EDITORIAL

Musings

RIK FARROW

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

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

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

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

File Systems

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

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

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

Object Stores

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

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

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

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

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

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

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

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

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

The Lineup

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

References

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

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

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

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

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

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

Publish and Present Your Work at USENIX Conferences

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

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

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

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

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

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

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

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

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