Musings

RIK FARROW

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

While watching Brent Welch present his storage technologies tutorial right before FAST '15, I found myself thinking about the topic of the storage hierarchy: registers, various caches (SRAM), memory (DRAM), then the much slower non-volatile storage mediums: flash, disk, and tape. Brent shared the table he used in his talk (Figure 1), which really got me going in this direction. Again.

I realized that I have taken the path of discussing the enormous disparity between processor speed and even DRAM too many times already. Of course, I am not the first to think about this problem. Admiral Grace Hopper used to hand out one-foot-long pieces of wire and explain that each piece represented one nanosecond, the maximum distance light could travel during that brief moment of time. Her point was to make the most of every instruction, as even the speed of light was a real limitation.

Of course, Hopper worked in an era where a single computer filled a room, long before the beginning of integrated circuits. And the wires she handed out, actually about 11.8 inches long, represented the speed of light in a vacuum: in copper, light only travels about half that far in a nanosecond. In the fiber optics I'll soon be addressing, light travels about eight inches in a nanosecond.

Kimberly Keaton of HP Labs also caught my attention. I was congratulating her for winning a Test of Time award (see the FAST '15 conference report in this issue), along with a crew of other HP Labs employees, and Keaton suggested that I take a look at "The Machine." I'd heard of The Machine, but I'd also heard of something that sounded similar in a ParLabs paper six years ago. Keaton pointed out that this new project was more like the FireBox, a project that Krste Asanović spoke about in his keynote during FAST '14 [2].

The Machine

So I visited the Web site for The Machine [3], expecting to find the usual marketing there. And there really wasn't much, so I started watching the videos. I skipped the one by HP CTO Martin Fink at first, but he actually does spend more time on the details than the other two videos found on that page (as of March 30, 2015).

But I wanted to know more, so I asked for "someone technical" who could talk about The Machine. HP PR set me up with Kirk Bresniker, an HP fellow and the architect who helped to develop the entire category known as blade servers. Kirk was now working on The Machine, as well as on something called HP Moonshot, which is involved in the project.

First, here's the meat of Fink's talk, at 16:10 into the video: electrons, photons, and ions. Electrons compute, photons communicate, and ions store. Okay, I hope that sounds weird, or at least trite, but there is meat there. There are really three big pieces to The Machine that Fink tells us about, and I want to discuss the last two first.

Like FireBox, The Machine will rely on photonic communications. By using light over fiber, instead of electrons over wires, you can get faster communication. Eliminating wires also helps to deal with design issues because wires have capacitance, which makes communicating over wires slower in more ways than one.

L1 CPU Cache	4 cycles (~1 nsec)	32K
L2 CPU Cache	10 cycles	256K
LLC CPU Cache	40 cycles	1 MB
DRAM	240 cycles	16 GB
RDMA Read	6K cycles (2 usec)	16 GB
FLASH Read	150K cycles (50 usec)	128 GB
FLASH Write	1500K cycles (500 usec)	128 GB
HDD Write min	1500K cycles (500 usec)*	4 TB
HDD Read min	15000K cycles (5 msec)	4 TB
HDD Read max	75000K cycles (25 msec)	4 TB
Tape File Access	150000000K cycles (50 sec)	6 TB

Figure 1: Brent Welch's table showing storage latency and example capacities from his FAST '15 storage technology tutorial

Fink said that The Machine, when working at datacenter scale, can access memory in any rack in just 250 nanoseconds, which, if we ignore issues like switching times, limits the length of fiber that can be used to half of 167 feet, without actually including the reading or writing time for the remote memory involved. There are other problems with photonics, mostly because photonics are still largely research projects. But photonics will be real one day, and I've already heard talks about photonic switching at NSDI.

Next, let's discuss the ions that store. Fink is referencing memristors, a non-volatile memory (NVM) store developed by HP years ago, but still not in production. Memristors are very simple: they use a layer of titanium dioxide at the intersection of nanoscale wires to store bits, which suggests that memristors could reach densities far exceeding that of flash, as well as being byte-addressable in the way that DRAM is—a cache line at a time in practice. Memristors are also expected to be eight times faster than DRAM, making them a lot faster than flash. Looking at Figure 1, that would make memristors close to LLC cache in performance.

Memristors replace both DRAM and long-term storage in The Machine. And I like that idea, of having storage class amounts of NVM with the performance of L3 cache. The balance between memory access times and CPU performance has been very skewed for decades now [4], and this would do a lot to redress that issue. For example, instead of running memcached sharded over many servers, the entire database could fit into a single system with memristor memory and run *faster* than it can run in DRAM. Of course, your SQL database could also fit into NVM as well, presuming NVM becomes as common as multi-terabyte disks are today. Any of today's big data applications would truly benefit from memristors.

If they existed.

The Reality

And there's the rub: neither memristors nor photonics are available today.

I spent a lot of time listening to Kirk Bresniker talking, as he obviously is used to giving presentations to analysts, and getting him to stop long enough for me to ask questions wasn't easy. But Bresniker presented a more practical, short-term view of The Machine.

For example, Bresniker mentioned that early versions of The Machine wouldn't necessarily be using memristors. HP Moonshot, a very large rack-mountable drawer that can contain up to 45 cartridges, is the basis for the datacenter version of The Machine. Moonshot already exists, and the cartridges currently contain Intel Atom or ARM 64-bit processors, DRAM, and storage. The storage can be hard drives or SSDs, and the drawer contains several different means for communication: a toroidal mesh, a neighbor-to-neighbor interconnect, and TCP/IP switching. Moonshot allows people to work with something that provides some of the features of The Machine today, although without the performance of memristors and photonic communications. If you read the review of the Atom-based Moonshot [5], you will see that configuring it is like working with a rack of servers in a box, where the top-of-rack switch is also in the box.

Bresniker told me that the plans are for The Machine to really eclipse Moonshot, allowing the building of thousands of heterogeneous cores connected by a photonic fabric and backed by hundreds of petabytes of NVM. The vision is an ambitious one.

HP has planned on releasing a version of Linux, called Linux++, designed to allow people to start working with an operating system that will support The Machine. Linux++ will be open source and, hopefully, a new platform for research. And there will need to be research, as having Linux working on something like The Machine, with a variety of hardware platforms as well as vast amounts of NVM arranged as a unified pool, will be very different from what researchers and developers are working with today.

Bresniker repeated a line from Fink's talk: that data will live in the peripheries, not centralized, and that we need algorithms that can access and analyze that data. When Bresniker slowed down, I asked him whether there were plans for operating systems other than Linux. Bresniker agreed that supercomputers and the Internet of Things won't be running Linux but, more likely, will be running custom operating systems, such as library OS for supercomputers and microkernels for IoT. When I asked about how to deal with crashes in a future where all memory is non-volatile, so rebooting no longer erases the problem that caused the crash, Bresniker responded that we need to build systems that are trustworthy: programs that are formally proven not to have side effects. Bresniker said that HP already has done a lot of work with fault-tolerance, and Bresniker mentioned having fault containers.

I took a couple of days to digest what I had heard from Bresniker and Fink. The Machine is an admirable concept, something that may be built, something worth doing. If HP can provide more than concepts—for example, Linux++, or better yet, working memristors—then developers and researchers will have something concrete to begin working with.

The Lineup

The opening lineup for this issue was inspired by my time at FAST '15. I encountered people presenting a poster comparing the performance of NFSv3 with NFSv4.1 in the first poster session, and asked the presenters to share their results with *;login:* readers. Chen et al. tell us about their experiments comparing both versions working on a Linux server. I think you will be pleasantly surprised, especially if you have planned on trying NFSv4.1.

I also met Christoph Hellwig during FAST, as well as during the Linux FAST workshop (see the conference report in this issue) that occurred after FAST. This wasn't the first time I had met Christoph, and I decided that I would try and uncover the path he had taken to become one of the top Linux kernel developers.

Dean Hildebrand and Frank Schmuck of IBM Research, Almaden, wrote about putting the "general" into the General Purpose File System. Their 2002 FAST paper won a Test of Time award several years ago, and they wanted to share how they had helped take a distributed file system that was designed for HPC and turn GPFS into a system that handled many different types of loads over multiple interfaces well.

I also met Carl Waldspurger during a poster session. I had already heard the presentation of their paper, where he discussed the technique they are using to calculate miss ratio curves (MRCs) that require much less time and very little memory. Not only is their work interesting, but I think that their sampling technique, using a spatially distributed hash to select their samples, could be relevant in other areas as well.

We begin our section on system administration with an article by Zbigniew Jędrzejewski-Szmek and Jóhann B. Guðmundsson on using systemd to run daemons. Systemd has become the accepted replacement in Linux for initd, the first process to start after the kernel has booted, and in this article the authors focus on the types of tasks that had traditionally been handled by a separate daemon: inetd. I learned a lot more about systemd in general, including that its capabilities are vast.

Chris Jones, Todd Underwood, and Shylaja Nukala have written about the process used for hiring SREs at Google. While this might seem an awfully narrow topic, I found myself thinking that the hiring of technical staff might function much better if managed in a fashion similar to how Google hires their SREs.

Dave Hixon and Betsy Beyer write about being a Systems Engineer (SE). If you thought becoming an SRE was tough, how about a job where there is no training or career path? I found myself wishing I were 20 years younger (for more reasons than just this), as I found the position of SE both challenging and appealing to the problem solver in me.

Andy Seely has some surreal tales from his work as an IT manager. The ideal tech manager has to balance the quirks of his reports against the oftentimes resistant framework of the larger organization. But sometimes those quirks create problems that take real creativity to solve, while at other times just a phone call helps to sort things out.

Peter Salus continues his special column on history by delving into the deep past of computer user groups. Salus starts with mainframe user groups, then takes us to the very beginning of USENIX via meetings of early UNIX users.

David Blank-Edelman and Dave Beazley synched up this issue, both writing about parallelism. Blank-Edelman shows off examples of simple parallelism in Perl, through the use of forks and threads. Dave Beazley takes a deep dive into threading in Python, with demonstrations of how the Global Interpreter Lock (GIL) affects how many threads can run and complete.

Dave Josephsen explains how to start using jmxtrans to monitor Java Virtual Machines (JVM). Dave tells us how to install, configure, and use jmxtrans so you can collect the metrics of your choice from JVMs.

Dan Geer and Dan Conway have decided to analyze the use of keywords within 12 years of articles for *IEEE Security and Privacy*. The authors search for trends, revealed through the prevalence of these keywords, that might provide hints about the security topics keeping people awake at night, as well as threats that no longer appear to be as interesting.

Robert Ferrell reflects on USENIX history, specifically, his own adventures as the SAGE historian. Warning: events that appear in his mirror may be further away than they appear.

Mark Lamourine has contributed three book reviews for this issue, on graph databases, creating visualizations within your browser, and a new book on Scratch.

We also have some conference reports in this issue. Erez Zadok, one of the FAST '15 co-chairs, wanted three of his students to have the experience of writing reports, and those reports appear in this issue. I attended Linux FAST, and did my best to capture the dialog that went on during the five-hour meeting. My goal is really to help people understand, by recounting the interaction of CS researchers and Linux kernel developers, how these two groups might work together better.

It might seem to you that I have forgotten the "electrons compute" from earlier in this column. I really haven't. I've just been saving that bit for now.

If you examine the history of computers, you might notice that early computers were not flexible at all: they were designed for performing very specific tasks. Over time, computers became more programmable and more able to work on general problems. And perhaps we have lost something by having made computers so general, as being general means that we have given up specific abilities, like the vector processors of '80s supercomputers. And perhaps we really need some of these special processors, as seen in the increasing popularity of using GPUs to complement the general purpose CPU. GPUs are, after all, special-purpose processors, but ones that never stand alone, reminding me of the floating-point coprocessors once common in early PCs.

The design described in the Tessellation paper [1] was a method of partitioning a manycore system, with heterogeneous cores, into groups based on the requirements of code executing on that group. In a way, this is not unlike one of the concepts in The Machine: heterogeneous cores connected together to form processing groups on demand.

But, by far, what I find most interesting of all about The Machine is the embrace of large amounts of non-volatile memory. Dealing with such a large pool of fast memory is quite alien to the way we in CS do things today. We've become quite accustomed to having manageable containers of memory, usually hierarchically organized, as part of servers, SAN or NAS devices, or JBODs connected to servers. In the world as imagined for The Machine, all memory forms a giant, flat space.

My guess is that we will have a difficult time wrapping our heads around such a design. Even HP spokesmen talk about moving "computation close to the data" and that "data will be distributed," which hints that there won't be a single, flat storage, but, at the very least, distributed groupings of data.

At the very least, we will be in for an interesting time indeed.

Resources

[1] Rose Liu, Kevin Klues, Sarah Bird, Steven Hofmeyr, Krste Asanović, and John Kubiatowicz; "Tessellation: Space-Time Partitioning in a Manycore Client OS": https://www.usenix.org/legacy/events/hotpar09/tech/full_papers/liu/liu_html/.

[2] Krste Asanović, "FireBox: A Hardware Building Block for 2020 Warehouse-Scale Computers," USENIX FAST '14: https://www.usenix.org/conference/fast14/technical-sessions/presentation/keynote.

[3] The Machine: http://www.hpl.hp.com/research/systems-research/themachine/.

[4] J. McCalpin, "A Survey of Memory Bandwidth and Machine Balance in Current High Performance Computers," *(circa 1995):* http://www.cs.virginia.edu/~mccalpin/papers/balance/.

[5] Tom Henderson, "First Look: HP Takes Giant Leap in Server Design" (review of Intel Atom-based Moonshot): http://www.networkworld.com/article/2174138/servers/first-look--hp-takes-giant-leap-in-server-design.html.