Technical Disclosure Commons

Defensive Publications Series

October 2020

Adaptive Data-Coldness Classification For Solid State Drives

Bin Tan

Narges Shahidi

Ricky Benitez

Adaptive Data-Coldness Classification For Solid State Drives

ABSTRACT

In solid-state drives (SSDs), data is not generally updated in place. Rather, data update in SSDs involves garbage collection, in turn facilitated by overprovisioning of storage capacity. Overprovisioning 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 throughput and the life of the SSD. This disclosure describes techniques to adaptively classify data based on its coldness, e.g., recency and frequency of access, and dynamically allocate overprovisioning rates for each coldness category to achieve optimal write amplification and throughput.

KEYWORDS

- Solid-state drive (SSD)
- Write amplification
- Write throughput
- Data coldness
- Data temperature
- Adaptive data classification
- Dynamic overprovisioning

BACKGROUND

Data can be read from solid-state drives (SSDs) one page at a time; however, data can be erased only in blocks, known as erase units (EU) or recycling units, that are larger, e.g., kilobytes or even megabytes in size. Moreover, in contrast to random access memory (RAM), updating or writing a single page or byte of SSD storage involves erasure of the entire corresponding EU followed by re-writing of the entire EU, including both the unchanged and updated bytes of the

EU. Writing to an SSD is performed in units of user data blocks (UDB) which are typically 2-to-16 kilobytes in size. An erase unit comprises several user data blocks.

Due to the constraint in an SSD that writing can only be done in units of UDBs, and due to the constraint that an entire erase unit comprising a large number of UDBs must be erased to write even a single UDB, data in SSDs is not updated in place (unlike RAM). Rather, data update involves garbage collection, facilitated by overprovisioning of storage capacity. Overprovisioning (OP) reduces the 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 effective data throughput as well as the life of the SSD.

Based on frequency of updates, as an example, data is classified (typically) into three coldness (or temperature) categories, e.g., cold, warm, and hot. Cold data is data that is rarely updated; hot data is data that is frequently updated; and warm data has an update rate between that of cold and hot data.

Rate of update	**Classification-A**		**Classification-B**		**Classification-C**	
	Data Class	Over provisioning	Data Class	Over provisioning	Data Class	Over provisioning
Updated more than twice (83%)	—	20%	Hot	10%	Hot	12%
Updated twice (6%)	—	20%	Warm	8%	Warm	6%
Updated once (4%)	—	20%	Warm	8%	Cold	3%
Updated never (2%)	—	20%	Cold	2%	Extreme cold	1%
Write amplification	40 TB		27.6 TB		21.67 TB	
Write throughput	200 MB/s		260 MB/s		294 TB	

Fig. 1: Write amplification and throughput depend on accurate classification of data type and appropriate overprovisioning

Consider a 1 TB SSD with a 1GB/s raw write bandwidth. Every 30 days 10 TB of random data is written into the drive, of which, as illustrated in Fig. 1, 83% is updated more than twice; 6% is updated twice; 4% is updated once; and 2% is never updated. Fig. 1 also illustrates example temperature classifications for the data.

- Under classification A, all data is treated as having the same temperature and granted the same OP, 20%.
- Under classification B, data that is updated more than twice is classified as hot and granted a 10% OP; data that is updated once or twice is classified as warm and granted 8% OP; etc.
- Under classification C, data that is updated more than twice is classified as hot and granted 12% OP; additionally, data that is never updated is granted a distinct temperature class, extreme cold, and granted 1% OP.

Fig. 1 illustrates that data classification can affect write amplification and effective write throughput. In particular, granting excess overprovisioning to relatively cold data classes impacts write amplification and throughput. On the other hand, even relatively cold data classes must have some overprovisioning, however small, to accommodate updates, however occasional. A balance must be struck. Fig. 1 also illustrates that a greater number of data classes can provide improvements with respect to write amplification and throughput, and that allocating OPs that increase with data temperature (as do classifications B and C) reduce the overall OP rate (e.g., as compared to classification A). In real situations, the data-update frequency is unpredictable. Furthermore, existing techniques record the post-data-update frequency, but not the future update frequency. Accurately measuring data temperature and arriving at the appropriate number of data

classes is difficult. On the other hand, pre-configured coldness categories and static overprovisioning rates are seen to be sub-optimal.

DESCRIPTION

This disclosure describes techniques to adaptively classify data coldness and dynamically allocate overprovisioning rates for each category of data to achieve optimal write amplification and throughput. In particular, the disclosure provides the following:

- A technique to accurately measure data coldness to reflect future data-update frequency.
- An adaptive technique to split and merge data coldness categories to dynamically optimize the number of data coldness categories.
- A technique to dynamically find the optimal OP distributions over coldness categories.
- A procedure to integrate the above techniques to optimize data-storage performance without increasing overall overprovisioning of raw capacity..

Coldness timestamps

The coldness timestamp (CDTS) of an erase unit is a derived timestamp based on the timestamp of data-writes into the erase unit and the update frequency of the erase unit. The earliest coldness timestamp boundary and the latest coldness timestamp boundary are similarly defined. For a class of user data blocks, the earliest, latest, mean, and variance of the coldness timestamps are defined using the usual statistical meanings of minimum, maximum, mean, and variance.

Erase unit metadata

As mentioned earlier, an erase unit (EU) is the smallest unit of storage in SSDs that can be erased and readied for writing new data. An EU comprises several user data blocks (UDBs), a

UDB being the smallest unit that can be written to. An EU can be kilobytes or megabytes in size, and updating a single page or byte involves the erasure of the entire corresponding EU followed by re-writing of the entire EU, including both the unchanged and updated bytes of the EU.

Field	Significance
EE_CDTS_EU	Earliest coldness timestamp boundary in the erase unit
MEAN_CDTS_EU	Mean value of the coldness timestamps of the user data blocks in the erase unit
D_UDB_EU	The number of dirty user data blocks in the erase unit

Fig. 2: Metadata for an erase unit

Per the techniques of this disclosure, a metadata structure, as illustrated in Fig. 2 and updated dynamically, is defined and associated with an EU. The number of dirty user data blocks, e.g., blocks that store data that has been invalidated due to the erase-and-rewrite process, are updated as follows. Upon arrival of a UDB write request, the map between the logical block address and the physical storage address is looked up to determine if the logical block is storing the UDB. If si, D_UDB_EU, the number of dirty user data blocks, is incremented by one.

User data frequency-domain spectrum (UDFDS)

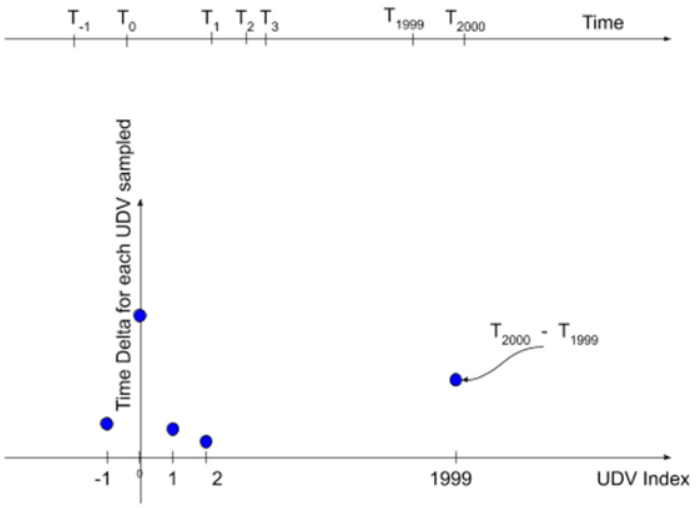

Fig. 3: An example UDFDS. Top figure shows time points at which an incremental UDV (=1 GB) has been written. Bottom figure shows time difference between successive UDV samples.

The UDFDS captures the frequency with which user data is written into a SSD. It is a measure of data temperature. Formally, as illustrated in Fig. 3, UDFDS is defined as a time sequence $\{T_i, 0 \leq i < M\}$ where M is the total amount of data to be captured by the sequence, a number preconfigured, e.g., to 2000, and T_i is the timestamp when a certain amount of user data volume (UDV), e.g., set to 1 GB, has been written into the SSD. The smaller the UDV is, the finer the granularity with which data behavior can be captured. For example, for an SSD with capacity 1TB, if M=2000 and UDV=1GB, the UDFDS will capture the timestamps of data-writes (or arrivals) over 2000×1 GB = 2 TB, which is equivalent to two complete SSD writes. The UDFDS is similar to a rolling time-sequence window; in the above example, at the 2001st timestamp, T_0 is popped out and T_{2001} timestamp is pushed into UDFDS at the location of the former T_0.

User data time-domain spectrum (UDTDS)

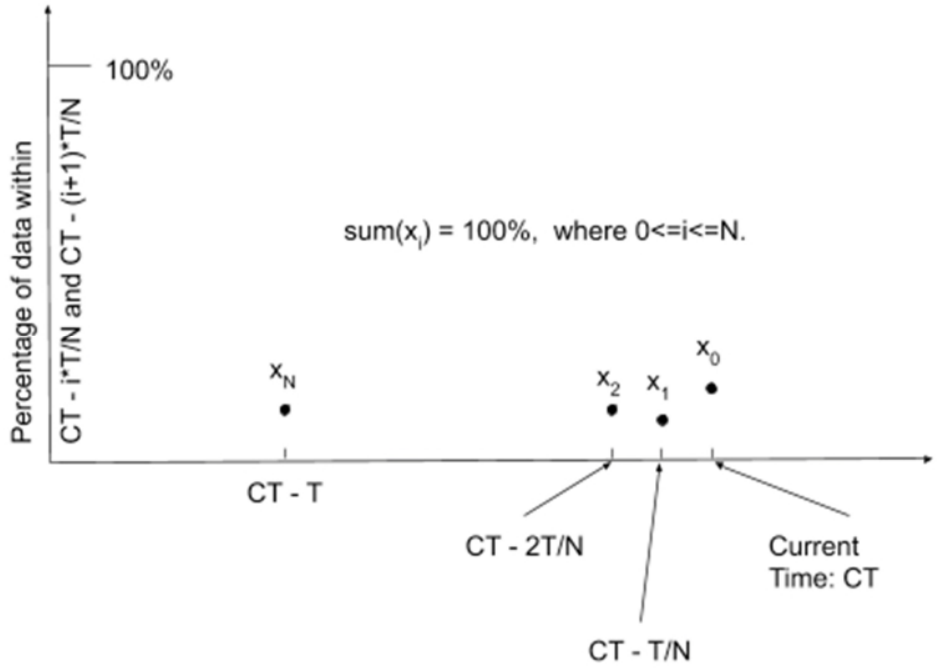

Fig. 4: User data time-domain spectrum

The UDTDS measures the recency of data-writes. Formally, as illustrated in Fig. 4, given the current time CT and a time interval T, it measures the fraction of data volume that has been written in the intervals (CT, CT−T/N), (CT−T/N, CT−2T/N), …, (CT−(N−1)T/N, CT−T), where N is a granularity parameter.

Data coldness class metadata

A data coldness class is a dynamic set of EUs with approximately the same current data temperature, e.g., whose data changes at nearly the same rate.

Field	Significance
EE_CDTS_CLASS	Earliest coldness timestamp boundary in the class
LE_CDTS_CLASS	Latest coldness timestamp boundary in the class
MEAN_CDTS_CLASS	Mean value of coldness timestamps of the UDBs in the class
POINTERS{1, .. K}	Pointers to the erase units within this class
UDTDS	The user data time-domain spectrum for this class
WRITE_BUF_PTR	Pointer to the write buffer (which stores data just before it gets written to an EU) for this class.

Fig. 5: Metadata for the data coldness class

Per the techniques of this disclosure, a metadata structure, as illustrated in Fig. 5 and updated dynamically, is defined and associated with a data coldness class.

User data space-domain spectrum (UDSDS)

Due to the aforementioned out-of-place update characteristics of SSDs, at any given time T, the number of dirty UDBs of each programmed EU (erase unit or recycling unit) can be non-zero.

For a given set of EUs, denote it as EU_Set = {EUs}. Consider a partition of EU_Set with M subsets and denote it as {EU_Subset_i} where $0 \leq i < M$ and

$$EU\_Set = \{EUs\} = \{EU\_Subset\_i,\ 0 \leq i < M\}.$$

For each EU_Subset_i, the UDSDS (user data space-domain spectrum) is defined as

$$UDSDS\_EU\_SUBSET\_i = \{V\_UDB\_EU, \quad \text{for each EU in EU\_SUBSET\_i}\}.$$

$$= \{UDB\_EU - D\_UDB\_EU, \quad \text{for each EU in EU\_SUBSET\_i}\}.$$

Here, V_UDB_EU is the number of valid UDBs in the EU, D_UDB_EU is the number of dirty UDBs in the EU, and UDB_EU is the total number of UDBs in the EU.

For EU_Set with partition {EU_Subset_i, 0≤i<N}, the user data percentage based GUDSDS (global user data space-domain spectrum) is defined as

$$P\_GUDSDS = \{(\text{Valid data blocks in EU\_Subset\_i})/(\text{valid data blocks in EU\_Set}),\ 0 \leq i < N\}.$$

$$= \{Pi\_GUDSDS,\ 0 \leq i < N\}.$$

The user data capacity based GUDSDS is defined as

$$C\_GUDSDS = \{\text{capacity of valid data blocks in EU\_Subset\_i},\ 0 <= i < N\}.$$

$$= \{Ci\_GUDSDS,\ 0 \leq i < N\}.$$

With the above definitions in place, specific procedures for adaptive data-coldness class selection, class splitting, class merging, dynamical overprovisioning, etc. are described.

Data coldness class selection upon the arrival of a user data write request

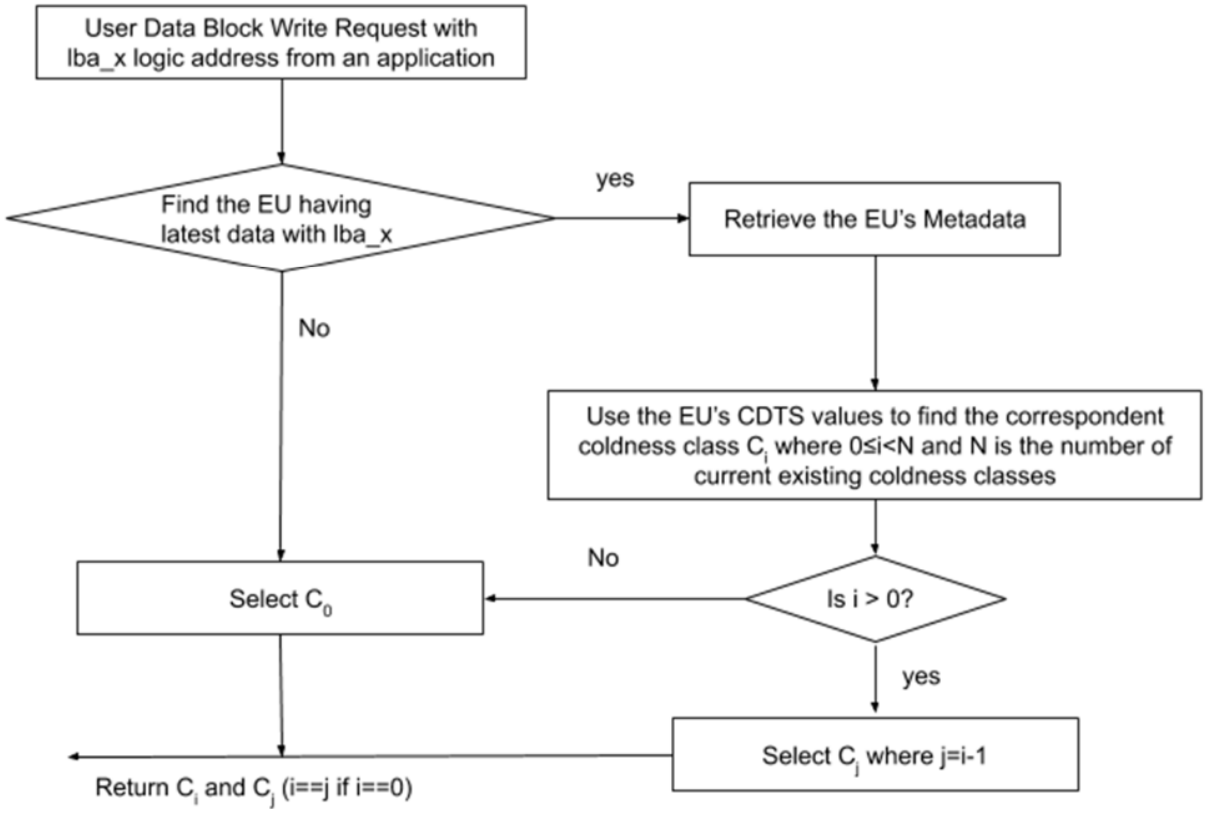

Fig. 6: Data coldness class selection

Fig. 6 illustrates the selection of data coldness class $\{C_0, C_1, \ldots, C_{N-1}\}$ upon the arrival of a user data write request. In this figure, lba_x stands for a logical block address. The classes are in ascending order of data coldness, e.g., C_0 is the hottest class and C_{N-1} the coldest. The newly arriving data is written in the coldness class selected per the procedure illustrated in Fig. 6.

Assigning data coldness timestamp for an arriving UDB write request

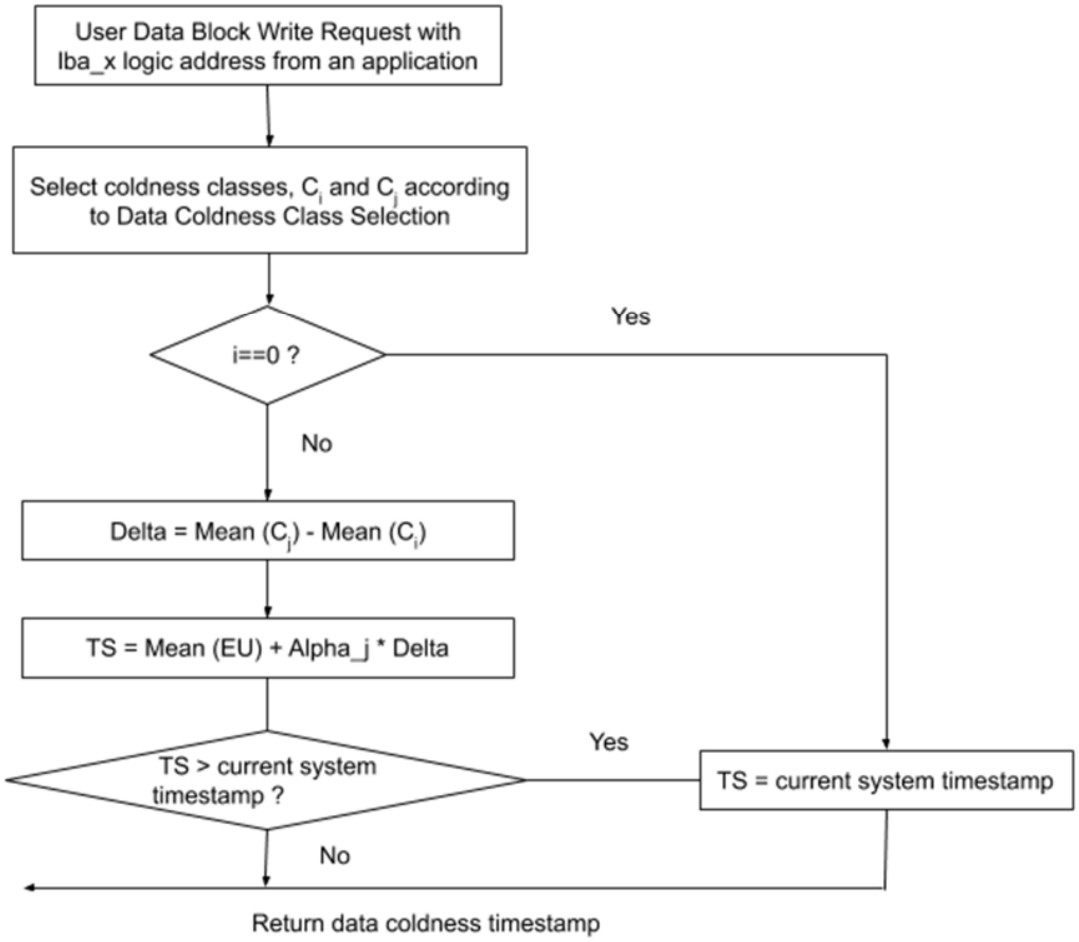

Fig. 7: Assigning data coldness timestamp to a user data block (UDB) write request

Fig. 7 illustrates the assignment of a data coldness timestamp to a UDB write request, per the techniques of this disclosure. In this figure, Alpha_j is an acceleration coefficient, applied at class j. Acceleration coefficients can be used to expand the range of timestamps across coldness classes, e.g., by using larger Alpha_j for smaller j (hotter data classes). C_i and C_j are coldness classes such that $0 \leq j \leq i < N$, N being the current number of coldness classes.

Updating data coldness timestamp metadata for UDB that are recycled

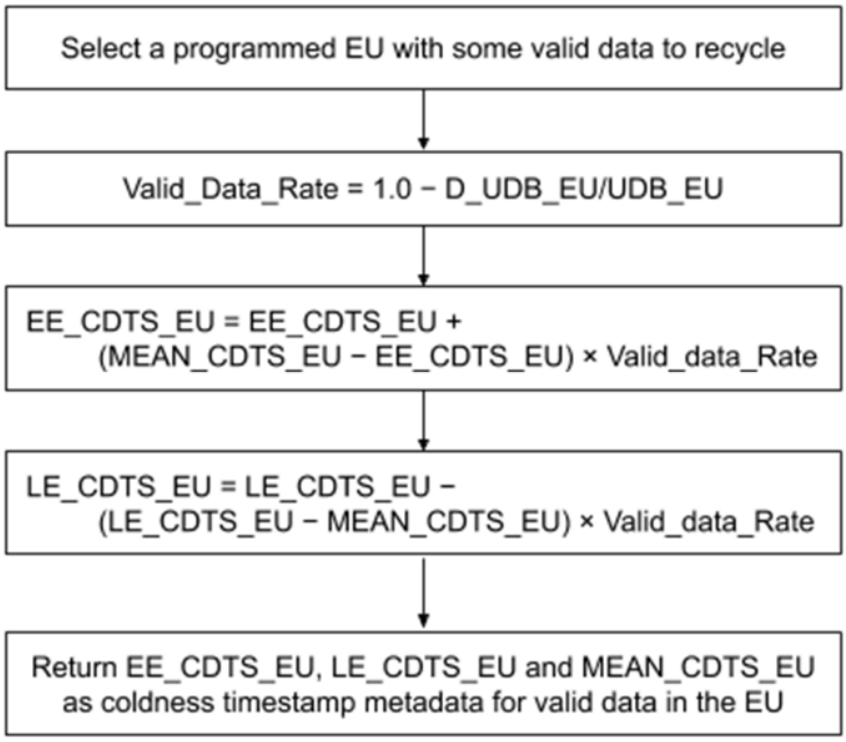

Fig. 8: Updating data coldness timestamp metadata for UDB that are recycled

Fig. 8 illustrates updating data coldness timestamp metadata for UDB that are recycled. Recycling refers to a procedure wherein a number of UDBs, each only fractionally occupied by valid data, are erased and valid data across the UDBs are consolidated into a fewer number of UDBs. The symbol UDB_EU denotes the total number of user data blocks in an erase unit. The remaining symbols are defined in the data structures for the EU metadata and the data coldness metadata.

Splitting a coldness class

A coldness class is split if its cardinality (number of constituent erase units) becomes too large (as a fraction of the total number of erase units) and/or shows a bimodal distribution that can be split using a threshold test.

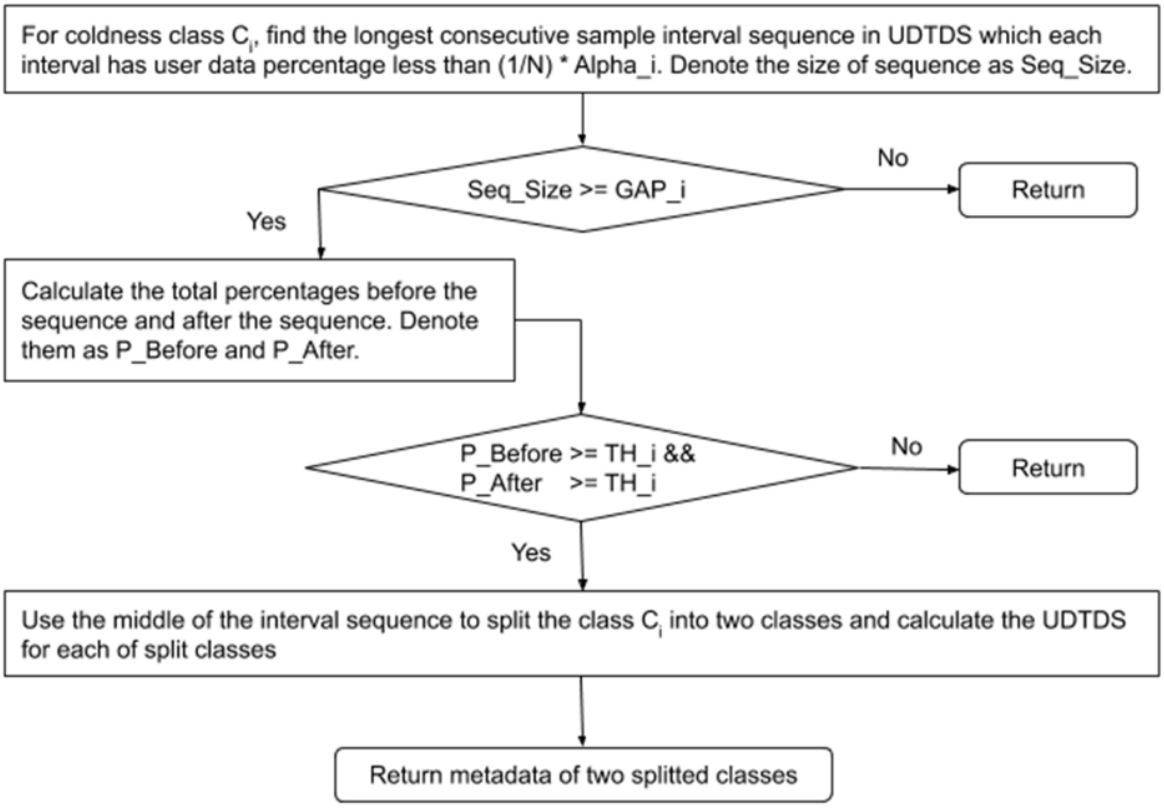

Fig. 9: Splitting a coldness class

Fig. 9 illustrates the splitting of a coldness class C_i. UDTDS is user data time-domain spectrum, as explained earlier. GAP_i represents the timestamp gap between the ith (C_i) and (i+1)st (C_{i+1}) coldness classes. The TH_i are threshold parameters.

Merging of two coldness classes

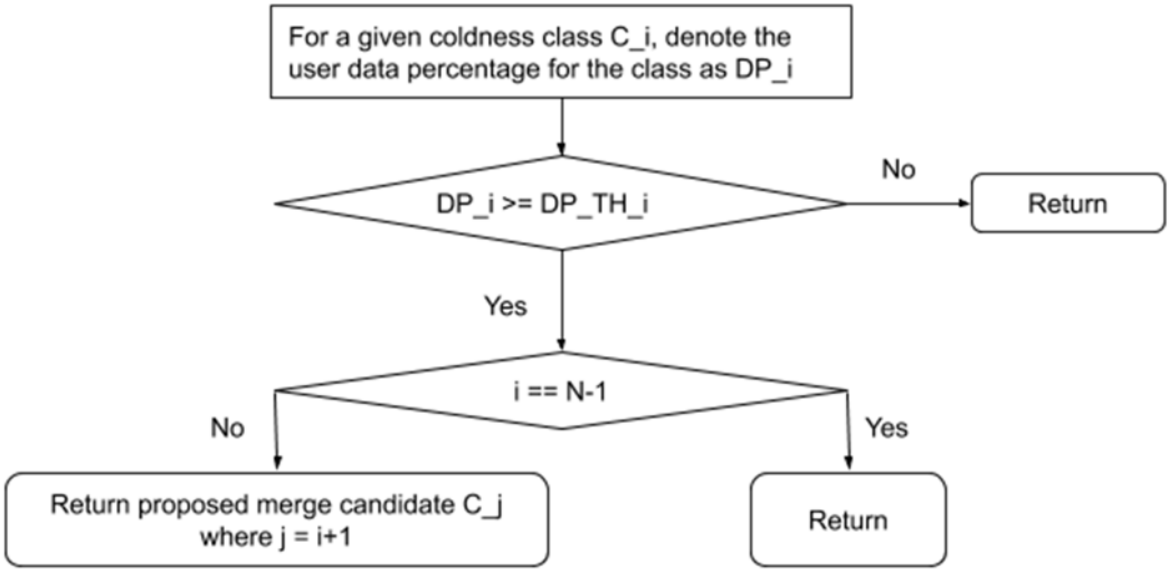

Fig. 10: Merging of two coldness classes

Fig. 10 illustrates the merging of two classes, which can happen when either or both have a relatively small fraction of the total number of erase units and are close to each other in data coldness. DP_TH_i is a threshold parameter that acts on the size of a coldness class (expressed as a percentage) that triggers merging.

Dynamic optimization of overprovisioning

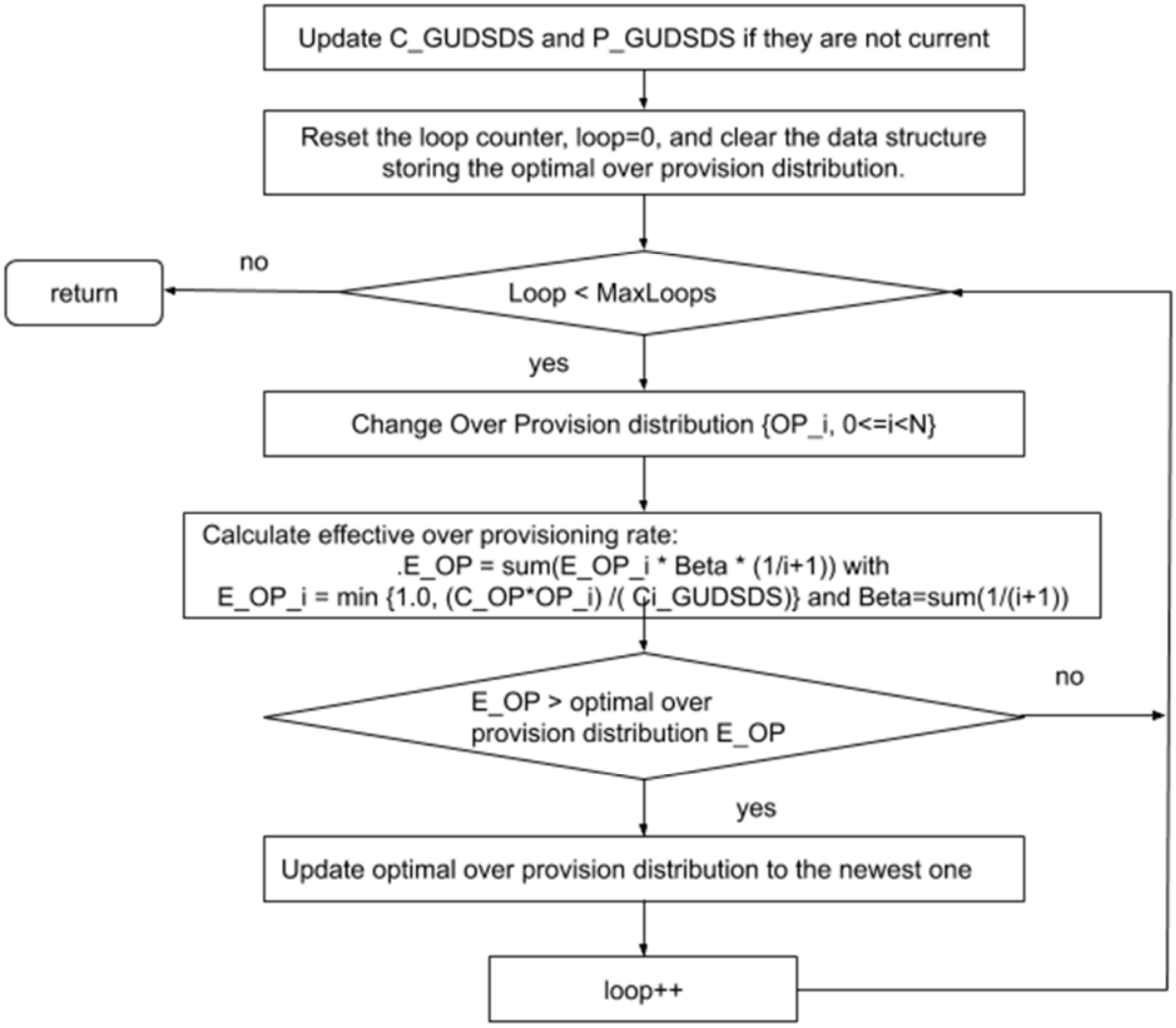

Fig. 11: Dynamic optimization of overprovisioning

Having dynamically defined the coldness classes via class selection, splitting, merging, etc. (as described above), each coldness class is assigned an adaptive overprovisioning, as illustrated in Fig. 11. In this figure, C_OP is the pre-configured overprovisioning for a user data capacity. OP_i is the fraction of C_OP distribution in the ith coldness class, e.g., the sum of OP_i over i is unity. N is the current total coldness classes, and $0 \leq i < N$.

Adaptive data coldness classification and dynamic overprovisioning

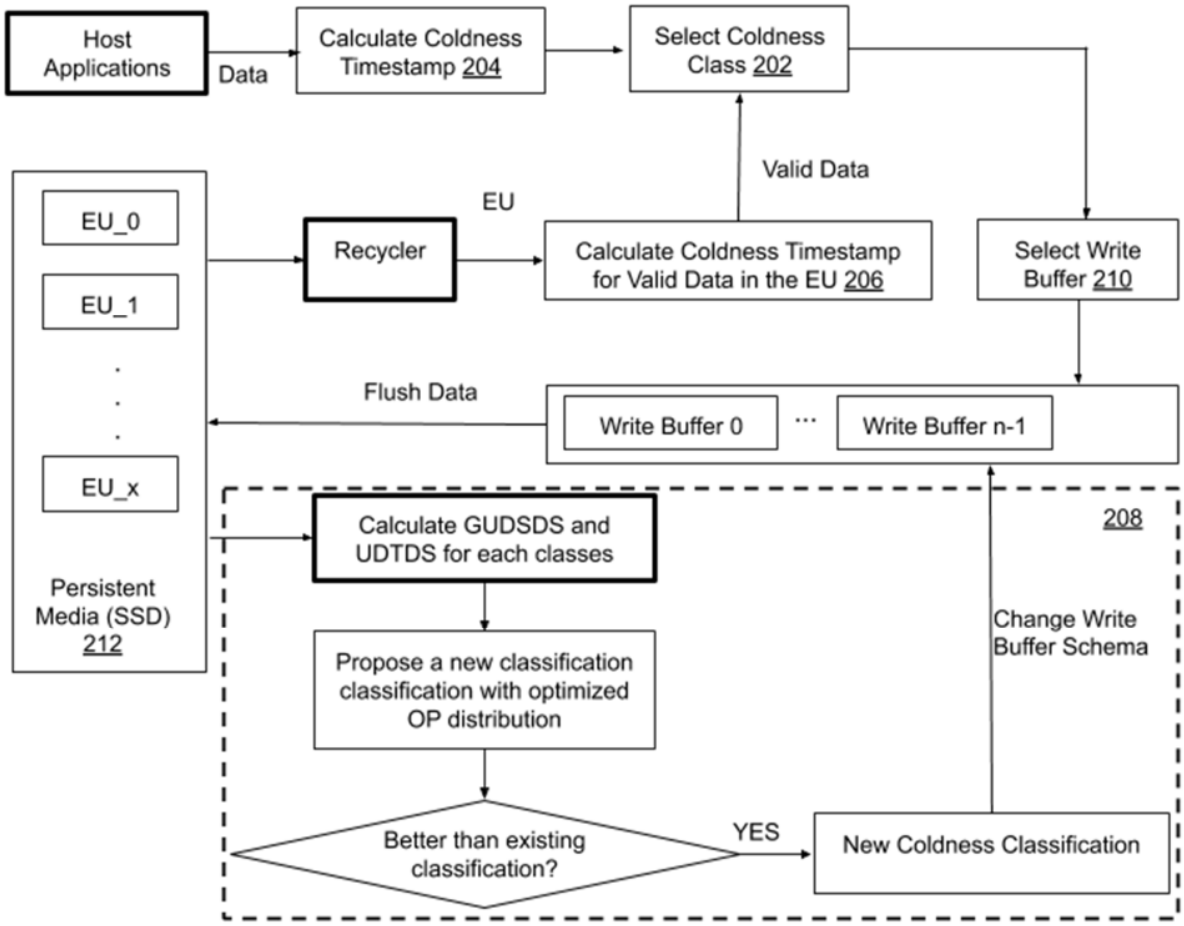

Fig. 12: Adaptive data coldness calculation and dynamic overprovisioning

The techniques of selecting coldness classes (202), assigning data coldness timestamps (204), updating coldness timestamp metadata for recycled UDB (206), dynamically optimizing overprovisioning and coldness classification (208), explained in detail earlier, are unified into a procedure, illustrated in Fig. 12, for adaptively calculating data coldness, selecting the write buffer (210) corresponding to the selected coldness class, and writing (via the selected write buffer) to the corresponding erase unit (EU_0, EU_1, …, EU_x) within the SSD (212). In this figure, boxes with thick borders represent processes that can run in independent threads.

CONCLUSION

This disclosure describes techniques to adaptively classify data based on its coldness, e.g., recency and frequency of access, and dynamically allocate overprovisioning rates for each coldness category to achieve optimal write amplification and throughput in solid-state drives.

REFERENCES

[1] https://www.samsung.com/us/labs/pdfs/2016-08-fms-multi-stream-v4.pdf accessed on Oct. 16, 2020.