Exam Assignments 13

Cells, Pages and Blocks

Cells

Cells are the smallest units in an SSD. Depending on the type of SSD, they can store one to four bits of information in the form of electronic charge.

SSDs with cells that store more bits of information are cheaper, since more storage space can fit on the same chips. However, they also lose some performance and may be more prone to degradation than SSDs with fewer bits per cell.

Pages

Many cells are organized together into a page. A typical page size is 4KB. Pages are the smallest unit on an SSD that can be read or written.

Note that pages cannot be updated. After they have been written to once, they need to be erased before they can be written to again.

Blocks

Many pages are organized together into a block. A typical block size is 512KB or 1MB. Blocks are the smallest unit on an SSD that can be erased.

Pages can't be updated; if the user wants to update some information on a page, what actually happens is that the updated page contents get written to another page while the original page is marked stale. Stale pages are basically dead until they get erased. When the garbage collection algorithm deems it time to erase a block (probably when most of the block is marked stale), the block gets erased and all pages on it can be written to again.

Purpose of Garbage Collection

As explained in the previous section, blocks need to be erased so that they can be written to again. Thus, someone needs to decide when which blocks get erased. This role is filled by the garbage collection algorithm.

This algorithm is crucial to the performance of the SSD, as block erasure has a significant latency.

Purpose of Wear Leveling

As cells in an SSD get written to and erased, some latent extra charge builds up inside them. When this charge gets too high, it becomes difficult to tell bit states of the cell apart and the cell becomes useless. As even a few broken cells cause significant problems, the wearout should be spread evenly across all cells so that the SSD can function as long as possible.

This is what wear leveling addresses. It tries to spread the wearout evenly across all cells by periodically erasing and rewriting some rarely used data to a different block.

SSDs with an M.2 Form Factor

M.2 is a form factor for SSDs. SSDs with the M.2 form factor have a long rectangle shape (typically 22mm wide and between 30mm and 110mm long), compared to SATA SSDs which are nearly square shaped.

They may be SATA-based, PCIe-based with NVMe support or PCIe-based without NVMe support. A PCIe connector generally offers more bandwidth than a SATA connector, but to get the most out of it, the NVMe protocol needs to be supported by the SSD.

M.2 form factor SSDs also come with one of three types of keys (edge connector shapes). The three types are "B key", "M key" and "B & M key".

The compatibility of these different connector types, protocol supports and edge connector shapes needs to be considered when buying an SSD or choosing parts to use with it.

Influence of Garbage Collection and Wear Leveling on Write Amplification

Garbage collection causes write amplification, the phenomenon that the SSD needs to write to more pages than the user actually asks them to. This happens because when garbage collection finds that a block should be erased, the block's contents that are not stale get written to a different block, thus incurring more write operations.

Wear leveling also increases write amplification, because it periodically erases and rewrites rarely used data. The user did not ask for the data to be updated, it is being rewritten to prolong the life of the SSD, while incurring some additional write operations.

Recommendations for Writing SSD-Friendly Code

Adopt Compact Data Structures

The smallest unit for reading and writing in an SSD is the page. If we scatter our data among multiple small files and read each individually, then each access will read/update a whole page and waste some bandwidth. This is why we should use compact data structures to avoid scattered accesses and improve performance and also make our SSD live a little longer.

Avoid Full SSD Usage

The fuller an SSD is, the more write amplification is increased. One cause for this is that when many blocks on an SSD are full, more refreshing needs to be done for wear leveling. Another cause is that garbage collection needs to work a lot more if the SSD is full, as more blocks with valid data need to be moved to free a block.
In extreme cases, the garbage collector needs to work hard for every single operation, causing a massive slowdown.

Overprovisioning (SSD has some more blocks than it needs to have, used only for garbage collection) can help to stave off this performance drop for a bit longer. But in general, we should only fill our SSDs to a reasonable level.

Use Multiple Threads for Small IO

Because an SSD has multiple levels of internal parallelism, it can handle parallel requests. This means that using a single IO thread for a task that could be parallelized wastes some bandwidth.
If we, for example, have multiple small files we wish to access, we could create a thread for each one to better utilize the SSD's capabilities.

Reducing CPU Load

When accessing IO, especially in parallel, asynchronous calls should be used instead of blocking calls. This allows the CPU to keep working productively while waiting for the IO operation to complete.

CPU load can also be reduced by disabling OS buffering. This causes data to be loaded without saving to RAM inbetween.

Solving Problems That Do Not Fit In DRAM

Using memory mapped files (Numpy example: `numpy.memmap`) one can work on data structures saved on a file without loading them into main memory. On SSDs this incurs

much less of a slowdown than on a hard drive.
This provides a simple alternative to distributed Big Data systems for problems that work on huge datasets.