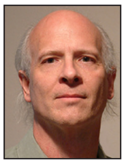
2017 USENIX Research in Linux File and Storage Technologies Summit (Linux FAST Summit '17)

RIK FARROW



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

Ric Wheeler (Red Hat) chaired the Linux FAST Summit '17. There were 50 attendees, the most yet, with 60% from large companies, 20% from universities, and the rest consultants or from smaller companies. According to Ric, 33% of the Linux FAST attendees did not attend FAST '17.

After introducing ourselves and briefly explaining why we were attending, discussion of issues with block I/O began. Someone mentioned that the latest Linux kernels can handle as many as 40 million IOPS. Ted Ts'o (Google) suggested that it's time to start considering techniques used in high-speed networking to further improve performance.

Erez Zadok (Stony Brook University) wondered how multiple write queues to the same device affected order handling. Christoph Hellwig (consultant and Linux file system hacker) said ordering isn't handled; it's an unsolved problem. Most devices behave as if they are non-volatile, returning completion codes while data is still buffered in on-device RAM. And devices perform out-of-order writes as they see fit. That pretty much guarantees that anything done by an OS, such as write barriers, can't work.

Andrew Morton (Google) then began the "how to work with the Linux kernel" section, a tradition at Linux FAST. Andrew suggested sending him your first patch (for file system patches) rather than just posting your patch to the Linux-kernel list. Andrew pointed out that the kernel developers had gotten a bad reputation for being harsh, but now "we're pretty professional."

Ted Ts'o put this another way. Suppose someone unknown to the developers sends an email, which is like cold calling. You want to work through introductions if at all possible, just as you would in any social situation, and it's also important to use the most recent kernel possible. You can get the most recent build at kernel.org, but if you are working with a specialist in some area, ask that person which build to work with. In general, choosing a stable release means you will be working with a kernel that will be supported for some time.

Ted also mentioned that he has created some regression testing tools for file systems. You can find these tools at https://github.com/tytso/xfstests. Ted, who co-authored the FAST '17 paper "Evolving Ext4 for Shingled Disks" (in this issue), tried the patches written for improving SMR performances against his regression testing tools. The patches failed, although they were good enough to run the benchmarks used to write the paper. Those patches will eventually be cleaned up and merged into the upstream kernel.

Editor's Note: This report includes some summaries from Mai Zheng (zheng@nmsu.edu) and Om Rameshwar Gatla (omram@nmsu.edu)

George Amvrosiadis (student at Carnegie Mellon University) mentioned having three thousand lines of code that he shared with members of the file system group. He said he got lots of feedback and started to develop a relationship with this group of kernel hackers. He also wanted a particular tracepoint added to the kernel and hasn't succeeded yet. But he wasn't discouraged by the process.

Ric then shifted the focus to FUSE by asking Sage Weil (Red Hat, key author of Ceph) about his experience working with FUSE. Sage said that although writing user-space software is easier, you still run into kernel issues. For example, you don't control the page cache or writeback queue.

Erez mentioned a paper he co-authored for FAST '17 (Vangoor et al., "To FUSE or Not to FUSE: Performance of User-Space File Systems"), where they played with lots of switches in FUSE to see how those affected performance. He was surprised there was so little documentation for FUSE. George mentioned that the patch he wanted was a tracepoint that would let them know when metadata had been modified. Sage pointed out that with FUSE, the kernel is still doing a lot of work "under the hood" and that FUSE performance has gotten a lot faster over time.

Another person from Red Hat mentioned that one big advantage with using FUSE is that you can run your file system without having to patch a certified kernel. Jeff Darcy (Red Hat) agreed and added that trying to run non-standard kernels in the cloud was a non-starter.

John Grove (Micron) said his group was developing a new file system and that being able to work in FUSE for prototyping was a great help.

The next topic covered had to do with writing "dirty" buffers back to disk. Jonathan Amit (IBM Israel) has a problem with a project that allows customers to write many gigabytes, using multiple threads. But there is just one kernel thread serving the write-back cache, and to get the best performance they just bypass the page cache. Ted answered that using O_DIRECT is the way people who are passionate about performance handle this problem. Jonathan said it was not always easy to use O_DIRECT, and Ted agreed.

Mai Zheng (New Mexico State University) mentioned two cases where bugs in the Linux kernel affected devices' behavior. In one case he tested dozens of SSDs under power faults, and many devices exhibited corruptions in the tests (see "Understanding the Robustness of SSDs under Power Fault" presented at FAST '13). However, after several years, the same tests were performed using a newer kernel. It turns out that a bug patch (by Christoph Hellwig) changes the corruptions observed on some devices (published in 2016 in *ACM Transactions on Computer Systems*). In another case that happened at Algolia datacenter, Samsung's SSDs were blamed for data corruption initially. However, Samsung's engineers eventually found that it was a kernel bug that caused the trouble (http://www.spinics.net/lists/raid/msg49440.html); the bug was patched by Martin K. Petersen.

Ted commented that only enterprise-class SSDs can be relied upon (at all) for safe behavior on power fail. The enterprise-class SSDs have super-capacitors that store enough power to write all data stored in the RAM within the SSD on power fail, and vendors charge three times as much as they do for consumer class SSDs. Some vendors do certify their SSDs, but you should check them under real power-fail conditions, like pulling the plug. Peter Desnoyers (Northeastern University) suggested using an Arduino with a relay for experimenting with cutting off power.

Jonathan then changed the topic to ask about NVME device performance. Christoph replied that he had rewritten that device to make it simpler: no waiting, no polling, and this should be in the 4.9 kernel.

Om Rameshwar Gatla (New Mexico State University) raised a question regarding how robust the local and large-scale file system checkers are besides e2fsck. Christoph replied that even the XFS repair utility is as vulnerable to faults as e2fsck is, and this could be the same with the repair utility of B-tree file system (btrfs). In regards to the robustness of checkers for large-scale file systems, developers of Ceph said that their file system includes many fault-handling techniques such as journaling, data replication, etc. by which this situation may be mitigated.

Ric Wheeler commented that many repair utilities, such as XFS repair, consume a lot of memory and that this problem could serve as a good research topic. The other topic discussed regarding `fsck` was its running time. Ric suggested running all file system checkers of an aging, fragmented file system on a hard disk whose sizes are on the magnitude of terabytes and observe the memory consumption and total run times. The results from these experiments may provide a good research opportunity. Ted added that the problem that e2fsck's slowness is because EXT file systems maintain lots of bitmaps to track information on all the inodes, direct and indirect blocks, etc., but the overall memory consumption of e2fsck is far less than any other file system checker. To support his argument, Ted gave an example where they ran e2fsck on a 6 TB hard disk that was 80% full and had the Hadoop layout. e2fsck consumed less than 9 MB of memory to complete. Ted added that having a large number of hard links creates the greatest challenge for `fsck`.

Niels De Vos (Red Hat) mentioned that GlusterFS uses extended attributes (xattrs) in ext4, and if users edit the attributes, you really get into big trouble. Of course, there's no way that an fsck could check for that. They also do erasure coding for files, which means that checking involves reading files on multiple servers.

Om also asked about the error reporting mechanism from file systems or lower layers. He wanted to know more details when facing some errors (e.g., why a volume is reported "unmountable"). Ted, Christoph, Ric, and some others commented that the current mechanism relies on error numbers (errno). The overhead of passing more detailed information around might be high. Also, `dmesg` is a good place to look for more detailed error messaging in current systems.

There was some discussion about mapping and providing low-level block information to higher level software. Ted commented that debugfs (https://www.kernel.org/doc/Documentation/filesystems/debugfs.txt) provides such a mechanism. Mai commented that in his project about analyzing the bugs in databases and file systems, debugfs has helped a lot for examining the relationship between the corruption at low-level I/O blocks and the impact on database logs.

Jonathan asked about why mmaping two terabytes of memory takes so long. Andrew pointed out that populating two terabytes working with four-kilobyte pages was always going to take a long time, leading Jonathan to wonder whether the Persistent Memory (PMEM) driver supported huge pages.

Pankaj Mehra (Western Digital) said that people so far don't understand PMEM, as they are not using `mmap` (see Andy Rudoff's article "Persistent Memory Programming" in this issue). Ted agreed: you don't want a POSIX layer, you want to `mmap` PMEM into your process memory. You can treat PMEM as superflash, but there's lots of overhead there.

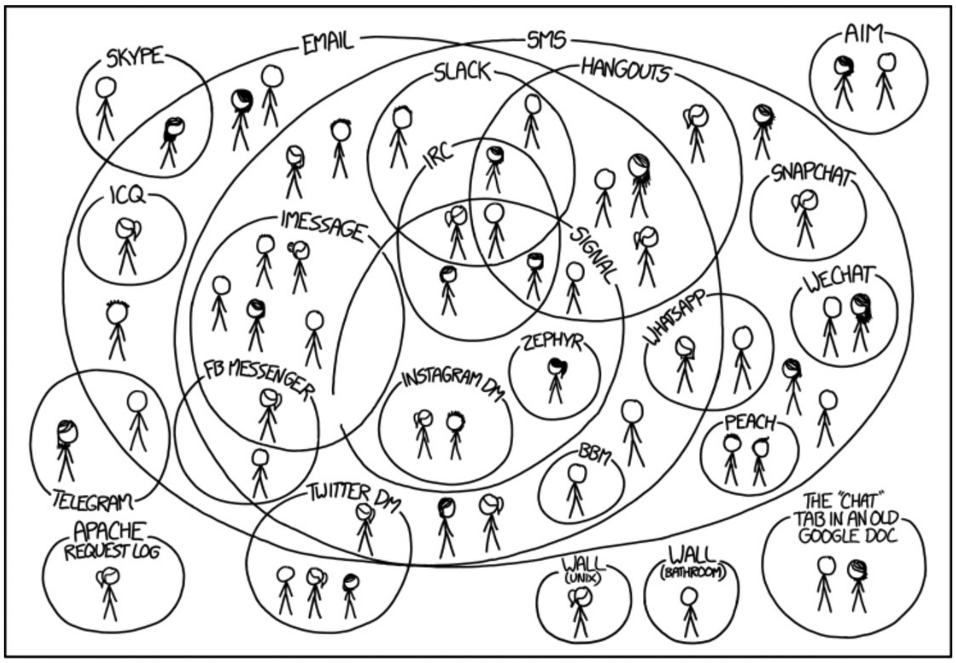
Pankaj replied that if you have PMEM, you are going to want to manage it, which includes encryption, snapshots, naming, permissions, and free space. Sam Fineberg (Consultant) pointed out that the traditional way of dealing with memory errors in Linux is to use ECC or to crash. Ric mentioned that the Micron-Intel XPoint PMEM will be able to report bad memory. Mai mentioned a paper published in EuroSys '13 which makes the `msync()` system call robust ("Failure-Atomic `msync()`: A Simple and Efficient Mechanism for Preserving the Integrity of Durable Data"). Christoph confirmed that the idea as well as the findings in a follow-up paper from the same group have been incorporated into the Linux kernel.

Pankaj continued: "When we first came up with the term PMEM, we were very careful. The way we handled this is the way Rudoff describes it: one instruction per address. When you do a store, we will store. If you want PMEM to do transactions, you lose the performance benefits."

In the (near) final topic of the day, Ted said that he is currently working on data encryption at the file system level and that there are many challenges to it, such as how to provision crypto keys for encryption and decryption, and where to store them securely. Ted also said that the efficiency is highly architecture-dependent, with Intel Skylake able to encrypt one word per cycle, but ARM CPUs having no native support.

The final topic concerned tuning the page cache, and Ric pointed out that there is a tool called `tuned` that helps with picking appropriate sets of tuning for storage, and that you can actually find tuned profiles for different use cases.

XKCD xkcd.com



I HAVE A HARD TIME KEEPING TRACK OF WHICH CONTACTS USE WHICH CHAT SYSTEMS.