Write Amplification Reduction in Flash-Based SSDs Through Extent-Based Temperature Identification

Mansour Shafaei
Northeastern University

Peter Desnoyers
Northeastern University

Jim Fitzpatrick
SanDisk Corporation

Abstract

We apply an extent-based clustering technique to the problem of identifying "hot" or frequently-written data in an SSD, allowing such data to be segregated for improved cleaning performance. We implement and evaluate this technology in simulation, using a page-mapped FTL with Greedy cleaning and separate hot and cold write frontiers. We compare it with two recently proposed hot data identification algorithms, Multiple Hash Functions and Multiple Bloom Filters, keeping the remainder of the FTL / cleaning algorithm unchanged. In almost all cases write amplification was lower with the extent-based algorithm; although in some cases the improvement was modest, in others it was as much as 20%. These gains are achieved with very small amounts of memory, e.g. roughly 10 KB for the implementation tested, an important factor for SSDs where most DRAM is dedicated to address maps and data buffers.

1 Introduction

The performance of log-structured storage systems such as Flash Translation Layers (FTLs) is often dominated by cleaning costs—additional internal copy operations which serve to compact active data and free log segments or flash erase units for re-use. Early works [13, 14] established that when the write frequency of a workload is skewed, resulting in short-lived *hot* pages and long-lived *cold* pages, the performance of simple cleaning methods (i.e. FIFO, Greedy) suffers due to excessive copying. To improve performance for realistic non-uniform workloads it is necessary to take (estimated) data lifetime into account when selecting segments for cleaning, either via methods such as Cost-Benefit [13] or CAT [3] which consider data age when making cleaning selections, or by segregating hot and cold data into separate pools with different write frontiers [7].

The best performance is achieved by systems using multiple write frontiers [5], requiring a method to track write accesses and categorize data as either *hot* (short expected time to overwrite or TRIM) or *cold* (long expected data lifetime). Practical methods for doing so are memory-constrained, as though even consumer-class SSDs have a fair amount of memory, the LBA space to be tracked is huge (e.g. 256M 4 KB pages for a 1 TB SSD) and other functions such as the address map itself have high memory requirements. Yet accurate identification of hot data is important as well, in order to group data with similar lifetimes in the same erase unit and thereby achieve better cleaning efficiency.

A number of different techniques have been proposed for hot data identification; however we assert that much previous work suffers from the following deficiencies:

- Use of per-LBA tracking. As drives get larger and faster, the number of blocks written grows faster than the number of distinct IOs, resulting in workloads comprised of long (yet unaligned) extents. We assert that interval or extent-based methods may be more effective for efficiently monitoring such workloads.
- Measurement of "accuracy" of hot data operation, rather than performance of the hot data-informed FTL.

In the remainder of this paper we provide more background on prior in this area, justify our choice of extent-based methods with selected trace measurements, and then describe and evaluate ETI, the Extent-based Temperature Identification algorithm.

2 Background and Prior Work

Methods for identifying hot data for cleaning purposes fall into several classes. Implicit algorithms [7] use LRU-like queues where infrequently-updated data is pushed to the bottom; explicit algorithms track per-LBA data access frequency and/or recency; finally the *length heuristic* [8] makes use of the fact that short host writes frequently correspond to shorter-lived data. In this work we focus solely on explicit hot data identification algorithms.

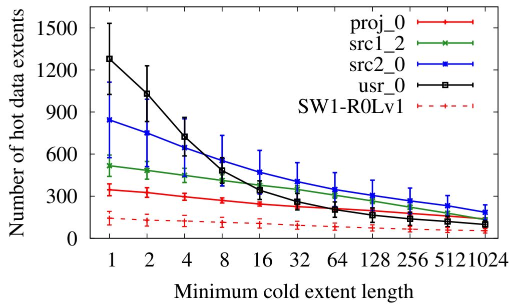


Figure 1: Hot extents after merging. Extents separated by less than k cold sectors (X axis) were merged; remaining number of hot extents is shown. Sector size=4 KB

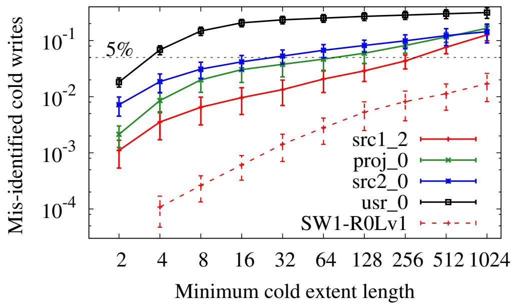


Figure 2: Cold data mis-prediction rate after merging. Value shown is the fraction of all cold writes which were mis-classified due to extent merging.

The simplest of these, the *direct* algorithm [7], maintains a per-page exponentially weighted moving average (EWMA) of writes to that page, requiring one counter per page and periodic *aging* where all counters are multiplicatively decreased, e.g. divided by two. This algorithm is effective, but clearly requires excessive resources. Hsieh's Multiple Hash Functions [7] algorithm approximates the direct algorithm using a counting Bloom filter with saturating counters and periodic aging. Park et al.'s Multiple Bloom Filter [12] algorithm instead uses an array of k Bloom filters $(f_1, f_2, \cdots f_k)$ corresponding to trailing intervals of duration τ, where an LBA is marked in filter f_i if either one or more writes to that LBA occurred in the period i (i.e. $t-(i+1)\tau \cdots t-i\tau$) or at least i writes to that LBA occurred in periods $1, 2, \cdots i-1$.

3 Clustering for Hot Data Identification

Data temperature identification can be considered as a density-based stream clustering problem, where writes are located in a 1-dimensional space (LBAs) and the density of each given address is determined by the number of writes it receives. Although clustering techniques have been widely studied in the artificial intelligence domain, such techniques do not appear have been used for storage workload characterization[1].

3.1 Existing clustering schemes

Density-based clustering is the procedure to find arbitrary shape clusters based on the density of data in a given address space, forming dense clusters separated by sparse areas. These algorithms are typically categorized as either *grid-based* e.g. [10] and [2] or *micro-clustering* e.g. [6] and [9] schemes; in this research we examine micro-clustering techniques due to their (usually) higher accuracy. These techniques use a set of statistical metrics such as center, radius and weight to determine dense areas[2]. The weight is adjusted by a decay function to give more weight to recent inputs, and final clusters are identified as those with weight above a defined threshold. Denstream [1] is an example of such a scheme; it defines two types of micro-clusters (outlier and potential) based on their weight. A new point is added to the micro-cluster with the closest center, as long as it does not expand the cluster enough to violate a maximum defined radius; otherwise an outlier cluster is created. Outliers are promoted to potential micro-clusters when they pass a certain threshold; if aging causes potential micro-clusters to drop below another threshold, they are removed. Intuitively these methods appear appropriate for identifying hot data for cleaning algorithms, as these data accesses are both sparse (in part due to today's large LBA spaces) and highly non-uniform, with significant spatial locality.

3.2 Applicability to storage workloads

To help determine whether clustering techniques are in fact applicable to storage workloads, we perform a series of trace-driven tests, using the well-known MSR block traces [11] (*proj_0*, *src1_2*, *src2_0*, *usr_0*) as well as a trace provided by an industry partner ("*SW1*"). We approximate the limited timescale examined by an online algorithm by splitting each trace into fairly short sequences (1 GB of writes each) and taking measurements on each segment, looking at (a) how many extents are needed to accurately represent the hot data map for a sequence, and (b) how much accuracy is lost by merging extents separated by small amounts of cold data, treating the merged range as entirely hot.

Traces were split into sequences, each containing 1 GB of writes, and statistics (assuming a 4 KB sector size) were measured for each sequence, with hot pages defined as

[1]Chiang's DAC (Dynamic Data Clustering) [4] is a technique for segregation of hot and cold data, not clustered measurement.

[2]Most of the literature refers to 2 or 3 dimensions; we will instead work with 1-dimensional LBA *extents*.

Trace	hot pages	cold pages	hot writes	cold writes	write length
proj_0	23500	39800	181700	62100	10.2
src1_2	21200	41700	172900	67800	8.2
src2_0	7100	29300	157800	55000	1.9
usr_0	6100	36000	163100	63800	2.6
SW1	2100	198000	9600	222000	21.6

Table 1: Mean per 1 GB data written: unique hot (4 KB) pages, cold pages, writes (4 KB) to hot pages, writes to cold pages, mean write length (4 KB pages).

those written to 4 times or more during a 1 GB sequence. Hot pages were grouped into compact extents (i.e. with no intervening cold pages), and a map constructed of these extents; the map was then used to classify data in the same trace sequence. We then merge adjacent extents if they were separated by less than k cold pages, with k ranging from 2 to 1024; when the resulting approximate maps are used to classify accesses within the section, some cold accesses will be mistakenly classified as hot.

In Figures 1 and 2 we see the mean map size (i.e. number of extents) and inaccuracy (fraction of cold writes mis-classified) as k increases from 1 (no map compression) to 1024. The hot data map is small to begin with—a thousand or two, or less—and shrinks down to a few hundred as nearby segments are coalesced. Accuracy is high, and in all but one case the number of map segments can be cut in half with less than 5% misclassification[3]. We can compare these results with the page-level statistics in Table 1. Although tracking independent pages may be done more simply (e.g. via a Bloom filter) than extents, we see that page tracking must track orders of magnitude more items.

4 Extent-Based Temperature Identification

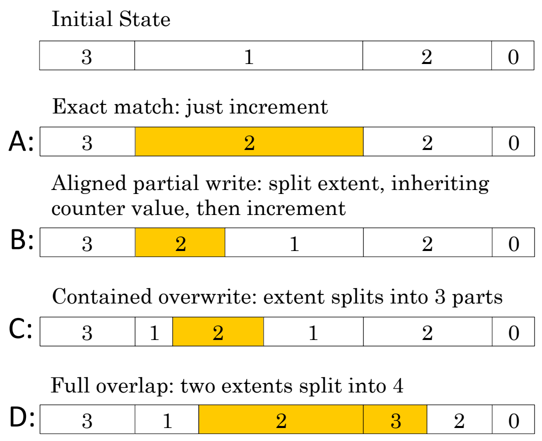
Extent-based Temperature Identification (ETI) is a 1-dimensional density-based clustering scheme which splits the address space into up to N hot and cold extents of different sizes. It begins with a single extent covering the entire address space, then splits extents in response to host writes, and merges extents in a periodic aging process. The algorithm is shown in Figure 1; more intuitively, given a write to an extent of length L starting at LBA A:

- If the write matches an extent exactly (case A in Figure 3): increment the extent counter.
- If the write does not match an extent and the number of extents is less than the maximum N: split extents at A and $A+L$, so that the extent is now matched by 1 or 2 new extents, and increment appropriate counters. (see cases B, C, and D in Figure 3).
- If the write does not match exactly but the number of extents has reached N: for each extent overlapping $[A \cdots A+L)$ by a fraction greater than α: increment the extent counter.

ETI periodically decays counter values and merges extents as follows:

- Divide all counters by two
- Merge all adjacent hot extents with equal counter values
- Merge all adjacent cold extents (i.e. those with counter values less than β)

[3]This does not imply that online algorithms using an extent map can *predict* hot accesses with such accuracy; however we believe it is strong evidence that an approximate extent map can efficiently *represent* the information needed by a realistic online algorithm, with small or negligible loss in accuracy.



Initial State
3
1
2
0
Exact match: just increment
A:
3
2
2
0
Aligned partial write: split extent, inheriting counter value, then increment
B:
3
2
1
2
0
Contained overwrite: extent splits into 3 parts
C:
3
1
2
1
2
0
Full overlap: two extents split into 4
D:
3
1
2
3
2
0

Figure 3: ETI Extent/Counter Update (see description in text)

Algorithm 1: Updating the latest found extent's info while updating extents

```
if Splitting an extent or more then
    for Every split before latest found extent do
        latest_index+=1
    for Splits within the latest found extent do
        latest_upper_bound-=(2nd sub-extent's
        coverage+probable 3rd sub-extent's coverage)
else if Merging extents then
    for Every merged extent
    composed of extents before latest found extent do
        latest_index-=(number_of_its_initial_extents-1)
    for Merged
    extent composed of the latest found extent do
        for Its
        every initial extent before latest found extent do
            latest_index-=1
        for Its
        every initial extent after latest found extent do
            latest_upper_bound-=extent's_coverage
```

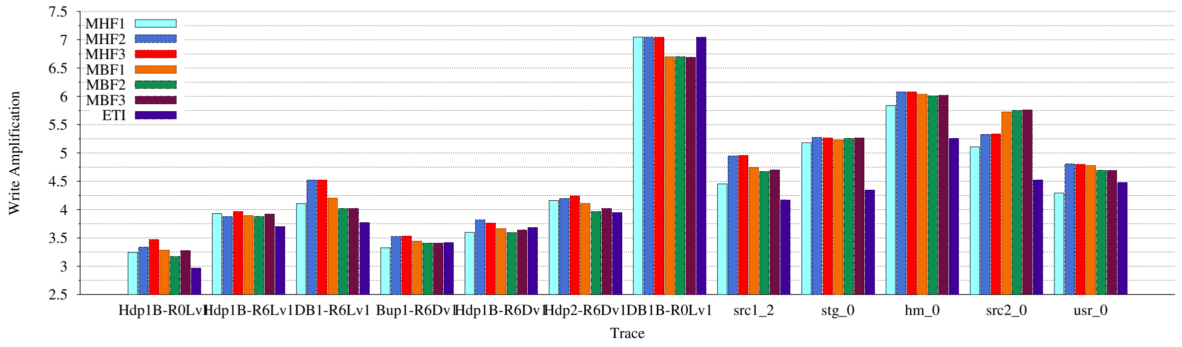


Figure 4: Write amplification seen for different traces using implemented temperature identification schemes

Note that where prior algorithms such as MBF and MHF operate once per block written, ETI is executed once per write operation. With write lengths in the traces examined ranging from 2-5 4 KB pages in the MSR traces, from 2007, to 25-30 pages in the more recent partner-supplied traces, this is a significant reduction in execution frequency. The algorithm is slightly more complex, requiring an ordered search structure for extents, although one of limited length. The lookup operation may be further optimized by taking advantage of spatial locality in write workloads, and beginning each search at the location of the last update.

Table 2: Default parameters for implemented schemes

Scheme	ETI	MBF	MHF
Extent (Table or BF) size	Max of 2048	2048	2048
Counter Width	4-bit	4-bit	4-bit
Decay Period	4096	4096	4096
Hot/Cold Threshold	4	4	4
Splitting Threshold (α)	100%	N/A	N/A
Merging Threshold (β)	4	N/A	N/A
Number of Hash Functions	N/A	2	2
Type of Hash Functions	N/A	CRC	CRC
BF Weight Difference	N/A	0.5	N/A

5 Experimental Results

We measure the efficiency of ETI in terms of write amplification reduction it can provide. Hence, we implement it in a high-level SSD simulator [5] and feed it with 1) selected MSR traces as well as 2) a set of traces provided by an industrial partner. The partner traces represent traffic received by several devices in three different RAID arrays—RAID-0 and two RAID-6 variants—under different workload conditions—two backup applications, a database, a Hadoop HDFS cluster, and an unspecified software application. The simulator is configured with 32K of 128-page blocks, 7% spare factor, and a global greedy cleaning algorithm with hot and cold data segregation. To get stable results, we sequentially write the entire address space once and then uniformly twice before running the target trace. In each experiment the total volume of data written was 2× the volume size; workload traces were truncated or repeated in order to reach the required length.

We compare ETI with Multiple Hash Functions (MHF) and Multiple Bloom Filters (MBF), implemented in the same simulator. Default parameters for each algorithm, including hash functions, are given in Table 2. In some cases MHF and MBF seemed sensitive to the particular hash functions used; we thus provide results for three variants of each.

Figure 4 shows the write amplification for different traces using each implemented scheme. As it can be seen, ETI

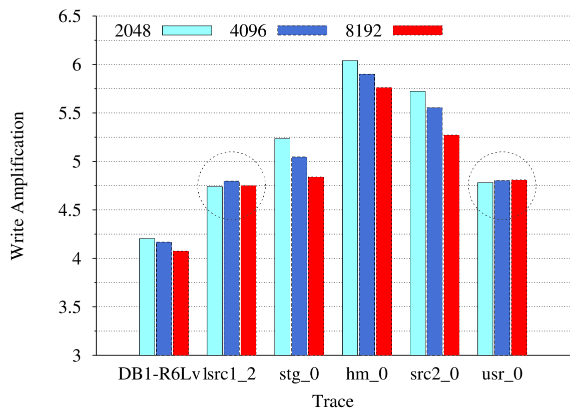


Figure 5: Effects of increasing bloom filter size on write amplification reduction

reduces write amplification of traces such as src2_0 up to 21.3%, performing no worse than MHF in all cases and only slightly worse (4%) than MBF for one trace, Hdp1B-R6Dv1.

Although ETI promises better (or at least equal) performance compared to hashing based schemes, a valid question is how MBF or MHF would behave in case the extra memory used for keeping ETI's metadata was spent on creating larger hash tables or Bloom filters. To answer this question, we increase the size of the MBF Bloom filters by factors of two

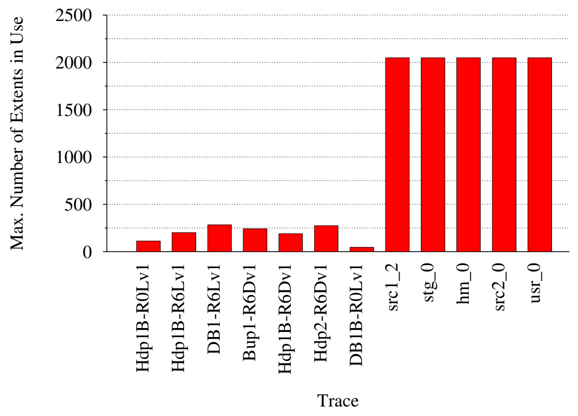


Figure 6: Number of extents being used for different traces with maximum possible extent set at 2048

and four, examining the traces on which ETI achieved the highest gains; results are shown in Figure 5. In four cases this resulted in modest improvements, significantly less than that achieved by ETI, while in two cases write amplification actually increased by negligible amounts.

To further examine the behavior of ETI we measure the average number of extents during each test run, as shown in Figure 6. The shorter partner traces with high mean write lengths result in small extent tables, with a working set of a few hundred extents. These traces show low write amplification and little improvement from ETI relative to MHF and MBF. In contrast the more complex MSR traces make use of the full extent table; these are also the workloads with highest write amplification, and on which ETI achieves more significant performance improvements. From this we infer that in these high write amplification cases ETI is able to identify hot data with increased accuracy, resulting in reduced cleaning costs.

Finally we evaluate the sensitivity of ETI to its extent split ratio and merging parameters, α and β. Setting α to 90% and 80% and changing β from 4 to 3 and 2 show negligible ($<2\%$) changes in experimental results.

6 Conclusion

Much prior work on identifying frequently updated data has focused on the measured ability of proposed algorithms to detect addresses which fit the author's definition of "hot". We assert that since the entire purpose of identifying hot data is to segregate hot and cold data, thereby reducing write amplification, that the appropriate metric by which such an algorithm should be measured is in fact that reduction in write amplification.

Using this criteria we present an extent-based hot data tracking algorithm, ETI, which when combined with a dual frontier, page-mapped Greedy translation layer is shown to improve write amplification by up to 20% over both Multiple Hash Functions and Multiple Bloom Filters.

7 Acknowledgments

This work has been supported by the National Science Foundation under grant CNS-1149232, and by a NetApp Faculty Fellowship.

References

[1] CAO, F., ESTER, M., QIAN, W., AND ZHOU, A. Density-based clustering over an evolving data stream with noise. In *SIAM '06* (2006), pp. 328–339.

[2] CHEN, Y., AND TU, L. Density-based clustering for real-time stream data. In *Proc. SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (2007), pp. 133–142.

[3] CHIANG, M.-L., AND CHANG, R.-C. Cleaning policies in mobile computers using flash memory. *J. Syst. Softw. 48*, 3 (1999), 213–231.

[4] CHIANG, M.-L., LEE, P. C. H., AND CHANG, R.-C. Using data clustering to improve cleaning performance for flash memory. *Software: Practice and Experience 29*, 3 (Mar. 1999), 267–290.

[5] DESNOYERS, P. Analytic modeling of SSD write performance. In *SYSTOR* (2012), pp. 1–10.

[6] FORESTIERO, A., PIZZUTI, C., AND SPEZZANO, G. A single pass algorithm for clustering evolving data streams based on swarm intelligence. *Data Min. Knowl. Discov. 26*, 1 (Jan. 2013), 1–26.

[7] HSIEH, J.-W., KUO, T.-W., AND CHANG, L.-P. Efficient identification of hot data for flash memory storage systems. *Trans. Storage 2*, 1 (Feb. 2006), 22–40.

[8] IM, S., AND SHIN, D. ComboFTL: Improving performance and lifespan of MLC flash memory using SLC flash buffer. *Journal of Systems Architecture 56*, 12 (Dec. 2010), 641–653.

[9] ISAKSSON, C., DUNHAM, M. H., AND HAHSLER, M. Sostream: Self organizing density-based clustering over data stream. In *MLDM'12* (Berlin, Heidelberg, 2012), Springer-Verlag, pp. 264–278.

[10] JIA, C., TAN, C., AND YONG, A. A grid and density-based clustering algorithm for processing data stream. In *WGEC '08* (Sept 2008), pp. 517–521.

[11] NARAYANAN, D., DONNELLY, A., THERESKA, E., ELNIKETY, S., AND ROWSTRON, A. Everest: scaling down peak loads through I/O off-loading. In *OSDI* (San Diego, California, 2008), pp. 15–28.

[12] PARK, D., AND H.C. DU, D. Hot data identification for flash-based storage systems using multiple bloom filters. In *MSST '11* (2011), pp. 1–11.

[13] ROSENBLUM, M., AND OUSTERHOUT, J. K. The design and implementation of a log-structured file system. In *13th ACM symposium on Operating systems principles* (Pacific Grove, California, United States, 1991), ACM, pp. 1–15.

[14] WU, M., AND ZWAENEPOEL, W. eNVy: a non-volatile, main memory storage system. In *ASPLOS '94* (1994), pp. 86–97.