acmqueue Deduplicating Devices Considered Harmful

A good idea, but it can be taken too far

David Rosenthal

During the research for their interesting paper, “Reliably Erasing Data From Flash-based Solid State Drives,”¹ delivered at the FAST (File and Storage Technology) workshop at San Jose in February, Michael Wei and his co-authors from the University of California, San Diego discovered that at least one flash controller, the SandForce SF-1200, was by default doing block-level deduplication of data written to it. The SF-1200 is used in SSDs (solid-state disks) from, among others, Corsair, ADATA, and Mushkin. In hindsight, this sentence from the SF-1200’s marketing collateral is a clue:

“DuraWrite technology extends the life of the SSD over conventional controllers, by optimizing writes to the Flash memory and delivering a write amplification below 1, without complex DRAM caching requirements.”

It is easy to see the attraction of this idea. Because a flash block needs a time-consuming erase operation before it is written with new data, flash controllers use a block remapping layer, called the FTL (flash translation layer). This layer translates from logical to physical blocks, allowing the controller to write data incoming for a logical block to a previously erased physical block and then update the map rather than having to wait while the physical block is erased. The FTL also mitigates the limit to the number of times a block can be written. Flash devices have more physical than logical blocks so that worn-out blocks can be replaced and writes distributed evenly across the set of blocks. By enhancing this layer to map all logical blocks written with identical data to the same underlying physical block, the number of actual writes to flash can be reduced, the life of the device improved, and the write bandwidth increased.

Deduplication is a good idea, but like all good ideas, it can be carried too far. Deduplicating in devices that host file systems, especially doing it unannounced by default, is not a good idea.

File systems write the same metadata to multiple logical blocks as a way of avoiding a single block failure causing massive, or in some cases total, loss of user data. An example is the superblock in the UFS (Unix file system). Suppose you have one of these SSDs with a UFS on it. Each of the multiple alternate logical locations for the superblock will be mapped to the same underlying physical block. If any of the bits in this physical block goes bad, the same bit will go bad in every alternate logical superblock.

A PROBLEM FOR ZFS

In brief, that devices sometimes do this is very bad news indeed, especially for file systems such as Sun’s ZFS, intended to deliver the level of reliability that large file systems need.

Based on discussions with Kirk McKusick and the ZFS team, the following is a detailed explanation of why this is a problem for ZFS. For critical metadata (and optionally for user data) ZFS stores up to

three copies of each block. The checksum of each block is stored in its parent so that ZFS can ensure the integrity of its metadata before using it. If corrupt metadata is detected, then it can find an alternate copy and use that. Here are the problems:

- If the stored metadata gets corrupted, the corruption will apply to all copies, so recovery is impossible.
- To defeat this, you would need to put a random salt into each copy so that each block would be different. The multiple copies, however, are written by scheduling multiple writes of the same data in memory to different logical block addresses on the device. Changing this to first copy the data into multiple buffers, then salt them, and then write each one once would be difficult and inefficient.
- Worse, it would mean that the checksum of each copy of the child block would be different; at present they are all the same. Retaining the identity of the copy checksums would require excluding the salt from the checksum, but ZFS computes the sum of every block at a level in the stack where the kind of data in the block is unknown. Losing the identity of the copy checksums would require changes to the on-disk layout.

This isn't an issue specific to ZFS; similar problems arise for all file systems that use redundancy to provide robustness. The bottom line is that drivers for devices capable of doing deduplication need to turn it off. One major advantage of SSDs, however, is that they live behind the same generic disk driver as all SATA (serial ATA) devices. Using mechanisms such as FreeBSD's quirks to turn deduplication off may be possible, but that assumes that you know the devices with controllers that deduplicate, that the controllers support commands to disable deduplication, and that you know what the commands are. ¶

REFERENCES

1. Wei, M., Grupp, L., Spada, F., Swanson, S. 2011. Reliably erasing data from flash-based solid state drives. Presented at the Ninth Usenix Conference on Flash and Storage Technologies, San Jose (February 15-17).

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

DAVID ROSENTHAL has been an engineer in Silicon Valley for a quarter of a century, including as a Distinguished Engineer at Sun Microsystems and employee #4 at Nvidia. For the past decade he has been working on the problems of long-term digital preservation under the auspices of the Stanford Library.

© COPYRIGHT IS HELD BY THE AUTHOR