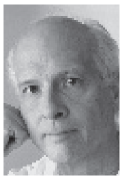
EDITORIAL

Musings

RIK FARROW



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

I have a new hat, one with a fancy feather in it, perhaps to help it feel lighter. I've become the tutorial manager, taking over for Dan Klein after he's done the job single-handedly for more than 20 years. I appreciate what Dan has done and will have to work hard, working with a team, to do as well and, hopefully, better. This change has led me on a quest for information about the future of system administration.

What I learned is nothing new, in a way: system administration has always been changing. But, as always, there will be the risk takers, the early adopters, and the adventurers—the ones who strike out in new directions expecting that others will surely follow them.

There are also those who, having been through several schools of hard knocks, would like to continue on the path that they have become adept at navigating. Those hard knocks were painful and time-consuming, and learning new ways of doing things means going through yet another learning process.

Being an older kind of guy, I too can resist change. Then again, I know it's good for the ol' noggin to stretch its limits by learning new things. I've watched what being really stuck does to older folks than I, and I surely don't want to go there.

System Administration, Then and Now

Like Elizabeth Zwicky writes in this issue, I am not a system administrator. Oh, I can still fake it, by managing my own systems (DNS, SMTP, DHCP, and HTTP) and occasionally even consulting locally. But I gave up on being a sysadmin when I found out that I wasn't very interested in a very important aspect of it: building scripts and systems to automate the work that needed to be done routinely.

What I liked most about being a sysadmin was solving problems. Although my skills are dwarfed by Brendan Gregg (also in this issue), I followed one of the techniques in his book [1], the scientific method. I didn't learn it from his book, but from Arthur Conan Doyle's fictional detective, Sherlock Holmes. I read all the stories about Sherlock, and what stuck with me was a methodical approach to problem solving: gathering evidence, creating a hypothesis, testing it, then either fixing the problem or iterating, depending on the result.

I think I also liked the effect I could have on people when I produced wizard-like successes. Often, simply by creating a clear problem statement, the answer to the problem would just pop out. I liked feeling and appearing smart.

What I didn't like was the boredom of repetition: adding a user to a handful of workstations or performing updgrades on that same set of systems. Like Elizabeth Zwicky, I am not a programmer who builds systems from scratch. So I contented myself with fixing systems, sometimes by building scripts for specific purposes. Managing a group of systems through a unifying set of programs, however, was beyond my thinking. I gradually drifted away from system administration.

For most of the ’90s, system administration remained very similar to what I had experienced. People managed each system separately, perhaps leveraging some network software (like NIS or LDAP) to solve part of the management issues. Someone who managed a lot of systems, say one or two hundred, would do so by making them all look exactly alike, whether they were part of a cluster or developer desktops.

Configuration management brought about some real changes, allowing administration of more systems without the need to make them all the same: now we could have groups of systems, and systems with sets of software on them. When I found myself installing 30 laptops for classes I was teaching, I couldn’t have done it without CFEngine. And I used CFEngine as an example of how to properly manage networks of systems in those very classes.

Scale

I’ve always been fascinated by big machines, whether they were harvesters, bulldozers, or computers that filled a large room. I got to visit MCI’s NOC, via connections to UUNet, and could marvel at the volume of data moving through a subterranean room in Northern Virginia.

But there was something else beginning to happen in the early noughts, the birth of services run on immense farms of distributed systems. Amazon and Google were experiencing tremendous growth, and dealing with this by building networks of computers on a scale never before seen or heard of. These networks might have been managed like clusters and supercomputers had been. But they weren’t. These systems weren’t there for running batch jobs, but for providing services to users who expected response times in terms of seconds, or less. The ’90s clusters weren’t designed for this, and although the tiered systems—load balancers in front of Web servers in front of a database—provided a rough model, they would freeze under the loads that these new companies were experiencing.

Todd Underwood, in his closing plenary at LISA ’13 [2] and in an article in this issue, addresses the human side of scale. Quite simply, you can’t, and don’t want to, scale human sysadmins at the same rate you scale servers. Doug Hughes explains (in this issue) how D. E. Shaw Research buys racks of configured servers at a time, and that they figured out how they could autoconfigure both the racktop switch and the servers as soon as they were plugged in. In my interview with John Looney, you get an even better notion of how scale affects how work gets done. Instead of spending months setting up a new datacenter, a Google team spent months building automation. That automation does in just three hours what took teams of people months to complete.

I don’t believe that system administration, as it has been practiced, is going to disappear. But I certainly know that system administrators are going to need new skills so they can work with online services. Debugging a distributed system where you have access to both the networks and the servers is very different from troubleshooting a distributed application running in the cloud where your access is limited. And working at scale, even modest scales, means the end of manually searching logs and monitoring: you need automation to help not just pick up problems but respond to them appropriately, and in a timely manner.

Will every service move to a cloud? I don’t think the NSA is going to, and neither will privacy-sensitive or regulatory-bound businesses. People will still be running company networks and servers, and in some cases, like D. E. Shaw Research, their own supercomputers. Cloud computing works very well for many things, especially Internet services. But office, factory, and many other workers will continue to rely on internally provided services—ones that will keep working even when their connection to the Internet has failed or a cloud service becomes temporarily unavailable.

And we will not all be working at Google or some place like it: not hardly, as working at these scales does take some skills that not everybody has or wants to spend time doing. But I know one thing for certain: system administration will continue to evolve.

The Lineup

We start off this issue with a deliberately provocative piece by Todd Underwood. Todd emphatically states that the practice of system administration needs to change, which, whether you agree with Todd or not, is definitely happening. What isn’t as clear is just how system administration will evolve, or how much it will follow Google’s lead.

Next up is an interview with John Looney. John is an SRE for Google and taught the SRE Classroom class at LISA ’13. I missed visiting this class, because I was teaching the same day, and I learned about it from hearing people talk about it over the next couple of days.

John provides the clearest explanation, through examples, of how being an SRE is different from what sysadmins, even those administering to large clusters, do. His description of how different things are within Google is crystal clear for me, and I think it will be useful to you as well.

Elizabeth Zwicky volunteers her own view: although Elizabeth started out as a system administrator, that was a doorway for her into many different jobs over the years, some related to system administration, and some barely at all.

Dinah McNutt, past LISA chair, writes about release engineering, a field related to system administration. According to Dinah, release engineering is critical for any company providing a service in a competitive market, and there are many subfields related to release engineering as well.

I interviewed Tom Hatch, creator and principal of SaltStack. SaltStack can be thought of as a grid execution engine, but it can do much more. I wanted to ask Tom about his use of ZeroMQ for message queueing and learn more about where he plans to take SaltStack.

Brendan Gregg provides this issue's troubleshooting feature. Brendan takes us through debugging a performance issue, explaining along the way how he performs troubleshooting. You can consider this a taste of what you'll find in his book [1].

David Lang continues his series on logging, focusing this time on Splunk. Splunk provides some wonderful features, but it does need to be set up and tuned if it is to provide the best performance. And, setting your Splunk servers up incorrectly can fool you into thinking you need more servers, when you simply need to do things differently.

Andy Seely begins a series of articles about managing system administrators and other IT staff. Andy explains how he handled a situation where his management needed someone to blame for a failure that was really outside the scope of what his own organization had control over.

Changing focus, Rob Sherwood updates us on Software Defined Networks (SDNs). Rob wrote for *;login:* in February 2011 [3] about safely using SDN in a production network. For this issue, I asked Rob to provide us with an update on where SDN is today, and why it is important to know about, even for those with moderately sized networks.

Doug Hughes and his co-workers have taken a careful look at the most commonly used DHCP daemon and found it lacking. They needed the DHCP protocol to do something they thought it could easily do, and they found that the easiest way forward was to do it themselves. Along the way, Doug explains DHCP, what it can do, and what they needed it to do to support their organization.

David Blank-Edelman stays on the theme of Perl and NoSQL databases. In this column, he explains MongoDB and explores its Perl interface. MongoDB is different in many ways from Redis, which he explored in the previous issue.

David Beazley discusses Python 3 from the perspective of whether you should now be using it or porting existing Python 2 code to Python 3. David provides a very balanced answer, as well as reminders for tools and techniques for moving to Python 3.

Dave Josephsen takes us on a graphic journal through a tool that helps in collaborative troubleshooting. Dave shows examples of ChatOps, a chat tool that includes the ability to include graphs and links, useful in operations.

Dan Geer and Richard Bejtlich explore a method for understanding the risk of data theft: counting and classifying digital security incidents, and measuring the time elapsed from the moment of detection to the moment of risk reduction. As usual, Dan and his co-author take a hard-line approach that is also much more realistic than common practice.

Robert Ferrell also writes about being a system administrator this time. He muses about his past as a sysadmin and pines for the power he has given up.

My usual book reviewers surprised me this month with just three book reviews. I've added one of my own, on a book that I really liked.

System administration isn't dead or dying. It's changing. Just like it always has, as technology has advanced. To be honest, you really do not want to do what I did in the mid-'80s: wire up a hub-and-spoke arrangement of serial connections to support UUCP. UUCP did relay email and support file transfers, but the truth was that it was all we had. Within a handful of years, TCP/IP networks became the new norm, and within 15 years, people would be discussing sharing DRAM over the network instead of using local disks.

In 1988, AT&T was releasing System V Release 4 (SVR4), with a focus on GUI interfaces for system administration. Under the hood, some of the commands for managing configuration had gotten extremely ugly (multiple lines with many strange options) to do what had previously been simple. The designers' argument was that system administrators shouldn't be hand editing these files, as it wasn't safe. I replied that hand editing files may not be safe, but GUIs and impenetrable syntax both restrict flexibility and cannot work with scripting.

Bay Networks was a competitor to Cisco in the early '90s, and they too had a very nice GUI for managing their routers. But their interface was also restrictive. Cisco had (and still has) their command-line management for network gear, which was tremendously more flexible than Bay Networks. Today, Bay Networks is just a memory.

SDN promises a much more flexible approach to network management, and I expect it will have a similar effect on old school network equipment companies.

In the world of sysadmin, we now have tools like Augeas [4] that turn the myriad formats of configuration files into tree-like structures, helping to hide their eccentricities.

I had considered updating the story of Chuck, my system administrator of the near future [5]. But I kept seeing Chuck getting bored and fat (from all the bon-bons he was eating) because his job had been taken over by automation.

I don't believe things are that bad. We still have time to shape our future, which we can do through the tools we build, and the philosophies behind those tools. Google employees have represented one view of the future, where automation is emperor, and I think that their view is not just relevant, but a likely future.

And, like Todd Underwood, I don't think humans will be forced out of the picture, at least, not for a long time. Not only are humans writing the automation tools, they are also troubleshooting what goes wrong with much more flexibility than exists in any automation tool.

References
[1] Brendan Gregg, *Systems Performance for Enterprise and Cloud* (Prentice Hall, 2013).

[2] Todd Underwood, "PostOps: A Non-Surgical Tale of Software, Fragility, and Reliability," 27th Large Installation System Administration Conference (LISA '13): https://www.usenix.org/conference/lisa13/technical-sessions/plenary/underwood.

[3] Rob Sherwood, "Safely Using Your Production Network as a Testbed," *;login:*, February 2011, vol. 36, no. 1: https://www.usenix.org/publications/ login/february-2011-volume-36-number-1/safely-using-your-production-network-testbed.

[4] Augeas, a configuration API: http://augeas.net/.

[5] Musings about the future of sysadmin, featuring Chuck, *;login:*, February 2010, vol. 35, no. 1: https://www.usenix.org/publications/login/february-2010-volume-35-number-1/musings.

Letter to the Editor

Dear Editor,

I want to congratulate you for the very nice and timely focus on file systems (and object storage in particular) in the *;login:* February 2014 issue.

I appreciate the first article's ("A Saga of Smart Storage Devices") overall analysis of the landscape of Object Storage but as someone who was closely involved in the design of Ceph I feel the need to express my confusion by how Ceph was portrayed.

The article describes Ceph as "a cluster of OSDs cooperating to perform automatic replication, recovery, and snapshots" that "builds on previous work," presumably NASD, "decentralized placement of objects on devices" and "the RUSH family of algorithms."

I'd like to make two clarifications:

1. Ceph is a parallel file system and, as such, much more than a cluster of cooperating OSDs. The following papers give a more accurate account (none of them were cited in the article): [1-4]. The authors appear to conflate "Ceph" with the Ceph's object storage subsystem which is called RADOS (Reliable Autonomic Distributed Object Store, described in detail in [5], also not cited) and mention the term RADOS only in the context of a protocol.
2. Similarly, while Ceph's data placement (aka CRUSH) is in part related to RUSH, RUSH turned out to be an insufficient solution in practice. CRUSH fully generalizes the useful elements of RUSH while resolving previously unaddressed reliability and replication issues, and offering improved performance and flexibility. The authors do write about the various features of CRUSH but without mentioning it or citing the paper describing it [6].

Thanks,
Carlos Maltzahn
carlosm@soe.ucsc.edu

References
[1] S. A. Weil, K. T. Pollack, S. A. Brandt, and E. L. Miller, "Dynamic Metadata Management for Petabyte-scale File Systems," in SC '04 (Pittsburgh, PA), ACM, Nov. 2004.

[2] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A Scalable, High-performance Distributed File System," in OSDI '06 (Seattle, WA), Nov. 2006.

[3] K. V. Shvachko, "HDFS Scalability: The Limits to Growth," *;login:*, vol. 35, no. 2, 2010.

[4] C. Maltzahn, E. Molina-Estolano, A. Khurana, A. J. Nelson, S. A. Brandt, and S. A. Weil, "Ceph as a Scalable Alternative to the Hadoop Distributed File System," *;login:*, vol. 35, no. 4, 2010.

[5] S. A. Weil, A. Leung, S. A. Brandt, and C. Maltzahn, "Rados: A Fast, Scalable, and Reliable Storage Service for Petabyte-scale Storage Clusters," in Proceedings of the 2007 ACM Petascale Data Storage Workshop (PDSW '07), (Reno, NV), November 2007.

[6] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, "CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data," in SC '06 (Tampa, FL), ACM, Nov. 2006.