in acting, especially in comedy, you have to give the audience what they want. Timing, listening, and understanding what they want is critical both there and in technology.

What to Follow Next Year

Next we had one of our traditional polls: "What's the thing you'll be following in the next year?" Answers included bring your own device (BYOD), OpenStack, Software-Defined Networks (SDN), changing federal appropriations for IT in the wake of the healthcare.gov launch, dealing with internal politics, developing a process to handle end-of-life (such as Windows XP), making things enterprise-worthy in their kludged ecosystem, managing people's expectation of "everything works and never crashes," the density of flash memory, the role of system administrators in national security, and the state of encryption. Several also talked about cloud technologies, from deploying smaller more-local cloud technologies, to the security (or lack thereof) in the cloud in the wake of the Snowden revelations, to the debate on using the public cloud versus a mixed (public/private) cloud.

Cloud

Since we mentioned the cloud, that was our next topic of discussion. For some people their professional world is all in the cloud, others are still all local and think "cloud'll blow over," and then there are people whose users are dealing with both. Technical integration and expectation management are questions. For example, one institution is moving mail to the public cloud, thinking "It'll all be the same, just on their servers not our servers," but really it's "Our 100,000 users are distributed to some vendor's cloud." How does the help desk handle that? This clobbers the customer expectations.

While outsourcing infrastructure to the cloud isn't different from outsourcing anything else to the cloud, in that you always have to take the service-level hit, and the tradeoff needs to be calculated in the initial deal, the decision-makers often just hear, "Oh, it'll save money" and didn't do any other research into the technology or its security. Marketing needs to happen: "Doing this to lower our costs" is the common refrain, and sometimes lower cost will have side effects like a lower level of service (to keep tuition down, to give raises, and so on). The help desk staff will receive service calls for things they can't control or affect, and have to handle the users. There may often be no recourse beyond "call the vendor and hope."

Someone else comes at this from a different perspective. A software company providing what became Software as a Service on premises, then became a cloud provider, and one of the biggest things they learned is that customer support is critical; customers are much less invested in your product than formerly, since switching cloud vendors is possible (and likely, if you screw up your service).

Enterprise Monitoring

One person had over a hundred alerts from his monitoring system today and is seeing over 700 per week on average for the past month. They're obviously getting ignored by the entire team. If you're getting alerts that clear themselves and users don't notice, should that monitor be disabled? What are some best practices in monitoring?

Answers included alerting on "the right stuff"; being selective about how you monitor, as "95% full" on a 1 PB disk array might not be relevant, and perhaps the rate of change is more important; only monitoring actionable items; monitoring services, not just components, such as caring about the end-to-end user experience and not that a particular process or server is down; only emailing for important issues and using dashboards for less-urgent ones; opening tickets to yourself when you see gaps in what should be monitored but isn't; paging the developers for their service alerts to incentivize them to fix their own problems (false positives); providing context, such as letting the users of a service receive alerts and telling the sysadmins if action needs to be taken; putting the sysadmins on change management emails, so they can tweak the monitors where appropriate; turning off monitors that are unused or ignored; and using easily filterable subject lines in the alert emails.

It was pointed out that in Nagios the retry-check-interval and number-retries can be changed ("It's not a real problem until it's been 30 minutes"). It was also mentioned that you should be monitoring services, what's about to break or whether the service is down, not so much the server or component. In Nagios 4, you can attach a "wait" to an individual check: "Don't tell me something's wrong until X is also wrong."

One person wants three categories in a monitoring system: thresholds (number of occurrences or time length of the event), priorities (e.g., critical, major, minor, information, warning, and wtf), and escalation policies. Some alerts are just for reporting and don't send email; "1000 memory errors" isn't a problem over a year but is over 10 minutes. Dependencies are also important; if the switch goes down, you don't care about alerts for the 20 servers behind it.

The original questioner was happy he's not the only shop seeing these sorts of problems.

To-Do List Tools

After the afternoon break, we held another poll about what tools we use to track our to-do lists and whether that tool actually works. Answers to the former included Cozi, email, Evernote, GitHub, GTD, GoTasks as hooked to Google Calendar, JIRA, Lotus Notes, OmniFocus, paper notepad, spreadsheet for tracking the team's tasks, text editor of choice, and trouble ticketing system such as RT. Answers to the latter were either "yes" or "mostly," with a couple of "meh"s thrown in.

Asperger's Syndrome

Our next topic was Asperger's Syndrome, which the DSM-V has collapsed with autism and other concerns into the "autism spectrum disorder" catch-all. The question was asked whether we have any coworkers that have (or that we suspect have) Asperger's Syndrome, what challenges we have in dealing with them, and what would most help us address those challenges. The goal is that if you understand how someone's mind works you might be able to work with them more efficiently.

Asperger's is an autism-spectrum condition characterized by difficulties in social interaction and non-verbal communications, and restricted behaviors and areas of interest (very depth-first). Also, there's atypical use of language; one individual (recently diagnosed with it) tends towards the pedantic and literal, and his sarcasm detector runs about 10 seconds slow. He self-describes as an idiot savant: there are some things he does really well yet some basic things he does very poorly. It has to do with how his brain handles associations. If someone says, "Something green on the ground," most default to "grass"; his brain goes breadth-first (green CD case, spray paint, etc.), so jumping to the obvious doesn't happen—which can be an advantage for evaluating possible options that the neurotypical brain might discard. Most people's brains track social cues and proxemics and kinesics; Aspie brains tend not to. The upshot is that he wants to improve awareness of this sort of thing, so he's wondering if we have experience with coworkers who may have it and would resources (such as a BOF) be helpful?

Several others identified that they or their families have some form of autism spectrum disorder or ADHD (five in the room), and others (nine attendees) know or suspect it in their workplaces. In workplaces in general, for this and other hidden disabilities, people won't feel bad if you point out or help with issues they know about. For example, reminding an ADHD person that something's time-sensitive helps them break out of a loop. The speaker's Aspie friends have said the same thing about social cues, and being reminded would be helpful. That doesn't relieve people of the social requirement to do that reminding in a socially acceptable way. Be sensitive providing that feedback.

In an engineering-specific work environment, one of our cohort is sure there's tons of it (e.g., "wear the same shirt for a week, but write bulletproof code"). Now as a parent he's more tuned into the possibilities and can nudge them along.

Programming Beyond Scripting

Our next discussion was on programming beyond scripting. Some believe that once a site reaches a certain size you have to use custom software to manage it, beyond just off-the-shelf software and a configuration management system. Given some uncited research studies and anecdotal evidence that implies some people simply don't and can't be taught to think algebraically (for example, "x=2, y=3, set y=x, now what's x?"), the question posed was if most sites have system administrators who code in a programming (not scripting) language, is there agreement that some people just can't make the shift from scripting to programming languages?

One expressed dissatisfaction with someone with a sysadmin degree from a particular institution who can't code, but another noted it might be luck of the draw as they have four people from that institution who can and do.

One noted that coding ability is a continuum, from none through some to all. One speaker thinks he's a pretty good programmer but doesn't have to code often (about 10%); he's happy with his physical, infrastructure, and monitoring gig. There's room for non-coders in the profession. Several people agree that they don't code often enough to feel comfortable with it.

The question can be applied more broadly to all sorts of skills: does a system administrator need a Computer Science degree? Within a group you need a little bit of everything; scripting is programming, just in a different language. A group should have experience and exposure to both sorts of languages. But when it comes to code, it's not that unusual for generalists to look at someone's code in a language you don't know, so exposure to the concepts is necessary. You have to be able to think logically and analyze while troubleshooting or looking for a bug; it's all problem-solving. There is, however, a difference between troubleshooting and debugging someone else's code and creating the code. While there are differences between debugging and creating, you still need to understand algorithms.

Given the advanced automation tools (like Hadoop, Nagios, CFEngine, and so on) that require some level of programming, is there something today that doesn't require programming? One environment has a requirement that all code pass code reviews and readability testing, and they have the same (strict) testing coverage for operations code as for the business-specific code. Another was trying a decade ago to remotely manage machines beyond turn on/off, and now that there are APIs that the hardware vendors support, he can. And now that those integration functions have software packages, it may not be necessary to write this stuff any more. As more and more automation tools have higher-level languages, there's less need for scripting tasks. In several of them you can say what to do, not how to do it. There's still a need for programming but perhaps not a strict requirement.

Workshop

The next subject was the workshop itself. We traditionally have some trouble hearing each other. One of the other workshops uses wireless microphones and speakers. Only about ten of the attendees favored using wireless microphones. One person was worried about germ transmission. Another suggested standing up to speak to project better.

Consensus was that there should be another ATW next year. One attendee wanted both fresh blood, people who haven't been to the workshop before, and fewer graybeards. Our recent new members have mostly been referred by existing participants; we as attendees need to get new people in by talking to them and Adam. Part of John Schimmel's (founder of the ATW) original purpose was for senior people to speak to each other without juniors interrupting. Most of our cohort, however, were against having a listening-only role for juniors. Someone noted that even senior people who see "position paper" in the sign-up are too intimidated to write one. Several thought that's still a low bar, as the CFP includes an example (which amounts to "write

a proposed problem, not its solution"). Others thought that the position paper was a hurdle, that people need to think they'll get a benefit from the workshop before they do the work of writing the paper. Consensus was that the writeup in the CFP should be different, and Adam asked us to send him suggestions, some of which are below.

One member noted that different people can contribute differently. One possibility is that already-insiders can nominate and sponsor one person per year. Several agreed that would be a good idea; however, there is a 30 seat plus 2 (Adam-as-moderator and Josh-as-scribe) hard limit for the workshop.

Another noted that some may be more wary of the image of ATW than the reality, and wondered whether it's the old boys' club. Several thought that there's a perception problem outside this room. The response was that the writeup is in ;login: every year and that there's more than just the CFP writeup to help someone determine whether to apply. One suggestion was to link to the previous year's writeup from the CFP description. Someone noted that this writeup is appreciated, but we often don't continue the discussion (either at or after the conference) after the writeup gets published. Helping people understand what we discuss may have value. Many people aren't self-confident enough to stand up to "20+ years of experience."

One first-time attendee said he'd been to some other LISA workshops and thought ATW was by far the most functional workshop he's been in; many of us shared new-to-someone ideas and agreed that the ATW is both run well and well-organized.

Security

Security was up next. The question posed was "Is the enemy of my enemy really my friend?" One person's environment has an IT Security group that dictates policy but doesn't actually interact with or seek input from the system administrators, and frequently in companies you have Security making a recommendation to fix a problem but Sales saying they need it so you can't fix it. Do the sysadmins get involved in security audits? How well, or not, do your sysadmin and security organizations work together?

In one case, there was animosity between IT Security and Sysadmin. By reading the actual security regulations one Operations person was able to better understand Security's motivation and could therefore help build the bridges between the two groups. The two groups need to support each other, but we need to recognize the importance of both sides. In another case, Security would invite specific Operations staff to a multi-day training course and then keep those people as liaisons.

One environment would use the position of Security Manager as a patronage because doing real security well is hard but faking it takes no effort at all. The more security theater a company has, the less inclined the sysadmins are to be helpful. In another environment, Security and Operations are co-located, eventually reporting to a common manager, and the relationship is very collegial and not at all adversarial. Someone pointed out that an adversarial relationship can develop from how each group measures or defines its success: if Auditing is measured on the number of issues identified, for example, but Operations is not measured on the number of issues resolved.

Some environments are very checklist-driven, some are balkanized as opposed to centralized, levels of enforcement often vary (does Security have the authority to turn off a compromised Operations box?), and it's not always clear what kinds of exceptions are allowed (e.g., "patch every 30 days" is at odds with "we have a 2-month release cycle").

What's Coming Up?

Finally, we had our last lightning round: what's the biggest or most important thing coming in the next year? Answers included being in a team doing automation and writing code, building a Web cluster, building out their cloud, changing the billing model, continuing to get better with politics, driving the IT manufacturing concept, evangelizing for and implementing configuration management for both infrastructure and application spaces, extending dependency resolution into spinning up machines, finding a new job, finishing the six-month Hadoop rebuild, getting a monitoring solution he doesn't hate, improving security in general, increasing staffing, revamping and modularizing his 20-year-old dot files, ripping out and replacing their build/automation/CM system and putting in something quality, solidifying DNSSEC, surviving a reorg at work, taking all the different opinions about CFEngine into a coherent plan, deciding whether to retire in March or June, working on the next generation of supercomputers, and writing a refereed paper for LISA'14.

Summaries from SESA '13 are available online at www.usenix.org/login/feb14/sesa13reports.

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

Donate Today: The USENIX Annual Fund

Many USENIX supporters have joined us in recognizing the importance of open access over the years. We are thrilled to see many more folks speaking out about this issue every day. If you also believe that research should remain open and available to all, you can help by making a donation to the USENIX Annual Fund at www.usenix.org/annual-fund.

With a tax-deductible donation to the USENIX Annual Fund, you can show that you value our Open Access Policy and all our programs that champion diversity and innovation.

The USENIX Annual Fund was created to supplement our annual budget so that our commitment to open access and our other good works programs can continue into the next generation. In addition to supporting open access, your donation to the Annual Fund will help support:

- USENIX Grant Program for Students and Underrepresented Groups
- Special Conference Pricing and Reduced Membership Dues for Students
- Women in Advanced Computing (WiAC) Summit and Networking Events
- Updating and Improving Our Digital Library

With your help, USENIX can continue to offer these programs—and expand our offerings—in support of the many communities within advanced computing that we are so happy to serve. Join us!

We extend our gratitude to everyone that has donated thus far, and to our USENIX and LISA SIG members; annual membership dues helped to allay a portion of the costs to establish our Open Access initiative.

www.usenix.org/annual-fund

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

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