Musings

RIK FARROW

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

Back in the late nineties, I found myself sharing a Silicon Valley hotel Jacuzzi with a sales rep from a big hard-disk company. The sales rep was telling me that they soon expected to be selling 10-gigabyte hard drives to consumers, and I was astonished. Why in the world would people need such large drives in their desktop systems?

Now we can buy six-terabyte consumer drives—600 times as much capacity as I found unfathomable around 1999—for under $300. And similar technology, in the smaller 2.5-inch form factor, fills server racks in many datacenters. Back when I thought that 10 gigabytes was a ridiculous amount, I wondered how mere mortals would manage all that storage. Well, turns out that I needn't have worried. Not only do most home users (and many businesses) not manage their storage, they can now expand out into apparently unlimited clouds of storage as well.

Life in the Clouds

To be honest, the problems with the storage surfeit is not just a cloud problem. During the presentation of one of my favorite storage papers [1], Dutch Meyer pointed out that many files found on Microsoft employees' desktops were "Write once, read never." Of the files that were modified, most were changed within one month of creation, then left alone. Forever. For storage vendors, this is a wonderful situation, as people will use ever larger amounts of storage since the supply appears endless. And now we can just store data "in the cloud," and if we forget about it, it is someone else's problem.

That's pretty nice compared to past methods for managing files that you or your organization had stored in case you might need the data again, some day. You've contracted with companies who keep redundant copies of your data, along with vague guarantees about how safe that is, but that's still better than the old days.

In the "old days," we managed our data by accident. Here's how that worked. We would store our data, carefully backing up what we considered important, until the hard drive (or RAID array or storage server) catastrophically failed. Then we would get out our backups, and see how well they worked when we tried to recover from the catastrophic failure.

Inevitably, much would be lost. But hey, that was storage management—the important data was either restored from backups, recreated from scratch, or the business or research project just failed. For home users, they'd just start over again with a brand new, mostly empty, and larger hard drive. See, storage management by accident.

We do have very serious uses for data, and I am purposely exaggerating. But there is more than a grain of truth in the problem of storage management, one that I believe still exists today.

The Lineup

We start out on the theme of storage with an article about measuring the size of the working set. The working set represents your hot data, the data you want to have ready for processing, within a relatively short interval. For programs working with big data, calculating the working set has been a real problem, as just collecting the block access data generates both a lot of data while adding a huge workload to the system under measurement.

Wires et al. describe a method for sampling block accesses and collecting enough information about those accesses to accurately measure the working set. I was impressed by this work and felt it was worth sharing with a larger audience than just those who read OSDI papers. I also have a hunch that their techniques will become ever more important as our storage requirements continue to grow.

Mark Lamourine offered to explain just how containers will complicate storage. When I read James Bottomley's "Containers" [2] article, I marveled at how we could now share resources safely without resorting to heavyweight VMs. I didn't understand the issues with long-term storage when containers are spawned in a farm of managed hosts, much like we fire up VMs in the cloud today.

I interviewed Steve Swanson about the past and future of nonvolatile memory. NVM has gone from being almost unused to commonplace, mostly because of the work that has been done with flash. Steve explains how he became involved with flash, the problems vendors have needed to solve to make flash reliable, as well as directions for future research.

Heading off in a different direction, I was intrigued by Ding Yuan et al.'s work analyzing catastrophic failures of cluster software systems, like HBase and HDFS. Honestly, it was what they found that was amazing: that empty, over catching, or nonexistent error handlers lead to most of the crashes in popular cluster software. Ding and his co-authors also produced (and shared) a tool, Aspirator, for finding bugs in error-handling.

Julian Bangert and Nicolai Zeldovich write about their tool, Nail, designed to build secure parsers. Many exploitable vulnerabilities, such as Heartbleed and bugs in the signature checking software in Android and iOS, involve failures in parsing. Nail solves these issues through being a tool for generating parsers, creating the data structures used while working with parsed data, as well as providing a method for correctly regenerating the processed data. Where almost all programs parse data, only Nail uses the same configuration for parsing, storing, and generating stored data.

Dave Hixson and Kavita Guliani continue the series of articles written about the practice of Google Site Reliability Engineers (SREs). Dave and Kavita have written about capacity planning, providing a very thorough approach that obviously comes from Hixson's hard-earned experience.

Andy Seely contrasts the worlds of the sysadmin and the technical manager. As he's worked in both positions, Andy does a great job of comparing the two viewpoints through illustrative stories.

David Blank-Edelman takes another look at REST, defining it and demonstrating several ways of making RESTful requests, including parsing the results, all with his trademark humorous style.

Dave Beazley takes a look at dynamic type handling. Python's flexibility can work against you because Python's ability to automatically convert types can cause unexpected behavior when calling functions. Dave has examples, including several ways to test that the expected types have been passed to functions.

Dave Josephsen continues on his theme of fat data. Dave doesn't want you to lose the richness of the data you are collecting through inappropriate ways of storing that data. In this column, Dave shows how to use a combination of Nagios, Graphite, StatsD, and Graphios to collect and save fat data.

Dan Geer takes on automation. While the future of computing (and manufacturing, farming, stock trading, and everything except service) appears to be automation, Dan questions the appropriateness of automation for dealing with crafty, and often state-sponsored, adversaries.

Robert Ferrell worked with the theme of storage and tells us what he considers will be the future of smart storage.

We have several book reviews by Mark Lamourine and myself. We also have summaries of OSDI '14 and two of the associated workshops.

Getting back to the topic of storage management, I believe that perhaps I've uncovered a region of computer science that is worthy of some serious research. Back when disk drive capacities were minuscule, another technique for storage management consisted of using a **find** command to list all files not accessed within the last chunk of time, say, six months. A bit of shell scripting later, users would receive a list of these "aged" files by email and need to request that they not be deleted. After these email warnings were ignored, the system administrator could then delete the files, then wait for the few frenzied requests for restoration of the missing files. Ah, those were the days!

I like to imagine that cloud providers actually do storage management at scale. I know that Facebook does, because they had a paper [3] about how they handle warm BLOBs (Binary Large OBjects) by reducing the effective duplication factor. Note that Facebook is not throwing away rarely watched cat videos, just saving fewer copies of them. But how do you think Amazon, Google, and Rackspace handle their customers' warm, or cold, storage? They bill their customers for them.

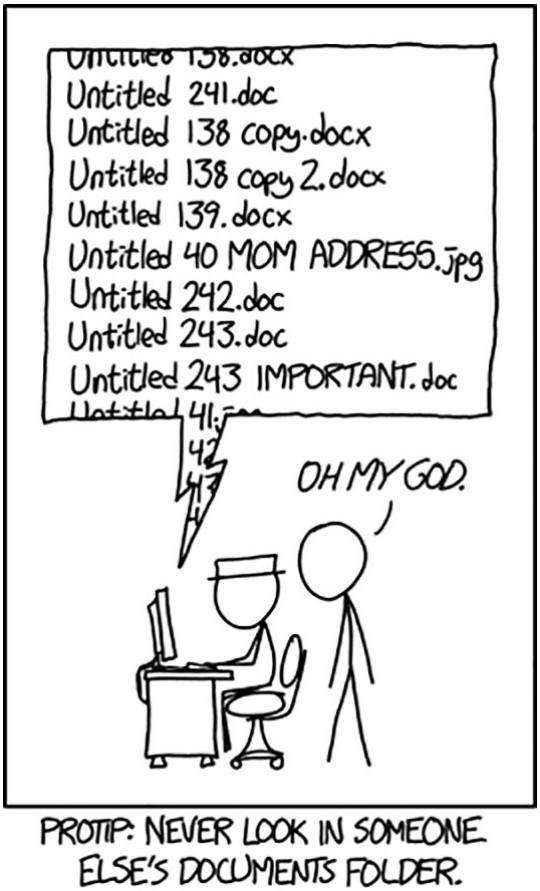
References

[1] D. Meyer, and W. Bolosky, "A Study of Practical Deduplication," FAST '11: https://www.usenix.org/legacy/event/fast11/tech/full_papers/Meyer.pdf.

[2] J. Bottomley and P. Emelyanov, "Containers," *;login:*, vol. 39, no. 5 (October 2014): https://www.usenix.org/publications/login/oct14/bottomley.

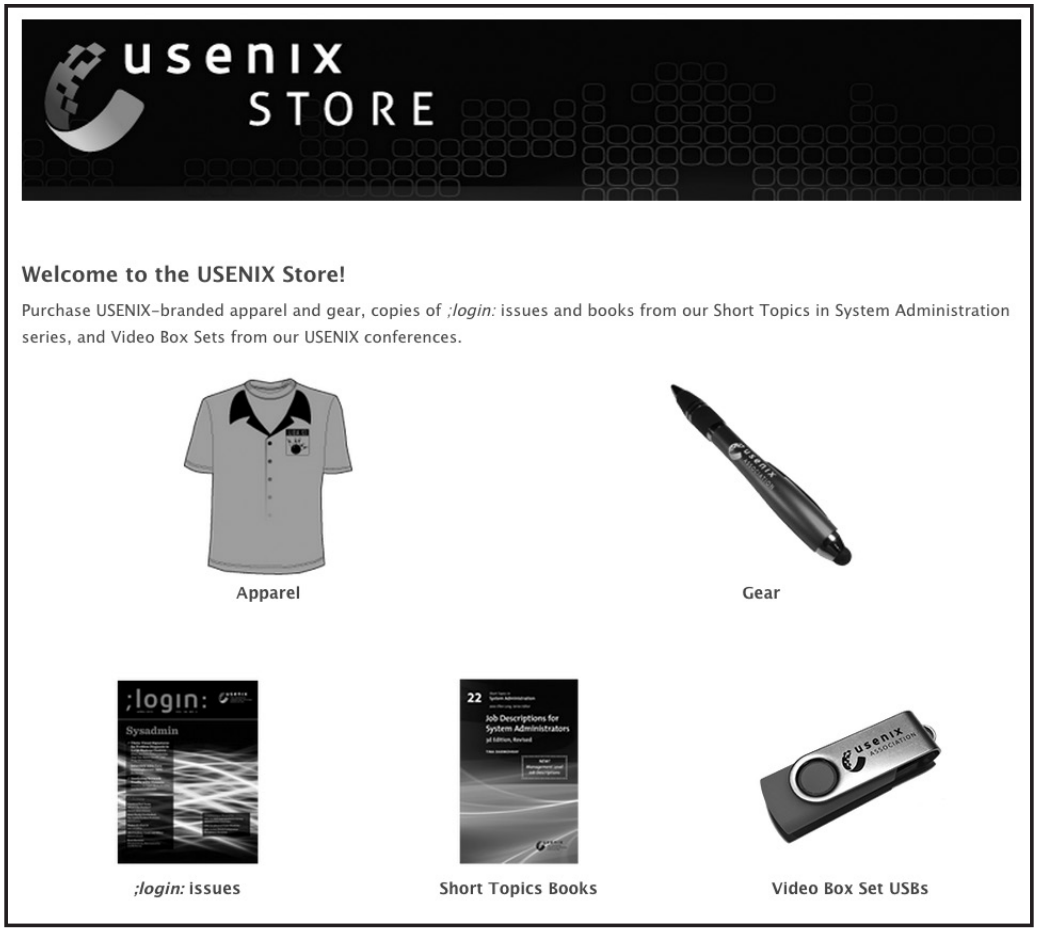
[3] Subramanian Muralidhar, Wyatt Lloyd, Sabyasachi Roy, Cory Hill, Ernest Lin, Weiwen Liu, Satadru Pan, Shiva Shankar, Viswanath Sivakumar, Linpeng Tang, Sanjeev Kumar, "Facebook's Warm BLOB Storage System": OSDI '14: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/muralidhar.

XKCD



usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

Announcing the USENIX Store!



Want to buy a subscription to *;login:,* the latest short topics book, a USENIX or conference shirt, or the box set from last year's workshop? Now you can, via the brand new USENIX Store!

Head over to www.usenix.org/store and check out the collection of t-shirts, video box sets, *;login:* magazines, short topics books, and other USENIX and LISA gear. USENIX and LISA SIG members save, so make sure your membership is up to date.

www.usenix.org/store