August 12, 2019

Storing non-overlapped data streams in solid-state drives

Bin Tan

Narges Shahidi

Manuel Benitez

Storing non-overlapped data streams in solid-state drives

ABSTRACT

In solid-state drives (SSDs), data is generally not updated in place. Rather, data update in SSDs involves garbage collection, in turn facilitated by over-provisioning of storage capacity. Over-provisioning reduces actual available storage capacity. Garbage collection results in write amplification, an undesirable phenomenon where the amount of information physically written is a multiple of the logical amount of information to be written. Write amplification reduces the life of the SSD. Write amplification is partially a result of treating input data streams as random data streams and subjecting the streams to a pre-configured maximum data capacity.

Per the techniques described herein, input data streams, which may comprise sequential streams and/or random streams, are treated as non-overlapped data streams, e.g., streams that have non-overlapping logical addresses. The techniques thereby achieve near-zero over-provisioning with a close-to-unity write amplification and can provide SSD life and throughput that is significantly better than conventional techniques.

KEYWORDS

- Solid-state drive
- Non-overlapped data stream
- Random data stream
- Sequential data stream
- Log data
- Write amplification

BACKGROUND

In solid-state drives (SSDs), data is generally not updated in place. Rather, data update in SSDs involves garbage collection, in turn facilitated by over-provisioning of storage capacity. Over-provisioning reduces storage capacity and increases cost. Garbage collection results in write amplification, an undesirable phenomenon where the amount of information physically written is a multiple of the logical amount of information to be written.

Consider, for example, a server with 1 TB SSD storage. The server routinely handles spawning and termination of a large number of application-logging processes, e.g., a hundred processes. Each logging process generates a large amount of log data in a day, e.g., up to 11 GB of log data, while the total generated log data at any time does not exceed 950 GB. Upon termination of a process, the log data is sent to backup servers, and the local copy is invalidated. If 11 GB of dedicated storage is allocated to each logging process, then the total required storage is 100 ×11 GB = 1.1 TB, which exceeds the available storage space, and is therefore not feasible. One realistic alternative is to let all logging processes share the entire 1 TB SSD. However, in such a situation, even though data generated by each logging process is sequential, the overall data-writing behavior is random due to the randomness of both the log data size and the timing of the spawning/termination of logging processes. The over-provisioning being just (1TB - 950GB)/950GB = 5.26%, performance of the SSD is low and write amplification high.

Write amplification in current SSDs can be quite high, e.g., the number of physical writes can be as high as ten times the number of logically required writes. High write amplification and over-provisioning can partially be traced to the SSD treating input streams, even streams that don’t have an overlap in logical address space, as random data streams. Write amplification also

trends higher if the SSD requires a pre-configured maximum data storage capacity for the stream.

DESCRIPTION

Non-overlapped data streams (NOLDS) are defined as data streams that do not have overlapping logical addresses, at least prior to their invalidation points. An invalidation point can be established using, e.g., a tag-based trim command. An SSD can have two types of input streams, e.g., random data streams and sequential data streams with non-overlapping logical addresses. Per techniques of this disclosure, NOLDS are used to treat mixed input comprising both types of streams as sequential data. Additionally, the data streams are stored without preconfiguring a fixed user data capacity. The techniques thereby achieve near-zero over-provisioning with close-to-unity write amplification.

TAG_VALUE	Any distinct value, e.g., an integer, a string, etc.
STREAM_CAPACITY_POLICY	0 - ON_DEMAND_PRESERVED_POLICY: No pre-allocated EUs for the stream. If the stream needs more EUs, they are requested from the general EU pool. 1 - ADAPTIVE _PRESERVED_POLICY: A small, dynamical number of EUs are reserved for the stream so that the stream can get blank EUs with minimum latency. 2 - FIXED_PRESERVED_POLICY: A large number of EUs are allocated to the stream to accommodate the entire stream capacity. This requires that the stream user data capacity be known prior to the establishment of the stream tag.
PRESERVED_CAPACITY_EUs	If STREAM_CAPACITY_POLICY == ON_DEMAND_PRESERVED_POLICY, this value is 0. Otherwise, the value is pre-configured by the stream owner. The larger of the value, the smaller of the latency for the stream to get a blank EU.

BLANK_EU_THRESHOLD	The pre-configured minimum size of a list of blank EUs.
BLANK_EU_LIST	A field to keep a list of blank EUs so that when data needs to be written into an EU from the write buffer, a blank EU is picked up from this list.
IS_RECYCLE_PRESERVED	If true: recycled EUs with this tag retain the tag. If false: tag associated with the recycled EU is removed and the recycled EU is put back into the general pool.
WRITE_BUFFER_LEN	The size of RAM buffer for temporarily storing write data.
WRITE_BUFFER_POINTER	The pointer to the write buffer.
EU_INFO_DB_POINTER	The pointer to the data structure which tracks all EUs associated with this tag.

Table 1: The NOLDS_TAG data structure and fields thereof

Per the techniques, a data stream is associated with a data structure known as a NOLDS_TAG, the fields of which are substantially as shown in Table 1. In this table, the acronym EU stands for Erase Unit, which is a storage block in the SSD. Parts of a data stream can be held in different write buffers. The NOLDS_TAG is used to direct each stream, or part thereof, towards erase units for storage.

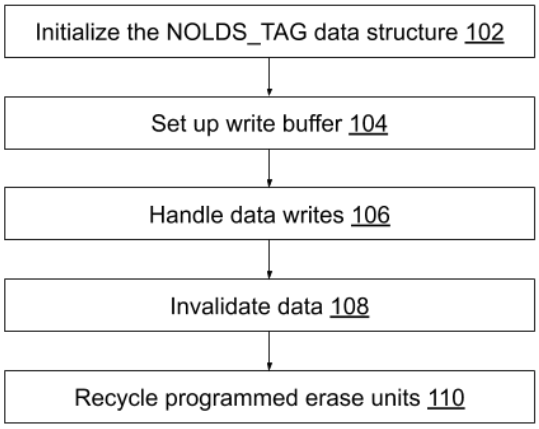

Fig. 1: Storing non-overlapped data streams

Fig. 1 illustrates storing non-overlapped data streams in SSDs, per techniques of this disclosure. The NOLDS_TAG data structure is initialized (102). A write buffer of size WRITE_BUFFER_LEN×EU is set up (104) and initialized. The write buffer can be configured as a ring buffer with WRITE_BUFFER_LEN units.

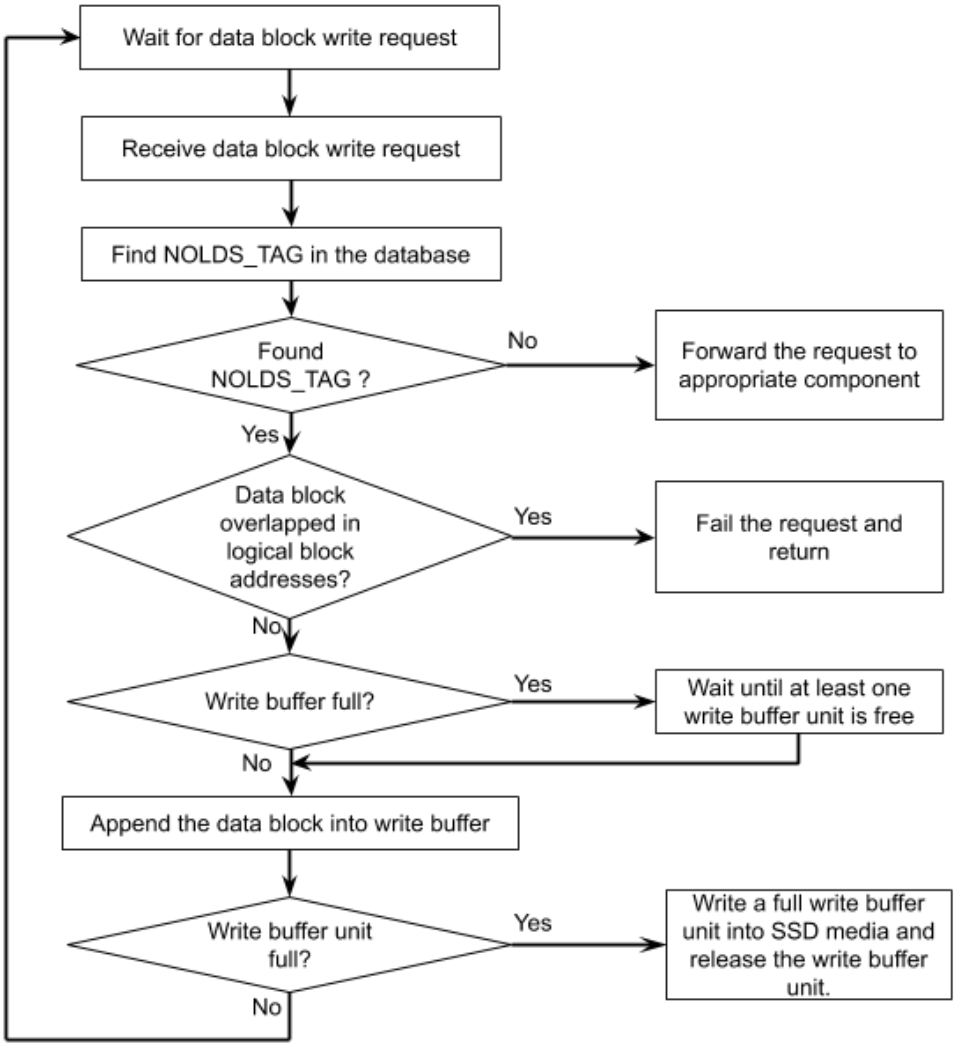

Fig. 2: Writing data

The writing of data (106), illustrated in Fig. 2, is handled as follows. When a data block write request is received without an associated NOLDS_TAG stream tag value (or the tag value is not found in the non-overlapped data stream tag database), the write request is forwarded to an

appropriate component. If the NOLDS_TAG is found, and if the logical address of the data block violates the non-overlap property, the write command fails and returns. If the NOLDS_TAG is found and the logical address of the data block is consistent with the non-overlap property, the data block is written contiguously with other blocks into the write buffer based on arriving order, and the write command is returned. If the write buffer is full, the SSD enters a wait state until at least one write buffer unit is free. If one or more write buffer units are full with data, a blank erase unit is selected from the BLANK_EU_LIST. A size of list smaller than the BLANK_EU_THRESHOLD triggers recycling of programmed erase units (110). The write buffer unit is written into the selected blank erase unit, and EU_INFO_DB is updated.

Data is invalidated (108), e.g., erased or written over, as follows. When a trim command for a non-overlapped data stream is received from the host, all erase units with valid data and an NOLDS_TAG value are invalidated. Alternative to a trim command, a customized command can be sent which can invalidate erase units with valid data and tag value. The data structure tracking the erase units with valid data and NOLDS_TAG is updated based on the fields of NOLDS_TAG as follows:

- If ON_DEMAND_PRESERVED_POLICY is set, tags are removed from just-invalidated erase units and put into the general pool.
- If ADAPTIVE_PRESERVED_POLICY is set, and if PRESERVED_CAPACITY_EUs is greater than the total number of erase units just invalidated, the excess erase units in the data structure tracking erase units with invalid data and tag are discarded, and the excess erase units are put back into the general pool. The field PRESERVED_CAPACITY_EUs is reset to the total number of EUs just invalidated, and recycling triggered (110).

- If FIXED_RESERVED_POLICY is set, all erase units with tag are kept, and recycling triggered (110).

Programmed erase units are recycled as follows (110). The size of BLANK_EU_LIST is monitored. If the size of BLANK_EU_LIST falls below BLANK_EU_THRESHOLD, then recycling proceeds along one of the following pathways:

- If ON_DEMAND_PRESERVED_POLICY is set, a recycling EU from general pool is selected, erased, and put into the BLANK_EU_LIST.
- If ADAPTIVE_PRESERVED_POLICY is set, a recycling EU with this tag is selected if available. If there is no more recycling EU with this tag, a recycling EU from the general pool is selected. The selected recycling EU is erased and pushed into BLANK_EU_LIST.
- If FIXED_PRESERVED_POLICY is set, a recycling EU with this tag is selected, erased, and pushed into BLANK_EU_LIST.

The recycling logic reverts to monitoring the size of BLANK_EU_LIST.

CONCLUSION

Per the techniques described herein, input data streams, which may comprise sequential streams and/or random streams, are treated as non-overlapped data streams, e.g., streams that have non-overlapping logical addresses. The techniques thereby achieve near-zero over-provisioning with a close-to-unity write amplification and can provide SSD life and throughput that is significantly better than conventional techniques.