Technical Disclosure Commons

Defensive Publications Series

May 2020

Efficient Storage of Data Chunks in Solid State Drives

Bin Tan

Ricky Benitez

Narges Shahidi

Efficient Storage of Data Chunks in Solid State Drives

ABSTRACT

The data update mechanism for an SSD involves overprovisioning, e.g., a storage footprint larger than the actual data size, and write amplification, e.g., a physical amount of information written being a multiple of the actual amount of information. Both these overheads are exacerbated when applications randomly generate data chunks of size that is small compared to the size of the designed erase unit of the SSD. Per the techniques of this disclosure, small, random data chunks of similar size are padded such that a sequence of such data chunks aligns to the erase unit boundary of the SSD. At the same write amplification, overprovisioning is thereby reduced.

KEYWORDS

- Solid-state drives
- Non-volatile memories
- Flash drives
- NAND flash
- Overprovisioning
- Write amplification
- Erase unit
- Data block alignment

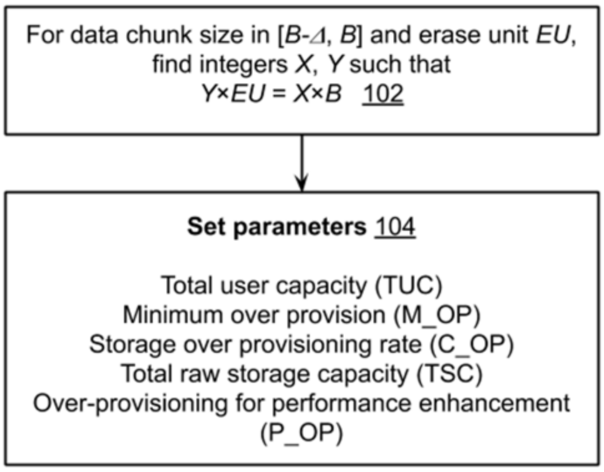
BACKGROUND

Due to the physical characteristics of solid-state drives (SSD), their data update mechanism involves overprovisioning (storage footprint larger than the actual data size) and write amplification (physical amount of information written being a multiple of the actual amount of information). Overprovisioning reduces available storage capacity. Write amplification reduces the life of the SSD. Overprovisioning and write amplification can be traded off each other to some extent; however, both these overheads are exacerbated when applications randomly generate and store data chunks of size that is small compared to the erase unit of the SSD. For example, if data chunks of sizes 0.96-1.0 MB are stored randomly and the SSD backend throughput is 1 GB/s, a 200 MB/s write throughput requires overprovisioning as high as 20%. Similarly, updating a small portion of a large erase unit involves significant write amplification.

DESCRIPTION

Per the techniques of this disclosure, small, random data chunks of similar size are padded such that a sequence of such data chunks aligns to the erase unit boundary of the SSD. Sequential erase unit alignment is preserved, thereby reducing overprovisioning at a given write amplification.

Consider a stream of random data chunks of similar size B, e.g., between 0.96 and 1.0 MB, that are to be stored on an SSD with block size 4 MB. The techniques of this disclosure find an integer multiple X of the chunk size that equals another integer multiple Y of the block size (also known as erase unit, EU, the smallest unit of storage that a NAND flash can erase at any one time). Mathematically, $Y \times EU = X \times B$. In this example, X=4 and Y=1. Incoming data traffic is tagged with its sequential alignment class (SAC), which is indicative of chunk size. The SAC tag determines the erase unit that the data chunk is directed towards. If necessary, the data chunk is padded to match the size of the selected erase unit and is then written to the erase unit. The techniques ensure that data chunks don't cross erase unit boundaries, e.g., preserve sequential erase unit alignment, while packing in as many data chunks as needed to fill up a block. Better candidates for NAND flash memory reclamation are thereby found during garbage collection.


For data chunk size in [B-Δ, B] and erase unit EU, find integers X, Y such that Y×EU = X×B 102
Set parameters 104
Total user capacity (TUC)
Minimum over provision (M_OP)
Storage over provisioning rate (C_OP)
Total raw storage capacity (TSC)
Over-provisioning for performance enhancement (P_OP)

Fig, 1: Data chunk Sequential Alignment Class (SAC) configuration

A data chunk sequential alignment class (SAC) configuration is performed for initial configuration of the SSD, as illustrated in Fig. 1. In this process, for a data chunk of size between $B-\Delta$ and B, integers $X \geq 1$ and $Y \geq 1$ are found (102) such that $Y\times EU = X\times B$, where EU is the size of the erase unit and $B > X\times\Delta$. The parameter $\Delta \geq 0$ represents the spread of the data chunk size. Also, if $Y > 1$, X can be set to 1. Data traffic tags TF_TAG_*j* are created, where *j* are predefined non-negative integers corresponding to sequential alignment classes (SAC). Parameters for the SSD are set (104) as follows:

- The minimum overprovision *M_OP* is set to

  $$M\_OP = (((X \times \Delta) / (Y \times EU)) + ((K \times Y \times EU) / TUC)) \times 100\%,$$

  where *TUC* is the total user capacity, and $K \geq 1$ is a pre-configured number to reserve $K \times Y$ blank erase units for the purposes of flushing write buffers and for persistent buffering.

- The storage overprovisioning rate C_OP is configured as $C\_OP \geq M\_OP$, and the total raw storage capacity TSC is configured as

$$TSC = TUC \times (1 + C\_OP).$$

- The overprovisioning rate for performance enhancement is set as

$$P\_OP = C\_OP - M\_OP.$$

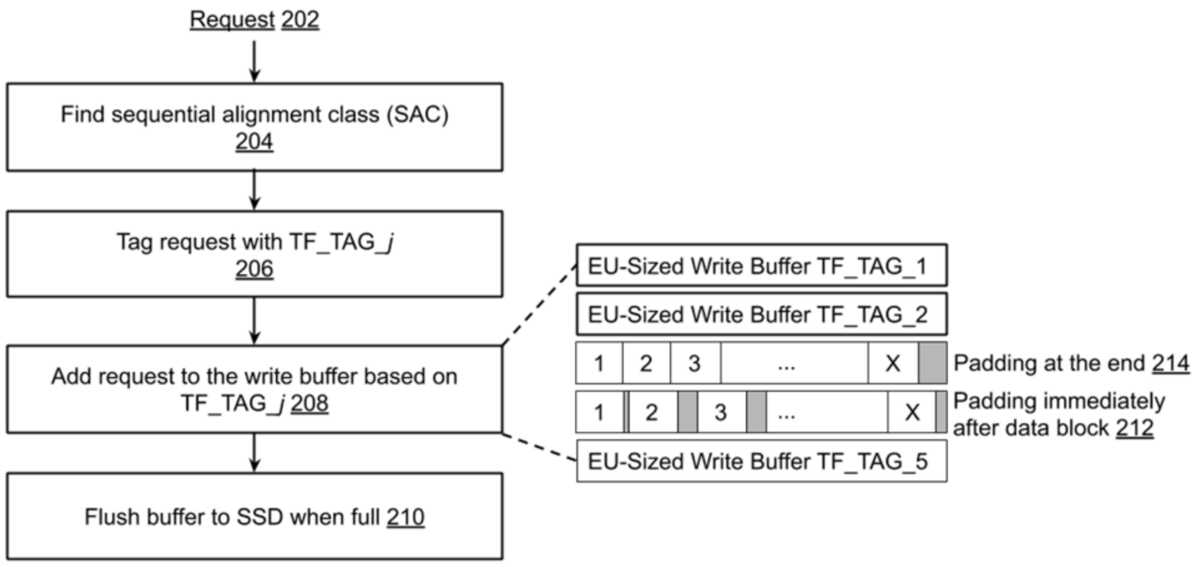

Request 202
Find sequential alignment class (SAC) 204
Tag request with TF_TAG_j 206
Add request to the write buffer based on TF_TAG_j 208
Flush buffer to SSD when full 210
EU-Sized Write Buffer TF_TAG_1
EU-Sized Write Buffer TF_TAG_2
1
2
3
...
X
Padding at the end 214
Padding immediately after data block 212
EU-Sized Write Buffer TF_TAG_5

Fig. 2: Storing data chunks to preserve SSD erase unit boundaries

Fig. 2 illustrates in greater detail storing data chunks to preserve SSD erase unit boundaries, per techniques of this disclosure. Upon receipt of a request (202) to store data in the SSD, the SAC is found for the data chunk request (204) based on the values of *B*, *X* and *Y*. The data chunk request is tagged with TF_TAG_*j* (206). A write buffer of size *EU* is set (208) for data chunks with tag TF_TAG_*j*. If $Y > 1$, the sequence of erase unit indices is tracked such that the tail erase unit is identifiable.

When a write data block arrives at the write buffer, padding is performed either immediately to make the data block size *B* (212), or until the tail erase unit is reached to satisfy

$Y \times EU = X \times B$ (214). This allows the logical address of the beginning of the block to have a misalignment of size B. When the writer buffer is full, the write buffer is tagged with TF_TAG and flushed to a blank erase unit in the SSD (210).

Per the techniques, the effective over provisioning (E_OP) is given by

$$E\_OP = (P\_OP + (1/X) \times 100) \%.$$

The techniques achieve $E\_OP \geq C\_OP$ and enable higher SSD capacity utilization, improve SSD input-output throughput performance, and can extend SSD life. For example, in an SSD with a 1 GB/s backend throughput, if the data chunk size $[B-\varDelta, B]$ is in the [0.99, 1] MB range and the EU size is 4 MB, a 250 MB/s write throughput can be achieved with just 4% overprovisioning, compared to a traditional 25% overprovisioning.

CONCLUSION

Per the techniques of this disclosure, small, random data chunks of similar size are padded such that a sequence of such data chunks aligns to the erase unit boundary of the SSD. At the same write amplification, overprovisioning is thereby reduced.