Conference Reports

In this issue:

USENIX LISA '11: 25th Large Installation System Administration Conference

Boston, MA
December 4–9, 2011

Opening Remarks, Awards, and Keynote Address

Summarized by Rik Farrow (rik@usenix.org)

The 25th Large Installation System Administration Conference began with the co-chairs, Tom Limoncelli (Google) and Doug Hughes (D. E. Shaw Research) tag-teaming their opening presentation. Carolyn Rowland will be the next LISA chair, with the conference occurring in San Diego in mid-December 2012. After many thanks to the organizers and USENIX, they announced the three Best Paper award winners: Herry Herry's "Automated Planning for Configuration Changes" won the Best Student Paper award, with his advisor, Paul Anderson, accepting the award for Herry. The Best Practice & Experience Report award went to a Google team for "Deploying IPv6 in the Google Enterprise Network: Lessons Learned" (Babiker et al.). Finally, Scott Campbell of Lawrence Berkeley National Lab won the Best Paper award with "Local System Security via SSHD Instrumentation."

Phil Kiser, LOPSA President, presented the Chuck Yerkes award to Matt Simon for his helpfulness and frequent activity on the mailing list. Matt said that he had never met Chuck, but that he must have been one heck of a guy.

David Blank-Edelman (Northeastern University and USENIX Board Liaison to LISA) presented the SAGE Outstanding Achievement Award to Ethan Galstad, the creator and principal maintainer of Nagios. Galstad said that he had created Nagios to prevent system administrators from being paged unnecessarily, and while he was pleased to receive the award, he is introverted and receiving it was not easy. He thanked his wife, family, friends, and the worldwide Nagios community, and the Linux community for providing him with a free compiler.

Keynote Address: The DevOps Transformation

Ben Rockwood, Joyent

Ben Rockwood, the Director of Systems Engineering at Joyent, gave a stirring explanation of the way he sees DevOps—not as a tool or a title, but as a cultural and professional activity. DevOps is not something that we do, but something that we are, said Ben. Ben declared that DevOps is more a banner for change (displaying a picture of a knight), not simply a technique. It is a journey, not a destination.

Ben went on to present both an interesting slide show and his detailed research into the history of systems studies. He emphasized that DevOps begins with the "Why," proceeds to the "How," and ends with the "What." The "Why" refers to the limbic system, the emotional seat of all animals, including humans, as does the "How." The "What" is the product, such as building awesome services. Another way to see this is that the "Why" represents motivation, such as quality through collaboration, and the "How" and "What" are the process and tools used. Ben said that DevOps does not mean starting with the "What," such as configuration management, and working backwards to the "Why."

Ben described Russell Ackoff's five contents of the mind: wisdom, understanding, knowledge, information, and data. He then tied these content types to levels of sysadmin, with System Architects related to wisdom and understanding, Senior Sysadmins with knowledge, and Junior Sysadmins/Support with information and data. Ben used these concepts as a way to shift into talking about systems thinking, with wisdom and understanding corresponding to synthesis and knowledge, information and data to analysis.

DevOps is about the entire system, not just Dev and Ops working separately, but with the two groups working together toward the goal of quality. While Dev and Ops are often silos, they truly are both part of the same system, but seeing this can be difficult. Ben quoted W. Edwards Deming: "A system cannot understand itself." This implied standing outside of both Dev and Ops to see how the two groups could work best together.

Ben was not content with spouting platitudes, but discussed methods for measuring quality in software and systems. You could tell that Ben had spent a lot of time studying potential metrics for measuring IT service, and after comparing many complex standards, he selected ITIL as the most complete and respected pattern for IT. Ben said that it is best to read the entire ITIL set of books, although he did suggest reading *The Visible Ops Handbook*, as a good place to get started.

At this point, Ben was just getting warmed up. He suggested Agile, said that the cloud has changed everything in the IT world (partially because it supports agility), and went into a history of operations management. In the next section of his talk, a section he said he feared might be boring, Ben explained where many of the ideas for scientific systems management came from. It turned out to be a fascinating look at the men who created this field and their contributions. He ended this section by suggesting that we stand on the shoulders of giants, and not ignore the lessons of the past. A simple summary of where we are today would include lean, Scrum, and agile operations concepts.

Ben concluded by bringing his focus back to DevOps, suggesting that boundaries between development and operations teams need to blur, and that both teams are fully accountable for both successes and problems.

Mark Burgess (CFEngine) opened the discussion by praising Ben for paying attention to great thinkers in history. He then mentioned Alvin Toffler's "Future Shock," and Ben agreed that people should read that book. Mark then asked if DevOps is an expression of wanting to make a difference, not just working in this monolithic tech environment. Ben responded that DevOps allows us to do what we always wanted to do, but were prevented from doing. Also, now more of the tools are present and we have moved beyond just building assembly lines.

Steven Levine (RedHat) commented on his world of tech writing, where he has to bridge the gap between marketing and engineering, and that DevOps concepts could address one of his lifetime issues. Ben answered by referring to Sun Microsystems tech writers, who had no access to engineers and just had to figure things out on their own—certainly not DevOps.

John Rouillard (Renesys) pointed out that if we don't know what went on before, we reinvent something else, perhaps worse. Ben responded by saying the problem is more fundamental than that—we don't know it exists! Ben told the story of walking his baby while at church, noticing a book on operations management (OM), then taking it home and devouring it. He hadn't even known OM had existed. Now he has an entire shelf in his office at Joyent devoted to OM books.

Perspicacious Packaging

Summarized by Carolyn Rowland (carolyn@twinight.org)

Staging Package Deployment via Repository Management

Chris St. Pierre and Matt Hermanson, Oak Ridge National Laboratory

Matt began the presentation by talking about their HPC environment and the concern for consistent security through automated package management. Using yum for package management did not allow for granular control of packages nor did it allow for a period of testing before rolling a package into production. The solution Chris and Matt presented included use of Bcfg2 for configuration management and a new tool called Pulp. Pulp is part of RedHat's CloudForms and a lean replacement for Spacewalk (minus the GUI and Oracle requirements). The authors wanted to control what was really in the package repository and how it moved into production, but they didn't want to do it manually. The repository has three areas: upstream—created daily from sources; unstable—automatically synced into from upstream filtering packages they don't want; and, finally, stable—copied from unstable after about a week. Development machines get their packages from unstable while everything else pulls from stable. For an HA server, one node pulls from unstable to test the package before the other node installs it. The only exceptions are security patches, which receive immediate attention. The authors said it was important to install the security packages right away.

Downgrading with yum was difficult, as was rollback. The authors examined and rejected a number of solutions, including functionality built into yum and Bcfg2 and other repository management systems. Pulp can do manual manipulation of blacklisted packages from upstream which can then be overridden with one command for machines that have an exception to the blacklisting rule. The authors' workflow uses the Pulp automated sync facility to promote non-blacklisted packages automatically. There are exceptions to the automated package management. Impactful packages (e.g., kernel packages) are not automatically promoted. The authors found that servers using Pulp were far more up-to-date with upstream package versions than the rest. Chris has also written Sponge, a Web front-end (written in Python/Django) to make configuration of Pulp more intuitive (Sponge is available on github: http://github.com/stpierre/sponge). Sponge is currently the largest instance of programming for the Pulp API outside of the CloudForms project. Matt and Chris aren't finished: they'd still like to add an age attribute to packages. Currently, packages move through the stages weekly, so a package may only be in unstable for a day before it is promoted to stable. They'd like to make sure that packages are at least seven days old before moving between unstable and stable. Currently, Pulp supports all RPM-based distros. The Pulp developers plan to support arbitrary content (e.g., Debian packages).

During questions, one person asked how the authors collected concrete evidence that packages were ready to go from unstable to stable. Chris responded that they depend a lot on HA to protect them, allowing the services to test themselves, but agreed that testing could use some improvement. Could the authors use RPM and yum to detect a rogue package installation? Bcfg2 could report these packages and they could be removed automatically; the authors are not currently doing this. How does the system know which packages are authorized? Bcfg2 has a list of base and specialized packages. Someone pointed out that a rogue package would only be detected if it were installed using RPM. The authors responded that one could use tripwire to detect such anomalies.

Does Pulp mirror the repositories or is there a separate tool to do that? In most cases, Pulp can do the mirroring; for RHN; the authors currently use mrepo. How can Pulp improve over the Cobbler+ custom scripts method? Chris admitted that Pulp was alpha code when they started (even as recently as a month ago it wasn't ready), but added, "We were also the first to contribute code back to it." How do the authors mirror packages from one site to another and still keep package consistency? Pulp has a content distribution system (CDS), but they hadn't used it in their efforts. How do the authors maintain an older version of a package even after that package is no longer available? They break the mirror and set that package aside. "We do have a couple of environments that demand that stability." Pulp uses hard-links to point to these packages.

CDE: Run Any Linux Application On-Demand Without Installation

Philip J. Guo, Stanford University

It's hard to package your software so that other people can reliably run it. You cannot predict someone else's environment. The Internet is full of forums trying to solve this problem. It's also tricky to create a package that runs across all Linux distros with no problems: missing dependencies and different environments can require a tiered dependency installation to resolve. Enter CDE, a tool that provides automated packaging of Code, Data, and Environment. According to Philip, there are three simple steps to using CDE to package a piece of software: (1) use CDE to package your code by prepending any set of commands with "cde"; (2) transfer the package to the new computer; (3) execute software from within the package on any modern Linux box using cde-exec

(root not required, installation not required). Step 1 creates a CDE package in the current directory that copies all necessary files into a self-contained bundle. This is similar to a chrooted environment, because every file or environment variable required to run the command is available within the local CDE directory. A second user of the program can also edit the script within the package and modify as needed, still running it with all included dependencies. CDE uses ptrace to hook onto the target process. It actively rewrites system calls. When a call is made to the kernel, CDE intercepts and copies the accessed files (e.g., "/home/bob/cde-package/cde-root/lib/libc.so.7") into the package, then returns control back to the program.

Addressing the impact of CDE on system resources, Philip said it depends on the number of syscalls the program needs to make. Having many files to access means more calls back to CDE and the package. A user can run CDE packages like any other command, so you can provide input or redirect output outside of the package. An example of this is cde-exec Python $CDE-package/var/log/httpd/access_log, which by default looks in $CDE-package/var/log/httpd/access_log first. If this doesn't exist, it then chops the path to look for the absolute path /var/log/httpd/access_log. You can stop CDE from looking for $CDE_package/path using rewrite rules to say ignore anything beginning with /var/log, which will cause CDE to look at the absolute path without prepending $CDE-package first. CDE also includes a streaming mode which allows the user to stream selected apps without installing anything: (1) mount distro (e.g., sshfs) and (2) cde-exec -s (streaming mode) eclipse. This will load from the cloud or server. CDE will cache a local copy into cde-root on the local machine. There have been ~4000 downloads of CDE up till LISA '11. (To find it, search for "cde linux".)

How does CDE deal with environment variables? CDE packages the environment variables first, from the original package build, and loads them as part of the cde-exec process. How does CDE keep the cache synced with the authoritative package source on a server or in the cloud? A checksum system would help this, but he hasn't yet implemented it in CDE. How does it rewrite the paths so that it knows where to go? Does it rewrite LD_LIBRARY_PATH? Philip responded that ptrace rewrites the strings to load the environment and said as an aside, "It's actually a big security hole," which made the audience laugh. How do you address signing? There is currently nothing built in and no verification of content in streaming mode. In capture mode, how do you know you've run the command long enough to capture all of the environment necessary to rerun the command? The solution is pretty low tech: you run some representative runs, do a find, and discover that most apps are well-behaved. You can rsync from run to run to manually copy missing files. How would things like plug-ins impact a package (e.g., eclipse)? Would it change the distributed version in the cloud? You would have to modify your own cached copy, or you could change the cloud copy. Does Philip plan to continue to develop CDE or is he done? Philip doesn't rule out further development—"I'm trying to graduate but maybe after that"—but he welcomed community support to make CDE more robust. CDE was really created to solve a specific problem but could have more broad applicability in the field.

Improving Virtual Appliance Management through Virtual Layered File Systems

Shaya Potter and Jason Nieh, Columbia University

Shaya started by asking some questions to make the audience think about how we manage virtual machines (VMs). How do we provision? How do we update? How do we secure? Virtual machines can increase complexity (sprawl) and this can introduce security issues, giving hackers a place to hide. Traditional package management works for a single system but falls down with many heterogeneous systems. Deploying virtual machines takes time; copying even a 1 GB VM takes a significant amount of time. Copy on Write (COW) disks/file systems are good solutions for provisioning homogeneous machines but are not so good in a complex environment. There are also no management benefits; once you snapshot/clone, you create independent instances, each of which needs to be managed.

Shaya introduced Strata, which models a Virtual Layered File System (VLFS). In reality, machines are administered at the filesystem level with large commonality even among disparate, diverse environments. Strata decomposes the file system into layers that can be shared out to VMs. There are three parts to Strata: layers, a set of file-like packages (e.g., mySQL, Apache, OpenOffice, Gnome); layer repositories, containing layers; and VLFS, which composes layers together into a single namespace.

Shaya's example of a VM used two VLFS layers consisting of a MySQL VLFS and an Apache VLFS. To use Strata in this example, perform the following three steps to build your VM: (1) create template machines;)2) provision machine instances;)3) maintain the VM. Updating the VM is easy. Shaya used the example of a security patch for tar. First you update the layer in the VLFS that contains tar, then the update occurs in all VLFSes that contain tar. All VMs get the update instead of having to update all individual instances of tar. Using union file systems, Strata composes the various layers together to form a single file system for a VM. The various layers have different rw/ro attributes depending on use. There is no copying of VM images, and the admin only

has to monitor a single, centrally maintained template for all VMs. The base layers are read-only (ro) while the user or local sysadmin has access to read-write (rw) layers for local system modification. These rw layers are sandwiched on top of the ro layers (hiding files in the ro layers that occur in both). This allows modification and deletion of files at the VM level. This layering also provides visibility into unauthorized modifications; they appear in rw layers and can easily be cleaned. This also reduces or eliminates the need for tripwire-like programs. Provisioning time is independent of data size. Even large numbers of layers provision quickly. An ssh server may have 12 layers while a desktop machine could have 404 layers. Updating traditional VM appliances takes time, while updates to Strata occur on system startup when VMs grab the updated layers.

A member of the audience asked where the shared layers were stored. Wherever you need, e.g., a SAN or NFS server. Through initrd, the client mounts the SAN/NFS file system, then determines the configuration and unions the layers. This becomes the root file system. Initrd does all of the work to connect the VM to the Strata VLFS. Another audience member asked if the unionfs was accessing the layers on the SAN to provide immediate updates. Shaya explained that updates means creating a new layer which then updates the configuration of the VM. The new layer becomes part of the VM and the old layer is marked as unavailable. Any program that is running currently will continue to run with the old layer, but new programs will not have access to the old layer. There was a concern from the audience that unionfs has had trouble getting into the Linux kernel. Shaya admitted that Stonybrook unionfs has been struggling to get support; there are others, but Shaya didn't know the status of these. Had the authors considered pasting a configuration management system into Strata? A CM system could be used with Strata but it wasn't part of the initial research. Is Strata limited to virtual machines? Could this be used with clustering? Shaya responded that he had presented a paper at USENIX ATC using Strata with a desktop machine. It certainly could be applied elsewhere.

Invited Talks I: Databases

NewSQL vs. NoSQL for New OLTP

Michael Stonebraker, MIT

Summarized by Timothy Nelson (tbnelson@gmail.com)

Michael began by reminding us what online transaction-processing (OLTP) looked like in the past. We bought our plane tickets on the telephone, and a human operator entered the transaction. Thirty transactions per second was normal, and 1000 per second was unbelievably large. Today the human intermediary is no longer around to be a rate-limiter; mobile phones and PDAs originate transactions and submit them directly via the Internet. OLTP systems today need to be able to consume a million transactions per second. More than just entering the transactions, the system must ingest them, validate them, and respond to them, and do so with low downtime. Broadly, there are three options available: "Old" SQL databases, which provide ACID (atomicity, consistency, isolation, and durability); NoSQL databases, which sacrifice ACID for performance; and the "new" SQL approach, which keeps ACID but uses new architecture to gain performance.

Traditional SQL products have problems, one of which is bloat. They are legacy systems, and once a feature has been added, they cannot easily remove it. They are also generic: when used properly, specialized software can outperform traditional SQL products by a factor of 50. Because of this fact, Michael encourages selecting specialized database solutions based on business needs. For data warehousing, column-store databases are 50 times faster than row-store, which is the standard approach. There are similar insights for OLTP. Most OLTP databases are under 1 terabyte in size, which is not an unreasonable amount of main memory.

Michael then discussed where traditional databases spend their time on OLTP transactions. Nearly a quarter of the execution time is used by row-level locking. Similarly, latching (to support multi-threading), crash recovery (maintaining a write-ahead log), and running the buffer pool (maintaining the on-disk store) each takes about 24% of the time. This overhead means that traditional databases are slow for architectural reasons, and we should rearchitect them when we can. Faster B-trees alone will not give us much.

NoSQL is not a cure-all. SQL gets compiled down to the same low-level operations that NoSQL requires, the compiler isn't a big source of overhead, and it's actually hard to beat the compiler these days. Giving up SQL will not give us much performance. Sacrificing ACID can improve performance (e.g., by abandoning row-level locking), but once an application gives up ACID, it's hard to recover if needed later. An application needs ACID if it has multi-record updates or updates for which order matters. For commutative, single-record transactions, NoSQL is appropriate. NoSQL is a great tool, and it is good at lots of things, but we should use the right tool for the right job.

Michael's company has a NewSQL product called VoltDB. VoltDB has no traditional record-level locking. It keeps the database (except for cold data) in main memory. Rather than keep a write-ahead log, it defaults to always failing over, which is what most OLTP applications do anyway. It handles multiple cores without latching, by partitioning

main memory between the cores. VoltDB can do a 200-record transaction in microseconds, and it beats an unnamed traditional SQL database vendor by a factor of 50. The product supports a subset of SQL; correlated subqueries are on the way. Finally, VoltDB is open source.

Is Amazon a good application for NoSQL and eventual consistency? Amazon implemented SimpleDB because, at their scale, they can afford to build their own solutions. The rest of us have to buy third-party engines. If there is no benefit to building faster B-trees, someone from Mozilla asked, should we work on improving latching instead? Better latching would absolutely be an improvement, and we should work on making our data-structures more concurrent, not just faster. Two audience members asked about benchmarking VoltDB in a data warehousing situation. Since warehousing is very different from OLTP, VoltDB would not do well in that space. For data warehousing, one should use a column store instead.

Eric Allman asked about the problem of moving data from OLTP systems to a data warehouse; wasn't it quite expensive? Michael agreed, but he said that you often need to duplicate the data anyway, since the warehouse schema may be different from the OLTP schema. Also, you often want to use different hardware for response-time reasons. So ETL (extract, transform, load) isn't going away, but there may be a market opportunity for seamless integration of separate OLTP and data-warehousing solutions.

What is the revenue model for open source software? RedHat has a good model: provide an enterprise edition with goodies that aren't in the free version, and provide support for the enterprise version. The free community version can be viewed as a vehicle for sales.

Invited Talks II: Newish Technologies

Summarized by Rik Farrow (rik@usenix.org)

Issues and Trends in Reliably Sanitizing Solid State Disks

Michael Wei, University of California, San Diego

Michael presented research from NVSL (Non-Volatile Systems Laboratory), first reported during FAST '11. The takeaway is simple: deleting data on solid state disks (SSDs) doesn't actually remove it. We all have confidential data on disk: private data, corporate and government secrets. We also known how to destroy data on hard disks, as we've been doing this for years with multiple overwrites. But SSDs are different because of the complex controller, the flash translation layer (FTL). The FTL maps page locations to physical pages, and there are always more pages than a device advertises as spares. So techniques used on hard disks fail to work for flash.

Recovery of data on SSDs is cheap: it costs a few hundred dollars. To do the same thing on hard drives, instead of unsoldering a few chips, requires hundreds of thousands of dollars. With SSDs, there is a lot of leftover data, which is easy to recover. It is also possible to recover data at the physical level, even overwritten data. You need to grind flash to destroy it. SSDs write to the first free block, and to overwrite it SSD writes to the next available block, leaving the original data untouched. This is unlike hard drives, where original data can be overwritten.

Their lab examined state-of-the-art mechanisms used in SSDs to remove remnant data. First, they wrote some patterns to SSDs. Then, they "'sanitized"' the SSD, removed flash chips, and used a custom hardware platform for reading chips, bypassing the FTL entirely. If a drive completes its sanitization successfully, there is no stale data left over.

Some drives worked correctly, but some drives didn't apply erase commands properly. They would report success, but data would still be present. Some drives could be mounted as if nothing had happened. Other drives could be reset and some or all data could be recovered. Some drives used a cryptographic scramble, which involves writing all data using some form of encryption and then discarding the key. But since they could not confirm that the key was actually discarded, they didn't trust this approach. Encryption is actually commonly used in SSDs, as it has the effect of randomizing data, which is desirable when writing to flash. For this to work, the key must be durably stored somewhere, and they could not prove that this key had been destroyed after a cryptographic scramble.

They also found that ATA security erase worked sometimes but not at other times *on the same drive*. They suggested combining cryptographic erase and erasing data. First, scramble the disk and delete the keys, then perform a block erase on the entire disk, which then can be verified by reading all blocks and checking for erasure. If done in parallel, a 256 GB disk could be erased in 20 seconds. Perhaps drive manufacturers would implement this.

If you try overwriting, you don't know if you have written to all blocks on the SSD. Their experiment showed that typically two passes were sufficient, but sometimes it took 20 tries. This had a lot to do with past errors on the drive and a lot of things that the FTL hides.

Sanitizing single files is an even bigger problem. You want to get rid of a confidential document, such as browser history or files with credit card data or passwords for customers.

When testing overwriting of a 1 GB file, they tried various standards, and all left at least 10 MB. The NIST standard left almost everything (overwrite once considered sufficient for this standard). Their suggestion for sanitizing single files involves adding a scrubbing command to the FTL. Scrubbing erases all stale data—that is, blocks that are unused but have yet to be erased. But this is not as easy as it seems. Flash is arranged in pages, and these are part of erase blocks. So erasing a block could also erase pages in the block that are still in use. In high reliability memory, they found they could overwrite pages instead of erasing the entire block. But in the more common MLC (a denser, less reliable flash), you have a limited overwrite budget before the write fails. MLC is the type of flash that is cheapest and most common.

They suggest overwriting using the scanning method, which overwrites just the page. If this fails, the entire block must be copied and the block erased (and likely linked to a bad block list as it is now faulty).

In conclusion, Michael pointed out that verification is necessary to prove sanitization effectiveness. Hard drive techniques do not work, and having drive-level support for sanitization is a requirement.

Someone mentioned a study that shows that you cannot recover data from flash disks after they have been microwaved. Someone else asked if the trim command works. Michael said they looked at that, as it appears very interesting. The standard describes "trim" as a hint, meaning the drive doesn't have to implement it, and, in fact, most drives do nothing when given a trim command. Is there any research on recovering data from erased SSDs showing that particular patterns work better? Flash is different from disk, and the way erasure works involves using a higher voltage level that makes distributions of bits impossible to recover.

Aren't some of these issues, such as relocating bad blocks, similar for hard drives? This is definitely an issue for hard drives. The difference is that hard drives don't do this very commonly—perhaps 30 blocks, after a lot of use. SSDs have as much as 10% of their capacity set aside for relocation of bad blocks. Someone else pointed out there has been a lot of work on extreme magnetic analysis on hard disks, where people could go back and see data patterns and infer some of the data. Michael repeated that flash is different.

Apache Traffic Server: More Than Just a Proxy

Leif Hedstrom, GoDaddy

Leif started with the history of traffic servers. Inktomi produced the first commercial version in 1998. Yahoo! acquired Inktomi but ignored the traffic server initially, as all they cared about was search technology. In 2005, Yahoo! began to build a traffic server, and once they had gotten the Inktomi solution under Linux, it was many times faster than Squid. They wanted to open-source their solution, but doing this with existing code required a huge amount of work because of patents. So they started over from source, and in 2010 they achieved this. Nginx is also an awesome proxy server, Varnish is good, and Squid is still around. When you look at traffic servers, what you want to do is comparison shop. Leif presented a table showing features of various traffic servers and suggested you pick the one that fits your tasks the best.

Rather than just consider requests per second, where most traffic servers can do around 100,000 requests (and Apache traffic server does best), you need to look at latency. If you have 2 ms of latency per request and each page consists of 50 requests, that's a total of 100 ms of latency, so latency is important and a better benchmark (again, Apache traffic server does better).

Leif then explained proxy servers. Forward proxy can rewrite URLs which control which Web sites users visit, and cache content. A reverse proxy is transparent to the Web browser and uses rules to redirect, or forbid, access to remote servers. A reverse proxy can also cache content and do keep-alives. The last type of proxy is an intercepting proxy, which Leif called a "mixed bag," where the firewall forces users to go through a proxy. This can be done through an ISP that wants to save bandwidth through caching content. In general, when you think about proxies, you want to take advantage of caching. But the content has to be cacheable.

Caching improves performance for three reasons. The first is the TCP three-way handshake, which multiplies any latency by three before any request can even be made. The second is congestion control. If the server can only send three packets without getting an acknowledgment, and there is high latency, then only three packets can be in flight over that 100 ms. Having the proxy server close to the clients reduces latency to the client. The proxy server can also use keep-alives with remote servers, avoiding the costs of the three-way handshake. When Yahoo! tried a proxy with a keep-alive in India, they got huge performance improvement. The final issue is DNS lookups, which also involve latency. In particular, if you have, say, http://news.example.com and http://finance.example.com, each requires its own DNS lookup. On the other hand, if you use URLs such as http://www.example.com/news and http://www.example.com/finance, only one DNS lookup is required. In this case, you put a proxy near the servers and it can redirect traffic to a particular server (news or finance) as required.

The real problem is when you have millions of users all with powerful machines. Varnish has a great proxy, but you have

to have a single thread for each connection. Threaded code is also difficult to write and to debug. Squid uses event processing, which runs within one process, so there's no locking or potential races. But this doesn't solve the concurrency problem, because it uses only one core. Nginx gets around this by starting multiple processes, but then it has to deal with resource sharing between processes.

Traffic server takes the worse possible route: it does both. Traffic server has multiple threads, one or two per CPU, and runs several different types of threads: resource, event handling, and connection handling. Traffic server has lots less overhead than we see in Varnish. Great coders are required to work with multiple threads.

Configuration looks difficult, because there are many config files, but Leif pointed out that you will only be concerned with storage.conf, remap.config, and records.config. records.config is a normal key-value style, with plenty of comments, used to set flags, for example, and most of the defaults will just work out of the box. You will have to change storage.config since the server cannot figure this out itself. You also need to configure remap.config, especially if you are using the server in a reverse proxy environment.

Apache traffic server can use compression for stored objects, only uses 10 bytes to map to stored objects, and uses raw disk instead of the file system for performance. Traffic server has full IPv6 support on the client side but uses IPv4 on the server side. In the longer term, they want to add SSD to their stack of caches, instead of having just RAM and disk.

In summary, traffic server, and proxy servers in general, are great general-purpose tools. Proxy servers are outrageously fast (except Squid).

Someone asked about two features that chew up resources: content rewriting and header injection. Leif said that headers get rewritten using marshaling buffers. For rewriting content, use a plugin in your data stream. The plugin can inform the proxy if the content should be cached. Plugins are tricky to write, but there are examples.

Clusters and Configuration Control

Summarized by Rik Farrow (rik@usenix.org)

Sequencer: Smart Control of Hardware and Software Components in Clusters (and Beyond)

Pierre Vignéras, Bull, Architect of an Open World

Pierre Vignéras has built a system that shuts off power to systems and devices in an order determined by dependencies. The system was designed to work with one of the largest clusters in the world, Tera-100, composed of more than 4,000 nodes. Nodes can be shut down using commands, such as "ssh root@node halt", while many devices (e.g., switches) can only be shut down by turning off the power, using power distribution units (PDUs), for example. Dependencies include not shutting down a file server before the systems it relies on have shut down, or not turning off network switches before all network-delivered commands have been sent. The sequencer also needs to be fast and robust.

His system has three stages: a Dependency Graph Maker (DGM), an Instruction Sequence Maker (ISM), and an Instruction Sequence Executor (ISE). The input to the system consists of a table with one rule containing a dependency rule per each row. The DGM accepts this as input and creates graphs in the mathematical sense, then prunes the graphs. The output is an XML file with <par> tags indicating which sections can be completed in parallel. In tests on the Tera-100, they could shut down the system in less than nine minutes, compared to more than 30 minutes previously. Vignéras said this is an open source project.

Paul Krizak (AMD) asked what they use for feedback when running ISE. Vignéras said they expected the written code to provide feedback, the way a script provides return codes. Paul then asked how they know that an instruction, such as shutting down a system, has completed. Pierre said that the sequencer assumes each action is atomic, but it may also poll to ensure an action has been completed. Paul asked what type of system they run the sequencer on. Pierre said they use a Bull S6000, a big machine.

Automated Planning for Configuration Changes

Herry Herry, Paul Anderson, and Gerhard Wickler, University of Edinburgh

Awarded Best Student Paper!

Paul Anderson ran a slideshow presentation, as Herry was defeated by visa issues and wasn't able to enter the US. Herry (in a voice-over) began by pointing out that most common configuration management tools have a declarative approach, and that poses a critical shortcoming. Declarative tools imply an indeterminate order of execution which may violate a system's constraints. He illustrated this with an example of wanting to switch a client node C from server node A to server node B, then shutting down server A. If the configuration management tool executes commands out of order, shutting down server A before client C has a chance to mount a file system from Server B, there will be trouble.

They developed a prototype that uses IBM's ControlTier and Puppet, where ControlTier schedules changes and Puppet implements them. Their system works by collecting existing system state, translating it into a Planning Domain Defini-

tion Language (PDDL), having an administrator specify a declarative goal state, a planner generate a plan, and a mapper generate a ControlTier workflow, which sends Puppet manifest files that complete the work in the correct order. They tested their prototype by migrating an application from a private to a public cloud to address spikes in demand, with the constraints that there be no downtime, that the firewall be reconfigured, and that this be a true migration, not duplication.

Paul Krizak asked about the tools that make up the library, and Paul Anderson replied that their prototype uses a combination of various tools that are glued together: the standard planner, Puppet, and lots of glue. Someone else pointed out that they were pushing work on people to write the actions and prerequisites in order to produce the output. Paul responded that someone has to define actions for Puppet, and that the additional work to create prerequisites can be done over time. Norman Wilson pointed out that getting used to their system would not be that different from an earlier Sun system he had used, and Paul agreed. Paul also mentioned that sysadmins can forget things, and writing it down helps. Norman replied that he had never seen a system where he couldn't screw up something by leaving out a step.

Fine-grained Access-control for the Puppet Configuration Language

Bart Vanbrabant, Joris Peeraer, and Wouter Joosen, DistriNet, K.U. Leuven

Bart Vanbrabant pointed out that configuration management replaces needing to have root access on each machine. You manage configuration files on a central server, which performs root actions. This implies that if an attacker can insert malicious actions into the configuration, the attacker can affect all systems without being root. Their goal is to add access control to configuration management, but this is difficult to do, as there is no one-to-one mapping from configuration files to actions.

This work builds on ACHEL, a tool presented at LISA '09 (Bart Vanbrabant et al., "Federated Access Control and Workflow Enforcement in Systems Configuration") that they built to prove that this works with a real configuration management tool, Puppet. They use Puppet's compiler to produce an abstract syntax tree (AST), then normalize the tree before deriving the actions to be authorized. The normalization is necessary because there are multiple ways to do things such as user account creation. They use the XACML policy engine to perform the checking. Bart said that their tool does not support some Puppet constructs, such as switch cases that are handled at runtime. But their tool does work.

Paul Krizak asked if they had considered performing access control at runtime, letting changes go into the system, but refusing forbidden things at runtime? Bart said that it was possible, but with constraint languages it is almost impossible to determine which rules generated a forbidden action. Paul Anderson followed up by asking if they store information in the configuration about who made changes, and Bart agreed, but said that they had that ability in ACHEL and here they just wanted to test the tool. Anderson asked if they just look at changes to configuration, and Bart replied that they use the entire file, create the AST, and determine what changes have been made. Bart said you can do anything in XML if you are willing to write huge amounts of XML.

Invited Talks I: DevOps: Chef

GameDay: Creating Resiliency Through Destruction

Jesse Robbins, Opscode, LLC

Converting the Ad-Hoc Configuration of a Heterogeneous Environment to a CFM, or, How I Learned to Stop Worrying and Love the Chef

Dimitri Aivaliotis, EveryWare AG

No reports are available for these talks.

Invited Talks II: Panel

Summarized by Michael Wei (mwei@cs.ucsd.edu)

How University Programs Prepare the Next Generation of Sysadmins

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

Panelists: Kyrre Begnum, Oslo and Akershus University College of Applied Sciences; Andrew Seely, University of Maryland University College; Steve VanDevender, University of Oregon; Guy Hembroff, Michigan Technological University

The main topic of discussion was how the university, which typically provides a rigorous formal education, could provide the kind of practical problem-solving skills that are necessary for effective system administrators. Each panelist brought with them the experience they had in training system administrators and the challenges they faced.

The first part of the panel covered the challenges of teaching and evaluating system administrators. In the words of Carolyn Rowland, "There's more than one way to skin a cat, and we all have the right answer." Kyrre Begnum agreed—sometimes a class can be about a specific way to solve a problem, but the student may have inherent knowledge on how to solve that problem in another way. However, the exam may test

only how well a student solves a problem in that particular way. As a result, a student who comes in with the inherent knowledge may actually be at a comparative disadvantage, and it may take time to get that student to accept that there are several ways to solve problems.

Andrew Seely felt that problem-solving in system administration is not a thing that can be taught. In the world of computer science, you can teach a student to build classes and data structures, and they can apply that knowledge to build a program. In the world of system administration, it is not clear what those basic building blocks are. Carolyn asked the rest of the panel whether they agreed. Kyrre Begnum and Guy Hembroff both disagreed. They gave the example of scripting and network infrastructure courses as similar building blocks that they teach at their universities, and felt that the problem was more that the literature for teaching these skills is limited and not necessarily practical.

The next section of the panel talked about the missing identity of system administration. Steve VanDevender pointed out that there was no standard that defined what a system administrator is, so attempting to figure out what skills are necessary to train good system administrators may be a futile question to ask until the identity of system administrators has been hammered out. Kyrre agreed, further pointing out that the lack of textbooks suitable for classroom use on system administration means that what system administration programs teach ends up being a hodgepodge of what the program administrators feel to be relevant at the time.

The panel ended with the difference between technical and theoretical education. There was general consensus that a good education in system administration would require both. Andrew Seely finished by explaining that other disciplines have fundamental theories, while system administration is composed mainly of best practices. If best practices could be changed so they did not expire with every new technology, then system administration could be furthered as a discipline with both a theoretical and a practical grounding on its own.

Security 1

Summarized by Timothy Nelson (tbnelson@gmail.com)

Tiqr: A Novel Take on Two-Factor Authentication

Roland M. van Rijswijk and Joost van Dijk, SURFnet BV

Roland van Rijswijk began by reminding us of a painful fact: we enter many username-password pairs throughout our lives. Entering a username and password remains the standard login paradigm, in spite of its risks: password re-use, for instance. Two-factor authentication is the act of logging in with two separate credentials, often something that you "have" plus something that you "know" or "are." For instance: a password and a one-time-use code, or a keycard and a fingerprint.

This work, tiqr, exploits the fact that nearly everyone has a mobile phone, and those who have them tend to carry them around. Even more, we tend to notice if they go missing, which is more than we can say for authentication tokens. However, most phone-based authentication solutions make their user retype complicated codes that appear on the phone screen. That requirement is an issue, especially for visually impaired users. tiqr makes progress in that direction.

Since the room had two separate projection screens, we were treated to a demo of tiqr. The screen on the left showed a Web browser, and the screen on the right showed an iPad display. Van Rijswijk showed tiqr scanning a QR code that appeared on the Web browser to generate an authentication response, which the iPad then transparently sent to the Web site via its own Internet connection (protected by 256-bit AES encryption). The speaker closed by showing us how tiqr can facilitate SSH login using an ASCII-art QR-code, and he briefly discussed an external security audit of tiqr. The external auditors said that tiqr's security was more than adequate.

What would happen if the user is already using the Internet on their mobile device? The QR codes contain a URL schema, and users merely have to click the code to switch apps. What about tiqr's API? They have a Web API, a demo version of which is available in PHP. They are working on making the API fully RESTful. What happens if the mobile device has no Internet access? In that case, tiqr will fall back to classic mode, giving the user a code to type into their browser manually.

Building Useful Security Infrastructure for Free (Practice & Experience Report)

Brad Lhotsky, National Institutes on Health, National Institute on Aging, Intramural Research Program

Brad Lhotsky, a recovering Perl programmer, is a Security Engineer at the Institute on Aging. Nobody likes getting "the call" from him, and his boss doesn't care about security. Lhotsky has found that security people need to justify their existence beyond saying that they make things more secure—to the organization, security is a minor concern compared to actually doing research. Their paper is about tools their team has built for useful security.

They do comprehensive, centralized logging of network events. They used to use syslog-ng, but some of its features

are not free. They now use rsyslog, which supports native encryption (among other things), but has a somewhat ugly configuration. They use postgreSQL for long-term storage of the events. Keeping logs in a RDBMS makes using the data easier. For instance, they can execute R (a language for statistical analysis) queries on the data.

This detailed logging lets them do correlation of data and provide it in a way that is helpful to their help desk operators. They answer questions such as "Where has a user logged in from?" "What devices has a user been assigned?" "Where is the user logged in now?" and "What is their network history?" They also feed the logs to Snort, an intrusion detection system, to discover potential data-loss risks.

An audience member noted that building infrastructure is one thing, but convincing auditors is another. They asked whether Lhotsky's team is externally audited. Lhotsky answered that they are indeed audited by health and human services.

Local System Security via SSHD Instrumentation

Scott Campbell, National Energy Research Scientific Computing Center, Lawrence Berkeley National Lab

Awarded Best Paper!

Scott Campbell's organization supports 4000 users worldwide, all of whom have shell access. NERSC doesn't want to interfere with their users' work, but does want to intercept intruders on their machines. They modified their SSH service to log sessions, authentication, and metadata, which involved identifying key points in the OpenSSH code.

To filter the raw data, they use the Bro intrusion-detection program, which is fed data from the captured SSH messages. Multiple data sources are mapped to one log file, which is then converted to a stream of Bro events. All the analysis is done in Bro, which looks at each event atomically to find what is considered hostile or insecure. For instance, they might want to detect someone remotely executing "sh -i". Their SSH modifications also allow them to collect soft data, listening in on intruders to learn their methods and skill levels. They actually captured a discussion between two intruders on their system.

Campbell's team plans to extend the work they have already done, to perform better analysis. For instance, they could analyze user behavior or pass more information (such as process accounting data) to Bro.

Someone asked about arbitrary whitespace, that is, what happens if an attacker uses "sh -i" (with extra spaces) instead of "sh -i"? Campbell answered that they do not currently catch that, but, with minor modifications, they could. Rik Farrow asked whether they capture passwords. Campbell replied that they can, but they try very hard not to. It is easier to capture passwords than not capture them, but they do their best to preserve their users' privacy.

Someone else asked what the false positive rate is. Campbell replied that his team is paged very rarely. Had they actually captured any intruders? While they have not captured anyone in the physical sense, they do intercept intruders 12 to 20 times per year. The full paper even contains some example conversations between intruders. Do they capture everything going on, because that would be a lot of traffic to log? They try to reduce unnecessary logging, stopping recording replies from the server after a period of user inactivity.

Invited Talks I: DevOps Case

The Operational Impact of Continuous Deployment

Avleen Vig, Etsy, Inc.

DevOps: The past and future are here. It's just not evenly distributed (yet).

Kris Buytaert, Inuits

No reports are available for this session.

Invited Talks II: Infrastructure Best Practices

3 Myths and 3 Challenges to Bring System Administration out of the Dark Ages

Mark Burgess, CFEngine

Summarized by Michael Wei (mwei@cs.ucsd.edu)

Mark Burgess began by describing the dark ages, a time when brute force was essentially the way problems were solved. In system administration, brute force is still widely used in many areas. For example, technicians are often attached to service tickets and thrown at problems wherever they show up. Mark proposed that there are three waves of innovation in system administration: the manual, brute force wave of the past; the automated wave that we are starting to move into; and the knowledge wave, where we build machines to serve us rather than just complete a singular task. Today we are moving from the second wave to the third wave, but we still face second wave myths that keep us tied to the first and second waves and third wave challenges that keep us from moving into the third wave.

Mark challenged three myths. The first is ordered sequential control. He believes that we are taught that sequentially is better, when it is not always the best solution. In fact, we sometimes artificially create sequentiality when no sequence necessarily exists. Mark gave an ordered XML document as

an example. The second myth Mark challenged was that of determinism and rollback. Mark said there is a belief that we can simply roll back changes to fix problems, but in reality this is a myth. If we accidently change our firewall configuration and a computer gets infected by a virus, we cannot simply roll back the firewall configuration; we need to repair the computer as well. This kind of rollback thinking is dangerous because it forces us to focus on the mistake instead of the outcome. The third myth Mark examined was "hierarchy or bust." The kind of hierarchical thinking that many system administrators thrive on creates many points of failure which are dangerous, while marginalizing problems.

After enumerating these three myths, Mark presented three challenges. The first is emergent complexity: we have to be able to accept that systems are becoming increasingly complex and accept diverse systems as they are. The second challenge is commerce alignment: we have to accept commerce's role in IT if we are to allow systems to grow. The final challenge Mark presented is knowledge management. We must be able to share knowledge efficiently, because individual skill is not replaceable.

Linux Systems Capacity Planning: Beyond RRD and top

Rodrigo Campos

No report is available for this talk.

Plenary Session

One Size Does Not Fit All in DB Systems

Andy Palmer, Global Head of Software and Data Engineering, Novartis Institute for Biomedical Research

Summarizedy by Ming Chow (mchow@cs.tufts.edu)

Andy Palmer talked about the changes and challenges of database fitting for biomedical research and scientific applications. The game has changed: the idea of one database engine (e.g., Oracle) fitting all research and scientific applications is no longer applicable. The reason for this is big data. Over the years, the amount of data and information has grown exponentially compared to storage of information, which is near flatline. The traditional DBMS architecture is roughly 25 years old and was designed mostly for write-based applications. The architecture has largely ignored CPU, disk, and memory trends. In short, while the RDBMS market became a billion-dollar market, traditional DBMS architecture has not caught up with the times. Vendors have addressed the issues by adding band-aids to their products such as bitmap indices.

The idea of OLTP (Online Transaction Processing) ran out of steam in the 2000s with the introduction of Big Table and Hadoop. Since then, there have been new database engines such as Mango, Couch, and Cassandra. Andy asked the question, "How do you pick the right engine for a specific workload?" He said we need frameworks and tools to characterize workloads to match engines based on empirical characteristics. This paradigm differs from "Oracle or MySQL is the answer for everything": now you need to stop and pick the right engine before starting a project. For scientific data, there are many options, including the traditional row store engine, column store, file-oriented, document-oriented, array-oriented, and federated.

Working with big data in a research and scientific environment is also much different from before: operations of new engines require integrated skill sets, including database, system administration, and clustering. Andy described the Novartis IT Data Engineering teams, which do hands-on application of new database technologies. The teams consist of an interesting mix of people who work at several different locations and frequently cross-train.

A significant issue with working with big data in a scientific and research environment is that the cost of fixing errors (e.g., quality) has grown exponentially. Andy stressed that data quality starts at the point of data creation. For an application, 70% of the work is creating or working with unstructured and structured data.

Andy noted a few myths, including "Oracle is the answer to everything," "one database for each app," and "one integrated database for everything." Scaling has been a huge challenge in working with petabytes of data. Andy noted that at Novartis, they have run out of space over 10 times this year alone. Unfortunately, given the importance of scaling, he has found it impossible to solve data challenges with Oracle. Working with big data involves numerous schema changes, and new data sets need loading and analyzing, which can be problematic in Oracle. At Novartis, a number of new database engines have been tested, including Vertica for gene expression, MapReduce, and CouchDB.

In-house, a few big questions have been asked as part of the framework to determine which is the best database engine to use in certain applications. The questions include the best performers, solubility, scalability, performance, and ease of adoption and integration. Andy and Novartis have taken a quantitative approach to answering these questions when working with new database engines.

Andy concluded the plenary by stressing the importance of considering up-front which database engine matches workload and that the operations of new engines require integrated skill sets.

From Small Migration to Big Iron

Summarized by Carolyn Rowland (carolyn@twinight.org)

Deploying IPv6 in the Google Enterprise Network: Lessons Learned (Practice & Experience Report)

Haythum Babiker, Irena Nikolova, and Kiran Kumar Chittimaneni, Google

Awarded Best Practice & Experience Report!

Irena's first slide summed up the move to IPv6: 96 more bits, no magic. She spoke of the IANA IPv4 exhaustion in February 2011 and how we are not reclaiming any of the current IPv4 address space. Sometime between 2012 and 2015 we will have assigned all IPv4 addresses. With the proliferation of smartphones, IPTV, virtualization and cloud, P2P, and network-aware devices, we are only increasing the demand for addresses. Google was motivated internally because they were running out of RFC 1918 addresses. They tried using NAT as a solution, but this just increased the complexity of the environment. Additionally, they thought that by implementing support for IPv6, they would help us break out of the chicken-or-egg problem: service providers in no hurry to deploy IPv6 due to lack of content, and content providers explaining the slow rate of adoption as being due to lack of IPv6 access to end users. To make a positive contribution to the Internet community, they knew they had to enable IPv6 access for Google engineers, to help them launch IPv6-ready products and services. Google has an internal process they call dogfooding—living the same experience as their users (eating your own dogfood). Since their internal teams wanted to work on IPv6 connectivity for Gmail and YouTube, etc., the networking team had a real need to provide them with a network that was IPv6-ready, so that they could help develop, test, and dogfood these products.

The Google approach was: IPv6 everywhere! They needed to build tools, test and certify code for various platforms, decide on routing protocols and policies, plan for IPv6 transit (WAN) connectivity, create a comprehensive addressing strategy, and request IPv6 address space. They began with dual-stack, using tunneling as a last resort. They assigned IPv6 address space as /48 to each campus, /56 at the building level, further dividing into /64 per VLAN. GRE tunnels were not a good option, because they created MTU fragmentation and other issues. This was pretty challenging, because lots of ISPs don't yet support IPv6. They tried DS-Lite to encapsulate IPv4 packets inside IPv6 networks. They also used SLAAC (stateless address autoconfiguration) instead of DHCPv6 for host address assignment. Sometimes they were the QA department for the vendors selling IPv6 products. The vendors were not eating their own dogfood. "Nothing could create a real-world scenario in the lab, so we discovered many IPv6-related bugs during deployment and testing." Also, IPv6 was still processed in software in many vendors' hardware platforms. Internally, training and education were the biggest challenges, although early information helped to fight FUD. They had some DevOps challenges as well; the engineers wanted to deploy new technology immediately, while the operations people were not so interested in early adoption. Words of wisdom: migration is not a Layer 3 problem, it is a Layer 7–9 problem. Migration is simple, but it takes time. A phased approach gradually builds skills and confidence. You want to make sure you design for the same quality standards as with IPv4.

Has Google worked with any VOIP technology? The Google Network Engineering Team is only responsible for migrating networks. Rik Farrow asked how organizations can allocate resources for an IPv6 migration. Irena admitted that most resource allocation is for IPv4 projects. If you have eight projects and two are IPv6 and six are IPv4, people are going to go for the v4 projects, because v6 is not a priority. An audience member said that now it's up to some other people to produce the other half of the equation. If you build it, they will come. So, are others starting to build stuff with IPv6? A lot of the Google external products are IPv6-enabled now. You can go to http://www.google.com/intl/en/ipv6/ to request access to Google products over IPv6.

Bringing Up Cielo: Experiences with a Cray XE6 System, or, Getting Started with Your New 140k Processor System (Practice & Experience Report)

Cory Lueninghoener, Daryl Grunau, Timothy Harrington, Kathleen Kelly, and Quellyn Snead, Los Alamos National Laboratory

Cory started his presentation with the obligatory HPC statistics slide. Cielo consists of 96 racks, 96 nodes per rack; two 8-core 2.4 GHz processors per node; 32 GB memory per node; a Torus network: 4.68 GB/s links; 142,304 cores; 284,608 GB total compute memory; and 1.11 PFlops measured speed. Cielo is currently used for simulations at Lawrence Livermore National Laboratory (LLNL), Sandia National Laboratory, and Los Alamos National Laboratory (LANL). This project was a collaboration between Los Alamos and Sandia (but lives in Los Alamos). Other computers are used to support the management of Cielo. Cielito is used for testing before scaling up to the big system; Smog is used to try out new Cray software stack releases, configuration management, and other challenges of bringing up the big system; and Muzia is Sandia's one rack system (similar to Smog).

Cielo can be used as one big system to run one job or can be used to run smaller jobs. They use CfEngine for configuration management; this adds a layer of abstraction to make the nodes all seem the same. Cray provided monolithic install

scripts, hard-coded RPMs, and versions. Cory said his team asked lots of questions to try to understand the Cray configuration tools (e.g., one of the default install scripts checked whether it was being run on an interactive terminal, so you couldn't automate the install with | yes). Cray used a couple of different schemes, such as /etc/alternative (Debian), module files (more flexible and dynamic), and default links (links within the software tree to different versions of software). The question Cory's team asked was, "Do we hack the scripts so we can make it easier to automate?"

They began tackling these challenges with some CfEngine hackery. This got them past 80% of the challenges. They were able to tweak rules used on other production systems for the Cray. They resorted to outside scripts for the parts CfEngine didn't handle. Still, CfEngine could run these scripts keeping some form of automation. Sysadmins could still work with module files and changes would trickle down. Cory said they were basically telling Cray "your way is wrong," which isn't the message they wanted to send. A lot of people use Cray's tools just as intended, but Cory's team decided to do it a different way. Despite this deviation from Cray's standard strategy, Cray was extremely helpful. Cory's team submitted bug reports, Cray was helpful with the weird stuff, "and there were rainbows everywhere," said Cory. Maintaining positive vendor relations helped them. Cory admitted that the team needed Cray's support to do what they needed to do; without them, the team would still be fighting about the answers. Getting early access to test systems (Smog and Cielito) allowed the team to solve problems early before moving to the big system. Configuration management is a worthwhile investment. The team was able to rebuild the whole system in a day or two instead of weeks. They did this several times when they were reviewing security.

One audience member pointed out that Cory had provided a long list of things Cray did wrong. What did Cray give them? Cray gave them a big integrated system that was able to run jobs across the whole thing very quickly, and an underlying control system to control the booting, the management, reliability, and serviceability. They provided a lot of the underlying pieces. How long did it take from first power-on to production use of the system? Cory estimated 3–5 months. It wasn't something they tracked; there was a lot of shaking out of the system. Did Cory know the cooling number per rack? Cory looked to his teammates in the audience and someone responded, "5 or 6 MWs total for the whole machine."

Capacity Forecasting in a Backup Storage Environment (Practice & Experience Report)

Mark Chamness, EMC

Mark pointed out that IT behavior is reactive. We react to 100% disk capacity, failed backups, and late-night alerts. In those reactionary efforts, we take shortcuts such as deleting files to make space on a volume or decreasing the retention policy on backups so that we can hit our window. The solution to this is proactive prediction. First, start by choosing a time frame (e.g., the past 30 days). Next apply linear regression (e.g., over the next 90 days), then choose the time frame for notification (e.g., the next 90 days). Finally, run your analysis and generate a report. However, using a fixed time frame results in poor predictions and doesn't adapt for behavior. They tried two time frames (60 days and 7 days) and picked the best one. This also failed miserably, because both were wrong. The optimal prediction occurs when you use all possible models and choose the best one, selecting the maximum value of R(squared) (regression sum of squares). The maximum R(squared) occurs at the change in linear behavior. This is the best model to fit the data and to use for forecasting.

Mark showed some example graphs of different scenarios. This painted a clear picture so that you could predict the date of full capacity. The majority of systems were able to be modeled using linear regression. Mark used an example of a system that was at 60% for a long time and then started to grow. The graph then displayed a rollercoaster behavior, where system capacity went up and down. The schizophrenic graph, where capacity varied all over the place, did not work so well with linear regression as a predictor. In order for a model to work, one needs the following: goodness of fit r^^2 > 0.90; positive slope; forecast time frame < 10 years; sufficient statistics (15 days); and space utilization minimum of 10%. Mark raised the gotcha that the last data point trumps all. You can model a system fairly accurately using this method, until the sysadmin deletes a bunch of data to create free space. Then the system is no longer behaving in a predictable way. You can no longer predict behavior if the error is too great between the last data point and previous data.

Were there models that could have been used other than linear models (e.g., logistic regression)? If you attempt to use a more complicated model and show it to sysadmins or customers, it becomes too difficult to explain. More complicated models also tended to behave erratically; they would often go off exponentially and produce some strange predictions.

Why did Mark decide to model the percentage of capacity instead of rate of growth? Linear regression provided growth in GB/day: "Here's how much it's growing per day; here's when it is forecasted to reach capacity." He had the

ability to capture that data to model it. Did Mark consider daily spikes? No, the model does not monitor daily spikes, but his group uses Nagios to alert for 90% capacity, etc. On what percentage of systems does this model work? He could accurately model 60% of his systems. Did Mark see this as a tool running all the time that you push out to a consumer or is it a sizing exercise for presales (at EMC)? All of the above. When people buy a system they get support; they can go to the customer support portal and configure alerts. EMC will soon be releasing a new product called DD Management Station. It will allow a customer to manage 60 Data Domain devices. The last questioner said he didn't see any control for human intervention. Mark reiterated his "Last data point trumps all slides" statement. This is the example where the sysadmin went in and deleted a bunch of data. The model would not publish a prediction because of the error between previous data and the last data point. It's important for a model to know when it doesn't know.

Content-aware Load Balancing for Distributed Backup

Fred Douglis and Deepti Bhardwaj, EMC; Hangwei Qian, Case Western Reserve University; Philip Shilane, EMC

Fred started with a primer on deduplication: basically, you can avoid storing data at all if it already exists on backup. Using hashes, the system can check for changes; if there are none, then the system won't transfer any data. Deduplication is common in today's backup products. But what is the impact of deduplication? You desire affinity, sending the same client to the same appliance so it will deduplicate well. If you send a client to a new system, then all of the data will be transferred, because there is no sameness. You can also send similar systems to the same appliance, because they will deduplicate well, due to similarities in the data. Doing this can reduce capacity requirements and can improve performance of backups, because you copy less data.

For performance reasons, you don't want to send too much data to one place, so ensure you support simultaneous backup streams. One gotcha is not sending everything to the same appliance just because it deduplicates well. When sizing a system for load-balanced distributed backup, look for cases of affinity with high overlap among a small number of hosts. Virtual machines are a good example of good overlap. If you have a lot of similar hosts, then you can probably put them anywhere and they will deduplicate well. There are small penalties for migration, but the biggest penalties come when a client doesn't fit on backup at all.

Fred posed the question, "What are some algorithms for load balancing?" Fred's team started with brute force; this works okay if the system isn't too loaded. The first approach is "random." They started at a random appliance and used it as long as the appliance wasn't overloaded. If it had no storage capacity, they searched round-robin from there to find one with capacity. They took the best of 10 random configurations, using a cost metric that considered capacity, throughput, client movement, and other factors. After that, they tried bin packing: assigning an appliance based on size from large to small. Next, they tried simulated annealing, which starts with the bin-packing configuration but then iteratively adjusts the configuration to try to find a better one. This model is willing to temporarily move through a worse configuration to try to avoid local minima. To evaluate the algorithms, they used a synthetic workload with clients added incrementally. Appliances were occasionally added as well, and 1/3 of the existing assignments were dropped each time an appliance was added, to avoid always starting with the older appliances already overloaded. "Make sure to stress overlap affinity." Cost was a first attempt at weighing the relative impact of different factors, but it really has to be evaluated in practice. All of the cases have a high cost, but simulated annealing was the best of the worst. Fred referred back to the paper for more information on overlap computation, more examples, overhead analysis of simulated annealing, how to penalize things for not fitting, and impact of content awareness.

Matt Carpenter asked if Fred's team looked at any analytics other than simulating annealing. Fred responded that the team started with bin packing. They wanted to understand the right way to deal with the least movement while still getting the best result. They admitted that there may be other heuristics. Had they considered network latency issues in terms of the cost, assuming flat space/local with no cost to move data? Fred said that there was no assumption that there would be a greater cost to move from one appliance to another. They assumed the same cost per appliance. There is a moderately high cost if you miss the backup window, because moving a client to a different server means recopying data. This is still considered a moderate penalty compared to not being able to perform the backup at all. Someone commented that one of the issues is to understand exactly what "cost" means. Fred replied that it would have to be tuned for a particular environment. Rik Farrow asked if Fred was looking at the size of the content, not the actual content of the data. Fred said that they were looking at the size of blobs to see how full an appliance was and at fingerprints of the content to decide if a chunk of data on one client matched a chunk on another client. He also said that you may need to run something that scurries through the file system and creates the fingerprints. If you are using Networker and it writes one big file for a client, comparing the fingerprints that correspond to that file would mean never having to go to the raw file. Matt Carpenter asked about skipping over

full devices: he said there might be value to moving stuff off a full device, because the remainder might deduplicate well with the data you need to store. Fred replied that this is what simulated annealing might do, by moving one client away from an appliance, then putting better data on the one that just freed up space.

Invited Talks I: DevOps: Core

SRE@Google: Thousands of DevOps Since 2004

Thomas A. Limoncelli, Google NYC

Summarized by Ming Chow (mchow@cs.tufts.edu)

Tom Limoncelli started his DevOps talk by revisiting the '80s, when there was no monoculture and the software engineering methodology was based on the waterfall method. Back in those days, developers didn't care about operational matters after shipment, software developers were not involved with operations, no bug tracking system was necessary, and new software releases were far apart. Then came the '90s, with the Web and the modem. Software moved to the Web, but servers required producers and operators. The users used Netscape and Internet Explorer. Despite the new software development shift, the waterfall method still kind of worked. With the 2000s, everything changed: speed mattered, there was pressure to be first to market, there was feature one-upmanship, shipping constantly to compete, and reliability mattered (which was also a selling point). The major shift also caused tensions between developers and operators, where developers stress changes while operators stress stability.

Tom introduced how DevOps works at Google. The goal is to improve interaction between developers and operators. He introduced the Google Site Reliability Engineer (SRE) job role. The model is based on extreme reliability, high velocity, and high rate of change. In a nutshell, developers run their own service, and SREs create tools and services. This creates a workforce multiplier effect. Tom described the process of product creation at Google. First there is creation, followed by a live readiness review, then a launch to production, followed by a handoff readiness review, considered only after developers have run the product or tool for six months. Products and services that receive SRE support are those that have low-operation burden or high importance (e.g., Gmail), and those that address a regulatory requirement.

Tom also described the product handoff process at Google. The process involves reviewing frequency of pages and alerts, bug counts, maturity of monitoring infrastructure, and production hygiene. If a product does not do well in some of these areas, then it goes back to the drawing board. Tom also mentioned that products that are approved the fastest for handoff are those that worked with SREs early.

In order to make DevOps work at an organization like Google, Tom listed management support and balance of power between SREs and developers. Skills required include deep understanding of engineering issues (system administration, software design). An SRE role is half engineering and half operations. Tom also presented how development life is made easier at Google with tools, frameworks, monitoring, and plenty of information resources, including training, launch checklist, and mailing lists.

Tom went on to describe the release engineering policy at Google. How it works is based on the idea of the canary in the coal mine. A product or service is first run on a test cluster. Then it is run on some percentage of machines in a cluster. Product teams are given a reliability budget (tokens). New pushes are based on budget. Tom has found that more pushes during this phase equates to more rollbacks. The reliability budget for release engineering is numbers-based and gives incentives for developers to test harder.

In summary, Tom said that the model has worked well at Google. First, there is joint responsibility even after adoption of a product or service. For SREs, developers are committed to fixing issues so they will not be supporting junk, and it gives developers production experience. It has even removed the adversarial quality of a lot of relationships. Tom advised those in the audience that in order to have this model in their organization, developers must be empowered, practices must be adopted, and there must be management support.

Deployinator: Being Stupid to Be Smart

Erik Kastner and John Goulah, Etsy, Inc.

Erik Kastner started with an overview of Etsy. Etsy receives over one billion page views per month, has approximately 100 engineers, values speed and agility, and tries to limit barriers. The company believes in turning ideas into code within minutes, in open source software, that optimizing now leads to happiness, and that sad engineers are bad engineers. The development process at Etsy is an embedded reaction to stupidity; there is no fear, they don't aim for perfection, and correctness the first time is not important. In fact, the idea is to be wrong as fast as possible. Etsy accomplishes this by good communication, trust, openness, and constant improvement. In 2009, there was a single deploy master, developers rolled back in fear, and all deploys took all day. Currently, anyone at Etsy can deploy, there are no rollbacks, and developers deploy all day. John went on to describe the culture at Etsy. Doing the dumbest thing possible lets you learn as much as possible, such as committing to trunk, putting configuration into code, and having blameless post-mortems. Etsy feels that what

makes this works is to graph everything. John displayed a graph on memcached and change-related incidents: there were six in 2010.

Erik Kastner then described the Deployinator tool, which is released on http://etsy.githib.com. The tool is based on the Staples easy button, for building scalable Web sites and for deployment and monitoring. The monitoring is done via a dashboard, and practically everything is graphed. The Deployinator uses a combination of technologies, including SSH, rsync, and a number of Web servers that are synced to the deploy host. Erik then discussed how Deployinator can be used for iOS: it has been used successfully in that iOS code was deployed to a Mac Mini, then to TestFlight. Finally, Erik challenged the audience, "What's stopping you?" Just know what you're optimizing for.

Invited Talks II

Summarized by Deborah Wazir (dwazir@gmail.com)

Fork Yeah! The Rise and Development of illumos

Bryan M. Cantrill, Joyent

Bryan Cantrill gave a lively presentation on the history and current status of the illumos operating system. Beginning with the transition at Sun from SunOS 4.x to Solaris, Cantrill described the struggles to develop a version of Solaris that worked well, which were hindered when management was in charge, but was finally accomplished once the engineers took over. At this point, radical engineering innovation ensued, with features such as ZFS, DTrace, and Zones invented because engineers felt they should be part of the OS. Once Sun open-sourced the OS, the OpenSolaris community flourished, until Sun became too hands-on. When Oracle bought Sun, introducing a totally alien organizational philosophy and mission, several key engineers left. These were followed by many more, after OpenSolaris was effectively killed by Oracle's internal decision to stop releasing Solaris source code.

Meanwhile, illumos development had begun, with illumos meant to be an entirely open downstream repository of OpenSolaris. After hearing Cantrill describe the engineer-led culture of innovation at Sun, it became easy to understand the exodus of engineers after Oracle took over, and why illumos represents a continuation of that same culture. The project greatly benefitted from the participation of engineering talent that had left Sun. Cantrill emphasized that illumos provides innovations and bug fixes that will *never* become part of Oracle Solaris. At this point, he brought up a terminal window and demonstrated some of the new features and enhancements.

After the demo, Cantrill discussed numerous specific improvements added to ZFS, DTrace, and Zones, and new features such as the port of KVM from Linux to illumos. He also named and thanked the individual engineers who worked on these features. He went on to discuss different illumos distributions that are already available, which complement each other by addressing different deployment environments: desktop, server, and cloud.

Cantrill branched off from his prepared slides from time to time, sharing his personal experiences and interpretation of Sun's engineering culture, the Oracle acquisition, and Oracle leadership. These digressions received enthusiastic appreciation from the audience.

Afterwards, Cantrill was asked about the state of illumos's collaboration with other distros regarding DTrace. He replied that the most complete port of DTrace is on the Mac, with the FreeBSD implementation not quite as far along. illumos is collaborating with other distros as well. Another audience member asked if recent innovations to DTrace would flow to Apple, and Cantrill said that he expected Apple to take all of them.

GPFS Native RAID for 100,000-Disk Petascale Systems

Veera Deenadhayalan, IBM Almaden Research Center

Veera Deenadhayalan presented a feature recently added to the GPFS parallel file system, GPFS Native RAID. The presentation slide deck contains excellent diagrams that clearly illustrate all the concepts discussed. He began by explaining that disk drive performance has not kept pace with improvements to other components such as CPU and memory. Therefore, to produce comparable increases in system performance, many more disks have to be used.

Before describing GPFS Native RAID, he covered some background concepts and characteristics of the GPFS parallel file system. He also discussed challenges to traditional RAID, which leads to performance and integrity problems in 100,000-disk petascale systems. When you have that many disks, disk drive failures are expected to happen daily, resulting in performance degradation when the disks are rebuilt. There is also more incidence of integrity issues.

IBM's solution to these issues is to remove the external RAID controller and put native RAID in the file system. This results in higher performance through the use of declustered RAID to minimize performance degradation during disk rebuilds. Extreme data integrity comes from using end-to-end checksums and version numbers to detect, locate, and correct silent disk corruption.

The physical setup is an enclosure with 384 disks, grouped four disks to a carrier. RAID is set up such that each declus-

tered array could tolerate three disk failures. In a declustered array, disk rebuild activity is spread across many disks, resulting in faster rebuild and less disruption to user programs. When only one disk is down, it is possible to rebuild slowly, with minimal impact to client workload. When three disks are down, only 1% of stripes would have three failures, so only those stripes need to be rebuilt quickly, with performance degradation noticed only during this time. Regarding data integrity, Deenadhayalan not only discussed media errors, but gave a thorough coverage of "silent" undetected disk errors with their physical causes: distant off-track writes, near off-track writes, dropped writes, and undetected read errors. To cope with these issues, GPFS Native RAID implements version numbers in the metadata and end-to-end checksums.

GPFS Native RAID provides functions to handle data integrity, such as disk rebuild, free space rebalancing upon disk replacement, and scrubbing. Disk management functions analyze disk errors and take corrective action. By maintaining "health records" of performance and error rates, the system can proactively remediate potentially failing disks.

To the Cloud!

Summarized by Rik Farrow (rik@usenix.org)

Getting to Elastic: Adapting a Legacy Vertical Application Environment for Scalability

Eric Shamow, Puppet Labs

Eric explained that elastic means using a cloud as Infrastructure as a Service (IaaS that they get machines to use, perhaps with an OS installed. He then said that elasticity requires automation. You first need automated provisioning of servers, at a minimum. From power-on to application, startup needs to be much more automatic. More complex issues include when to expand or contract the number of servers.

As sysadmins, we tend to see the big picture, but devs understand the metrics when an app is impacted. If you lose a server but the business keeps running without interruption, then you can wait until tomorrow morning. Eric then asked how many people share logs with development teams. About 10% raised hands. Sysadmins need to help make developers aware of the production environment so that they can make decisions based on fact rather than assumptions.

Eric suggested monitoring everything, but not focusing on anything at first. Latency is variable in the cloud. You need to be prepared for change. Impose sanity limits on builds and teardowns. How many and how fast can you provision/destroy? Consider having a pool of offline servers. If you have spare db servers ready to go, you don't have to wait 20–30 minutes for them to come up. To automate, you must have DHCP, PXE, DNS, OS, and Patch provisioning, all with APIs or script-based management.

Rik Farrow asked about their application. Erik responded that it published a number of newspapers. Rik then asked about the uncertainty involved in moving to the cloud. Erik responded by talking about what happened when Michael Jackson died, and one of their publications was the top hit on Google News. Their applications could handle the first spike, but it was the continuous pounding that led to slow death. You definitely want to wake up a human at some point. They also turned off parts of some pages, such as banners. Did they have access to the TCP stack, and other aspects they could optimize? They didn't have control of that in most environments, and generally just decided not to count on that. Another person asked how to protect their data. Eric said that there were lots of protection techniques, such as using encrypted tunnels for LDAP. But if your data really must stay private, don't put it in the cloud.

Scaling on EC2 in a Fast-Paced Environment (Practice & Experience Report)

Nicolas Brousse, TubeMogul, Inc.

Tubemogul has grown from 20 to 500 servers, four Amazon regions, and one colo, and requires monitoring of 6000 active services and 1000 passive. They like to be able to spin up or shut down servers as needed, but it is difficult to troubleshoot failures in Amazon. Nicolas described a single point of failure where the image would get stuck in fsck on boot; they had to revert to an earlier image, and that image didn't match their current configuration, requiring editing the configuration on each instance.

They had started with a Tcl/Tk script called Cerveza and rebuilt it in Python after the meltdown. Instead of using customized AMI, they used public Ubuntu AMI images to reduce maintenance, and cloud-init for easy pre-configuration of new instances. They used Puppet for configuration management, and Ganglia and Nagios for monitoring. In conclusion, Nicolas suggested using configuration management tools early, keeping things simple, and not forgetting basic infrastructure management, such as backup and recovery processes. They are hiring: http://www.tubemogul.com/company/careers.

Bill LeFebvre asked how their instances self-identified. Nicolas said they start a server and keep this info in SimpleDB to find the right profile. Bill then wondered what key info they used per server. Nicolas answered that they provide a recognizable hostname during the startup process. Someone else

asked if they were using trending data for Nagios monitoring, and Nicolas said they collected data into an RRD file, and Nagios watches that.

Invited Talks I: DevOps: Puppet

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

Building IronMan, Not Programming

Luke Kanies, Founder, Puppet and Puppet Labs

Luke Kanies described how he went to his first LISA conference in 2001 and got hooked on configuration management. Puppet was conceived at a LISA conference, and LISA and other conferences influenced Puppet over the years and were instrumental in its development.

Luke then said that he was not there to talk about Puppet, but about DevOps/. He started with a description of what DevOps is not. DevOps is not development. It's poorly named. It is not about developers becoming operations. It is not about operations becoming developers. It is not about "not operations." Operations is not going away; however, it will likely be transformed over the next few years.

DevOps is about improving operations, primarily through cultural change—not through new tools or the company changing, but by the people at the company changing. A major way to effect that change is by improving the sysadmins. Sysadmins are operations. You can't talk about things changing operations without talking about sysadmins changing.

Another way to improve operations is to minimize process through better tools. Sometimes you can trade off process for tooling. The main way to improve is through collaboration. If you are a sysadmin and you do not work with anyone else, you are not doing what you can for the organization and you are not doing what you can to get your job done better.

Automation shows up a lot in DevOps. In our jobs, automation is a big part of what we need to be thinking about; it's what we need to be doing. It's either there or it's coming, and we can't avoid it—we need to embrace it, and scale is the reason. We are dealing with numbers we couldn't conceive of 10 years ago; 100+ machines in 1999 was a decent-sized infrastructure. Today it's barely a blip, as people run thousands, even tens of thousands, of machines. Speed of scale is also an issue. You can now add 1000 machines a week. Zynga added 1000 machines a week for months on end—imagine adding that many machines from any vendor 10 years ago. Now we want them in a week and to have them up and running, not just sitting on the loading dock. Services are now critical. Remember when we had maintenance windows? Remember when you had time to figure out why something wasn't working? That method is no longer valid. A weak corporate Web site 10 years ago is now 90–100% of your business today. These are all things that lead to automation. Just because this is the reality, it doesn't mean automation will replace us.

You get to pick the kind of sysadmin you can be. You can be a mechanic, know the rules, follow the rules, and not make mistakes. That skill can be automated and eventually you will be replaced. Artificial Intelligence (deciding and thinking) is difficult. AI is 10 years out, just like it has been for the past 50 years. If you just know obscure items from the operating system, you will be replaced. If you just perform the mechanical aspects of the job, you will be replaced. You are not a key differentiator for your business. You need to be good at understanding and deciding. You need to understand what is normal in your infrastructure, why the infrastructure exists, why the services are running and what is important to the end users, the customers, and the employers. You need to be good at deciding what to do based on the information you have available. Your real value lies in making decisions, good ones.

For example, the financial trading industry would love to fully automate, but it can't. We are talking big data, rapid response, huge amounts of money. The skilled traders get paid, make bonuses, etc., and if their function could be automated, it would happen. Why are they not automated? A lot of what they do is, but this is a skill-based system. Consider it to be the best video game in the world and they are skilled players and get paid to play games for eight hours a day. The software systems provide huge amounts of information coming in all day. They absorb this wealth of information and then make quick decisions. The software then rapidly performs the trades based on the decisions. The software gets data to them and once a decision has been made, acts on it. It does not make the decision; they do. This is their value.

In a similar way, you need to be the kind of sysadmin who is good at understanding and deciding. This makes you valuable to the organization. If you are good at following rules and not knowing who those rules impact, probably you will be replaced with software. It is likely that we all know somebody who could be replaced with a shell script. As an observation, this year is the first year that the majority of attendees are running configuration management systems. The first workshop was 10 years ago. The scary part is that the LISA community is an early adopter and it's taken 10 years to get here. We've been moving very slowly and the world has moved quickly. If we don't start moving and move quickly, this will end up with the developers taking over the operations role.

In 2001, a group of developers wrote the Agile Manifesto. They all worked in a similar way and they were breaking all the best practices and doing dramatically better. They didn't do it because they had a plan; it was because they recognized their own behaviors. This is not scrum. It's not extreme programming. It's not a tool or practice. It's a way of thinking. Read the Agile Manifesto. It doesn't describe practice, but, rather, the way you think about the world and applies that to how you interact with your team and the teams around you, which is why it is related to DevOps.

The Agile Manifesto has four main principles:

1. Care more about individuals and interactions than about processes and tools. Processes matter and they exist for a reason. Having good tools is great. However, if the processes get in the way, they are not doing their job. If the tools interfere with why you exist as an organization, you have the wrong tools. This is important for agile development, as there tended to be good tools and tightly defined processes in old-school development. If you are good at making decisions, then you don't need inflexible processes or to use the world's best tools. If you are bad at decisions, then you need processes to double-check you and better tools to keep you on track.
2. Value working software over comprehensive documentation. Documentation is great and typically very valuable. However, if the infrastructure is down, the best documentation in the world will not help you. The software must be running and must work. If the software doesn't match the documentation, it has limited value. You are far better off building software than documenting software.
3. Value customer collaboration over contract negotiation. This seems more applicable to software developers than system administrators, since software developers tend to do more consulting, bring in a team, write the software, and then move on, but sysadmins also have contracts and customers. We tell the employers we will keep the services and systems up, we will deploy the software, we will keep the systems stable and secure. We have customers we need to help.
4. Respond to change over following a plan. No plan survives contact with the enemy. Reality often changes while we are doing something important. New technologies show up. Facts on the ground change and you need to be able to adapt to that. Change happens all the time. You need to be thinking about the change and not the plan. Build a plan but realize that the plan will not last. It is a guideline, not an absolute.

Waterfall development was good at building an application under budget and on time but building the wrong software. They never talked to the customer about what they wanted and never worked with the customer to understand what to build. Operations has a similar problem. We are very good at building secure, stable systems that take 18 months to deploy, often longer.

Kanies then talked about a customer that had invested in this process to the point that they had forgotten how to deploy software. Another customer had the problem of it costing more to deploy their software than to write it. This was a Web company and had been around a while.

If you can't change, you can't meet business needs. The company exists for a reason and the company is paying you to enable that. Operations needs to see the world in this light. Operations needs to find a way to do the job better. If operations doesn't, then the business will step in and tell you how to do the job. This would not be a good thing.

Process is the bugaboo of the systems world. We all hate it and we all follow it. Even more rarely is there a process owner who will call us on not following process. This may have happened when something went wrong and the business decided that it didn't want a repeat. This resulted in a process to ensure that particular failure did not happen again. That's how process happens. Change management is an example of these processes. The sysadmin writes up a technical change procedure that needs to be signed off, typically by a non-technical person. How can a business person really understand that? They sign off and the change is approved. Does this make any sense?

This is the fault of operations. Operations made it extremely difficult for business to be confident in the work we do. We didn't help them be confident in our work so they felt they needed to be part of the process. They needed to have a non-technical sign-off because we didn't find a way to give them the confidence they needed without that sign-off. A big part of the cultural change we need is to find a way to give them confidence that we are doing what they want us to do, as well as providing security, compliance, and stability. One of the best ways to build confidence is to trade out process for tooling. Version control is a great example of this. It wasn't all that long ago that developers didn't use version control. This inspires confidence that the versions in development and deployment are the same.

In the systems world we can also use tools to manage the change control process without requiring sign-off.

In general, sysadmins are very conservative, as in, nothing should be done for the first time. The thought process is, "This is new, so we need to move slowly until we are confident about the technology." We need to find ways to get that confidence so that we can move quickly and be confident

without sacrificing the things we have to deliver. Needs are changing faster than we do, resulting in higher pressure on the operations people.

There is a huge amount of discussion of cloud computing. Luke's opinion is that the two most interesting versions are Software as a Service (SaaS) and the self-service cloud.

Two SaaS examples are Salesforce and Zendesk. SaaS is where you take things that are not your core competencies and have someone else do it. Very few companies take CRM as a core competency. You need to be able to talk to your customers, but it isn't critical that you be able to maintain the CRM system yourself. SaaS providers have operations people and consider it to be a core competency. They know they have to be fantastic at operations, so they hire the best, train the best, constantly adopt new technology, and require operations to adapt quickly.

The self-service cloud—I want to be able to do it myself, I don't want to have operations do it. I want the developers able to get a new machine or a new operating system without involving operations. I want to get 500 new machines right now without having to deal with operations. Self-service is more threatening for operations.

Developers like the self-service cloud, since they get what they want on request. The best case is that the operations group has provided a system that allows developers to get what they want while ensuring that business requirements are met. This is about collaboration, finding ways to deliver what your business needs and what you need to deliver to your organization.

Remember that operations is not why your organization exists. You exist to enable people and fulfill the purpose of your organization. You need to understand who you are trying to help. You are in the service industry. You do not directly produce; you assist other people to produce. If you don't think of your job as helping people, you need to be afraid of automation. Every day you need to decide what kind of sysadmin you want to be. If you decide not to decide, it will get decided for you.

Tim Kirby (Cray Inc.) said that "somewhere along the line it started to feel here as though your stance was that we should do everything they ask without saying, 'That may not be the right thing to do.' And I just want to make sure that I wasn't hearing that, because I think we have a role, not necessarily as gatekeepers, but as facilitators." Luke replied that it's absolutely not about just doing what they tell you to do or doing what they ask you to do. If you aren't great at your job, then your choices are to follow the rules or to do everything the customer asks. You want to be in a place where you can make intelligent decisions about how to interpret what they're asking, whether what they are asking is complete bonkers, and of course you can't do that. It's very important that you understand what they are saying and that you decide how to act on that. It's a lot like design. One of the things we are doing at Puppet Labs is that we are really pushing our whole organization behind design and design is about the user experience. It's like the famous quote by Henry Ford: if he had done what people wanted, he would have built a faster horse. He knew what they needed and built that. It's about understanding. You have to trust yourself and you can't just mechanically follow rules. That's how you get replaced by software. Humans are great at making intelligent decisions based on complex situations

Someone said that he is scared of self-service going so fast and giving developers free rein, because it's very easy to get yourself into technical debt. Yes, it's very easy to get something functional very fast, but backups, documentation, and all these non-functional pieces still need to be done. Operations typically takes care of that, but it takes time. What are some ways that we can work faster to be able to deploy faster but still take care of all these nonfunctional bits that are important and that we know are important but which they may gloss over? Luke replied that the first thing is to figure out where your time is going. Most places I've been, people's time is going to things that aren't critical to business—a lot of what amounts to menial work that you can automate. To me this is the best place to automate. Where is time going and is it valuable time? Is it adding real value or is it time that is not that valuable? If you can automate it and make it go away right now you have more time. And this is the time you are reclaiming without increasing budget, which you can then reuse and add on to things such as building self-service infrastructure, building better monitoring, and building more information for the user.

Kent Skaar (VMware Inc.) asked if Luke had seen new areas where he was surprised to see self-service. Luke replied that he isn't surprised by very much of it. Most money is spent on maintaining existing resources, 80–85% vs. adding new resources. It's about agility and moving quickly. It's about responding to the needs of the organization. People want to move things faster.

Betsy Schwartz (e-Dialog Inc) asked if Luke could talk a little bit about the intermediate stage. Right now we have a lot of human beings in QA who sign off on accepting the QA stage and managers who sign off on rollouts. Luke said that there is no such thing as a non-intermediate stage. All this stuff is asymptotic. You can always be better; you can always move faster. Greg LeMond has a great quote: "It never gets easier—you just go faster."

Invited Talks II

IPv6, DNSSEC, RPKI, etc.: What's the Holdup and How Can We Help?

Richard Jimmerson, IETF ISOC

No report is available for this talk.

Honey and Eggs: Keeping Out the Bad Guys with Food

DarkNOC: Dashboard for Honeypot Management

Bertrand Sobesto and Michel Cukier, University of Maryland; Matti Hiltunen, Dave Kormann, and Gregg Vesonder, AT&T Labs Research; Robin Berthier, University of Illinois

A Cuckoo's Egg in the Malware Nest: On-the-fly Signature-less Malware Analysis, Detection, and Containment for Large Networks

Damiano Bolzoni and Christiaan Schade, University of Twente; Sandro Etalle, University of Twente and Eindhoven Technical University

Auto-learning of SMTP TCP Transport-Layer Features for Spam and Abusive Message Detection

Georgios Kakavelakis, Robert Beverly, and Joel Young, Naval Postgraduate School

Using Active Intrusion Detection to Recover Network Trust

John F. Williamson and Sergey Bratus, Dartmouth College; Michael E. Locasto, University of Calgary; Sean W. Smith, Dartmouth College

No reports are available for this session.

Invited Talks I: DevOps Case: Scholastic

Summarized by David Klann (dklann@linux.com)

Fixing the Flying Plane: A Production DevOps Team

Calvin Domenico, Marie Hetrick, Elijah Aydnwylde, J. Brandon Arsenault, Patrick McAndrew, Alastair Firth, and Jesse Campbell, Scholastic, Inc.

Seven members of the DevOps team at Scholastic, Inc., described their heroic effort to "Webify" an entire application set while simultaneously switching datacenters. In six months. With a single weekend of down time.

The team of 10, led by Calvin Domenico and Marie Hetrick, took control of more than 20 custom-developed applications for 10,000 school districts. The apps were originally developed as locally run client-server applications that had been "jammed into" a hosted, managed service provider (MSP) environment. At the time they moved the application set from the MSP environment, there were about 400 school districts using the apps.

The Scholastic team described their previous MSP environment as a nightmare: a virtualized environment in a managed-hosting datacenter with limited visibility to the backend systems, no access to network or storage configurations, and, of course, no administrative access to anything.

The environment from which they migrated included untenable support issues. Operations team leader Elijah Aydnwylde described a scenario where they decided to build a workaround to mitigate network trouble that the MSP wouldn't help troubleshoot: Scholastic customers were reporting network trouble when accessing an application. Scholastic troubleshooting pointed to the load balancer in front of the app servers. The MSP disagreed: "Nope, can't be the load balancers." End of story...almost. The team built and deployed a layer that bypassed the load balancer, and successfully mitigated the trouble. Unfortunately, the time spent troubleshooting (in person-weeks), and the ongoing poor performance resulted in lost revenue and lost customers.

The Scholastic team, led by infrastructure team leader, Patrick McAndrew, chose not to build a datacenter from scratch. But they wanted all the elements and access they were missing at the time. They described choosing a datacenter partner who would perform all the initial heavy lifting, and subsequently turn over management and administration of the systems to Scholastic.

The panel described several keys to completing this challenge. One was the concept of merging development and the operations groups. They actually thought they invented the term "DevOps" (or "OpsDev" as they called it). By way of example, their operations team would lob an operations task over to the development folks, often a difficult or inefficient one. The pain of having to do the manual task would get them to automate it.

The team, led by Jesse Campbell (who was also the lead coder for the team), wrote its own configuration management tools. With the goal of creating an automated control engine for the existing underlying technologies, they created a system using NFS, git, Puppet, VSphere, Bash, and Perl.

They discussed two key points about working in the "agile" environment they created. First, every project needs an advocate to usher it along. Second, communicate the requirements by actually doing the job. Software engineer Alastair Firth said that the developers were directly involved in writing specifications so that "nothing gets lost in the translation from the stakeholders." Firth also asserted that "personality matters" with respect to building teams. Pay attention to teams' personalities.

At the end of their six-month project, the team chalked up all kinds of wins. Use of their new tools and improved processes

resulted in time reductions in all aspects of provisioning and deploying services and applications. In some cases, such as patch application, the time savings were several orders of magnitude (4,500 hours reduced to 3 hours), and many processes were automated. Project manager and database designer J. Brandon Arsenault presented a slew of impressive statistics, dramatically showing the improvements in the entire application suite.

What recommendations did they have for those who can't start over in a new datacenter? Virtualization is your friend. Start small, get bigger. Build a small-scale functional model, then prove it out at scale (Scholastic didn't have a datacenter for the first three months of the six-month time frame). Someone else noted that it's relatively unusual to integrate Dev and Ops; how did that happen? Hetrick said it was a conscious decision. This was the only way that it would work. The operations team was four people; the core software was developed by many houses of outsourced stuff. Their only chance to get this done right and on time was to bring developers into the operations team. They were excited about the LISA theme and how their model fit the notion being presented at the conference. Hetrick described their concept of culture: they had a really large development staff, but the wrong culture; they had a month to turn a set of apps into the SaaS model—just hire new developers. Domenico said that it really comes down to culture. The Scholastic staff needed to be able to get developers into operations to understand the scale of the thing.

Someone else pointed out that there are different types of developers (applications, systems, etc.). They focus on different aspects of the systems. Application developers spend much more time doing UI than anything else, while system developers spend *no* time on UI. Application skills aren't really applicable to systems development. Domenico said that's a great point. The core software engine was more than six years old, but some parts of it were completely opaque; how *do* we run hundreds of these right next to each other? Aydnwylde said that it was mind-blowing that we had no control—a terrible, terrible place to be. Getting control of infrastructure was the game changer.

Invited Talks I: Case Study: Big Launch

Summarized by David Klann (dklann@linux.com)

Releasing 9/11 Data to Satisfy FOIA: It's Just a Simple Web Site, Right?

David Pullman and Carolyn Rowland, NIST

All they wanted was a photo gallery Web site. It seemed a simple request of David Pullman, Carolyn Rowland, Stephen Barber, and Andrew Mundy from their managers at the US National Institute for Standards and Technology (NIST). It turns out it was a bit more complicated than it first seemed.

The initial design requirements included a site for photos, videos, and PDF documents primarily to satisfy US Freedom of Information Act (FOIA) requests. The constraints included time and money. The deadline: September 11, 2011 (the tenth anniversary of the terrorist attack on the US). The budget: about US $25,000.

The team consisted of four systems developers who worked with a research team at NIST. Their usual workload included support for robotics, sensor networks, and related manufacturing projects. This request came during a significant structural reorganization at NIST. The request to create the Web site seemed a simple additional task while integrating two disparate IT organizations.

The Building and Fire Research Lab had collected a "pile of stuff" (1.5 terabytes of data spread over 350 DVDs) during their investigation of the World Trade Center attacks. This request was an afterthought that was proposed as a way of satisfying all the different labor-intensive FOIA requests. One significant constraint was that NIST staff was not permitted to alter the source material in any way (including adding metadata to files). This limitation, along with the fact that there was almost no text accompanying the images and videos, led to the realization that conventional search tools (such as a Google search appliance) would be of very limited value. The team settled on a different approach to the Web site.

They considered using an existing open source content management system. After evaluating several, they settled on Gallery version 2 because of Gallery 2's rich MIME-type support. Using the Model-View-Controller architecture of Gallery 2, they configured a MySQL database to hold and organize the metadata. On poring through the data, the team found that the data and the small amount of metadata they had "were a mess." Rowland noted that they found they had all the metadata for the objects (images, videos, etc.), but that they couldn't release all the objects themselves due to copyright and other restrictions. Only 13,000 out of 90,000 files had any kind of metadata or tags.

Pullman described the iterative series of tests they wrote (in Perl) to match metadata to objects. At the end of the exercise they were left with only 200 of the original 13,000 unmatched objects.

Pullman remarked on their perspective as system administrators working on a development project. They put the task into a sysadmin's perspective: "How will we rebuild this?

What if the import process crashes?" and similar questions. They described designing scripts and processes to survive the bottom falling out during batch runs, and having to start the processes over (which Rowland noted happened "quite a few times").

A few months into the project, the laboratory hired a new director. The new director involved a slew of additional players, who brought with them all kinds of different and competing requirements. They learned Agile development via "trial by fire."

While showing off the live site at http://wtcdata.nist.gov/, Pullman commented that the collection of data includes over 100,000 objects and continues to grow, as objects are released by their owners.

Someone asked if a third party could take this and Rowland answered that archive.org already has it. Rowland noted again that the primary requirement was the integrity of the data. "Integrity is of the utmost importance, not availability or confidentiality."

Pullman described the development path. The path started with the golden copy of the data on a private server. From this pristine copy they imported data and metadata into the database. From there they copied things to the development server for testing, and finally to the production server (a cast-off Dell server running CentOS). This (largely automated) process enabled them to incorporate changes and additions quickly.

By March of 2011 they felt the project was well on its way to completion. Then the NYPD "helicopter video" went viral. Suddenly ("Oh my god!") scalability, availability, and performance became part of the requirements. The team also re-evaluated the potential demographics of the site's visitors. They reconsidered their deployment strategy and considered options like the Amazon's EC2 service. They eliminated this option after receiving Amazon's quote: between US $200,000 and US $600,000 per year. Performance on a limited budget became the new top priority.

They settled on a hardware and software configuration of two load-balanced servers running nginx (which raised the eyebrows of the resident NIST security officer).

The team also considered various ways of restricting download traffic in order to avoid being "slashdotted." They ended up moving the site to a combined NIST and NOAA site in Boulder, CO. This move was based on a search for bandwidth: at their Maryland location they had only 10 Mbps available; the site in Boulder offered a full gigabit link out of the 11 Gbps total bandwidth.

Next, the team performed load testing with an Ixia traffic generator. They tuned the setup so that responsiveness would degrade under extreme load, but would recover gracefully as the load decreased. Great! The site was ready to go. Andrew Mundy, one of the team, was in Boulder preparing for the launch. The rest of the team gave the NIST director one last demo. The NIST director asked, "What about availability? Won't a successful DoS attack make the agency look bad?" He was right, but nobody had asked that question until then.

Add another requirement, but this one came with some cash. The NIST director was able to dig around and said, "Here's some money. Put the app in the cloud." Unfortunately, even with the additional funding, the security-certified cloud solutions were still too expensive, due to content delivery network add-on charges. They decided to go directly to CDN provider Akamai.

The Akamai deployment was easy; figuring out caching issues, not so much. They discovered lots of errors, and rework was needed for the shopping cart application. The lesson they learned was not to cache dynamic content. Having satisfied the layer upon layer of additional requirements, they were finally ready.

They performed some last-minute legal due diligence (copyright holder notices resent, legal review, etc.). More changes: two primary content providers asked the team to remove their content due to lack of embedded copyright notices. The FOIA people delivered a new 75 GB data set full of high-resolution video from the FBI to add to the repository (in file sizes up to 15 GB).

They launched the site on August 12, a mere five months after the original target.

Lessons learned:

"We're Ops, not Dev."

"Getting data from people is really hard!"

Getting requirements is also difficult. The team was driving requirements by showing work as it progressed.

Getting data from others is messy, but can be automated to normalize the information.

Many stakeholders. This was difficult, but useful in order to get as much input as possible before the launch.

Real stress-testing beats any theory (stress-testing was contrived, but useful).

Last-minute stakeholders: get buy-in from as many interested parties as early as possible; identify stakeholders as well.

Having an exit strategy from Akamai was helpful in order to migrate away from their CDN after the initial push.

"We're operations." This was the team's first real "development project." They got to see firsthand how to do DevOps. Final word: "The DevOps paradigm really works."

The first question at the end of the presentation was about boiling down lessons learned to a sound bite, such as "Never demo anything until you launch." Rowland replied that there are two sides to it: launching without showing to enough stakeholders may result in more changes after the launch. It may be better to demo late in the project and delay the launch a bit. Someone else asked about avoiding all the stakeholders popping out of the woodwork. Rowland commented that the team didn't really know the customers, due to the reorganization. Normally, they would have worked harder at this, and now they would know who the stakeholders are. Rowland further noted that there were two kinds of stakeholders: researchers issuing reports, and directors awaiting the site. Look more proactively for stakeholders. In the future, this team will get them involved sooner.

The next person wondered if they really need to go to Akamai. Pullman said, "It probably would have been fine." On September 11, 2011, they experienced a peak of 29,000 daily total page views, with almost 350,000 cumulative views. Total site volume has now exceeded 9 TB of downloaded data. Someone else wondered about the continue Link, and Pullman said that it was necessary. Without this mechanism it would be easy for other sites to link directly to images and videos. The NIST legal department wanted this "wall" to prevent sites from making these links. Pullman noted that there may be a better solution to this constraint, but the current implementation works fine. The final question was, "Where is the biggest performance bottleneck?" Pullman answered, "We haven't hit any bottlenecks at this point." Rowland added that not everything is cached, but Akamai helps a lot. Traffic spikes didn't cause "running hot." Pullman added, "We'll see how it performs when Akamai goes away."

Invited Talks II: Storage

My First Petabyte: Now What?

Jacob Farmer, Cambridge Computer

No report is available for this session.

Invited Talks II: Security

Can Vulnerability Disclosure Processes Be Responsible, Rational, and Effective?

Larissa Shapiro, Internet Security Consortium

No report is available for this session.

Network Security

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

Community-based Analysis of Netflow for Early Detection of Security Incidents

Stefan Weigert, TU Dresden; Matti A. Hiltunen, AT&T Labs Research; Christof Fetzer, TU Dresden

Stefan Weigert noted that attacks are especially insidious when they are targeted rather than random and stealthy rather than immediately destructive. They assume that attackers know what they want, that they have incentive not to be noticed, and that they want to compromise machines in more than one company at a time. These attackers are hard to detect! Finding a single infected machine in a large organization is a needle-in-a-haystack problem. This work focuses on attackers who target a community instead of a single organization: for instance, compromising multiple banking or retail establishments at once.

If an IP address communicates often with many companies inside a community, it may be suspect. The problem is the sheer volume of data involved. Weigert's group begins by discarding non-community traffic and allowing certain addresses to be whitelisted. They also borrowed a concept called community-of-interest graphs from telephony, which allows them to focus on the most important addresses. In these graphs, a weighted, directed edge is drawn from outside addresses to community addresses. Weights decline over time, and are reinforced by net flows from source to sink. In the end, the graph will be sparse, with edges from only the most common external addresses. They construct a separate graph for each determining factor (e.g., ports used or number of transferred bytes).

For this work, Weigert's group looked at a heavily sampled one-day segment of traffic. They manually examined the list of suspicious IPs that their analysis produced. Weigert showed us two suspicious examples. The first address connected to many different addresses within the community, but very few addresses outside. After looking at whois data, they concluded that that address was indeed acting suspiciously. The second address turned out to be an anonymous FTP server. Wegert underscored that only the community members will be able to truly tell an attacker IP from a

benign IP, and that the goal of this work is to provide data to the community.

Someone from EMC asked whether a clever attacker could produce random traffic as a countermeasure. Weigert said that they could do that, but their work isn't meant to detect all possible attackers.

WCIS: A Prototype for Detecting Zero-Day Attacks in Web Server Requests

Melissa Danforth, California State University, Bakersfield

Melissa Danforth presented an attack-detection method based on an artificial immune system. Immune systems are all about pattern matching. Just as a real immune system looks for certain protein shapes, Danforth's system looks for shapes in the syntax of URIs. Artificial immune systems are mostly worried about distinguishing "self" from "non-self." They extended the idea to detect specific classes of attacks: information gathering, SQL injection, read-only directory-traversal, buffer overflows, cross-site scripting, and attempts to execute scripts on the Web server.

Danforth's system uses genetic algorithms to breed sensors for these attack types. Each URI is reduced to a fingerprint involving features such as length and number of times "%" occurs, and this fingerprint is consumed by the sensors. The sensor-generation process begins by randomly picking features and initializing them with random values. To reduce the chance of false-positives, newly generated sensors that trigger for a random set of normal traffic are discarded and regenerated. The system then feeds the sensors a representative sample of a single attack type and begins to breed sensors with an affinity for catching that attack type, adding a small amount of random mutation. After several generations the process stops, leaving the system with a set of sensors likely to find attacks.

They were unable to test their system on a live environment; permission to do so arrived too late. Instead, they analyzed existing Web-server logs. Effectively, this means that they tested the accuracy but not the scalability of the approach. They found that their sensors were best at detecting scripting and traversal attempts, but had difficulty detecting passive information-gathering attacks. In the near future, Danforth's group hopes to use the live data that have become available. They plan to have students try to craft attacks that evade the sensors. They also discovered that it was hard to distinguish read-only directory traversal and script-execution, and may lump the two classes together. Finally, they plan to look at more of each request than just the URI.

Tim Nelson asked why they opted to use genetic algorithms. Danforth replied that traditional artificial immune systems use genetic algorithms, and they wanted to see how they could extend them. Tim also asked whether they were worried about obfuscation, since their sensors are bred to detect syntax. Danforth answered that they had considered obfuscation, which is one reason why their sensors can see the number of "%" characters. There should still be indications of attack left in the URI, even after obfuscation. Someone else asked how they determine what traffic is "normal" when removing sensors that misclassify normal traffic. Danforth said that is a challenge, and they had to work hard to develop their normal-traffic data set.

Invited Talks I: Networking

Ethernet's Future Trajectory

John D'Ambrosia, Force10 Networks

No report is available for this session.

Invited Talks II: IPv6

IPv6: No Longer Optional

Owen DeLong, Hurricane Electric Internet Services

Summarized by Thang Nguyen (thang@ccs.neu.edu)

Owen DeLong came to LISA '11 to speak about the imminent depletion of IPv4 addresses, and how prepared we actually are. APNIC has already run out, with IANA following closely behind. Several technologies are also not ready for IPv6, including current WiMax handsets/providers, DSL systems, most IT staff/management, and various others. Technologies that are ready include current operating systems, DOCSIS 3, the WiMax specification, LTE, CPE, and early adopters/industry experts.

Following his intimidating statistics and graphics about how IPv4 is running out, Owen began talking about how people should be preparing. "We need to get everyone to a fully production dual-stack environment before IPv4 runs out." To reach new participants, services will need to be able to provide IPv6. While workarounds do exist to make IPv4 work for a little longer, they come with bad tradeoffs—mainly NAT on a carrier level being an absolute nightmare to work with. Continuing on the IPv6 bandwagon, there are several advantages compared to IPv4, including not needing to count hosts, a better address issue methodology, no need for NAT v4, and a stateless auto-config. Owen also reviewed the strengths and weaknesses of several different relevant technologies, including 6to4, Teredo, and RA Guard and noting that 6to4 was one of the last solutions people should use.

Implementing IPv6 on a Global Scale: Experiences at Akamai

Erik Nygren, Akamai

Erik Nygren spoke about his experiences with implementing IPv6 on a global scale, and the upcoming transition. Erik pulled several figures from www.potaroo.net/tools/ipv4 to display APNIC exhaustion in 2011. "Think of IPv4 and IPv6 as two different Internets that don't have direct compatibility with each other."

Erik then segued into IPv6 user adoption today, showing native IPv6 preference vs. 6to4 and Teredo (google.com/intl/en/ipv6/statistics). Adding to that, he mentioned that very few home routers today properly support IPv6,never mind actual service providers. With potentially broken IPv6 being a real problem, "Happy Eyeballs" was born. "Happy Eyeballs" is a proposed Internet draft to work around broken IPv6, with rules to fall back to IPv4 when necessary. Akamai addresses this by providing a dual stack to deliver content to users regardless of protocol.

The final part of Erik's segment was about how IPv6 has widespread implications: people and training are important, along with feature gaps and bugs. In September 2012, the US government will issue a directive to upgrade their services to support IPv6. Erik closed by saying it is critical to make progress with IPv6, but to also remember that IPv4 will be here for a long time. Prioritize your IPv6 on the most important areas now, and focus on the rest when you need to.

Refereed Papers: Networking 1

Summarized by Cory Lueninghoener (cluening@lanl.gov)

Automating Network and Service Configuration Using NETCONF and YANG

Stefan Wallin, Luleå University of Technology; Claes Wikström, Tail-f Systems AB

Claes Wikström started the session by describing his team's experience using NETCONF and YANG to improve their network configuration efficiency. He began with a description of their problem: managing and automating a large network of devices is a difficult task, and previously they had been relying on a lot of screen scraping to make automation possible on interactively configured devices. When they went looking for more efficient ways to manage their network devices, they quickly settled on a solution using NETCONF (an XML-RPC-like network device configuration protocol) and YANG (a hierarchical modeling language).

To make managing their NETCONF and YANG system easier, the authors built a system named the Network Configuration Server (NCS) to act as a central place to manage NETCONF-compatible products. This system was implemented in Erlang on top of a configuration database that acts as the authoritative source of configuration data. When initially run, NCS connects to a device, finds out what YANG modules it has, grabs the device's configs, and stores a copy. From there the device's configuration can be changed within the NCS system, human-checked with a diff, and committed to the devices being changed. Bad changes can easily be rolled back to a previous revision, giving the admin a useful security net.

Noting that running large network tests in the Amazon cloud is much cheaper than building an actual large network, Claes finished by describing their performance tests on a network of 10,000 virtual routers. They found that performing an entire NCS configuration sync took less than three hours on the 10,000 devices, and that after the initial sync was complete they were able to perform day-to-day operations (such as adding an IP address to every device) in a handful of minutes.

Are the protocols vendor-specific and are general-purpose OSes that act as routers supported? Many vendors are getting behind the same standards, and it would be great if more general-purpose OS systems would start supporting it.

How generic are the configurations? For example, how does NCS handle Juniper and Cisco devices that call the same thing by different names? NCS handles this abstraction at its service manager level.

An attendee doing a lot of load balancing in his environment wondered whether YANG supported ACLs. Claes responded that it absolutely does, and that there is continuing work in IETF on that support.

Adventures in (Small) Datacenter Migration (Practice & Experience Report)

Jon Kuroda, Jeff Anderson-Lee, Albert Goto, and Scott McNally, University of California, Berkeley

Jon Kuroda described his group's experience moving a 600 square-foot machine room one floor down in a very short period of time. A group at his university decided to remodel half of a floor of their building, which included removing an existing machine room. When they were notified of this, Jon's group followed the usual steps one follows when kicked out of an apartment: first they filed a complaint, and when that didn't help they started looking for a new place to house their machines and friends to help move them. Out of the places they found, they chose the "least bad" one and prepared the move without the luxury of datacenter help, network admin, or offers of help from those doing the remod-

eling. He described the move as being like DevOps, but with facilities: FacOps.

Due to university class schedules, there was a very small window of time in which the move could happen. Jon gave a great description of what had to happen during this time: electrical work, cleaning, carefully orchestrated machine shutdown, move, reconfiguration, and bring-up. He also described some of the nice things about the move: the ability to make a more sensible layout in the new room, reinforce hot-aisle containment, and generally clean up their area.

The move went much as one might expect: some things went very well (the electrical work, for example, was done quickly), but other parts put them behind schedule. Eventually, the team had to do work that they had expected others to do (such as network reconfiguration) to get it done in time, and they succeeded. However, Jon noted that he would try his hardest never to let this happen again. His strongest suggestions after the experience were to maintain good relationships with those around you, to be ready for external delays, and to work on good collaboration tools before starting something of this complexity.

Several people related similar experiences during the Q&A. One attendee was surprised that they had enough server room elsewhere to get rid of one machine room. Jon noted that he was surprised by that too and that they need to do better space planning in the future. Another attendee wondered if Jon's group had the opportunity to clean up the new room before moving in, and he said that, thankfully, they did.

Experiences with BOWL: Managing an Outdoor WiFi Network (or How to Keep Both Internet Users and Researchers Happy?) (Practice & Experience Report)

T. Fischer, T. Hühn, R. Kuck, R. Merz, J. Schulz-Zander, and C. Sengul, TU Berlin/Deutsche Telekom Laboratories

The final talk of the session was given by Thomas Hühn about his group's experience providing researchers with a wireless testbed. He began by describing how the Berlin Wireless Network Lab was created to meet this goal, giving researchers a flexible testbed with realistic traffic and full-campus roaming abilities. In doing this they had to balance a developer's freedom for change with the operator's desire for a robust and reliable network.

Thomas continued with a description of their final product: a network with three levels of production (desktop development, indoor testing, and outdoor deployment) and a flexible configuration with multiple OS images. Their initial attempt at configuration management involved pushing new configurations to the routers, but after this didn't work well they fell back to a pull model. Their final model has the router nodes contact a central server at boot time or when their configuration is stale to check for updates. By deploying this setup across around 60 outdoor antennas, they were able to meet the needs of both users and researchers.

With the network deployed, Thomas described how the authentication layer on the testbed works. They used FreeRADIUS as the basis for their authentication system, and they found that authenticating against many accounts from several different organizations got complex very quickly. Troubleshooting the system was especially difficult, and Thomas ran through the process they used to debug one issue to illustrate this difficulty. After weeks of work, they were able to find the subtle problem: one server certificate on the university side had expired. From this experience they learned the importance of version control and making stepwise changes to their authentications system.

Thomas finished with a few lessons they learned through their testbed rollout experience. One suggestion was, when designing a research testbed, use it in a realistic way. That helps make the transition to real life easier. He also noted that robust autoconfiguration takes a lot of work and that they found that pulling configurations was much more reliable than pushing them from a central server. Finally, he concluded that having a general router image with individual configs is much faster and more flexible than having an individual image for each router.

There were no questions for this talk.

Invited Talks I: Panel

What Will Be Hot Next Year?

Moderator: Narayan Desai, Argonne National Lab

Panelists: Kris Buytaert, Inuits; John D'Ambrosia, Force10 Networks; Jacob Farmer, Cambridge Computer

Summarized by Thang Nguyen (thang@ccs.neu.edu)

Narayan started the panel by asking about the biggest changes people should be looking into. Jacob Farmer said that SSDs will significantly replace spinning disks. John D'Ambrosia talked about the chips that are driving all of the new technology. Kris Buytaert is eagerly waiting to see DevOps move into the enterprise world.

Narayan then asked how the industry is going to change for small- to mid-sized enterprises. Jacob asserted that the balance between performance and cost was highly in favor of SSDs. This launched a discussion about where people hit a plateau of diminishing returns in terms of networking, CPU, or storage speed. John noted, "By 2015 we will produce more

data than we can store. How much data do we need? What data do we need to back up?" The discussion shifted to future bandwidth requirements, and utility of the upcoming growth of data to users.

The issue of power consumption arose as well. John said that hardware is simply one aspect, but intelligent software design is also an important factor to consider. The topic shifted to the future of data mining with the rising popularity and plausibility of SSDs. Better disk performance will allow faster indexing, enabling users to read and sort through information faster.

The panel continued on a broader topic: the different and exotic things we will be experiencing in the future. The consensus seemed to be that the future is already here, and not much is going to change. Narayan pointed out that pervasive computing, sensor networks, distributing computing nodes, etc. would be changing storage, power, and networking needs for future infrastructures. Kris made a valid point in that these sensor networks exist and significant data collection is happening today, but we still need to turn this into salient information for the user.

An audience member posed a final question about the future of supply chain manufacturing, highlighting recent natural disasters in Thailand and Japan which have interfered with the supply of hard drives and tape media. Jacob spoke of those events potentially being enough of a catalyst for SSDs to succeed hard drives, as the entry barrier for SSD production is lower than hard drives.

Invited Talks II: Beyond Technology

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

Customer Service for Sysadmins

H. Wade Minter, TeamSnap Inc.

Wade Minter got off to a quick start by asking, "How many people here think they are good at customer service?" and followed up with a few questions and comments that set the tone for the talk. Wade mentioned that his style is a little more interactive than the standard tech talk.

At TeamSnap they say, "You only have to talk to the customers you want to keep." Unlike most Web startups, they effectively started with a dedicated support person. They stumbled into it by accident. The wife of one of the founders was looking for some extra work, and she was hired to help out with customer support. That helped them quite a bit, as they were able to carve a niche in customer support. The competitors were large, well funded, and targeted to sell to big organizations. They took the opposite tack. They started with individual teams and grew the business with customer support and word of mouth. That set the stage for some success, and they built on this. They believe that they stumbled onto something that Web startups do not do well.

Wade went on to observe that the audience was composed of people who supported both internal and external customers. Regardless of the type, customers are not only the people who pay the company money (internal customers also pay, usually through some other mechanism), but they are the people whose lives you must make easier.

Wade continued with the question, "What is great customer service?" starting with examples of bad customer service from the audience. They included problem-report black holes (no response), reading from the script, automation hell, rigidity, and not taking responsibility for the problem report. This was followed by good customer-support examples, including setting expectation levels, making a connection with the customer, admitting to the customer that they did not have the answer right now, fast turnaround, and non-scripted follow-up. There are trends in both categories.

During this discussion, Wade had a slide up that paraphrased Clarke's Third Law. The slide said, "Any sufficiently advanced technology is indistinguishable from crappy customer service."

Wade went on to ask, "What can you do even if you are not directly answering external customer calls?" You can get engaged, treat the customer like a real person, follow up, and have a real dialog. Those are the positive things we remember from our own good customer-service experiences. You should strive to give this experience to your customers. Examples of customer comments from actual TeamSnap customers were shown that indicated that they follow this methodology.

He observed that we (the audience) are good at technology, even things we are unfamiliar with. He went on to describe a few situations where we may not be expert and how we'd feel about situations where we would need support. That is how our customers feel. Put yourself in their shoes. Empathy is a key skill in this area and we don't value it sufficiently. This will resonate with the customer and they will want to do business with you. Don't forget that business models are easy to copy, so you need to have a differentiator that attracts customers when a well-funded startup comes in to compete with you. People will go with the folks they want to deal with. The business or group that typically wins is the one the customers want to deal with.

Wade went on to ask why sysadmins are typically bad at customer service. In addition to lack of empathy, we are impatient with other people's technology issues ("It's just a

computer: how hard can it be?"), we have a propensity to say no, we tend to be overworked, we'll get to it later, we are reluctant to call people, and we resent being asked about things that can be answered with a short search. We value expertise and we need to remember that not everyone can be an expert.

A summary of customer types was shown: power users—you can respond with short technical answer); regular old everyday users—you need to respond, but they will accept "I'll get back to you easily); reluctant users, those using tech because they are forced to—empathy works well here; the totally clueless, whiteout on the screen, crayon on the big screen, etc.—they are difficult to deal with and you may not be able to give them an answer that works, so be patient, prompt response, etc.; asshats, the people who hate you personally, etc.—since you don't seem to be capable of giving them an answer they will accept, refer them to your manager. Wade then told a story about one customer who was less than impressed regarding language support and compared them to Gaddafi. With a little care and discussion on the issue, this customer was converted to a raving fan. The point being that if they can get upset enough to complain, they have an investment in success and want to use your product. This is an opportunity to create a fanatical supporter, sowork with them.

Wade then said that you should build tools for your users, make them nice to use; Twitter's bootstrap (http://twitter.github.com/bootstrap/) is useful, simple, and incapable of doing damage. This can reduce your nuisance calls. Another item that has good mileage is to make it super easy to interact via email. Make this happen even if you have a good phone system or a fantastic ticketing system. This is how a lot of people expect things to work.

Another TeamSnap practice is to put everyone in the company on customer support, where feasible. Have the CEO, marketing people, etc., do a customer-support shift once a month. You don't understand the customer's problems unless you are on the front line. Everyone should have a stake in the customer. Don't withhold support from free customers. If you can impress them with your support, then they are likely to think "these are the people I want to work with" when they are in a position to purchase support. You don't want them to get the idea that you will nickel and dime them to death. While they may want the world, you just need to manage expectations. When the answer is no, say no, but nicely. You don't need to be a jerk about it. When you say no, give a reason, be sympathetic.

Wade finished the talk with a short summary: be empathetic with your customers; listen to your customers; treat everyone the way you want to be treated; don't be the BOFH. Your career and your company will benefit from this.

Colin Higgs (University of Edinburgh) pointed out that from personal experience on automation vs. live interaction, it costs more to have live support.

Wade replied that having people interacting with people vs. having machines interacting people does cost more money. You are investing in people and there needs to be buy in on this from the top down; if there's no buy-in from the top down to support this type of methodology, you're just going to have to do what you can. If there's only one of you and the support load is going up like this, you're probably not going to be able to take 20 minutes for every person and make them feel like a unique snowflake. Apply some of the principles. If you have to blow them off, blow them off politely.

Jay Faulkner, Rackspace, said, "I am a Racker, and top-to-bottom customer support has to be in everything we do. People tend to overlook that uptime is the primary method of serving your customers. If you build stable systems that do not crash and your customers never have to call, then you have reached the nirvana of customer support. Your customers are happy and you are happy, with no interaction required. Wade said that's an excellent point; you help yourself quite a bit by making systems that do not cause people pain. Eric Radman, D. E. Shaw Research, asked how you manage feature requests. Sometimes feature requests contradict what you just said. Wade said that they do, and if you are like TeamSnap you have three developers and a feature list of 200. You won't get to everything. Generally, say we are strapped for resources, we have serious uptime considerations, and we will consider the request. Promise that you will review the request and then really review it—don't just blow it off.

Marc Staveley asked, Isn't it true that this is one reason people like open source? You can track your feature request. You know what happens to it. Wade replied that open bug tracking may be a way to give your customers a view into what is really going on. If you are not going to get to it for years, or it's a single user request, mark it as such. Be honest. Other things may have higher priority, it may not be a good fit, etc. Marc then asked if Wade thought that visibility of the process is important. Other people chimed in and also said they would listen. Wade replied, "I think for us it is, but not for the Muggles. They will lob it over the wall and forget about it, unless it is critically important to them. It's a nice thing to have, especially if you are an internal person and you have technical customers, it would be a great thing to let them know things are being worked on; in a non-technical environment, not so much." Someone commented that it can backfire. He had an open bug open. There are a thousand people watching this bug that goes nowhere.

Somebody else asked for Wade's thoughts on transparency. He said he erred on the side of being open personally in terms of telling people, we don't have the money to do this, or, I can't do this because it has repercussions beyond what you know you're asking for. You're asking me to change the database. The more transparent you can be, the more users will think, "Okay, I'm not just getting a canned script blow-off. There is actually a reason behind why this is the way it is." If you give people a reason, the vast majority of the time they will be cool with that. People like knowing things. People like knowing that there's a reason behind something.

Jay Faulkner, Rackspace, said that the nutshell version is not just having empathy but also inducing empathy in others. In most cases, we want to fill that feature request but can't. Wade agreed, saying that by being open and honest, you allow them to understand that there are constraints. It helps your case and people no longer consider you to be an automaton.

Marybeth Griffin (Carnegie Mellon) pointed out that people in larger organizations have distributed responsibilities and everybody is a customer of everybody else. Due to the nature of the bureaucracy, things take time, and sometimes the customer service sucks. Tickets opened can sit in the queue past when the response time has elapsed. Shouldn't respect for each function be a two-way street? Wade responded that if you have groups that do not practice good customer service it will frustrate everyone and could severely limit your ability to do the same for other folks. That's a tough one to solve. His advice was to take care of your stuff and be an example, and other people may take notice.

Playing the Certification Game (No Straitjacket Required), a.k.a How to Become Certified Without Becoming Certifiable

Dru Lavigne, iXsystems, PC-BSD Project, FreeNAS Project, FreeBSD Foundation, BSD Certification Group

Dru Lavigne provided her background, which includes system administration, training, and writing and developing certifications (she's Chair of the BSD Certification Group). The tenet of the talk is that a person can derive value from certification. The session concentrated on system administration certifications, the message being that it's not all bad and ugly. There are good certifications out there.

Dru started with the bad. "Paper" certifications are a prime example of bad certification. What gives certifications a bad name is the idea that the certification itself is not worth the paper it's written on and that people who don't know what they are doing are getting certifications.

She used the "Dummies" book series as an example. The books themselves are not all bad. The basic idea is that you can take someone who knows nothing about a topic and they can be brought up to speed. The scary part is when we are dealing with system administration. You don't want someone who knows absolutely nothing about a topic very quickly coming up to speed, getting the piece of paper saying, "Oh yes, I know how to do this."

Brain dumps are also bad. If you search, you can find a list of the questions and answers for many certifications. Just by reading and memorizing you can be certified for something you have no ability to do.

Then she started getting into the ugly. What is involved in a particular program? The only way to pass an exam is to take the official training program, typically only available from the vendor, usually a week long, and costing several thousand dollars. This portion is not ugly, as who knows the product better than the vendor? The ugly part is if the only way to pass the exam is to take the program. Either the information is unavailable elsewhere or the official published information material has nothing to do with the exam, which you do not find out until you actually take the exam. The training is actually training on how to answer the exam questions. This type of exam is really an expensive brain dump.

More ugly are psychometrically invalid exams. Psychometrics is the science of assessment. The point of psychometrics is to see that if you have published a list of skills that are required to pass an exam, then the exam should test to see if you are able to perform the list of required skills, not to trick you into not passing the exam. The exam should test the skill level, not your ability to determine what question is being asked. That is not psychometrically valid.

Dru proceeded to give a number of examples of psychometrically invalid exams. Exams with pick lists, badly translated from another language or written by someone who is not a native speaker of the language the exam is written in, technically inaccurate, or possibly written by the sales team after a few beers.

More ugly, the current "hot" certification is required. This is seldom about content but, rather, the current trend. Experienced people have seen cycles of technologies go around. New terms for old concepts, new acronyms, and shiny new marketing spin generate new certifications that are unnecessary—clouds, virtualization, etc. No actual increase in your skill set results from the certification.

Dru then went on to talk about costs, not just for the holders of a certification, but for the maintainers of the certification programs. Psychometrics is expensive to achieve and not marketed as a value for certifications. This should change. Exams are expensive to maintain, and the bank of questions

will eventually get leaked. A lot of programs only have one version of the exam and it doesn't change. Over time, this cheapens the exam as more people are familiar with the content. Rectification in order to remain valid/current is expensive, especially when all that is changing is the feature list.

Having said all of that, Dru started covering the good side of certification. There is value in quantifying the tasks that make up a skill set. If you are aiming to become a good system administrator, you have a catalog of skills to measure against. If you are missing skills, then you have a learning map. If you are hiring people and they have a certification and there is a list of tasks that make up that skill set, then the candidate will have to prove that they can do these tasks. Chances are you learned your system administrator skills by doing the job. Chances are that there are knowledge gaps due to lack of exposure to some tasks. This skills catalog and a good certification program will help you fill in those gaps.

Dru described what to look for in a good certification program: Are the objectives available? Are they skill-based? Are there third-party reviews that indicate that the exam adheres to the objectives? What forms of training and study material are available?

If you are looking at certifications, you want to receive value. Sometimes the goals are different. Why are you doing it? Is this something your boss told you to get? Do you need it to get a job? Few training programs are geared toward skills, so build a lab setup. Find others who are skilled in this area. They can help you. Check IRC, forums, the local user groups, and system administrator groups.

You can gain from any certification, even the bad ones. Your employer wants you to have it, HR requires it, the vendor requires so many certified people, etc. You might even learn something new or develop the skills to wrestle that system into submission.

Dru advocates reinventing the certification game. The first system administrators didn't learn from a training program. Skills were developed by doing. How will the next generation of system administrators learn their skills? There are few practical programs, and while certification programs have improved, they are mostly vendor-specific. Look for and promote quality certifications. If you have the time, contribute to them. Most training is boot-camp based—three to five days training and you become certified. This is not sufficient time to be good or proficient at a task.

A good certification program is a tool that can be used to bridge the post-secondary knowledge-skill gap or bring new hires up to speed.

Aleksey Tsalolikhin asked if there is a list of the good programs. Dru replied that she thinks the BSD certification is a good program. Aleksey then asked her to tell us briefly about it. The BSD certification program was founded seven years ago. Prior to that, on BSD community mailing lists somebody would bring up a thread about every six months asking why there was no certification program for BSD, and that would immediately result in a flame war where people would list all the reasons why certification is a terrible thing. And some people would pipe in and say, "But you know there is some value in that." That went on for a couple of years. Finally, a couple of us got together and said, "We know there's some value; somebody just needs to sit down and do something," so we contacted all the people who said good things for years and we formed a nonprofit group.

That group was composed of system administrators and academics, and trainers and people who write programs. We had no idea what we were doing, but we just knew that this needed to be done, and we learned a lot along the way. We've run it like an open source project; all of our processes have been out there in the open, and we've written them down so other programs know what to do. Obviously, the questions themselves aren't open source and transparent, but everything else is.

We've always worked with the system administration community and psychometricians to find out how to make an exam very practical. What is it that people actually do in their day-to-day jobs, and what is it that employers and HR people are looking for in their employees? That's what we use to build the exams. Everybody's a volunteer, so we are on a shoestring budget. The only person we pay is a psychometrician, as I've never found a psychometrician who worked for free, but we pay for the exams through the cost of the exams themselves. And we offer a DVD where people have the tools to set up their own labs and practice skills they need to know.

Aleksey then asked if they found it necessary to keep updating the exam. Dru answered yes. The first step in creating exams is to define skills, and out of that there is a process for you to turn those skills into exam questions. After a set period of time, or after a certain number of people have taken the exam, you need to take all the psychometric data on how people respond to questions and see if there are any questions that need to be rewritten because they are too easy or too hard. You also have to look at those skills and say, "Since the last time we defined those skills, have new tasks been required of system administrators? Are there new features they need to know what to do with?"

Have they created training programs as well for this certificate? They decided they were only going to create the ques-

tions, the objectives. They have open sourced their objectives so that anybody can take them and create their own training programs and either contribute those freely or sell them commercially.

Christian Bauernfeind, Freudenberg IT, said that they are hiring, and he's been realizing that each interview is a kind of free-form certification exam that he's coming up with on the fly. How would psychometrics, doing proper exams, and basically testing for skill and not the ability to pass an exam apply to the interview process? Dru suggested two things: look at your own exam objectives and go through them, as they are basic system administration tasks. Their exam concentrates on BSD systems, but a lot of that would translate into any system administration. She suggested finding a subset that is important to you and have that as part of your process to make up your own mini exam. Second, if you find that a lot of the objectives would apply to the skills that you would want to see, the certification group could set it up to have a proctor come in and you could offer the exam to new hires. So do a dozen or so at a time and maybe make it a requirement of employment to be able to become certified.

Christian then wondered if they have a good source on how to turn a given skill set into a good set of interview or exam questions. Dru replied that they haven't looked at it that way. They have had some people in their group who actually work for very large companies that deal with a lot of new hires and have taken what they start with, something they call a JTA (a Job Task Analysis), and they have used the JTA list basically to see how people respond to those tasks. It's usually a list of 200 tasks, and if an applicant could answer 120 of these, that would indicate they have a very good skill set.

Eric Radman, D. E. Shaw Research, asked if they found they have to give some people hints that you can type "man ls" or "man intro" to find the answer. Dru said that's an interesting question; even when they put together their program they have two levels of exams. The first-level exam is very introductory, for junior-level sysadmins, and is a paper-based exam. It's multiple choice, but it was important that they put together a program that wasn't promoting memorization over understanding. You wouldn't get a question, for example, saying, "Which switch to ls do you use to do X?" That really is not for information. In the real world nobody memorizes those, and in the real world you've heard of "man" before. You do a "man ls" and you find your switch.

Their second level of exam is actually going to be lab-based and more involved with concept testing. The rules of the test will be that we won't tell you which operating system you use and we won't tell you what tools to use. They will ask test takers to meet an objective: for example, set up a mail server with certain attributes using whatever tools you want; at the end of the day, they want to see your number of users, does spam filtering work, etc. A lot of certification programs are either promoting memorization or they're promoting how you get to a certain screen. There's really no value if you end up with monkeys who don't know what they're doing. They try to test more concepts.

Aleksey Tsalolikhin wondered if anybody has experience with the O'Reilly certification for Linux/UNIX system administration and could talk about it. He has a new hire promoted from the help desk that he's apprenticing, is trying to make it go a little faster, and is looking at what resources are available. Dru asked Aleksey if they published any objectives. Aleksay said they have a course syllabus posted, but he didn't see any objectives. Someone suggested that Aleksey check out LPI (Linux Professional Institute); Dru said that their program is very similar to BSD's. LPI started before they did, it's Linux system administration, and they also started very open source. Their exam objectives are very well detailed, with the skills you need to know.

Migrations, Mental Maps, and Make Modernization

Summarized by Ming Chow (mchow@cs.tufts.edu)

Why Do Migrations Fail and What Can We Do About It?

Gong Zhang and Ling Liu, Georgia Institute of Technology

The goal of their paper was to understand the cause of incorrect migration. They hypothesized that the cause of most incorrect cloud migrations has to do with incorrect configuration. Gong said that his tool, CloudMig, analyzes configuration errors. Gong first discussed the cloud, which is utility driven, pay-as-you-go, and has elastic scalability. However, unlike in the past, physical nodes are connected to virtual nodes, and there are even virtual nodes to virtual nodes in datacenters. Thus, system migration is non-trivial, considering we are now dealing with a multi-tier, multi-server architecture. Alas, single host migration is not enough. Component checklists and knowing dependencies are more important than ever, as migration is a multi-step process and error rates are reportedly high and time-consuming. To make matters even worse, there are a plethora of dependencies and implicit things that occur. Currently, manual processes are used to fix migration errors. Unfortunately, this is very error-prone and the larger the data set, the longer it will take to fix them.

Their paper proposed a policy-based migration validation. The setup for the experiment included physical machines, one Hadoop server, and one Rubis machine. The aim was to

observe migration errors. Their paper put forth a categorization of errors: dependency preservation (which includes things like typos in dependency files), platform differences, network connectivity, reliability, shutdown and restart, and access control and security. They found that 36% of the migration errors are due to dependency preservation. Tools to manipulate and check configuration errors are critical, and this is the goal of CloudMig.

CloudMig is based on the idea of policy validation. It helps operators to weave important configuration constraints into continual query-based policies and periodically to run these policies to monitor the configuration changes, detecting and alerting for possible configuration constraint violations. CloudMig is semi-automated migration, and the architecture is two-tier: one server and one client. CloudMig has been tested on the same setup described above. Gong illustrated a configuration policy layer and an installation layer on the server. The experiment eliminated a good number of network and performance difference errors and mitigated platform, software, and hardware issues in Hadoop. The big lesson learned was that implicit and hidden errors are paramount in distributed apps.

Provenance for System Troubleshooting

Marc Chiarini, Harvard SEAS

Marc provided an overview of troubleshooting in a nutshell: troubleshooting is hard and frequent triage is detrimental to the construction of good mental models. The ideal way to develop such models is exposing hidden dependencies between components and build modes of component interactions that one can query. To accomplish this, Marc introduced the idea of provenance. "Provenance" means collecting and maintaining a history of interactions over time (i.e., where do things come from?). A provenance for system troubleshooting is a recorded history of digital process creation, including environment variables, execution time, parent process, arguments, and process destruction.

Marc described the use of an acyclic graph to organize all the information, where nodes are objects and edges are potential dependencies. It is important to note that the content passed between objects is not analyzed. Marc illustrated a simple example called wire_test that started with resolv.conf and led to net manager, which in turn led to netman socket endpoint on the left and other inputs on the right. Netman socket endpoint led to DBUS, which had two child nodes: dhclient socket endpoint as the left child and other inputs as the right child. The dhclient socket endpoint led to the dhclient, which led to the dhclient.conf file. Opening the dhclient.conf file revealed that someone commented out domain-search, host_name line. But what if a provenance graph has hundred of thousands of nodes? Marc proposed ranking edges based on the likelihood that the inputs affect expected behavior. A statistical approach is used: the number of times input F is read by program P divided by the number of times program P is invoked. As time progresses, rank is refined.

Marc also noted a few caveats with this approach, including that inputs of many programs can be changed by options or shell redirections, thus skewing the ratios. The solution is to treat invocations in which these differ as separate programs. In addition, for new executables such as different versions of applications, Marc proposed using a stack approach. In his work, many first-order dependencies for programs have been ranked (e.g., ssh). Edge-to-files residing in well-known directories may be increased (e.g., /etc/), while edge-to-files in log or temporary directories should have rank decreased. Popular objects are less likely to be the singular cause of problems. Using this statistical approach, users can explore "what if" scenarios by ranking paths and subgraphs. That is, take the arithmetical average of all of its edges.

Marc also introduced the PQL (pronounced "Pickle") language to query the graph. He showed an example to retrieve all the processes' output objects that use sendmail.

Marc concluded by reiterating building a model of interactions between system-like components. There are a number of works in progress, including performance, integrating the query warehouse with trouble ticket systems, exploring other methods for extracting patterns in provenance graphs, and making this work on a system with hypervisor.

Debugging Makefiles with remake

Rocky Bernstein

There was no presentation of this paper.

Invited Talks I: Security

Surveillance or Security? The Risks Posed by New Wiretapping Technologies

Susan Landau, Visiting Scholar, Department of Computer Science, Harvard University

Summarized by Erinn Looney-Triggs (erinn.looneytriggs@gmail.com)

Historically centralized technologies such as the telephone network lent themselves easily to wiretapping. As technology has progressed, certain facets, such as decentralized point-to-point networks, have removed that ease, while others, such as cloud architectures, have increased the ease of interception. The US government, attempting to keep pace with a perceived growing threat, has enacted laws broadening the scope of wiretapping and easing oversight on wiretapping.

The legal framework that underlies both the right to privacy and the laws that enable wiretapping starts with the Fourth Amendment, which grants the right to privacy. The first wiretapping laws were not enacted until 1968, with a revision in 1978. Since the 1994 advent of CALEA, there have been an escalating number of laws enhancing and broadening wiretapping capabilities.

As the use of wiretapping has increased, equipment from manufacturers such as Cisco has commoditized wiretapping abilities. This has in turn increased the attack surface for illegitimate use of said equipment for nefarious purposes. In short, with the increasing ease of wiretapping has come the increasing ease of illegal or unwanted wiretapping from third parties.

These increased risks have to be balanced against the need for wiretapping, just as increased encroachment onto privacy needs to be carefully balanced against the legitimate needs for encroaching upon said privacy.

How is law enforcement coping with the use of encryption? It appears that other tools, including pattern analysis, are able to extract enough information in some cases; in others, law enforcement will simply have to spend more time and money. The use of BlackBerry devices in India and how they were coping with RIM's end-to-end encryption was also discussed. The banning of BlackBerries and the changes that RIM is putting into place specifically for India should assuage the country's concern. Concern was also voiced about the costs of CALEA enforcement on ISPs; in a world using carrier-grade NAT, a larger ISP may generate up to a terabyte of tracking data a day. Susan responded that CALEA may have to be reworked to take this concern into account.

Invited Talks II: Sysadmin in/and the World

Copacetic.

David N. Blank-Edelman, Northeastern University College of Computer and Information Science

Summarized by Deborah Wazir (dwazir@gmail.com)

David Blank-Edelman presented several ways sysadmins could become happier at work, despite the stress, setbacks, and roadblocks we all experience in this line of work. The techniques were organized around the themes of mindset, motivation, and making change.

First, drawing on work published by Dr. Carol Dweck, Blank-Edelman covered the differences between a fixedand a growth mindset.

Dr. Dweck explains (http://mindsetonline.com) that people having a fixed mindset believe that basic qualities such as intelligence or talent are just fixed, so they focus on documenting these qualities rather than developing them, and they believe that success and perfect results will come from innate talent alone, without effort. Mistakes and great effort are then viewed as signs of failure.

People with a growth mindset view innate talent as the foundation, so abilities can be developed through hard work. Since working hard and making mistakes are considered to be necessary for improvement, they are viewed as signs of progress toward mastery, encouraging further effort.

Blank-Edelman emphasized that making mistakes is the way to innovation. He summarized the way that individuals can practice replacing fixed mindset thoughts with growth mindset actions.

Intrinsic vs. extrinsic motivation was discussed next, referring to Daniel Pink's book *Drive*. Although sysadmin work can contain many algorithmic (repetitious) tasks, quite a bit of it is more heuristic and can contain a lot of intrinsic reward. External rewards can motivate for a while, but the effect will wear off and result in poorer work quality overall. Elements of a work environment that can boost motivation are autonomy, mastery, flow, and having a sense of purpose.

Keeping these elements in mind, the discussion turned to making change. Blank-Edelman suggested changing just one thing as a way to start, such as increasing autonomy by negotiating to own a whole project rather than just helping with part of it. Mastery of new technology could be developed by using virtual machines to create an environment safe for experimentation that would not affect production. To effect change in the organization, the focus should be on persuading the large group of people who are neither advocates nor opponents of the new idea, as this would give a clear majority in favor.

Finally, Blank-Edelman explained ways to change your own perception—especially necessary as a sysadmin, where continually changing goals can become frustrating. He gave examples of ways to turn tasks into games and to increase motivation to do them. Several books were displayed for further reading and inspiration.

After the talk, one audience member recommended the "Quantified Self" Web site as an additional resource.

Project Cauã

Jon "maddog" Hall, Linux International and Project Cauã

No report is available for this session.

Closing Session

What Is Watson?

Michael P. Perrone, Manager, Multicore Computing, IBM T.J. Watson Research Center

Summarized by Rik Farrow (rik@usenix.org)

Michael gave the closing talk, thanking his audience for sticking around. He explained that although he had worked on Watson, he wasn't responsible for most of the algorithms that provided the magic sauce. He then went on to tell his audience how Watson is different from other forms of search.

Michael began by asking how many people in the audience had seen the PBS show, then if anyone was not familiar with *Jeopardy*? One man from Scotland piped up, another person said he lived under a rock. Michael dodged, then went on to say that in the past, grep was his search tool. Next, Google became the tool of choice, but using Google requires putting some thought into the proper search terms.

Jeopardy poses much more difficult problems to solve. Michael provided some example *Jeopardy* answers, using them to illustrate the importance of being able to parse natural language and tease out the important parts of each answer. Winning at *Jeopardy* also requires broad knowledge and quick answers.

Over 350 TBs of text were parsed to create syntactic frames, and the results processed again to create semantic frames. A semantic frame might be "Water is a fluid (.9)" or "Ships sink (.5)," where the number represents the degree of certainty. As more text is processed, it is cross-correlated with existing frames, increasing or decreasing certainty. Simply matching frames with answers doesn't work, because finding matches may involve temporal or geospatial reasoning, statistical paraphrasing, decomposition, and synthesis.

Michael took a break for questions, most of which he said he would answer later in the presentation. Then he presented some examples of early responses by Watson, provoking laughter and applause because certain key engines were lacking. For example, under the category Milestones, the answer "In 1994, 25 years after this event, [one] participant said, 'For one crowning moment, we were creatures of the cosmic ocean.'" While the correct question was, "What was the Apollo 11 moon landing?" Watson posited, "What is the Big Bang?" as Watson had no engine at that point that took time into account as a constraint. There are hundreds of engines in Watson.

Someone asked how they measured reliability, and Michael sighed and said they don't have a good answer for that. One way was how many times some particular evidence appeared and whether that evidence appeared in more documents. How did they monitor such a complex system? There was a team of about 20 people working on various aspects of certain engines, and if something was wrong with an answer, they could direct the result to the person or persons handling the related engine. An audience member pointed out that categories themselves can have puns built into them, and Michael agreed, saying that categories have to be analyzed by pun engines, just like words surrounded by quotes. Someone asked how much they studied humans, and the answer was "a lot." There are lots of strategies in *Jeopardy*, and they compared their strategies to human strategies. Humans frequently start at the top and work down. Watson doesn't. Watson goes to the bottom rows, going for the Daily Doubles, which allow players to double their winnings.

Michael then displayed a graph showing Watson's progress compared to human winners. Over the four years of the project, Watson went from very poor to the territory of the best human players. Michael said that he was worried that Watson would get too good, and that could result in a backlash against devices like Watson. On a single node, a single question would take about 2 hours on a 2.6 GHz to run. They parallelized the task over 2880 cores, reaching 2.6 seconds, which is what they needed to compete with humans. Adding more cores may not help much, as there is a certain amount of overhead.

The real goal is to make Watson useful. One of the first areas of interest would be in answering health questions. Other areas could be tech support, business intelligence, and improved information sharing in government. Someone quipped, "Skynet," at the mention of government, but the goal is improving citizens' ability to get answers quickly from government. Someone else asked how much time it would take to specialize Watson for something else. Michael answered that they have built tools for analyzing data and could reuse those tools. The same person pointed out that the *Jeopardy* version had betting that relied on the confidence that an answer is correct. Michael said that the betting engines wouldn't be needed, but a lot of other engines would prove useful.

Michael saved the hardware slide for last: 90 Power 750 servers, 2880 POWER7 cores at 3.6 GHz, with 16 TBs of memory and 20 TBs of disk, in 10 racks. These run SUSE Linux, UIMA (Unstructured Information Management Architecture) software, their own software, and Hadoop. Watson is good, but it takes 80 kW of power and 20 tons of cooling. The human brain fits in a shoebox, can run on a tunafish sandwich, and can be cooled with a hand-held paper fan. We have a long way to go, Michael said.

How cost-effective was this project? Michael said he didn't know, but the publicity was priceless. Does Watson teach us anything about human brains? Michael answered that he likes to think about this. The algorithms they use are statistically driven, and he wouldn't want to tie this to human brains too tightly. Michael said perhaps he could discuss this over a beer later.

Someone asked about the buzzer. Michael showed the setup for *Jeopardy*, and explained that buzzers are not active until the game host has finished reading the answer. Watson has control of a solenoid that presses its button. Michael also pointed out that humans can hit their button when they intuit that they know the answer, while Watson will not answer until it has calculated the answer. Were linguists involved? No, that while natural language experts were involved, what was most important was to create a system that could learn. How many sysadmins were used? Just one or two, as the whole project was run on a shoestring. Had they rated provenance? Yes, they ranked their import sources, where an encyclopedia was rated with more confidence than Twitter, as an example. How had they handled software updates? Michael didn't know for certain.

Workshop Report

Advanced Topics Workshop

Summarized by Josh Simon (jss@clock.org)

Tuesday's sessions began with the 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 new folks (more than in any past year). Then we went around the room and did introductions. Businesses (including consultants) outnumbered universities by about 9 to 2 (up from 4 to 1); over the course of the day, the room included 6 LISA program chairs (past, present, and future, the same as last year).

For the third year in a row, our first topic was cloud computing. We still don't have a common definition, though the room seemed to agree that we're moving toward the "Whatever as a Service (WaaS)" model with software, platform, and infrastructure as the most common. One problem is the relatively low amount of data on the scalability of the services; when the cloud is abstracted away from our control, there can be problems if production has capacity or bandwidth or speed requirements. When you grow in 18 months as big as the current cloud, that won't work. Anything with growth may not be appropriate for the cloud, though that's not necessarily true for well-understood and well-behaved Web applications. For some, the consumer view of "A place somewhere out there to keep my data" is a good definition. This isn't new to most of us. Businesses with certain regulatory requirements (FERPA, HIPAA, and SOX) may have requirements preventing them from moving certain data (and thus the processing thereof) to the cloud. The ability to spin up a machine and provision it via some configuration management system without having to do actual work (racking, connecting cables, and so on) is a good thing. We'll probably come up with different terminology in the industry.

Next up we discussed increasing regulation of Internet activity. Governments don't seem to have a clue about the Internet; each country doesn't seem to understand that they don't control the whole Internet. There's a lot more censorship on national boundaries (Australia, China, and Egypt were mentioned, and before we went to press the USA had legislation pending as well), and we're concerned where this might be leading. SAGE-AU managed to get the Australian legislation put on hold. The room seemed to be split on whether lobbying would really have any effect, though stewardship (such as ARIN for IP addressing) might be a good thing. This was a fairly gloomy discussion. One person noted that we have to give politicians an alternative; they'll take the most expedient thing. We need more companies to help enable the environment we want.

Our annual lightning round of new-to-you tools seemed to fall into two categories: software (Augeas, cloud-based VM systems, Dropbox, Evernote, f.lux, git, Google+ Hangouts, Internet in the pocket (any smartphone), OneNote, OpenStack, Puppet, Trac, vimdiff, vpnc, WordPress) and non-technological (changing jobs, getting engaged, new mattress, paying others to do work for you, taking vacations, and team-building exercises at shooting ranges).

That segued into career paths. The general question was how to move out of a too-stable, unchanging environment where there's no opportunity for growth without going into management; several believe they're in that kind of workplace. Companies increasingly claim, "We're interested in people who've been around for a while," but the reality is that they're hiring younger, inexperienced people who are more willing to work ridiculous hours for less money. Becoming senior often leads to becoming siloed. We took a straw poll: one person has been at his job for 17 years; about half a dozen were at seven years or more. Having a technical growth path is important. The problem with even tall technical tracks is they get narrow pretty quickly. Having other senior people around (even in different silos) to learn from can be helpful.

On the subject of interviewing, understanding the deeper concepts is much better than trivia; make the interview questions open-ended so you can see how the candidate thinks.

When you interview for a new job, always target the job *after* that. Remember that you're interviewing them as much as they're interviewing you. It's also probable that you know more than you realize. Practice interviews are good. Honing your higher-level thinking and problem-solving skills is also useful. We all have contacts; use your networks and possibly bypass the formal recruiting process.

On the subject of hiring, several have had problems finding enough high-quality people in the pipeline. Finding those who're interested in looking at the big picture is problematic and frustrating. One person believes that intelligent companies don't care so much about you knowing everything already but just being "clue-ready" to pick up their oddities (although HR has been filtering a lot on the specifics). Hiring managers working with HR to build the filter may be helpful. However, another is seeing the opposite: word on the street is that managers want very specific things. This may be region-specific. Any company needs to understand there's a learning curve, and hire people with clue so they can learn the specific technology. It's not a bad thing to come in understanding the space even if you don't have specific expertise. Demonstrate proficiency on the stuff you're doing now and how you've been able to pivot in the past. However, it may depend on where you're going: Big outsourcing organizations nowadays seem to look for the specifics so they can hit the ground running at the client, whereas research organizations or universities may be more willing to hire clue-ready people without specific skills or experience in a specific technology. One person had a senior position open for six months; they'd find a candidate they liked, but would take too long to get back to them and the candidate would slip away. They wanted to hire a candidate to revitalize and reinvigorate the team. They got a new recruiter on the HR team and within a month they filled all three open positions with amazing people. Sometimes you really do need a good recruiter.

The next major discussion was about what the DevOps and sysadmin community needs but doesn't have. We already have contacts, national and regional conferences, some sorta-magazines, a mentoring program (LOPSA), and mailing lists. One immediate response was lobbyists, linking back to the previous discussion on regulation. Some disagreed, believing that improving public perception of what we do would be helpful even without political lobbying. One believes that "sysadmin" is too narrow a term; many of us do more than just systems. We'd be better served as a community if we had better labels (for example, service administration): it's systems, databases, services, networking and connectivity, and so on. DevOps is another facet of the whole, and it's being integrated, but names are important and the "sysadmin" name may be too restrictive. One possible problem is that a universal professional identity is missing from the field. The medical profession (doctor/nurse) was brought up as an example. However, humans only work in specific known ways; IT can work in many different ways, so it's more complicated.

One tangential discussion was on the term DevOps. Some see it becoming as much a buzzword as cloud. We're not integrating the big DevOps communities into the USENIX/SAGE/LOPSA community. Is it "deploy multiple times a day to Production"? "Continuous integration via Hudson or Jenkins"? It should also be remembered that what works (or not) for Web sites definitely won't for larger enterprises. Even configuration management hasn't penetrated as much as people seem to think it has. We don't have best practices for CM yet. We don't have best practices for code review yet. There are no white papers on this.

After our lunch break we took a quick poll: only 11 of the 26 present at the time still run their own email service (either at home or offsite), and nine more have stopped doing so in the past year. One hasn't outsourced because it keeps his skills sharp. Another has his public blog adminned by someone in Romania for $50 every time the blogging software needs to be updated.

Our next discussion was on large scalable clustered storage. One company represented generated a lot of data (1 TB/day) that they need to keep forever, and they see that growing to 10 TB/day. The question was, what are people looking at for data? Are they staying with spinning media or moving toward flash or other solid-state drives? Much depends on your use profile; one site uses EMC Celera for non-parallelized storage, but their profile is user home directories and scientific data in an NFS model. Most people with large storage needs seem to be using GPFS. Other mentioned products include Fusion I/O cards, Infiniband, NetApps, and Violin. The network wonks present wondered about the network behind this large storage; consensus seems to be to use dedicated networks, though some have upgraded their network switches to terabit backplanes. On the subject of failover, most seem to be failing over the servers but not necessarily the storage independently from the servers.

Next we took a quick lightning round asking what the next useful fad will be in the next two years. Answers included configuration management, death of tape, decline of social networking, increasing use of app store-like software distribution within companies, infrastructure as a service (IaaS) increasing, IPv6 deployment, JSON APIs, mobile security, more private cloud products, moving away from big iron databases toward NoSQL/MongoDB, moving away from running machines toward providing APIs, moving away from

the cloud back to local, SSD not spindles as primary storage, statistical analysis about systems, UI improvements (facial recognition, motion detection, and Siri- or Watson-like interfaces), and virtualization.

We next discussed workstation replacement. Only four people said they use virtualized desktops. Some environments reimage the workstation on logout (mainly in public labs), and most seemed to prefer physical workstations, due to performance issues. Environments that use spare CPU cycles for processing (such as Condor) prefer physical to virtual workstations for performance reasons. Virtual desktops assume high-bandwidth and low-latency networking between the user and the physical hardware, which is not universally true. Furthermore, most seem to think virtualized desktops don't save money; hardware costs are falling and local processors and capabilities are getting cheaper, so centralizing the services for anything other than administrative overhead may not have a benefit except in areas where power and cooling are your expensive limiting factors.

Next we discussed life balance and stress management. IT culture seems to still be 60- to 80-hour work weeks, which leads to a lot of burnout. Some places bought toys like ping-pong tables ("startup mentality"), but we should change the culture more toward mentoring the younger or juniors, learning how to say "No" despite pressure, and educating management to have them cause less stress. There's a difference between good stress ("I bet you can't do this over the weekend...") and bad stress ("... or you'll be fired on Monday"). At one represented employer it's good to hit or be within some percentage of the service-level agreement, but bad to be outside that percentage, even if it's responding too fast. In other words, meet but don't exceed your SLAs.

IT often manages to pull a magic solution when backed into a corner, so expectations are set (perhaps unreasonably) high. One method of pushback is to say, "Here's what I'm working on; what do you want me to drop to work on this new thing?" and let management make the call. If your work runs into your personal time, you can use some of the work day to recover (such as running errands, making doctor's appointments, etc.). One person noted that adding a fitness regime can help with stress as well, though not even half of those present have a regular fitness routine. Another person pointed out that there are strict rules for what overtime is allowed in Europe, and there was a brief tangent on cultural differences between US and European time expectations.

One person's employer allows everyone to take one day per month to not come to work (managing their time); the requirement is to remain in town and available if you're needed. They tend to use it for kids' functions or doctor's visits. The company president's general attitude is that if there's a crisis of technology, he asks whether anyone's going to die if it isn't fixed immediately. The trick is convincing your management of that. However, if management doesn't support having a work-life balance, you're working in the wrong place. The final comment was that you get more respect by respecting yourself and enforcing your own work-life balance.

We had a very brief discussion about patents. There have been a lot of lawsuits about technology. One person was subpoenaed in a patent suit between two companies he'd never heard of; having written a PAM module a decade ago was apparently evidence of prior art. Do software patents help or hinder innovation? The way the (US) law is written and the decision is done, except for clear prior art the Patent Office has to grant the patent because something isn't prohibited, which can hinder innovation. How sysadmins look at things is different from how the law is written. LISA is important because papers help show prior art. A couple of years back a commercial company tried to patent configuration management, but Anderson's 1994 paper was prior art such that the patent was denied. The best way to fight this is publish your work; once it's published, it's prior art. However, many are held back by fear of being sued for violating someone else's patent.

Next someone asked if there was management-level publicity about sysadmins going away. Some are seeing a lot of this, in part because developers see that cloud services let them jump right to release. Others noted that the thought of some new technology making sysadmins obsolete has been around for the past decade, such as with autonomic computing. One person suggested that we could change the sysadmin role away from "operations drone" toward "architect." With the automation and configuration management tools we have today, many of the "mindless" tasks can be automated away, and the sysadmin can take on more of a higher-level architect, designer, or decider role and improve the service and infrastructure. Another idea was to have a gatekeeper between Development and Production, selling that their knowledge of security, process, scalability, and so on is important and relevant. One person's environment bills every product and service back to the requesting department. It was noted that the real answer depends on the actual cause. What's tickling management's nerves? If it's cost, argue about the cost to the business in the event of outages, in terms of financial impact, publicity, and goodwill.

After the afternoon break, we discussed women in technology. One of our participants is involved in a number of research areas and focus groups. They asked if we're seeing women in technology, if they are showing up in applicant pools (under- or overqualified), if we have any outreach

ideas, and so on. One environment has a lot of women in both tech and leadership roles and is seeing qualified candidates of both genders, although women tended to be more on the development than the sysadmin side, and there were almost no women DBA candidates. Another environment has a lot of female developers and project and program management, but practically none in service engineering/SA.

Some say that IT in general has a lot of unfriendliness toward women. One person observed that when we say, "We're not creating a hostile work environment," it may be untrue. We need to treat candidates or colleagues solely on their technical merits, not on their gender. In the past, women needed to be aggressive enough to get past the Old Boys' Network to get in. Also there's the culture of "She's not that good" from guys, which may be subconscious from many men. One person noted that the unconscious gender-biased behavior is learned. One job he was at had 40% women in technology. They felt little to no bias against them because so many were there it was considered "normal."

One person has been interviewing students for a decade and in that time he's had all of three female applicants. He was able to hire one; the other two weren't the best for the job at the time. That one has since left, in part because she didn't have the skill set yet. It's too late if we wait until they're in industry; we need to get them involved earlier. We need to instill the interest in technology at a younger age (college is too late). He sees no non-US females and only a small number of women overall. Most in the room seem to agree that we need more women in the field; we need to get more women (girls) involved in science, technology, engineering, and math (STEM), especially where there are no tech classes. However, we've observed that SEM is easier than T.

Tangentially, someone was triggered to think about statistics by a previous discussion. We're likely to look at more complex metrics over time that are statistically defined (95% of requests under *n* milliseconds, 99% under *m* milliseconds, and so on). Running large-scale services, you can't use "10% above peak" as a metric. It's also an educational problem. We need to think statistically about latency and capacity and what's "good enough." Similarly, we need to move from "I ran this test 10 times and got results of x" and toward "I have a 95% confidence that...," which is a better metric. We're also seeing a drive for comparisons (such as this week versus last week) and trending analysis. Several people think this could make a good Short Topics book. The final comment was that you have to know what's *normal* before you can define *abnormal*.

We ended the workshop with our final lightning round, asking what is going to be new-for-you in the next year. Answers included architecture design and development, career planning (finding better jobs, increasing team visibility in a good way, managing the existing job, moving between generalized and specialized), data mining and statistical research, decommissioning old hardware, delegating everyday tasks, doing more DevOps type work, hiring more people, managing more data, publishing new books, recreating one's environment from the ground up, and training interns. However, for some, not much is changing that quickly.