Hot and Cold Data Identification for Flash Memory Using Multiple Bloom Filters

Dongchul Park* and David H.C. Du
Department of Computer Science and Engineering
University of Minnesota–Twin Cities
Email: {park, du}@cs.umn.edu

***Abstract*—In flash memory, hot data identification has a critical impact on its performance (due to a garbage collection) as well as its lifespan (due to a wear leveling). Although it is an issue of paramount importance in flash memory, little investigation has been made. Moreover, all existing schemes focus almost exclusively on a *frequency* viewpoint. However, *recency* also must be considered equally with the frequency for effective hot data identification. In this paper, we propose a novel hot data identification scheme adopting multiple bloom filters to efficiently capture finer-grained recency as well as frequency. In addition to this scheme, we propose a Window-based Direct Address Counting (WDAC) algorithm to approximate an ideal hot data identification as our baseline by using a sliding window concept. Our experimental evaluation demonstrates that our scheme not only consumes 50% less memory and requires less computational overhead up to 58%, but also improves its performance up to 65%.**

I. Introduction

Recent technological breakthroughs in flash memory and dramatic reduction of its price enable flash-based storage systems to hold sway in the storage world. This flash memory retains a distinguished feature: in-place update (i.e., overwriting) is not allowed. To resolve this issue, Flash Translation Layer (FTL) has been developed and deployed to the flash memory [1]. The FTL consists largely of an address allocator, garbage collector and wear leveler all of which are fundamentally based on hot data identification. Therefore, the effective hot data identification has a critical impact on the performance as well as reliability of flash-based storage systems.

We can simply classify the frequently accessed data as hot data. Otherwise, they are regarded as cold data. This definition is still vague and takes only *frequency* (i.e., the number of appearance) into account. However, there is another important factor–*recency* (i.e., closeness to the present)–to identify hot data. In general, many access patterns in workloads exhibit high temporal localities; therefore, recently accessed data are more likely to be accessed again in near future. This is the rationale for including the recency factor in hot data classification. The definition of hot data can be different for each application and also can be applied to a variety of fields such as data caching, B-tree indexing in sensor networks, a garbage collection and a wear leveling in flash memory, and SLC-MLC hybrid SSD. In addition to these, hot data identification has a big potential to be exploited by many other applications. Although this hot data identification is an important issue in flash memory, it has been least investigated. Existing schemes either suffer from large memory space requirements or incur huge computational overhead.

Considering these observations, an efficient hot data identification scheme has to meet the following requirements: **1)** effective capture of recency as well as frequency, **2)** small memory consumption, and **3)** low computational overhead. Based on these requirements, in this paper, we propose a novel hot data identification scheme based on multiple bloom filters (for short, BFs). The key idea of this scheme is that each BF has a different recency weight and coverage so that it can capture finer-grained recency. The main contributions of this paper are as follows:

• *An Efficient Hot Data Identification Scheme:* A BF can provide computational and space efficiency. Our proposed scheme takes advantage of the BF thereby adopting multiple BFs. The multiple BFs enable our proposed scheme to capture finer-grained recency information so that we can achieve more accurate hot data classification. Multiple and smaller BFs empower our scheme to require not only less memory space, but also lower computational overhead.

• *A More Reasonable Baseline Algorithm:* Our proposed approximation algorithm named Window-based Direct Address Counting (WDAC) adopts a sliding window. Whenever a write request is issued, the LBA is stored in the head of the window and the oldest one is evicted like a FIFO (First In First Out) queue. WDAC assigns different recency weights to all LBAs in the window according to the closeness to the present. Thus, when a new request arrives, all LBAs are shifted toward the tail of the window and all their recency values are reevaluated. Consequently, WDAC can catch precise recency as well as frequency.

II. Multiple Boom Filter-based Hot Data Identification

• ***Operation:*** As shown in Figure 1, our scheme adopts a set of V independent bloom filters (for short, BFs) and K independent hash functions. Whenever a write request is issued to the Flash Translation Layer (FTL), the corresponding LBA is hashed by the K hash functions. Then, K hash values set the corresponding K bits in the first BF to 1. When the next write request comes in, our scheme chooses the next BF in a round robin fashion to record its hash values. In addition, it periodically selects one BF in a round robin manner and erases all information in that BF to reflect a decay effect.

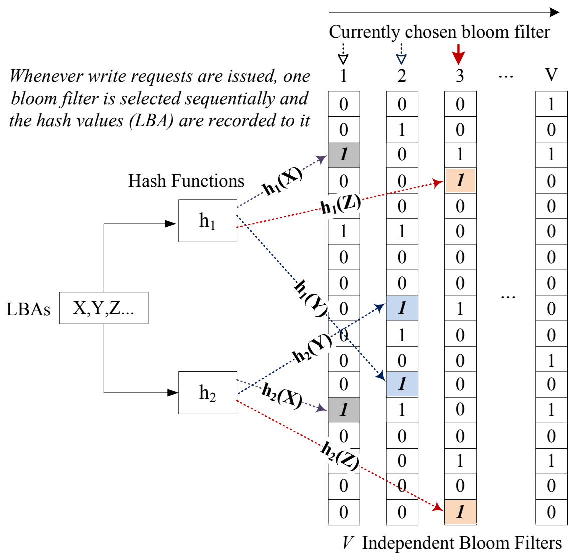
Currently chosen bloom filter
Whenever write requests are issued, one bloom filter is selected sequentially and the hash values (LBA) are recorded to it
Hash Functions
LBAs
X,Y,Z...
h1
h2
h1(X)
h1(Z)
h1(Y)
h2(Y)
h2(X)
h2(Z)
V Independent Bloom Filters

Fig. 1. Our Framework and Its Operations

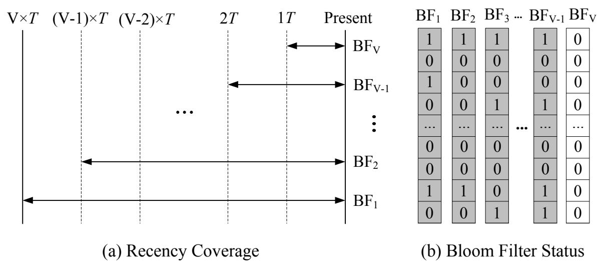

Fig. 2. Recency Coverage for Each Bloom Filter

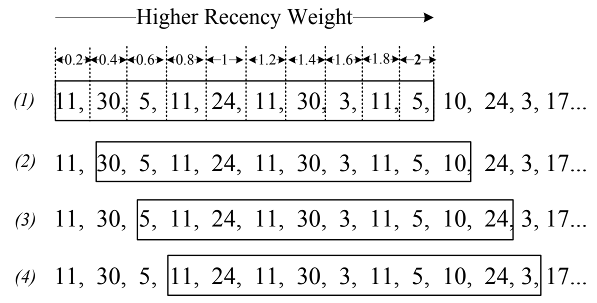

Fig. 3. Working Process of WDAC Algorithm. Here, the window size is 10.

• ***Frequency Capturing:*** Unlike the multihash function framework [2] adopting 4-bit counters, we do not maintain the specific BF counters for each LBA to count the number of appearance. Instead, our scheme investigates multiple BFs to check if each BF has recorded the corresponding LBA. The number of BF retaining the LBAs can show its frequency. Thus, assuming the hash values of an LBA appear in *r* $(0 \le r \le V)$ numbers of the BFs out of V BFs, we can say the corresponding LBA has appeared *r* times before.

• ***Recency Capturing:*** Since our scheme does not maintain LBA counters, we need to devise a different aging mechanism to capture recency information. After the interval *T*, the BF that has not been selected for the longest time interval is selected and all bits in the BF are reset to 0. Similarly, after the interval *T*, the next BF is selected in a right cyclic shift manner and all the bits are reset as time goes on. Figure 2 shows the recency coverage after the interval T as soon as (BF_V) is reset. The reset BF (BF_V) can remember LBA information accessed during only the last one interval *T*. The previously reset BF (BF_{V-1}) can record the LBA information accessed during the last two intervals. Similarly, the BF 1 (BF_1) which will be chosen as a next reset BF after this period can cover the longest interval $V \times T$. This means BF_1 records all LBA information for the last $V \times T$ intervals.

Our proposed scheme assigns a different recency weight to each BF so that recency value is combined with frequency value for hot data decision (called *hot data index*). The reset BF (BF_V) records most recent access information; so highest recency weight has to be assigned to it, whereas lowest recency weight is allotted to the BF that will be chosen as a next reset BF (BF_1). Consequently, even though two different LBAs have appeared once in BF_V and BF_1 respectively, both frequency values are regarded differently. Thus, if the value of hot data index is greater than or equal to a predefined threshold value, we regard the data as hot.

• ***WDAC:*** We propose a more reasonable baseline algorithm to approximate ideal hot data identification named Window-based Direct Address Counting (WDAC). As shown in Figure 3, WDAC maintains a specific size of buffer like a sliding window. In addition, it maintains total hot data index values for each LBA. Within this window, *all* elements have a different recency value according to their access sequences: the closer to the present, the higher recency weight is assigned to the LBA. Whenever a new LBA comes in, all recency values are reassigned to all the LBAs shifted in the window and the last one is evicted from the window. Consequently, our proposed WDAC can properly identify hot data thereby using very fine-grained recency information.

III. Experimental Results

We made experiments in many respects under diverse real traces including real SSD traces. All hot data identification results (i.e., hot ratios) of our scheme display much closer results to those of the baseline scheme than the other one. Furthermore, to make up for the limitation of a hot ratio-based analysis, we also compared the number (or rate) of false identification by making one-to-one comparison of each identification result. This pinpoint evaluation not only enables us to make a comparative analysis of each performance, but also demonstrates that our scheme more precisely identifies hot data. Lastly, we carried out experiments on variable memory size, window size, and the number of a bloom filter in our scheme to explore their impacts.

References

[1] D. Park, B. Debnath, and D. Du, “CFTL: A Convertible Flash Translation Layer Adaptive to Data Access Patterns,” in *SIGMETRICS*, 2010.
[2] J.-W. Hsieh, T.-W. Kuo, and L.-P. Chang, “Efficient Identification of Hot Data for Flash Memory Storage Systems,” *ACM Transactions on Storage*, vol. 2, no. 1, 2006.