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

Conference Reports

27th Large Installation System Administration Conference

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

Summarized by Jonathon Anderson, Ben Cotton, John Dennert, Rik Farrow, Andrew Hoblitz, David Klann, Thomas Knauth, Georgios Larkou, Cory Lueninghoener, Scott Murphy, Tim Nelson, Carolyn Rowland, Josh Simon, and Steve VanDevender

Wednesday, November 6, 2013

Opening Remarks and Awards

Summarized by Rik Farrow (rik@usenix.org)

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

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

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

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

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

Keynote

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

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

Jason Hoffman, Founder, Joyent

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

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

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

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

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

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

Invited Talks 1

Summarized by David Klann (dklann@linux.com)

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

Jonathan Geibel and Ronald Johnson, Walt Disney Animation Studios

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

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

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

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

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

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

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

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

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

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

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

Becoming a Gamemaster: Designing IT Emergency Operations and Drills

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Invited Talks 2

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

A Working Theory-of-Monitoring

Caskey L. Dickson, Google

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

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

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

Effective Configuration Management

N. J. Thomas, Amplify Education

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

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

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

Invited Talks 3

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

Our Jobs Are Evolving: Can We Keep Up?

Mandi Walls, Senior Consultant, Opscode Inc.

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

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

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

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

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

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

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

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

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

Invited Talks 1

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

User Space

Noah Zoschke, Sr. Platform Engineer, Heroku

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

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

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

Observing and Understanding Behavior in Complex Systems

Theo Schlossnagle, CEO, Circonus

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

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

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

Invited Talks 2

Summarized by Cory Lueninghoener (cluening@gmail.com)

Building a Networked Appliance

John Sellens, Syonex

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

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

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

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

How Netflix Embraces Failure to Improve Resilience and Maximize Availability

Ariel Tseitlin, Director, Cloud Solutions, Netflix

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

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

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

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

Invited Talks 1

Summarized by Rik Farrow (rik@usenix.org)

Storage Performance Testing in the Cloud

Jeff Darcy, Red Hat

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

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

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

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

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

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

Surveillance, the NSA, and Everything

Bruce Schneier, Fellow, Berkman Center for Internet and Society

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Invited Talks 2

Summarized by John Dennert (jdennert@iupui.edu)

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

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

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

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

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

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

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

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

What We Learned at Spotify, Navigating the Clouds

Noa Resare and Ramon van Alteren, Spotify

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

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

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

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

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

Lightning Talks

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Thursday, November 7, 2013

Plenary Session

Summarized by John Dennert (jdennert@iupui.edu)

Blazing Performance with Flame Graphs

Brendan Gregg, Joyent

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

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

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

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

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

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

Women in Advanced Computing Panel

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

Moderator: Rikki Endsley, USENIX Association

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

An audience member asked whether the panel thought the "BOFH" mentality would disappear in the next few years. Forinash succinctly observed, "Assholes won't disappear." McNeil thinks it's easier for women to work in IT today than it ever has been, but society still doesn't encourage it, and she sees the number of women entering the field decreasing. Rich confessed she secretly loves the BOFH stories, but they are from another age, and the best way to get things done is not to be controversial, saying, "I like movies where people get murdered, but I don't go out and murder people."

An audience member asked, "How do you be a role model instead of a figurehead?" Forinash said she felt this was up to managers, and that peer awards reflected respect from peers rather than giving people awards for being in a minority. The audience member continued her question by observing that in academia, she was asked to do outreach because she was a woman and not because she was doing serious technical work. Forinash said to be good at what you do regardless of expectations. Straughan recommended talking about technical work rather than about being a woman. McNeil summed it up as "You're a woman in engineering, rock it!"

Invited Talks 2

Summarized by Cory Lueninghoener (cluening@gmail.com)

Hyperscale Computing with ARM Servers

Jon Masters, Red Hat

Jon Masters, chief ARM architect at Red Hat, spoke on how advances in ARM processor technology are making it possible to fit 1,000 to 10,000 nodes in a single datacenter rack. He began with a description of what the ARM architecture is, noting that it is a design that can be licensed by anybody, is not tied to a specific vendor, and can create low-power systems. After showing a picture of the current ARM celebrity, the Raspberry Pi, he went on to describe server-class systems that also use the same processor design.

Jon told the audience that the ARM architecture makes it possible to fit thousands of server nodes into a single rack that uses only a couple of hundred watts, which increases server density and reduces datacenter PUE. He showed a picture of an existing rack drawer with many nodes within it and described how the architecture encourages a fail-in-place operational model, where failed nodes are replaced in bulk (if at all) instead of on a case-by-case basis. He also touched on the newly released 64-bit ARM architecture, noting that 64-bit systems are currently much more expensive than 32-bit systems.

For the next section of his talk, Jon described the software stack that Red Hat has been working on for the newest generation of ARM servers. He focused on the 64-bit ARM architecture, describing the work needed to get Fedora 19 ready in advance of the 64-bit release. He spent some time on how the new 64-bit architecture uses standards such as ACPI and UEFI to be more consistent across multiple vendors, and how Red Hat has used this standardization to speed up the Fedora port. After describing details on the porting process and the package build process,

Jon noted that there is now a Fedora 19 remix that boots natively on 64-bit ARM hardware and contains more than 12,000 packages. Before taking questions, Jon pointed attendees to #fedora-arm on Freenode IRC for more information.

One audience member asked whether there are low-cost 64-bit ARM systems that hackers can play with. Jon answered that $5,000, the current cost for a system, is indeed a lot to spend on the hardware and that he is working with companies to push that below $500 in the relatively near future and cheaper after that. Another audience member asked about commercial adoption of the new 64-bit systems and whether Red Hat will support them with their RHEL product. Jon carefully chose his words when answering, noting that if the market responds there will be RHEL support, but that he has nothing to announce right now. Finally, an audience member asked how 64-bit ARM will handle peripherals. Jon told us that the new architecture is moving toward ACPI and enumerable buses.

As a grand finale, Jon asked the audience "Who wants a Beaglebone ARM board?" When a large number of hands were raised, he threw a small Beaglebone box out into the audience and an enthusiastic attendee caught it.

LEAN Operations: Applying 100 Years of Manufacturing Knowledge to Modern IT

Ben Rockwood, Joyent

Ben Rockwood began his talk with a video showing President Obama addressing the country because of Web site problems with the new Affordable Care Act. He used this as an example of how the work that system administrators do has changed a lot in the past ten years: we are now at a point in which a Web site like Twitter or Facebook being down is treated like a national or world catastrophe. With this basis, he went on to speak about how to use already-existing manufacturing knowledge to improve modern IT practices.

Ben next talked about several models of system management. He described the DevOps flow, showing a diagram that demonstrated the flow from developers to operators to customers and back again. He encouraged a holistic view of systems design, noting that value in systems comes from the sum of the behaviors of the parts.

After spending some time describing how manufacturing has become increasingly automated and efficient in the past 100 years, Ben described 11 practices that easily can be translated from manufacturing to system administration. These included building trust relationships, sharing vision, standardizing work, eliminating waste, process improvement, and spending time where the work happens, among others. In each case he showed how manufacturing has used the practices to make their processes more efficient and increase productivity, and described how the same practices can be used by system engineers to improve their own workflows. Ben closed with words of wisdom: know who you are, know where you are, and know where you want to be.

An audience member noted to Ben that eliminating waste needs to tie into global optimizations, to which Ben replied that he was correct and that doing anything just for the sake of doing it is wrong. Another person wondered whether the way the talk tied manufacturing to software systems was a good fit, because software is a lot easier to change than a factory. Ben suggested that the questioner should spend more time in a factory, as they are a lot more malleable than he thinks. He then noted that the principles behind the manufacturing practices are actually the important parts.

Invited Talks 1

Summarized by Cory Lueninghoener (cluening@gmail.com)

Systems Performance

Brendan Gregg, Joyent

Brendan Gregg was originally scheduled to speak about Flame graphs, but that talk was moved up to a plenary session due to a cancellation. In the talk's place, Brendan spoke during this session about how systems performance monitoring has changed since the 1990s. He started by defining systems performance: the analysis of applications down to bare metal. He noted that this is an activity for everyone, from casual to full-time system analyzers.

With that definition introduced, Brendan presented a series of comparisons between how performance analysis has changed in the past 20 years. First he looked at systems performance as a whole: in the 1990s, text-based tools were used with inference and experimentation to gain insight into systems; in the 2010s, we have a lot more insight into the systems and can use standard methodologies to instrument them. In the 1990s, operating system internals were studied in textbooks; in the 2010s, operating system internals are studied via dynamic tracing. Systems in the 1990s had murky observability, with parts of the system hidden; systems in the 2010s are incredibly transparent. The 1990s had text-based or simple line and bar graph performance visualizations; the 2010s has heat maps and Flame graphs that display more information more easily. Finally, in the 1990s, we used ad hoc checklists and the tools we had at hand as performance methodologies, while in the 2010s we have workload characterization, drill-down, and other methodologies at our disposal.

Hacking Your Mind and Emotions

Branson Matheson, SGT

Branson Matheson gave a highly engaging talk about social engineering, covering both how to use the practice to your advantage and how to prevent yourself from being taken advantage of. This wide-ranging talk started out with some keys to being a good social engineer (confidence, ability to think on your feet, knowing your limits, theatrics) and some indications that you may not be so good at social engineering (cracking a smile when trying to pull a prank, freezing up when confronted, not being comfort-

able with play acting). He then talked about some of the basics of social engineering: making what you're doing look normal, keeping your target goal in mind, adapting quickly to change, and establishing a false trust relationship.

Focusing on the concept of trust, Branson described ways social engineers can build that false relationship. These involved such tactics as making yourself look like your target would expect, making yourself or your techniques look official, and exploiting emotions. He then described how some day-to-day social engineering takes place: advertising, businesses, car salesmen, and the police. From there, Branson went through a quick, but thorough, description of how to pull off a social engineering caper.

Branson finished the talk by showing how to avoid being the victim of social engineering. He described defensive actions to use on both the phone and in person, focusing on how to avoid disclosing too much information about yourself or your company. This included a number of concrete suggestions, including how to recognize leading questions and how to still get the perks of giving away information without actually giving away information.

Invited Talks 2

Summarized by Tim Nelson (tbnelson@gmail.com)

The Efficacy of Cybersecurity Regulation: Examining the Impact of Law on Security Practices

David Thaw, Visiting Assistant Professor of Law, University of Connecticut; Affiliated Fellow, Information Society Project, Yale Law School

Thaw is both an attorney and a computer scientist. His research investigates how cybersecurity legislation affects what security admins do in practice. Surveys can be annoying and often don't ask the right questions, so Thaw's group interviewed chief information security officers at several large companies. Privacy concerns prevented him from explicitly naming the companies involved.

His interviews found that most were unaffected by laws such as Sarbanes-Oxley, but that others, especially the new breach-notification laws, had a significant impact. The impact was not always positive: because the laws mandated reporting a breach only if the lost data was unencrypted, they resulted in a flurry of investment solely in encryption. In some cases, this disrupted other aspects of security practice by diverting time and money into encryption, regardless of whether they might have been better spent elsewhere.

Thaw's group went beyond interviews and quantitatively measured the effect of these new laws. To do this, they examined online breach reports from 2000 through 2010. Although using breach-reports as a metric is not perfect, at least it addresses whether an entity is focusing on security. He found that the number of breaches reported increased in August 2004 (when the laws took effect) but began to subside by September 2008. The decrease in reports was much sharper for previously unregulated entities, which suggests that the breach-reporting laws did have a positive effect on security practice.

Thaw presented his results to Congress and helped prevent the passage of a bad data-breach reporting law. The study was performed more than two years ago. Because the world changes rapidly, Thaw would like to do a follow-up study with more sysadmins. He opened the floor for discussion early. Someone asked how badly these laws were written and whether they might allow people to encrypt poorly and ineffectually. Thaw replied that the laws varied by state, but that in many cases the wording was quite weak. For instance, "renders unreadable" may be satisfied by ROT13. Further, nothing prevents a law from hard-coding specific encryption algorithms, rather than references to current best practice.

Eric Smith noted that Thaw's research largely looked at consumers of security products, and asked whether he had investigated producers as well. Thaw answered that he would like to in his next study. Rik Farrow asked about the distinction between prescriptive and standards-based legislation, noting that HIPAA was much more flexible than the breach-reporting laws. Thaw favors standards-based legislation because it forces companies to think about security and come up with a plan, rather than following specific narrow requirements. He went on to say that security is only one-third technical, because the administrative and physical aspects are also important.

Someone else asked whether in a small organization a flexible law would boil down to a directive. Thaw answered that HIPAA, and similar legislation, takes size and capability into account. When asked about the process of reporting breaches, Thaw mentioned that very few breach laws actually have a centralized repository for reporting, and that the Open Security Foundation's data has become difficult to obtain, making his research more difficult.

The Intersection of Cyber Security, Critical Infrastructure, and Control Systems

Sandra Bittner, CISSP, Arizona Public Service, Palo Verde Nuclear Generating Station

Bittner began her talk stressing that safety is the first concern at a nuclear power facility. The US nuclear industry is required to apply lessons learned in one facility (called operating experience) to all US commercial facilities to prevent similar situations from occurring in other US facilities. The Nuclear Regulatory Commission (NRC) requires, in the area of cybersecurity, that all US commercial facilities protect their facilities against nuclear sabotage and the release of radiological material. The facility is free to choose the cybersecurity techniques, methods, and implementations they use but are subject to rigorous inspections and approval by the NRC. Her facilities are some of the largest generators of electricity in the US, and are meant to have a lifetime of 60 years.

She spoke about the facilities, but had to remove some pictures and material from her presentation upon being told that her talk would be recorded. She summarized some challenges inherent in securing control systems such as power plants, wastewater treatment plants, etc. These systems often contain both analog and digital components, a plethora of sensors, and electronics that run nonstandard protocols. Because certification is required, and recertification may take as much as a year, sometimes critical infrastructure operators must decide to delay software patches, or even not to patch at all. They are therefore heavily dependent on physical and network isolation. Stuxnet presented something of a wakeup call to the industry, because the cyberattack used on an Iranian nuclear enrichment facility made use of portable media to carry the infection past the "air gap."

In IT an outage is often the worst thing that can occur, but in their situation, an outage is far less worrisome than losing control of the facility. Critical infrastructure is also revisiting the topic of responsible vulnerability disclosure rules to exercise caution in areas that could incite public panic. Another topic of interest was forensics challenges for critical infrastructure because facilities are not easily shut down without public risk of loss of life. Bittner also noted that there are many cybersecurity "dialects" spoken, especially at a large facility. There can often be a terminology gap between the IT staff and the Operations staff (OT), and it is vital to send IT staff to plant training so that they understand the entire system.

Rik Farrow asked about frequency of software updates, mentioning that later versions of Windows have improved significantly. Bittner answered that they have a time-to-live on all of their equipment, and they can often only replace and install equipment during an outage, which may occur years apart. Thus, the last good version of software installed may be past its formal end of life by the time it is replaced. Farrow also asked about certifications necessary to work in a nuclear facility, and whether there was concern about validity. Bittner replied that there are required certifications and that in the area of cybersecurity she expressed concerns that established public certification programs should work hard to discourage the dilution of their credentials, and pull credentials when necessary.

Mark Bartlet noted that some recent compromises of control systems were done remotely, and asked what rationale could possibly lead to these systems being connected to the public Internet. Bittner replied that in those cases typically there was a lack of understanding about cybersecurity for the control system and a drive for business efficiency that conflicted with maintaining isolation barriers. Overall business will always want quick access to production control system information, but this should be weighed against the cybersecurity risks.

Invited Talks 1

Summarized by Tim Nelson (tbnelson@gmail.com)

A Guide to SDN: Building DevOps for Networks

Rob Sherwood, Big Switch

Sherwood began his talk by observing that network configuration has changed very little since the mid-'90s. In that time, we've made incredible strides in managing our server systems, but the networks themselves lag behind. Companies build vertically integrated networking appliances, and we lack the equivalent of Chef or Puppet for our routers. Software-Defined Networking (SDN) is a step in this direction, and is largely enabled by the OpenFlow protocol—SDN is to OpenFlow as the Web is to HTTP. Using OpenFlow, programs running on a centralized controller machine instruct the switches, making true network-programming possible. The need for automation and custom development lead to the DevOps trend, and Sherwood compares SDN to DevOps for networking.

Sherwood was careful to explain that SDN controllers are only logically centralized; they may actually reside on a cluster for fault-tolerance and load-balancing. Thus the controller does not actually represent a single point of failure.

As a motivating example, Sherwood brought up backup management. Large-scale network backups require reservation of bandwidth, ports, and devices, as well as a keen understanding of future bandwidth requirements. Configuring this consistently can be difficult. In contrast, an SDN-enabled application can (in a sense) talk to the network and schedule an appropriate time for the backup operation. This is analogous to scheduling in an operating system.

Switch hardware is already fairly flexible, and so we need only to leverage it intelligently to implement SDNs. Current work involves bringing OpenFlow up to the pre-SDN level of features and performance. OpenFlow-enabled switches are inexpensive, and, more importantly, one can Google to find prices, because the switches need not be proprietary hardware. There are also numerous software switch programs available, such as Open vSwitch. Openflow has also enabled or inspired many new tools, including the network simulator Mininet; benchmarking tools for SDNs like cbench and oflops; and Onie, which allows OpenFlow switches to net-boot and even dual-boot.

Someone asked why there wasn't an "OpenFlow for generalists" akin to Chef or Puppet. Sherwood replied that OpenFlow isn't the right level of abstraction for that, but that languages are already being built on top of OpenFlow. Another person asked how well OpenFlow works with legacy hardware. Sherwood answered that while switches need to be OpenFlow-enabled to use OpenFlow, such switches usually have a hybrid mode that enables bridging between OpenFlow and traditional networks. When asked what happens if the switches lose their connection to the controller, Sherwood replied that since the controller pushes down rules that persist on the switches, the network

won't simply stop. Instead, how well the network works will depend on how good the rules themselves are.

OSv: A New Open Source Operating System Designed for the Cloud

Nadav Har'El, Cloudius Systems Ltd.

Nadav Har'El spoke on OSv, a new virtual operating system. OSv is intended for guest processes in the cloud, and Cloudius hopes that it will become the default operating system for virtual machines. Har'El began by showing the virtualization stack with hardware at the base up through the host operating system, hypervisor, guest OS, JVM, and finally the running application at the top. He pointed out that many of these components duplicate effort. For instance, the OS, hypervisor, and the JVM all provide protection and abstraction in the interest of virtualization. OSv removes this duplication from the OS.

Mainstream operating systems were written with many assumptions that no longer fit the most common uses of virtual machines. Today, a VM often runs only one process (like memcached), has no users, and requires few services. Even the file system is not always needed. OSv was written from scratch to take advantage of this.

In OSv, while multithreading is allowed, there is only ever one process running at a time. It therefore has no separate kernel address space, and system calls are just function calls. It also provides a ZFS file system, an efficient (netchannels) TCP/IP implementation, and provides enough Linux emulation to run binaries. It can be run in KVM/XEN now, and VMware compatibility is expected soon.

Narayan Desai asked how one would debug applications in OSv if there is only ever one active process. Har'El replied that, for the moment, they attach debuggers to the guest process itself on the development machine. He also pointed out that OSv is still young, and they are still adding features to support debugging.

It was also mentioned that OSv's one-process architecture looks remarkably similar to Novell NetWare from 15 years ago. Har'El replied that OSv is about more than just the single process. For instance OSv is a virtual OS, and NetWare was not.

Rob Sherwood asked what a smarter hypervisor might do if it could assume that the guest processes were all running OSv, and whether we would just end up with containers at that point. Har'El answered that he does not think that the hypervisor will be going away anytime soon, but if OSv becomes predominant, he envisions hypervisors taking advantage of the guarantees OSv provides.

Invited Talks 2

Summarized by Jonathon Anderson (anderbubble@gmail.com)

Enterprise Architecture Beyond the Perimeter

Harald Wagener, Systems Engineer, Site Reliability Engineering; Jan Monsch, Security Engineer, Google Security Team, Google

Jan Monsch and Harald Wagener presented an Internet-based network security model that serves as an alternative to a more traditional perimeter-based model.

They described a baseline enterprise network security model as a system of "moats and castles." Security in this system is traditionally based on a trusted, privileged intranet zone that is logically separated and protected from other intranets and the public Internet. In this model, access to privileged services is granted based on access to the trusted network that is protected by "high walls" and "strong gates." Remote access is granted to trusted users by granting access to the protected network, often by way of a virtual private network connection. Attackers have evolved to address this specific security model, requiring them to exploit only the weakest point in the security perimeter in order to gain access to the network and its private services.

Google has addressed the problem of internal network security by looking "Beyond Corp." In this network security model, no privilege is associated with having access to the internal network. The perimeter itself is replaced with client certificate authentication that can be coupled with existing user authentication systems. This has improved their user experience by allowing their users to work from anywhere on the public Internet, with the only limits to internal service based on configurable policy. This has also allowed more user workflows to be moved to cloud services, as physical and network location is no longer a factor in authentication and authorization.

The Overcast architecture blueprint combines an authentication certificate database with a meta-inventory of system state. This allows service authorization to be based on the metadata associated with the client system. Does the system use full-disk encryption? Is it a desktop or a laptop? What software is installed?

The workflow patterns supported by this new security model have presented a few challenges. Access to network-attached storage is less viable on the now pervasive wide area network, and users should be moved to asynchronous local storage wherever possible. Certain legacy workloads have proven difficult to port, necessitating some traditional VPN access; but these networks can be reduced in scope around only these legacy services.

Further implementation details and examples, as well as the discussion of code availability, was deferred to more specific talks the following day. For more information, see "Managing Macs at Google Scale" and "OS X Hardening: Securing a Large Global Mac Fleet," below.

Drifting into Fragility

Matt Provost, Weta Digital

Matt Provost used themes drawn from the books *Drift into Failure* by Sidney Dekker and *Antifragile* by Nassim Taleb to reframe and redirect efforts to improve stability in complex systems.

As an example, he opened with a story of a seemingly simple deployment of an NFS cache. In keeping with best practices, the service was tested and deployed in stages to progressively larger sets of the production environment. Yet, within two hours of deploying the seemingly good service to the entire facility, all production operations had ceased: the deployed cache had been compiled as a 64-bit system, but the application using that cache was compiled as a 32-bit binary. Despite testing, the problem only became apparent when the cache delivered an inode with a number larger than 32 bits.

Systems are often fragile, denoted by disruption in the face of variability. One response to such fragility is to make the system "robust," often by introducing redundancy; but redundancy often impairs the ability to detect systemic problems during testing by hiding problems within individual components.

Public root-cause postmortem reports provide further insight into the difficulty of troubleshooting complex systems. Examples from Amazon, GitHub, and CloudFlare all showed ample preparation and past experience that seemed to guarantee success yet failed to anticipate the interaction of components in the system.

Matt advocated the creation of "antifragile" systems that improve in response to variability rather than deteriorate. One approach to creating antifragility is to generate artificial variability in the system, making this variability routine rather than exception. As one example, he referenced the Netflix "Simian Army" (see "How Netflix Embraces Failure to Improve Resilience and Maximize Availability," above) and provided his own simplified "Cron Monkey" implementation: by rebooting development servers automatically and regularly, he prevented these servers from drifting into production use. Also, single-points of human failure can be detected by simply going away from the system for some period of time.

He also advocated the simplification of systems wherever possible, even going so far as to disable file-system snapshots to ensure that backup services are reliable. Legacy systems should be actively decommissioned, though, in Q&A, he admitted that variety in user culture can make it very difficult to remove a service once it has been deployed.

Papers and Reports

Summarized by Carolyn Rowland (unpixie@gmail.com)

Live Upgrading Thousands of Servers from an Ancient Red Hat Distribution to 10 Year Newer Debian Based One

Marc Merlin, Google, Inc.

Many of us in the community keep an eye on Google because they're doing what we're doing but at scale. Often the lessons from Google show us what is possible when you must automate because it is too expensive not to do so. This presentation is no exception. Marc presented Google's solution to updating and maintaining thousands of Linux servers with minimal effort.

He started by talking about their in-house Linux distribution: ProdNG, self-hosting and entirely rebuilt from source. All packages are stripped of unnecessary dependencies and libraries making the image quicker to sync, re-install, and fsck. Google chose Debian because it had a better base of packages than Red Hat. With previous distributions, they spent too much time packing software that wasn't included in the distribution. They considered Ubuntu, used on their workstations, but it wasn't ideal for the servers.

The first step was to check the golden image into revision control. When they were ready, every server started booting the same image at the same time. They also found that they could do a side-by-side diff of images as needed for testing. Ultimately, there was no way to guarantee that they could seamlessly switch distributions from Red Hat to Debian. It was hard to find beta testers or to convince their internal users to run production services on a very different Linux distribution.

They had to find all packages that inherited from RH 7.1 that were never needed (X server, fonts, font server for headless machines without X local or remote, etc.). It's amazing how many packages come with a distro that you don't really need on servers. They converted the remaining Debian packages (debs) to rpms, keeping both distributions libc/binary compatible.

Sometimes it's the little things that bite you.

The coreutils upgrade was scary due to amount of breakage created upstream. The day they removed /etc/redhat-release is the day they broke a bunch of Java that parsed this file to do custom things with fonts. One rule on the team was whoever touched something last was also responsible for fixing the breakage, revert the change, fix the bug, try again later.

Then they started the painstaking upgrade of around 150 packages carefully stripped and built from scratch in a hermetic chroot environment. The process was to upgrade a few at a time by building them in ProdNG and putting them in both ProdNG (not used yet) and the current image.

After upgrading the packages, they would throw a regression tester at the new image, run all of the daemons, make sure they come up, test crash reboots, and test reverting to the old image for completeness.

There were some lessons learned along the way:

- Maintaining your own sub Linux distro in-house gives you more control (if you have in-house expertise).
- At large scale, forcing server users to use an API and not to write on the root FS definitely helps with maintenance.
- File-level syncing recovers from any state and is more reliable than most other methods.
- You'll be able to do crazy things like switch distributions.
- Don't blindingly trust and install upstream updates, because they could conflict with your config or even be trojaned.
- If you can remove services/libs you don't need, that leaves you with fewer things to update and fewer security bugs to worry about in the future.
- Running the last Fedora core or Ubuntu from five months ago is more trouble than it's worth. Full updates of servers every three years is reasonable, so Debian stable or RHEL are the way to go for servers.
- Depending on your server setup, partial upgrades may be better for you. Choose a distribution that allows this, such as Debian.

David Ehle, Argonne Laboratory, asked why they built everything from source. Marc responded that when you take binaries, you only have signatures, and you're trusting Red Hat. You're not sure there's not a trojan in the binary. If you build from source, you can have your own optimizations, and you can update with some stack protections. You can use the compiler options you like. You have more luxury (control). We do modify some daemons (e.g., better command logging).

Managing Smartphone Testbeds with SmartLab

Georgios Larkou, Constantinos Costa, Panayiotis G. Andreou, Andreas Konstantinidis, and Demetrios Zeinalipour-Yazti, University of Cyprus

The number of mobile devices in use today is now exceeding the number of laptops. There is the tablet, Raspberry Pi, smart glasses, smartbook phones, and 53% of smartphones in 2016 will be running Android. People are also not changing their phone OS when a newer version is released because manufacturers are not supporting the older phones. This means a lot of old phones running old operating systems are still in use. How can a smartphone developer cope with this software/hardware fragmentation?

Georgios Larkou presented SmartLab (http://smartlab.cs.ucy.ac.cy/), a comprehensive architecture for managing mobile and virtual smartphones through a Web browser. You can manage devices, issue shell commands, transfer files, see the debug information, and do sensor mock-ups as well. You can even go to SmartLab and rent a smartphone. Imagine you are a developer and you want to test your app on 50 different smartphones. You can do that with SmartLab. It also gives you a platform to manage all of our gadgets in the future. You might need to handle a fleet of devices installed on 1000 buses. SmartLab will support that next year. The SmartLab developers are talking with a local ISP and will have smartphones running on buses in the future. They will monitor all of them to get data for other experiments. Additionally, people tend to change smartphones more often than they change computers. It would be advantageous if SmartLab could use all of these smartphones to build Beowulf-like clusters.

This architecture has been evolving since 2009. SmartLab provides fine-grained control over ARDs (Android Real Devices) and AVDs (Android Virtual Devices). There is remote file management, and a home directory and /share directory, and access to all SD cards from the devices. If a user drag-and-drops a file from one device to another, it will copy through the SmartLab architecture. If the person drops it in the /share directory, then it will be distributed to all Android devices. The same thing happens with the remote shell; the user can send commands to multiple devices.

SmartLab builds upon standard open tools for its architecture. Benefits of the SmartLab open architecture include experimental repeatability, urban-scale deployment, federation issues (similar to PlanetLab) so others can contribute, and personal gadget management.

David Lynn (Vanderbilt University) asked whether the cell radios in the actual devices are active. Georgios said no, because they don't have SIM cards on the devices. They are talking with an ISP to provide mobile data. Marc Merlin wondered how the virtual devices compare to simulators that have good native speed. Georgios answered that the real devices are not good for I/O-intensive experiments, as it takes too much time to push and install an app on multiple devices; on the other hand, it takes much longer to install the app on the simulators. Marc then pointed out that if you're running out of memory because you have a virtual disk, then you are fighting for a disk head. This is still useful for some applications but for others you might be better off running simulators. Georgios responded that they want everyone to be able to have access to real devices.

YinzCam: Experiences with In-Venue Mobile Video and Replays

Nathan D. Mickulicz, Priya Narasimhan, and Rajeev Gandhi, YinzCam, Inc., and Carnegie Mellon University

The YinzCam team started with wireless capabilities in the football stadium in Pittsburg. It was a distraction-filled work environment, and inefficient because they had to have someone on-site at the venue every game. How would this have scaled if they continued to do on-site operations for multiple teams in multiple cities?

They took on four new installations in 2011 and had to fly across the country to San Francisco for every one of their home games. The costs would have continued to increase, especially the travel costs for the on-site person.

The team thought they could centralize their operations in a single remote location. This would eliminate the distractions from the on-site environment and allow efficient communication between staff. Plus, they could cover for each other. It also

meant a reduction in travel costs because everyone would be together at the remote site.

They would need to provide reliable access to remote resources. Using a VPN would provide secure connectivity between all sites, allowing the team to monitor content remotely. They are only allowed to publish content inside of stadiums because of licensing restrictions, but they can play that content over the VPN without breaking those restrictions.

They got to the point of having 18 in-venue installations, 83 source video streams, and 98 physical and virtual machines.

They leveraged Amazon Web Services (AWS), connecting to the nearest AWS region (east or west), then established a connection back to the other AWS region. Now they could have a remote ops workstation through a remote management gateway into the servers inside of the stadium. They reduced costs in 2012 and 2013 because they only had to go to the actual site for a one-time installation', and that was it for the rest of the year.

Another challenge was how to quickly and accurately generate replays of game footage. How it would work: a fan goes into the application and pulls up the highlights list. Then s/he selects the replay after choosing the camera angle. Originally this was a manual cutting workflow that worked like this: an interesting event occurs during a game, a remote YinzCam operator clicks a button at the end of the event, a tool searches back in time for the start of the event, the system produces a video file, the operator annotates this file with metadata, and, finally, the system publishes the video in the mobile app.

This method was time-consuming, error prone, and repetitive, and the manual metadata generation took time. It required full operator attention, and there was no pipelining (e.g., there was only one replay at a time).

The team resolved this by using resources for what they do best: machines are well-suited to repetitive tasks such as approximating replay boundaries and encoding video. Humans perform well on visual tasks such as visual verification of the instant replay window. They also needed to allow remote override of automated decisions because these decisions might be incorrect. Humans would verify replays.

The automated replay cutting architecture included two data feeds: one was video, the second included the statistics about the game. Luckily timing rules allowed the team to estimate the start and end times for every replay in the game. Video streams could also be easily duplicated in cache. The team set up a proxy server that split the content to all existing clients. Segments of video were served over HTTP while standard proxies were used to cache video data. Public clouds have ample bandwidth; you pay per GB, but the bandwidth is practically infinite. The requirement was for five hours of uptime on game day only with rapid scale-up and scale-down on demand.

Once the stream was in the cloud, it was migrated to a live streaming server. The team produced a copy of content to the cloud so it could be served from the cloud using the bandwidth available there. A copy of the replays and livestreams were sent to the in-venue WiFi network which operated just as it did before.

Some of the lessons the YinzCam team learned from this effort include:

- Manage operations remotely
  - extract ops from field environment
  - ensure easy, reliable access to remote systems
- Invest in system automation
  - automate error-prone manual processes
  - allow overriding of automated processes
- Take advantage of the cloud
  - short-term CPU-intensive processes
  - bandwidth-intensive processes and services

David Lynn (Vanderbilt University) pointed out that the Tennessee Titans are about three miles from Vanderbilt, and cellular coverage at the LP field sucks. Sprint is being used for wireless, apparently. Nathan replied that in Sprint's case, DAS (distributed antenna system) is often done on carrier-specific lines. When a DAS is installed, it is often installed by a single carrier, and not all of the major carriers will be on it at once. Sprint may not even be on the DAS but some of the other carriers may be. DAS handles a lot more traffic than a single cell tower would outside of the station.

Andrew Myers (Lean Logistics) asked whether they used blackmagic or matrix cards, and how much COTS was used. Nathan replied that they used blackmagic cards and software encoders like ffmpeg. Andrew also asked if they used commercial streaming software or their own, and Nathan responded that they wrote a little bit of software that takes a TCP stream and copies it to a different endpoint. Someone asked how support is handled for the deployed equipment. Nathan said that clients purchase and maintain the hardware.

Friday, November 8, 2013

Invited Talks 1

Summarized by Jonathon Anderson (anderbubble@gmail.com)

Rethinking Dogma: Musings on the Future of Security

Dan Kaminsky, Chief Scientist, White Ops

Dan Kaminsky, perhaps best known for his work pointing out a fundamental design flaw in DNS, offered his thoughts on the security landscape predicted for 2014.

He started by framing the current state of security as that of a hacked-in afterthought: design the system, then make it secure against exploits. In today's globally Internet-worked world, security has become an engineering requirement of its own, but we don't know how to deliver security at scale; we don't know how to authenticate users; we don't know how to write secure code;

and we don't know how to punish the "bad guys." Instead, when system security fails, blame falls on the victim.

In the absence of any existing solution, Dan says that we have no room for religious arguments about how to do security. We need new science with new data and new instrumentation.

Dan likened existing security models to static puzzles. Any puzzle of arbitrary complexity can be solved given sufficient incentive and time; and with no consequence for failed attempts, any security system can eventually be broken as well. Instead, security should be approached as a game: an interactive challenge where the security professional has the advantage of superior information. In essence, the defending player "doesn't have to play fair." Security professionals have a wealth of data signals to mine, including system logs, network activity, and honeypots; and the time it takes an attacker to probe an unknown system for vulnerabilities provides ample time to respond with a consequence. The remaining challenge is in managing the signal-to-noise ratio, reducing false positives as much as possible.

Dan touched on a series of recent trends and events in the security community, including the increasing difficulty of securing virtualized systems; reports of "badBIOS" exploits; the Microsoft and Facebook Internet Bug Bounty (IBB) program; improvements in sandbox technology (particularly asm.js); and controversies arisen from recent revelations regarding the United States National Security Agency (NSA).

The larger issue in securing systems, however, is usability, both for the end-user and the administrator. Key management is always more difficult than the cryptography it uses, and security is more easy to circumvent when it is more difficult to implement correctly.

Dan fielded several questions from the audience. He defended his preference for Windows 8, calling all modern operating systems "essentially single-user machines." He responded to recent research into application sandboxing, calling efforts to extend the Capsicum project into hardware "one of the most interesting things [he's] heard." When asked about Bitcoin, Dan marveled both at the currency's existence and at the fact that it has remained unbroken despite the intrinsic financial incentive to defeat it. When asked about the upcoming termination of support for Windows XP he responded that he's more concerned that Web browsers for the OS might stop receiving security updates, calling the browsers the "gateway to the world." He closed with anecdotes from Google's experience running honeypots, providing recommendations for deploying them as part of a site's overall security strategy.

Invited Talks 2

Summarized by Scott Murphy (scott5@ovsage.org)

ZFS for Everyone

George Wilson, Delphix

George Wilson of Delphix started with a poll of the room checking to see how many are using ZFS. It seemed that pretty much all common platforms are in use and represented by the audience. He then gave a short bio of his involvement from the first introduction of ZFS to Solaris 10 up to his current OpenZFS activities and then gave a quick overview of the ZFS timeline.

He went on in a little more detail regarding OpenZFS. It was launched in 2013 as the truly open source successor for ZFS. When ZFS was being developed, most of the work was done by Sun/Oracle. Now the contributions are coming from across the spectrum. Basically, OpenZFS is an umbrella project founded by Open Source ZFS developers representing multiple operating systems. The stated goals are to raise awareness of the quality, utility, and availability of OpenZFS; to encourage open communication about efforts to improve OpenZFS; and to ensure consistent reliability, functionality, and performance of OpenZFS. This is all coordinated through the project Web site: http://open-zfs.org.

He then displayed a slide showing the involvement of companies with OpenZFS, stating that OpenZFS is becoming a big driving factor for adoption by companies for their products.

Moving on to the management of OpenZFS, he explained how they deal with features across platforms by using "feature flags." He then gave an overview of the feature flags available in OpenZFS. These include the Pool feature, LZ4 compression, compressed L2ARC, enhanced write throttle, per I/O device queues, and imbalanced LUNs enhancements.

Someone asked why the per I/O queue was processed in the order described. George answered that the queues are processed in priority order and checked to see if any of the queues have not gotten their minimum number into the active I/O pipeline. If a minimum has been met for any higher priority queue and there is a queue without anything in the I/O pipeline, then the next highest queue will be processed and the next until they are empty or have reached their minimum. This effectively guarantees that all queues get processed and blocking is reduced.

George went on to say that upcoming features would include enhancements for pool fragmentation and zero block compression. There are many more features in the pipeline.

George concluded with an appeal to get involved with the project. If you are making a product, let them know, if you are a sysadmin, spread the word and contribute to the wiki.

Someone asked, "Is there a feature that would allow you to remove a fragmented imbalanced LUN and redistribute the data across any new LUNs?" George mentioned that there is a block pointer rewrite project driven out of Oracle, but they don't

want to implement it in ZFS; it was a grandiose idea and would ultimately have killed development by making it difficult to add features later. He added that there is an idea to introduce a device removal logic that would allow you to effectively remove the LUN and to recreate a new virtual device that looks like the LUN but then gets data redistributed to it. There is a writeup on device removal that is being tweaked to include both redistribution and device removal which would cover this case.

Someone else asked about the new combined development process and how it feeds back into the various things we actually use. George said that the idea behind the dev process is to take all of the platform-specific layers and tease them apart to create a common ZFS repository in order to allow any platform to take that wholesale. They have also created a testing facility, which is available today in the open source community, that will make it easy to test regardless of the platform. You test on any platform with the same test structure, ensuring it will be available on all supported platforms.

Someone else asked, "Do you have any advice for those still on Solaris who may want to move to this platform?" George answered that for those who are doing migrations, it will be difficult. The easiest way might be to use a ZFS send, but Oracle may have added a feature which will make this incompatible if you are in a newer revision. If you are running Solaris 10 it should be easy. Someone asked a related question, whether there is a version number to avoid exceeding. George responded that if you are running anything less than version 28, you can just import the pool. He also said that they will be putting some migration notes on the Web site and you should check there for more details.

The final questioner asked whether Oracle is interested in the OpenZFS efforts or were they doing their own thing. George said that he was not sure about interest, but they are definitely doing their own thing. The OpenZFS folks have talked to developers about adding feature flags so there would be compatibility, but Oracle has its own agenda and business needs. The community is interested in what Oracle is doing, such as larger block support. Good ideas can come from everywhere.

Manta Storage System Internals

Mark Cavage, Joyent

Mark Cavage, software engineer at Joyent, gave a talk on the Manta Storage System, starting with a short history lesson.

Back in the '80s as part of a magazine article series, Jon Bentley asked Don Knuth to demonstrate " literate programming" using a classic problem, word frequency analysis. He also asked Doug McIlroy (inventor of UNIX pipes) to review the solution. Knuth's solution was done in ten pages using a custom algorithm in WEB, his own literate programming language, with a from-scratch implementation of everything. McIlroy did a one-liner with standard UNIX commands and went on to make several points to illustrate why his solution was acceptable, including the following two: not everyone has a Don Knuth on staff and not all problems are the same, but they may be similar.

This illustrates the UNIX philosophy, which is specifically creating small programs that do one thing and do it well. This also influences the approach to building programs.

Moving the time machine forward to today, the aggregate term "Big Data" provides the same class of problem, illustrated by Google's MapReduce paper. Mark wanted to apply the UNIX philosophy to it. To that end, you need a few items: parallel execution capability, the compute interface to be a raw OS with the standard OS tools, cloud vendors to provide multi-tenancy, and the ability to store an arbitrary amount of data.

Beginning with storage, you have three choices: block, file, and object. Object storage is similar to file, no partial updates, no exposed volumes, and a universal protocol. The challenge is how to make UNIX work with an object store efficiently. Next, you virtualize the OS—in this case with zones. This provides one kernel on bare metal and many virtual containers with their own root file systems. This is much more efficient than hardware-based virtualization. There is also a rich interface between the global zone and the individual tenant zones.

Putting it all together, Manta is a scalable, durable HTTP object store with a namespace that looks like a POSIX file system with in situ compute as a first-class operation.

Mark then did a successful demo, the Manta version of "Hello World," demonstrating that the file system was available on his laptop talking to the Manta cloud. The demo showed a prototype UNIX environment running on an object store.

He went on to describe the Manta design parameters. They chose to be strongly consistent vs. eventually consistent. This is based on the CAP theorem (Brewer, 2000) which can be paraphrased as "on any storage system in the face of network failures, you can choose for the system to be consistent or available, but not both." They chose strongly consistent because in an eventually consistent system, nothing up-stack can ever be consistent again. If you build strongly consistent, you can always put an eventually consistent layer up top. An object store needs to be strongly consistent for maximum functionality. In HTTP terms, system up returns 200s, system down returns 500s. Given these parameters, their system is built to be highly available. They can tolerate everything up to a single datacenter failure—every object in Manta must be a contiguous file.

He then showed a diagram that resembled the classic three-tier architecture, in this case front-end, metadata, and raw storage. The front-end consists of Stud (SSL terminator), HAProxy (HTTP terminator/Load Balancer), a Web API (restify on Node.js), and Redis (authentication cache).

The metadata tier consists of Moray (custom node.js key/value interface), metadata copies on 2+ Postgres databases, replica-

tion topology managed via ZooKeeper, and a consistent hash on directory names.

Raw storage is a custom bare metal system called the Manta Shrimp, which consists of 73 TiB in a 4U 256 GB DRAM RAIDZ2 box, SmartOS (ZFS, Zones), storage interface (nginx), and the ability to support compute jobs.

The final part of storage is how they implement group membership, which is through a custom DNS server in Node.js. Participants write an "ephemeral node" in ZooKeeper on startup. This "mostly" works, as the name service caching daemon (NSCD) and ZooKeeper have issues.

Mark continued with a discussion of the compute flow. Users submit jobs which specify pipelines to run either on each input separately (Map) or all inputs together (Reduce). Inputs are objects accessed as regular files, outputs are saved as objects. User programs run inside transient zones managed by the service. Resource usage is capped but makes allowance for bursty activity. Input is taken via objects mapped in as read-only files (for Map) and redirected as stdin. Upon completion, ZFS rollback is performed and the zone is rebooted.

Mark revisited Bentley's challenge from earlier showing the Manta variant of McIlroy's solution to the challenge with parallelism and did a demo using a word find on all the works of Shakespeare. It was fast.

Customer usage tracking (aka metering and reports) uses Manta. Every *N* minutes, they upload database snapshots. Once the snapshots are uploaded, a job is kicked off to aggregate usage by customer. There is nothing special here, just MapReduce.

Garbage collection is an interesting problem—links in Manta are copy-on-write UNIX hard links. On every delete, the record's state is recorded and then Manta crunches the delete logs and all live records by hourly sending all records about an object ID to the same reducer. The reducer can then determine whether the object is truly garbage; all garbage objects are emitted to Manta as a list per backend server. The backend server then polls Manta for new objects to delete.

Mark concluded with a series of implementation lessons. The modules node.js, bunyan, restify, fast native-dns—and, particularly, MDB, DTrace and Flame graphs—were handy for debugging; Stud is very buggy. If you are going to use nginx, make sure you have the version with the fsync patches. nginx resource requirements will need tweaking. They use ZooKeeper for now, will hopefully find a replacement soon. Postgres is running with lots of churn with a 24/7 duty cycle, so there will be a lot of tuning, vacuuming, analyzing, and table fragmentation.

There were no questions.

Invited Talks 2

Summarized by Georgios Larkou (glarkou@cs.ucy.ac.cy)

Apache Hadoop for System Administrators

Allen Wittenauer, LinkedIn, Inc.

Allen started the presentation by asking about Hadoop adoption among the attendees, continued with a brief MapReduce and Hadoop introduction, and provided some real examples of Map and Reduce functions in Perl and Python. Next, Allen provided some background information about Hadoop genesis and how Hadoop evolved over time, from the Google's MapReduce Framework and file system to the open source version we all use today.

Hadoop binds the MapReduce programming layer and the HDFS file system together in order to provide support for file-system-agnostic behavior since MapReduce does not care if you are running HDFS on Amazon's S3, for example, or GlusterFS, or any other file system (e.g., ext4 or ext3). HDFS uses the NameNode to provide a dynamic metadata store for each file, similar to an iNode server. Additionally, Hadoop implements a scheduling framework and uses a JobTracker to schedule tasks on a distributed cluster. The bad news is that Hadoop is built by developers for developers, thus it is not designed for system administrators and/or support staff.

Allen pointed out some of the Hadoop problems (e.g., single point of failure) and provided some suggestions for (junior) administrators, including not to always trust the logs since most of the time, tailing the logs will not tell you the whole story. Allen asked the audience to remember just one character, a "%" (percentage), since that character is the key to making Hadoop work. In order to better explain why, he provided an example from nature, an ant colony. He said that if the colony loses an ant, that is not a big deal, the colony will survive without any problems; but if the colony loses the ant queen, it will probably face a variety of problems and might not be able to recover. Consequently, he asked the audience to think of the master node in the Hadoop framework as if it were an ant queen.

The solution that Allen suggested to the audience was to use a monitoring system to monitor the master nodes, and he presented a custom build of Nagios that they use at LinkedIn. He also suggested performing health checks on the NameNode and the JobTrackerNode every "x" amount of time. Additionally, he suggested extensively testing everything before running the framework, and noted that even if you introduce simple tests such as moving files to HDFS, performing simple MapReduce tasks can save a large amount of time since you will be able to detect most of the errors before you turn on the framework for production. Finally, he suggested monitoring resource usage and mentioned that developer teams or individual developers will not understand any Java heap problems, but they should always write their code as if they had limited resources. For this, Allen suggested the introduction of a quota system.

In terms of Hadoop security, Allen suggested always being aware of any open ports and noted that since it is difficult to revoke a user-level certificate, administrators should use Kerberos. Kerberos supports the use of Active Directory certificates and provides mechanisms for easily revoking user certificates. As a result, the administrators will have more time for building "cool infrastructures" than doing boring user management.

Lastly, Allen presented the "White Elephant" open source project contributed by the LinkedIn team. White Elephant provides tools for monitoring an individual user's usage by analyzing the JobTracker logs for every job submitted to the system. The tool allows the Hadoop administrators at LinkedIn to ensure that every developer obeys the fair usage policy.

Someone asked whether putting a quota on everything might introduce big hits on metadata operations. Allen responded that they did not observe any kind of metadata problems since Hadoop actually kind of cheats; Hadoop actually knows the block size when you ask for file storage and thus can calculate the quota you may potentially use, based upon the default block size and the number of blocks the user asks for.

Someone else asked for advice regarding the naming scheme of the file system or machines, especially if you are going to have more than one cluster. Allen responded that they use generic names for everything. He also provided an example of how LinkedIn had chosen the naming scheme for their machines and some best practices based on their experiences.

Jennifer from Yahoo!, noted that Allen is one of the active Hadoop committers but not one of the project management committee (PMC) members. Jennifer expressed her concern that Hadoop has a lot of operability issues and asked whether there are any people from the PMC who are operational-side focused and, if not, how can Allen get on the PMC and how can she vote for him. Allen responded that he does not believe that anyone from the PMC is operational focused. Regarding how he can get elected to the PMC, Allen said that it is a much more difficult question which he cannot publicly answer.

An attendee asked Allen whether he knew any framework dedicated to automatic health checks for Hadoop. Allen responded that he is not aware of an existing framework but it would be advantageous if anyone could provide directions on how to build one or contribute to developing an open source framework for health checks.

The last question concerned security and how an administrator can actually secure a Hadoop cluster. Allen responded by providing some examples from their configuration at LinkedIn. 'LinkedIn uses iptables for firewalling their clusters, and when they finish configuring the whole cluster they usually export the resulting iptables and provide them to the network administrators to implement in the network stack. Allen also expressed concern regarding the fact that switches might not have enough memory to implement the whole iptables configuration.

Optimizing VM Images for OpenStack with KVM/QEMU Fall 2013

Chet Burgess, Senior Director, Engineering, and Brian Wellman, Director, Operations, Metacloud, Inc.

Brian presented Metacloud's experiences running production clouds for their clients and, as a first step, he presented all software versions of the tools covered during the presentation. He continued by presenting the most common disk formats, RAW and QCOW2, and a VM container format, AMI. RAW disk format provides a direct representation of a disk structure; it can be sparse, is widely supported by hypervisors, and can be treated as a block device. On the other hand, QCOW2 (QEMU Copy on Write version 2) format is part of the QEMU project and supports pre-allocation as well as on-demand allocation of blocks. QCOW2 supports a wide range of features such as read-only backing files, snapshots, compression, and encryption. Additionally, Brian presented a short comparison between the two formats and some suggestions based on their experiences. Finally, Brian presented the AMI (Amazon Machine Image) container format, which consists of three different files: the AMI raw disk, the Amazon Kernel Image (AKI), and the Amazon Ramdisk Image (ARI).

During the second part of the presentation, Brian illustrated how a user requests and launches an instance by selecting the flavor of an instance (e.g., quantity of CPUs, RAM, disk sizes). All users' requests are scheduled to the nova-compute node which is responsible for making the image available locally. Additionally, Brian provided some information regarding the procedures that nova follows in order to allocate a disk file of either RAW or the QCOW2 format where the default disk format is nova.

During the third part of the presentation, Brian presented best practices for preparing an image OS for a cloud environment. He started by presenting a bunch of essential tools such as QEMU, cloud-init, etc. Additionally, he showed how these tools can be easily installed and configured on Ubuntu and CentOS.

Brian presented best practices on how an administrator should prepare a cloud image regarding authentication, networking, and hotplug support. He suggested that, as a first step, administrators should disable SSH password-based logins and disallow remote root logins via SSH. Administrators should create a single user and add this user to the administrators group and allow root login without a password from the virtual console. At this point, he noted that there is no actual security benefit if you introduce passwords, and it will introduce a lot of overhead in case you would like to change the administrator password on all pre-distributed machines. Moreover, Brian suggested that administrators should allow MAC addresses to be generated dynamically each time an instance is spawned and eliminate MAC address binding. Additionally, he recommended configuring DHCP clients for persistence. The reason behind this is that a VM should never give up trying to obtain a DHCP lease, because otherwise the machine might fall off the network and require administrative intervention. Finally, he demonstrated

how an administrator should configure a machine for hotplug support in order to prevent rebooting while attaching or detaching a hard disk drive.

Brian concluded his presentation by summarizing the contents of his presentation and providing his contact details for the audience.

An attendee from the University of Maryland noted that since Metacloud provides everything as a service, they have access to their customers' data and asked if they have any policies or auditing mechanisms. Brian answered that Metacloud is actually building and operating the cloud using their clients' datacenters, and thus they are not using their own infrastructure or datacenters. Everything resides inside their customers' firewalls, so from that point of view they follow the same guidelines that their customers follow regarding data protection and auditing.

Invited Talks 3

Summarized by Carolyn Rowland and Rik Farrow

Managing Macs at Google Scale

Clay Caviness and Edward Eigerman, Google Inc.

If you work at Google and you want to use a personal platform other than Mac, you have to make a business case for it. Google has made a big push away from Windows, to using Macs for desktops. Google has 64 offices around the world (http://www.google.com/about/company/facts/locations/) and 42,162 permanent employees as of the third quarter of 2013, as well as lots of contractors, temps, and vendors. As of the most recent month where they have login statistics, they are managing 43,207 Macs.

Google has a MacOps team charged with keeping those machines secure and operational. They use none of Apple's management tools, as Apple hasn't had a major release of their management tool since 2006. And since the release of the iPhone, they've lost their attention as far as enterprise management tools. Instead, they use open source tools and a few commercial products.

Google is very security and privacy conscious and has a very strong bias toward open source. If they don't find an open source tool, they have a very strong bias toward building their own tool. They are trying very hard to move away from the corporate network. They believe that all of their tools should work on the Internet, not only on the Google network. They support everyone from software engineers and SREs to masseuses and chefs.

They are also moving away from the corporate network to the Bag of Infinite Holding. They encourage their users to keep their data in the cloud and do everything they can in the cloud.

Google uses a single monolithic image for all users. They can do this because their package and config management are so good that they can produce a single image and customize it through other management tools. And Google is open sourcing the projects we use to do this.

Google uses Puppet for configuration management. They only do enough Ruby to get confused about it, and have clients pull configuration with the master. They use a sibling product called format that gathers information and sends it up to the server. You define the state of the machine as you want it to be and then Puppet makes it happen.

crankd is a Python script that registers system config events, Cocoa events. They use it every time the network state changes, to see if, for example, you're on the corporate network, a VPN, behind a captive portal, etc. They can also track application launches and quits, logging it for usage which is good for licensing and security.

Users do terrible things. They once had a user describe Puppet as a virus to a team mailing list including a list of ways to disable it and work around it. Occasionally the Google team does terrible things. So they needed Plan B.

Plan B is a management watchdog, and is just a shell script because they've seen cases where the system, Python or Ruby, breaks in interesting ways. They figure if the shell is broken then nothing is working. Plan B runs periodically and reports whether the management agents are running. If not, Plan B tries to download and re-start them.

Google uses three open source tools to manage their software life cycle from beginning to end. Munki is an open source project out of Disney animation. The client end is made to look like an Apple software update, and it manages software update for users. Munki can push Apple software updates or their own packages. They can prepackage things with optional software, and users will get a package that Google has okayed for them to install. The backend of Munki just looks for an HTTP server. They built the backend on Simian and that gives them fine-grained controls on how packages are pushed out. They manage hundreds and hundreds of packages without issues.

For security and monitoring, they built a tool called Cauliflower Vest (an anagram of "filevault escrow"). Cauliflower Vest is open sourced, allows for key escrow for full disk encryption (FileVault 2), and uses App Engine internally to Google. All of the access to the client's key is logged and audited. The user can get the recovery key or incident response, or someone on the MacOps team can get access to the locked hard drive.

They owned up to not having a great story on inventory management and have another open source for it. They get most data from factor (the tool that works with Puppet) and a few other sources. Most of the data they get from factor (related to Puppet) and a few other sources. This data gets reported to App Engine and sent up to other inventory management systems at Google.

To summarize security, encrypt all things, log everything, and log everything centrally. Make sure everything is in source control, make sure anything you do can be done again exactly the same. You don't want to do something 40k times.

Someone pointed out that some of their users have a negative opinion of Puppet managing their Mac for them. Assuming the existence of independent hacker types who want to manage their own Macs, how do they balance that? They try to go with a carrot-stick approach. The access to internal network resources is dependent on a machine certificate. In order to get that certificate, certain internal checks have to pass, demonstrating the machine is up to date. If it isn't, they can revoke the certificate. They also give users flexibility to opt-out, but not a simple way to opt-out. There are always some edge cases, so they try to create as much freedom as they can.

Another person wondered whether Google doesn't have the ability to ask Apple "Can you just make that work?" The team replied that they ask Apple lots of things. Joking aside, Apple takes a lot of their suggestions. Google saw Lion 10.7 was great and asked for key management, and a year later they got some sort of key management. They're easily one of Apple's biggest enterprise customers. But Apple easily sell 3x as many Macs at the Palo Alto Apple store than they sell to Google, so Apple knows which side their bread is buttered on. With the original FileVault 2 in 10.7, you could give the security question and key encryption and send it up to Apple. But there's no way for that person to get it back, and Google needed a way for the enterprise to get the data back.

Mario Obejas (Raytheon) asked about Pymacadmin (https://code.google.com/p/pymacadmin/) and they replied that Pymacadmin is mostly crankd. Mario continued by pointing out that many of us have all kinds of edge cases. Those of us with lots of engineers have the same problem. What does Google do for those repeat offenders where they don't have a business case, but who are repeatedly defeating the controls Google has in place. Does Google have escalations? A divergence report? Contact their manager? The team responded that they try to fix things technically up to a point. If that doesn't work, it's an HR problem...sort of. If there's a reason they're doing this constantly then maybe there's a workaround they can give them or a knob they can tweak. It doesn't happen very often. Hypothetically, they might have a developer who was installing the latest OS before they approved it, but that's also rare. Mario wondered what they did with the guy who actively posted on the mailing list about how to defeat controls, and they answered that they shamed him and his manager. He was good about getting feedback.

Another person asked how they manage user preferences (bookmarks, local prefs) and the team answered that they trust the users to back up. Their users use Chrome which backs up its preferences to the cloud. They encourage people to use iDrive. They discourage users from keeping data locally. They don't do backups at this level.

Someone asked since they let users keep their own backups, how do they make sure they don't just keep an unencrypted backup that may contain something valuable to the corporation. Their answer was that it is hard to keep users from doing bad things if they really want to. They provide guidelines to using Google tools that they trust. They try to keep an eye on tools running on the machines, like Apple's Time Machine which they have not been able to encrypt. They'll contact them and encourage them to use something else. You can't stop users from plugging a USB drive in and copying stuff. Instead, they have guidance that says "don't be an idiot."

Finally, someone asked why they use Mac OS when they have Chrome OS, and the team answered that Chrome OS is young. They have a lot of engineers, some need to run MS Office. They have needs that go beyond what Chrome OS can do right now.

OS X Hardening: Securing a Large Global Mac Fleet

Greg Castle, Security Engineer, Google Inc.

Greg explained that Google hardens Mac OS X using Apple built-ins and add-ons. To start with, full disk (FileVault 2) encryption is critical for an enterprise fleet of laptops. They use Cauliflower Vest for key escrow, and the user gets notified if anyone requests a key to unlock the files on their disk.

The next step is to keep badness off the platform. Apple provides Gatekeeper, a tool that taints downloads via Web browsers. Anything that is not signed is blocked, and you can added your own keys for your enterprise developed apps. If you call malware from cmdline or POSIX APIs or shellcode, then Gatekeeper doesn't protect against those cases.

XProtect (File Quarantine) is similar to Gatekeeper, except that it works more like AV. There are currently only 40 signatures and the update frequency is pretty low—seven updates in six months. Apple has introduced minimum version requirements that allows you to stop people from running older versions of software (e.g., Java). The format of the signatures is so simple, it seems that Google could extend the signature list themselves. Greg mentioned that you must keep the "Report to Apple" checkbox ticked, or there is no logging when XProtect blocks something. Also, it would be useful if Google could have a separate plist (configuration file), or get Apple to include signatures that Google creates.

Application sandboxing is used by Chrome, Adobe Reader, and all app store apps. All major software vendors are using sandboxes. Apple has made this simple for developers with Xcode, and Google has been working to sandbox their own management daemons and tools. The most simple case is just to trace execution and log, then you can run sandbox-simplify to dedup the information and make it more usable.

Sandboxing is not perfect, and Greg mentioned Meder's ruxcon 2013 presentation.

Greg then said that Java is no longer installed by default. Also, XProtect enforces a minimum version of Java. A Java bug was exploited by the Flashback attack in April 2013. The nssecurity plugin wrapper (also works on Linux and other plugins) Java now has includes whitelisting of versions (this old version of

Java with this Web site is okay, but no others), and JRE expiration dates (if you try to call the plugin after the expiration, you can't).

Greg said that the other elephant in the room is Flash. Given that most of Google users' Macs have Chrome installed, and Chrome includes Google's own version of Flash, one that is sandboxed, why not try uninstalling Flash? They started with a small group, moved to a larger group (which he called the canaries), and Google eventually was able to get 94% of Mac users to uninstall Flash. The rest presented use cases for keeping Flash (separate from that in Chrome).

The Google philosophy is no corporate network, and for this to work, they need to be able to apply patches, tweak configurations, and collect logs. The configuration and patching were discussed in the previous talk. Google collects one billion lines of log info per day for its Mac fleet using an internal tool. They also use a tool called GRR (Google Rapid Response, http://code.google.com/p/grr/) so they can handle incidents anywhere.

They also use the auditing that comes with Macs. OpenBSM (auditd) is an implementation of Sun's Basic Security Module written by McAfee Research under contract to Apple. OpenBSB is very different from the auditing in Linux in that it audits classes of events. Greg called the auditing information invaluable for detection and incident response.

David Lynn (Vanderbilt) asked since they are accepting log info from any machine on the Net, how do they keep nation states from blowing out their log server? Greg said they use machine certs and SSL. Someone else asked whether they are trusting Puppet certs to be their end-all of security. Greg replied no and went on to say that exposing your Puppet server is not something he would recommend. They have a shim that sits in front of Puppet that uses a cert they issue themselves, and they proxy through an SSL connection to their Puppet server. Clay and Ed in the previous talk discussed where Google is using distributed Puppet to send the manifests down to the client; in that case they don't need these sorts of proxies.

Someone asked about the OS X firewall. Greg responded that they haven't replaced it, but doesn't believe the security value of firewalls is enormous.

Rik Farrow (USENIX) asked whether they watched for anything in particular, such as when Apple changes the underlying system tools operation (moving events to launchd).

Greg said that they use the OpenBSM framework to monitor that sort of thing. Rik also asked whether they had fixed the problem where Firewire could be used to access memory, even in a screenlocked system, and Greg responded that Apple had fixed that with their 10.7 release (a firmware update).

Invited Talks 1

Summarized by Georgios Larkou (glarkou@cs.ucy.ac.cy)

Cloud/IaaS Platforms: I/O Virtualization and Scheduling

Dave Cohen, Office of the CTO, EMC

Dave started the presentation with some background information regarding the explosion of data growth and how we can use I/O virtualization in order to address it. Dave noted that the transition from static terminals to mobile computing introduced the need of data services and anywhere/anytime connectivity. He discussed the notion of economies-of-scale, an economy where factors cause the average cost of producing something to fail as the volume of its output increases. As a result, datacenter operators have to bring the operational-cost-per-megawatt-of-power down via sharing and automation. Additionally, the explosive growth in the size and diversity of data-under-management has forced datacenter operators to look for ways to decrease the data redundancy rate.

In order to reduce the data redundancy rate below 2x, multi-site erasure coding is being adopted by datacenter operators. The adoption of multi-site erasure coding is predicated on scale and caching. Dave provided an example of distributed storage with dual-site mirroring, which improves reliability and availability, with redundancy roughly equivalent to RAID5, and another one with distributed storage with multi-site erasure codes and improved reliability and availability of dual-site while reducing redundancy below 2x.

During the second section of the talk, Dave presented I/O forwarding, which bridges the gap between compute and storage in horizontally scaled environments and introduces an intermediary tier that provides impedance matching between compute and storage. In addition, Dave presented some existing libraries that the audience can potentially use in order to build a complete I/O forwarding engine. He presented a complete architecture of a I/O forwarding engine and briefly discussed most of the modules.

Dave then discussed IP Internetworking and how network virtualization can be used as the means for supporting I/O forwarding. He provided an example of how a datacenter operator can evolve the datacenter network to support scalability and mentioned that almost every cloud provider he is aware of is currently transitioning to a more scalable datacenter network architecture. Dave commented on the idea behind programmable switches supporting this transition and the explosive (almost infinite) amount of bandwidth needed to support the newly introduced architecture and noted that this bandwidth can be treated as a schedulable resource. In order to justify his suggestions, Dave presented four illustrative deployment scenarios, Google's Data Center to Data Center Connectivity, Microsoft's BGP-Only/SDN-Controlled Data Center Network, Open Network Lab's BGP-Only/SDN-Controlled Internet Exchange Point, and BGP-Only/SDN-Controlled Network Architecture recently posted by Ivan Pepelnjak. Additionally, he provided a survey of

Network Virtualization Controllers and highlighted the Open Network Operating System (ONOS) developed by the Open Network Lab.

Dave ended by presenting Autovias, a network-partitioning scheme for I/O forwarding. Autovias is not a product but it is a technology that allows a user/service manager to schedule network bandwidth for their service. The Autovias concept was conceived in 2012 and derives from Google's B4 Data Center to Data Center solution as presented at the Open Networking Summit 2012. Additionally, Autovias employs a Network Virtualization Controller such as VMware's NSX controller. Autovias was first demonstrated and discussed publicly at EMC World in May 2013. In conclusion, Dave presented a top-down view of a network, that we couldn't have before, which allows the administrator to be able to monitor, model, and modify over time from the controller.

An attendee asked how an administrator can manage the bandwidth between multiple sites/groups and whether it is based on VMs or source-destination IP addresses. Dave responded that it is based on subnets that ideally provide IP failover and use standard BGP techniques for moving things around.

Cluster Management at Google

John Wilkes, Google

John started the presentation with a map of Google's datacenters and commented that the location of each datacenter is as near as possible to their customers. Additionally, he showed one of Google's datacenters in Finland. He also defined the terms cluster, a set of machines connected with a high speed network, and cell, a set of machines managed as one entity and went on to explain why cluster management is important. A cell runs jobs, made up of one to thousands of tasks for internal users, and a manager, one per cell, is responsible for allocating these jobs and assigning them to machines.

John presented the two kinds of jobs Google receives: batch jobs which are things you submit that produce a result and, if they succeed, exit. Batch jobs regularly run and produce a result in a few minutes. Service jobs, the second type of job, start and stay up until there's a software update. Google Docs and Gmail are examples of service jobs. Additionally, John presented job interarrival time, small batch jobs that continuously arrive at the cell and services which remain and consume resources for a larger amount of time. Afterwards, he presented some graphs derived from a cell's logs and illustrated that the jobs that arrived in each of the three cells did not actually consume all the available RAM or CPU cycles. As a result, a significant amount of resources were just idling. He provided a public cell workload trace (search for "john wilkes google cluster data") consisting of a 29-day cell workload trace from May 2011 and asked the audience to download the traces and try to write a better schedule manager in order to achieve better resource utilization. Of course, John mentioned that better resource utilization is not the only goal of a good scheduler since there is a variety of other SLI/SLOs that the scheduler has to satisfy: for example, availability (e.g., max outage duration), performance (e.g., time to schedule and start a job), and evictions (e.g., percentage of jobs with too high an eviction rate).

Google's current cluster management system was built in 2003-4 and it works pretty well. But it is difficult to extend it and add extra features, and it is difficult to add new people because of its complexity. As a result, John presented a brand new system, coined Omega, which is currently being deployed and prototyped. The main goals of the new system are to be easy to use and flexible. Additionally, John presented Omega's architecture and briefly described its components. He also briefly commented on some of the advantages introduced by Omega, such as the distinct parameterized schedulers for batch jobs and service jobs, cell reconfiguration without breaking SLO promises, a calendar that can answer questions about future events (e.g., "Can I execute this batch job on this cell without breaking SLOs?"), and sharing decisions made by a scheduler with other schedulers. Finally, John presented the results of a performance experiment (simulation study) between the old monolithic scheduler, UC Berkley's Mesos scheduler, and Omega. The results showed that Omega performed better than the two other schedulers and, more specifically, provided 3–4x speedup for MapReduce jobs.

In conclusion, John presented some open issues—for example, smart explanations. According to smart explanations the system should provide reasonable information to the user: why the job did not run and some advice on what the user can do in order to fix these problems. John predicted that cluster management configuration may be the next big challenge.

The first questioner noted that most of the existing configuration systems are declarative and asked what is the next direction in configuration management systems and whether we can learn from biological systems. John answered that we need a mix of declarative and imperative, with the introduction of some constraints for safety reasons. He also stated that he believed that machine learning will emerge in the area of building smarter configuration systems.

The second questioner wondered how good people were at giving estimates about what resources they needed. John answered that if people know about every job that requires more resources than what they have requested, they will be really good and they will provide a really good estimate of what they need. Of course, systems tend to be much better at estimating the amount of resources a job might need since they can take into consideration logs from previous executions or pre-calculated parameters, but the current systems might fail to provide a good estimate for newly submitted jobs without supervision.

Someone asked why Google is implementing its own configuration system/scheduler when some open source solutions already exist. John answered that he can hypothesize why...but he won't.

Invited Talks 3

Summarized by Ben Cotton (bcotton@funnelfiasco.com)

Managing Access Using SSH Keys

Tatu Ylönen, SSH Communications Security

Tatu Ylönen, the inventor of the widely used SSH protocol, began with a brief introduction to SSH before launching into the thesis of his talk: unmanaged SSH keys are a security problem. Most organizations don't manage or remove old SSH keys. Since servers trust clients, compromising a client can allow access to more important servers.

As an example, a global top-10 bank that Ylönen worked with had over 1.5 million authorized SSH keys. Over 2 million daily key-based logins occurred to the 2000 core production servers. In many organizations, over 90% of the possible access credentials to production servers are SSH keys.

Unmanaged SSH keys are an excellent attack vector. Organizations don't know who can access what and who is accessing what systems or information. SSH keys can bypass other access management mechanisms. They allow unattended file transfer and escalation from development to production environments, non-PCI to PCI environments, etc. Most critically, SSH keys lack effective termination of access.

Current and pending regulations address SSH key management, often indirectly. Proactive management is important to prevent regulatory issues before they happen. The first step is to establish a controlled provisioning process that documents the purpose and owner of each authorized key. Once provisioned, SSH credentials should be continuously monitored for automated and interactive access. Since some legacy systems will break if the keys are changed abruptly, careful remediation of those systems is necessary.

In Ylönen's experience, the process of remediating one million keys can be a multi-year project. Some tools are available, but they don't necessarily cover the full process; many only do discovery. SSH Communications Security offers two tools and consulting services.

Ylönen concluded by saying that most organizations with large UNIX or Linux environments have a serious compliance and security problem, the scope of which is not widely understood. He finished with time for a single question. An attendee asked whether it was possible to automatically expire SSH keys. The answer was no.

Secure Linux Containers

Dan Walsh, Red Hat

Containers are a hot topic in Linux and security, but people don't always agree on what a container is. Walsh based his presentation on his definition of containers as a user space, not a kernel-space construct.

A container has four key elements: (1) process isolation, giving processes their own private corner of the world; (2) resource management, controlling how much of a given resource a process gets; (3) security, where tools like SELinux enforce security policies to keep the containers from overstepping their authorization; and (4) a management component that allows administrators to control the container. In the Red Hat Enterprise Linux container architecture, namespaces provide the process isolation, cgroups provide resource management, SELinux provides security, and libvirt is used as the management tool.

Because containers have some similarities to virtual machines, including being managed by libvirt, Walsh examined the two concepts side-by-side. Containers provide benefits like improved startup and shutdown speed, ease of maintenance and creation, and scalability. Virtual machines, specifically KVM, offer the ability to boot different operating systems, thus providing full separation from other clients, and supports features like live migration.

Walsh also presented a container product called "Docker." Docker bundles a micro-operating system with an application. This allows developers to bundle their application without regard for what host the application will run on. Red Hat has recently announced an effort to contribute to Docker development to allow containers to work on either Docker or Red Hat's OpenShift PaaS offering.

Walsh closed with a live demonstration of building, using, and tearing down containers. The audience was briefly surprised to see he wasn't running SELinux inside the container, but that turned out to be a feature of the container. The talk ran past time, but the audience was content to watch the demonstration until it ended.

Closing Plenary

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

Post Ops: A Non-Surgical Personal Tale of Software, Fragility, and Reliability

Todd Underwood, Google

Todd started off by saying that system administration as we currently know it is over, and that the time has come to quit feeding human blood to machines. He noted that machines currently eat system administrators' times, passion, and imaginations. Todd noted that he shares the same history as many system administrators who started off at ISPs, but that it might be time to revisit common views of system administrators and their nostalgia for the past. Todd noted that common objections he has heard are that Google's solutions to problems may not apply at smaller scales, that Google has massive amounts of software engineering available to it, and that Google has already solved all the problems you are going to encounter. Todd rebutted these arguments by noting that they may solve common problems for everyone and that open source solutions will allow system administrators to focus on tasks which are more interesting to them. Todd then described a number of "war stories at scale" to

show that Google has its own share of problems which it has had to address at all.

Todd then began describing Google's approach to Site Reliability Engineering by providing a "mythical" history in which he described Google as frugal and argued in favor of costs which don't scale linearly. He said that one example of this type of solution was developers deploying their own code in production through automated processes so that they would not be responsible for repetitive and tedious tasks which machines can perform better. He also talked about the importance of being able to balance competing demands and improve availability and efficiency while helping improve the deployment times of new services. And he described systems developing emergent behaviors during their post-deployment evolution.

Todd then described DevOps at Google, which he sees as a cultural and professional movement. In a traditional infrastructure, development and operations are walled off from each other, he noted, whereas DevOps is the blending of development and operations together in to a single way of being and doing things. He said that DevOps strives to improve operations as a discipline and integrate operational concerns into business practices. Todd then quoted Adrian Cockcroft's definition of "NoOps," where software developers work directly with production and automation removes operational tasks entirely.

Todd went on to describe operations management as applied to software-based production while noting that operations research has a set of constraints while software production has its own set of constraints. Todd said that operations research may be relevant for the physical infrastructure stack (power, fiber, etc.) and for maximizing SLAs in the short run, but that it may not be as applicable towards software organizations. Todd said he would still like to maintain the ethic of caring about production and the ethic of privacy and security.

Todd closed by noting that he is advocating for a transformation from Ops, to DevOps, to NoOps/PostOps. Operations is still needed to identify new problems and prioritize development, but every other aspect of operations should be deleted. Todd said it is important for technical talent to demand that their management knows what is going on technically, because this breeds a better environment for everyone. He wants to work in an area which is constantly looking to improve, and that even though system administration is eventually going to go away, the world is moving towards distributed application development. The bathwater that will get thrown out is the repeatable work and the configuration management, while the baby that will stay is the production ethic, monitoring of systems, release engineering, and constructive cynicism.