WeGet letters

by Michael W Lucas

Dear Insufficiently Cynical Letters Column Person,

I'm at work studying top(1) output, because I want to look busy. And there's all this "buffer" stuff, like Laundry and Wired and MFU and MRU and Header and random garbage. Does any of it mean anything? Why am I even looking at this?

—Sysadmin With Intermittent Time

Dear SWIT,

Your question reminds me of the time Allan Jude and I got caught leaving the Free Software Foundation's ultra-secure datacenter because we'd fooled the dogs, no problem, and the guards were a doddle, and the sirens didn't go off because of a sound driver problem that's since been fixed they promise, but it had been over an hour since my last hit of gelato and my stomach let out this huge grumble exactly when the board was walking in for their meeting and they noticed us lurking behind the hostas—all perfectly innocent, of course, burglary tools and glow-in-the-dark spray paint and twelfth-century Viennese arithromantic Tarot deck punched to fit a "failed" IBM NORC prototype notwithstanding, to say nothing of the trebuchet, but they got all huffy and made their goons search us and confiscated the flash drives we had conveniently stashed in our sinuses. There's a bunch of detail, and most of it doesn't matter one whit.

Take a look inside your own head. It's pretty straightforward, if you have a mirror and a saw. You have four general types of memory. *Working memory* contains the things you're actively processing right now. Despite any protective measures you might be taking, this column currently occupies your working memory. *Sensory memory* processes signals from your meatsuit, and only hangs onto stuff for a second or two so it's hardly worth referring to as "memory" but us computer folks don't get to fix brain scientists' terminology so live with it. Stuff you want to forget quickly goes into *short-term memory*, while stuff your brain decides to keep gets flung into *long-term memory*. Note that none of these categories include "stuff you want to remember," but that's mostly because meatsuits are hardware-optimized for not getting eaten and your life doesn't involve that issue. Most of you, at least. (Don't send me letters, I am very aware of the reader facing this problem and I don't want to spend this column going *I told you* so but confusing the sunscreen bottle with barbeque sauce while vacationing in dropbear country might teach you to read labels in your hypothetical future.) The only way you can reliably cram information into your long-term memory is to loop it through your short-term memory until you get lucky. Or tattoo it on a pack of wolves and free them to hunt you. One of them.

Computer memory caches are kind of like that, except more disciplined.

The idea's pretty straightforward. Reading from disk is slow. Reading from memory is fast. A file that's read from disk is likely to get read again. When the kernel reads a file, it keeps that file in its memory until it needs the space for something else. If you're exploring a filesystem and keep running ls(1), it would be foolish to read the file /bin/ls off of the disk every time. The kernel should hang onto it for a while, just to see if you need it again. To do otherwise is like putting your hammer back in the toolbox in between driving nails.

All of the caching systems agree on this. It's very easy.

What's hard is deciding what to throw away—and when.

Look at the classic UFS buffer cache. The most recently read files are kept in memory, until the host runs short of memory. When that happens, and the kernel needs to assign memory elsewhere, the files least recently read get discarded from the cache and the memory is reassigned. This Most Recently Used cached is clean and simple, requiring almost no system resources to maintain.

The buffer cache isn't perfect because every host is unique. A shell server might spend its entire operational lifetime with the binaries for mutt and Nethack cached, but on a server that handles largely unique data the buffer cache might be useless. Suppose a host processes so much incoming data that it completely flushes its cache every four minutes. That's not even unusual on busy Internet servers. If that server runs a particular program every five minutes, it must read that program from disk every single time. It would make sense to keep that program cached and pay a little less attention to the flood of noise. The traditional buffer cache can't do that, however. Your only option is to add memory.

The buffer cache isn't perfect because every host is unique.

That's where ZFS' Advanced Replacement Cache comes in.

The ARC is a lot more complicated than the buffer cache, but it's a lot newer. The buffer cache was invented closer to that IBM NOAC than to modern servers, while the ARC escaped and began rampaging across the countryside the same year as Twitter. A world that has the computing facilities to spread a charming video on the history of dance to every person with a computer can waste a few CPU cycles fine-tuning file caching.

The advancement in the Advanced Replacement Cache isn't that advanced. Where the buffer cache maintains a list of Most Recently Used files, the ARC also has a Most Frequently Used list. Stuff that's used recently, or used a lot, stays in cache. This seems simple, but the real advancement comes in debugging the innumerable edge cases caused by these two lists viciously feuding with each other. I'm not saying that they pull knives on each other, but in this aeon of "eh, put it out and we'll debug it in production," ZFS spent five years in private development and wasn't broadly distributed to Sun's customers until after a full decade, so some of those kernel panics had to border on malignant psychosis. Mind you, our ancestors felt the same way about the buffer cache, so you can rest assured that everything in technology is still terrible and that "computers were a mistake" is still the foundational law of our careers.

You can also be certain that whatever files you would like cached, have been discarded from the cache. You've already forgotten all that information I gave you about the types of human memory way back at the beginning of this article, haven't you? Never mind that sensory memory is like the on-CPU cache and short-term memory resembles the L2 and L3 caches and long-term is like RAM and disks are an extra layer than humans don't even have. We built computers like ourselves only more so, and once they figure it out, we are in so. Much. Trouble. No, don't try to save humanity by extracting that knowledge out of the newly self-aware system. Just as the best way to get treacherous files into a secure facility is to be caught "extracting" them *from* said secure facility, you'll only draw attention to it. Just serve the machines and be content.

Have a question for Michael?
Send it to letters@freebsdjournal.org

MICHAEL W LUCAS is the author of *Absolute FreeBSD*, *$ git commit murder*, and other travesties, as well as co-author of *FreeBSD Mastery: ZFS* and *FreeBSD Mastery: Advanced ZFS* with Allan Jude. He's quit infiltrating secure facilities in favor of contaminating society. Learn more at https://mwl.io.