EDITORIAL

Musings

RIK FARROW

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

I've long been fascinated by hardware. That fascination was re-awakened by Tom Van Vleck's letter to the editor in this issue. Tom is a Multician (multicians.org), and he wrote to us with comments about my interview with Peter G. Neumann in the Winter 2017 issue of *;login:*.

When I was fortunate enough to assemble my first (nearly UNIX) system [1], it included a 34 MB Seagate drive with the ST-506 interface. The disk controller was not part of the hard drive, as it is today. Instead, the disk controller sent commands—such as seek-inward, switch to head two, read sector headers until sector 10 is reached, then write to sector 10, over a 34 pin control cable—and read or wrote data over a separate 20 pin cable. Device driver writers had to consider issues like how fast to seek, how many blocks to skip between reading or writing to allow the CPU to finish with the previous operation, and handling the bad block map. The latter was truly a PITA, as it appeared as a printed label on the hard drive case and had to been entered, as block numbers, when formatting the drive. Even worse, the controller actually was responsible for sending or receiving analog signals for writing or reading, and that meant that you could only read a hard drive with the controller that had originally done the writing.

By the time ATA [2] became popular, hard drives included their own controllers, and instead of two cables, we only needed a single 40 pin cable that could be used to attach two drives. Each drive had a jumper to determine whether it was a master or not (device 0 or 1), and getting this wrong meant your system wouldn't boot. But having the controller built into the drive was a huge leap forward, as you could now move drives between host adapters. The host adapter was just a 16-bit ISA bus relay for commands and data between the bus and the drive's onboard drive controller. As the ATA standards evolved, the drive controllers became more sophisticated, able to understand SCSI commands.

The Small Computer System Interface [3] (SCSI, pronounced "scuzzy") required an even smarter drive controller. Up to seven devices, plus a host adapter, could be connected to a SCSI cable, and each drive had to be capable of bus arbitration. The SCSI standard also allowed the drives to queue up multiple commands.

SATA, which means serial ATA, uses a four pin cable, with commands and data being sent serially rather than in parallel. Just as PC busses have moved from the parallel ISA bus to the PCI busses that support many simultaneous serial channels, SATA achieves higher data rates by moving away from parallel busses.

And somewhere along the way, disk vendors quietly changed how sectors were accessed. Except with really old interfaces, like the ST-506, disks presented an array of blocks. The operating system was responsible for writing blocks to the most appropriate free block, and the OS did its best to write sectors that would be read in sequence together later, or at least be located in nearby tracks, for better performance.

Since around 1999, hard disks accept Logical Block Addresses (LBAs) instead of block numbers. The hard disk then maps the LBA to a physical block address, a bit like flash translation layers (FTL) work. This change had two effects: the disk controller needed to become smarter, and the operating system no longer had control over disk layout. File systems like

ext4 and BSD's fast file system (FFS) create cylinder groups, and associate directories and free block lists within these cylinder groups, to speed up both reading and writing. But the disk controller is unaware of these distinctions and just stores blocks using its own algorithms. These algorithms can even move blocks later, for example, if a group of blocks is often read in sequence.

I certainly thought that the operating system should have control over block placement, but the disk vendors saw things differently. They wanted to create the cheapest, most efficient drives in order to remain competitive, and for them, that meant taking control away from the operating system developers.

Disk vendors continue to leverage the CPUs and memory on disk drive controllers, and that's led to some interesting developments.

The Lineup

Timothy Feldman wrote a *;login:* article [4] in 2013 about Shingled Magnetic Recording (SMR), a new technique for increasing drive capacity. But SMR introduces its own set of issues, including highly variable write latencies. In this article, Tim explains a huge change to SMR drives: hybrid drives that include both conventional and SMR partitions. A new API means that the operating system can control how much of a drive appears as a conventional drive and how much as SMR, and it even allows changing the proportion of these two formats on the fly, with the disk controller moving blocks between the two regions as required. Tim explains the plans for the new APIs, changes that will be implemented in the drive's on-disk controller, but also need to be included in device drivers.

Carl Waldspurger et al. in their FAST '17 paper explain how to use very small samples of cache misses to determine how best to configure full-size caches. Cache miss rates are crucial when determining the optimal cache size. The authors create hashes of block numbers and select cache misses to record by using a portion of the hash space. Their creative use of hashes for getting a random collection of samples caught my attention.

I searched through the CCS [5] program for papers that match my criteria for articles that will have broad appeal and, out of a huge selection of security research, found two.

Luca Allodi narrates how he infiltrated a Russian exploit market. But his real point is how examining what is bought and sold tells us about which exploits are likely to be used in untargeted attacks. I liked Luca's exploit, managing to gain access to the market, and also how he explains what the going prices for exploits can reveal about which exploits are likely to be widely used.

Frank Li and Vern Paxson describe how they determined that it often takes a very long time for open source software to be patched. It's likely that commercial software is similar, but with open source, they could trace the time a patch appeared in the code, the time it was announced or distributed, and compare that with the time the vulnerability first appeared in the Common Vulnerability Exposure (CVE) ratings. Much of their work involved crafting the means of trawling online Git repositories as well as info about vulnerabilities, a task that would have been much more difficult without techniques for winnowing the data.

AbdelRahman Abdou and Paul C. van Oorschot volunteered to share their work on secure geolocation. While geolocation is commonly used, often for secure applications, geolocation is just as commonly spoofed. Abdou and van Oorschot lay out their proposal along with examples of how well it worked using sensors in PlanetLab.

Diptanu Choudhury offered an article about using eBPF. The extended Berkeley Packet Filter has been around for a few years and provides a secure method for injecting code within a live kernel. Choudhury's particular example involves the Express Data Path (XDP), which can be used for moving network methods, like a firewall or packet forwarder, into code that can access a network device's ring buffer, avoiding slow memory copies. Diptanu explains enough about eBPF to be helpful to anyone interested in beginning to use eBPF, as its programs can use triggers throughout the Linux and BSD kernels.

Tapasweni Pathak discusses her research into flaws in Linux 3.x kernels. Extending prior work, and using some of the same tools for searching through source code, Pathak explains her process and shows us, via graphs, just how well the kernel source is doing when it comes to bugs that can cause crashes or be exploited. For the most part, things have gotten better.

I interviewed Laura Nolan, a Google SRE and a past co-chair of SREcon Europe. Laura had helped me find authors for SRE articles, and I hoped to learn more about what it's like to be an SRE. Laura was definitely forthcoming, as well as providing a humorous example, before she went to Google, of what it's like to be a woman in this field. Laura also answers questions about why Google chose to use Paxos.

Bob Solomon interviewed David Rowe, the developer of the Open Source Line Echo Canceller (Oslec). Bob asks David about the difficulties involved in building something as difficult as an echo canceller and the tricks David used for testing and debugging, while allowing him to tell us a bit about what it's like to be a successful open source developer.

Chris McEniry takes us on a journey through using gRPC and protocol buffers in Golang. Borrowing the certificate generation portion of his Winter 2017 column, Mac explains how to use protobuf, a non-language-specific library, with Golang, as well as how to use gRPC, Golang's version of Remote Procedure Calls. A lot of work for one column.

EDITORIAL

Musings

David Blank-Edelman wants to prove to you that Perl is alive and well. He takes us to a site that keeps track of the hottest, or currently most interesting, Perl modules. He also demonstrates a few of these modules.

Dave Josephsen returns from KubeCon and CloudNativeCon fired up about the Open Tracing API. Dave explains just why tracing requests traveling between microservices is a crucial part of monitoring these systems, tells us how Open Tracing works, and suggests several frameworks for getting started.

Dan Geer and Dan Conway examine the security risks involved with crypto-currencies. At the time their column was written, Bitcoin had exceeded $18,000/BTC, 100,000 times its value just five years ago. Geer and Conway discuss the failings not just of Bitcoin but of other crypto-currencies. The amount of "value" that's already been lost or stolen is enough to give any sane person pause.

Robert Ferrell muses about the past, and the dark future, of the Internet. A much more serious column than his usual, but totally fitting the times.

Mark Lamourine has reviewed two books. He has high praise for *Fluent Python*, by Luciano Ramalho, a book that will sit beside his copy of Dave Beazley's *Python Essential Reference*. Mark also reviewed *Once Upon an Algorithm* by Martin Erwig.

You might find yourself wondering whether disk vendors really have taken control over block placement on modern drives. I heard Dave Anderson of Seagate mention this during FAST '07, questioned him about it, and wrote about this in a Musings column later in 2007. I've since been asked to prove this a number of times, and the best I've been able to come up with involves the documentation for a Seagate Enterprise SAS drive in 2004 [6]. I'm sure there are better examples, and even a standards doc that explains this change. If you know about this, please let me know, because people still find this hard to believe.

In the meantime, enjoy your hard drives, which are gaining not only in capacity over time, but also in intelligence.

References

[1] Rik Farrow, the long version, Morrow Micronix: https://www.rikfarrow.com/about/.

[2] Parallel ATA: https://en.wikipedia.org/wiki/Parallel_ATA.

[3] Small Computer System Interface: https://en.wikipedia.org/wiki/SCSI.

[4] T. Feldman and G. Gibson, "Shingled Magnetic Recording: Areal Density Increase Requires New Data Management," *;login:*, vol. 38, no. 3 (June 2013): https://goo.gl/wj5Doi.

[5] ACM CCS 2017 Agenda: https://ccs2017.sigsac.org/agenda.html.

[6] Seagate documentation: https://www.seagate.com/www-content/product-content/enterprise-hdd-fam/enterprise-capacity-3-5-hdd-10tb/_shared/docs/100791103d.pdf.

Letter to the Editor

Great interview of Peter in the Winter 2017 *;login:*. I had the pleasure of knowing and learning from Peter for many years.

Rik asked, "What happened with Multics?" It was a moderate commercial success, until its hardware became obsolete and was not replaced. The operating system design and features, and the people who helped build them, influenced many subsequent systems, including CHERI.

I can amplify Peter's remarks on Multics in a few areas.

Peter said, "The 645 was pretty much frozen early"—in fact, Multics had a major hardware re-design in 1973 (after Bell Labs left Multics development) when the GE-645 was replaced by the Honeywell 6180. The 6180 architecture extended the Multics hardware-software co-design, providing support for eight rings in hardware (instead of the 645's 64 rings simulated in software), as well as better security. A later I/O controller ran in paged mode and supported Multics device drivers that ran unprivileged in the user ring.

The transition from discrete transistor implementation to integrated circuits gave us 1 MIPS per 6180 CPU rather than the 645's 435 KIPS. The later DPS8/70 was rated at 1.7 MIPS.

Another minor clarification: Peter said, "The buffer overflow problem was solved by making everything outside of the active stack frame not executable, and enforcing that in hardware." Actually, there were several features preventing buffer overflows in Multics:

- The PL/I language has bounded strings and arrays, not just pointers.
- CPU string instructions enforced bounds at no runtime cost.
- "Execute" permission is limited to code segments.
- The stack grows from low addresses to high.
- ITS format prevents use of random data as pointers.
- The segment numbers are randomized.

See http://multicians.org/exec-env.html#buffer_overflow for more on this topic.

Another clarification: Peter said, "In the early 1970s there was even an effort that retrofitted multilevel security into Multics, which required a little jiggling of ring 0 and ring 1. I was a distant advisor to that (from SRI), although the heavy lifting was done by Jerry Saltzer, Mike Schroeder, and Rich Feiertag, with help from Roger Schell and Paul Karger." There were several projects to enhance Multics security so it could be sold to the US Air Force. The MLS controls were done by a project called Project Guardian, led by Earl Boebert. A more ambitious project to restructure the Multics kernel, led by Schell, Saltzer, Schroeder, and Feiertag, was canceled before its results were included in Multics (http://multicians.org/b2.html#guardian).

In the mid-'80s, the NCSC B2 security level was awarded to Multics, after a thorough examination of the OS architecture, implementation, and assurance. The evaluation process found a few implementation bugs; much of the effort in attaining the digraph was documenting the existing product.

There are over 2000 names on the list of Multicians. I am mildly uncomfortable at being the only person mentioned by Peter as "heavily involved" in Multics—we all were. I did my part, but there were many others who made contributions more important than mine, and some who worked on Multics longer. I look back on those times and those colleagues with affection and awe.

Jeffrey Yost's interview with Roger Schell, a key person in the design of security features and TCSEC ("the Orange Book"), is also fascinating: https://conservancy.umn.edu/handle/11299/133439.

Regards,
Tom Van Vleck
thvv@multicians.org