DStream: Dynamic Memory Resizing for Multi-Streamed SSDs

Sangwoo Lim†‡, Dongkun Shin†

†Sungkyunkwan University, South Korea
‡ Memory Division, Samsung Electronics Co., South Korea

E-mail: {ssm17.lim, dongkun}@skku.edu

Abstract

To reduce the write amplification factor (WAF) and to improve the performance of flash memory SSDs, the multi-stream SSD (MS-SSD) has been recently proposed, which writes different streams into separate flash memory blocks. The previous studies on MS-SSDs focused on only how to divide user requests into different streams assuming that MS-SSDs support a fixed number of streams.

In this paper, we introduce a dynamic stream number adaptation technique, called DStream, which can improve the performance of MS-SSDs. By adjusting the number of streams adaptively according to the workload with k-means clustering, the remaining memory resources can be used for the map table cache to improve SSD performance. Experiments based on MS-SSD simulator show that DStream reduces cache miss rates by up to 82% and the number of flash memory I/Os by 24%.

Keywords: SSD, Multi-stream, k-means clustering, FTL.

1. Introduction

Solid state drives (SSDs) are rapidly replacing hard disk drives (HDDs) with low latency and high-throughput benefits and are being used in a variety of applications. Because NAND flash memory has special features, a firmware called flash translation layer (FTL) is embedded in SSDs. The FTL allows users to use the legacy block interface without any compatibility problems by logical to physical address translation. Also, FTL selects the victim block through garbage collection (GC) when insufficient free space, moves the valid page of the victim block to the free space. GC is an essential function of FTL. However, GC generates additional writes on the device in addition to the writes requested by the host making the write amplification factor (WAF) to be large. Therefore, GC reduces not only the overall system's performance but also the life span of SSD.

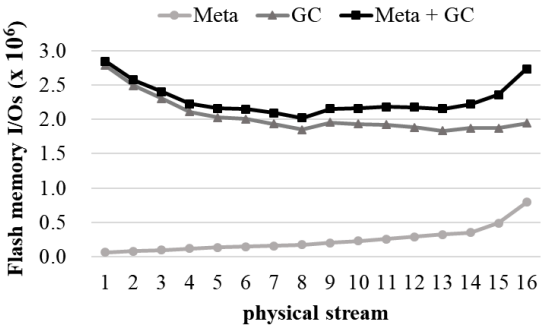

Figure 1. The number of metadata and GC I/Os observed as stream increase in Virt workload.

Many GC-related studies [1, 2] have been introduced to address this. In particular, the MS-SSDs that support stream isolation [3] proposed. Since then, several works announced with stream management techniques [4, 5, 6, 7]. All the previous work has focused on allocating streams to improve performance and reduce GC costs. According to the current SCSI / SAS [13] and NVMe [14] specifications, the stream identifiers (IDs) are in the range 1 to 65535. However, it is hard to support a large number of streams due to the hardware resource limitation in SSD. Hence, the number of streams supported by the device is not large in commercial SSDs [3, 4]. We propose to adjust the number of streams according to workload change and utilize the saved resource to improve I/O performance. We first observed the number of flash memory I/Os at different numbers of available streams with an SSD simulator using DFTL [8].

Figure 1 shows the number of metadata (Meta) and GC I/Os while increasing the number of streams from 1 to 16 with the SSD simulator (See section 4.1). As the number of streams increases, the amount of GC decreases. In contrast, metadata I/O grow. After all, we can see that the total number of flash memory I/Os are increasing. Using a large number of streams within a limited resource reduces cache memory. From the trade-off relationship between the number of streams and cache memory, we can

optimize the efficiency of memory usages explicitly changing the amount of memory for multi-streams.

Full-page mapping FTL include all mapping information that indicates the logical page number (LPN) to the physical page number (PPN). However, DFTL reloads mapping information to the cache when the cache miss occurs during the read or write processing. If the stream increase within a limited resource, more memory is needed to store user data (see section 2.2). This tendency motivates that we can optimize performance by adjusting the cache memory size of DFTL.

The rest of this paper is organized as follows. We describe the background and related work in section 2. Section 3 gives the detail of DStream and explains our technique. Section 4 shows the evaluation results, and we will conclude in Section 5.

2. Background and Related work

2.1. Multi-streamed SSD: In conventional SSDs (i.e., single stream SSD), the order of the writes decides by order of host request. In contrast, MS-SSD [3] can allocate data with similar lifetimes to the same stream IDs when it receives a write request from the host. As a result, when data with similar lifetimes collect in the same block, GC costs (i.e., WAF) are reduced. Also, this behavior can improve SSD performance.

2.2. NAND flash: NAND flash can divide into two types: single-level cell (SLC) and multi-level cell (MLC). SLC stores one bit per cell; however, MLC stores more than two bits. Recently, triple-level cell (TLC) is mostly used which is an MLC flash memory capable of storing three bits in storage systems. Also, quad-level cell (QLC) flash memory that can store four bits is also emerging. As such, the number of bits per cell tends to grow.

Figure 2 (a) shows the reprogram method that is sent in three steps at single page granularity, figure 2 (b) is an example of HSP [10]. In the case of the HSP method, programming complete in one step by sending three pages. The HSP method is faster than the 3-step reprogramming method. Also, it consumes less power. We emulate TLC NAND flash memory using HSP in MS-SSD simulator. Therefore, we collect three pages for user data per stream.

2.3. Related work: Recently, research using SSDs supporting multi-stream functions proposed actively. AutoStream [4] introduces SFR (Sequentiality Frequency Recency), and MQ (Multi-Queue) to automatically assign stream IDs based on data temperature. PC stream [6] uses the program context to allocate the stream IDs, and FStream [7] manage the stream IDs at the file system level. vStream [5] supports more virtual streams than physical streams. To classify virtual streams as physical streams, they use the k-means clustering method. All previous works focus on stream IDs allocation.

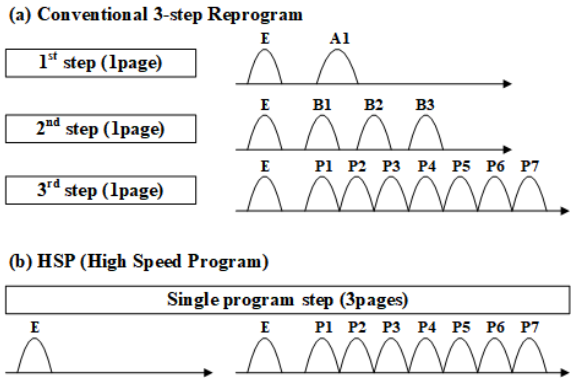

Figure 2. Conventional 3-step Reprogram method and HSP used in MLC NAND flash.

3. DStream

In this section, we introduce DStream which adjusts the number of streams adaptively using k-means clustering. Consequently, we utilize extra memory resources to the cache of DFTL.

3.1. Clustering: For assigning stream IDs, we use k-means clustering [9]. The value of k is the number of physical streams (Pstream). We define logical stream (Lstream) which is N-partitioned stream for LBA [4]. The Lstream can be considered similar to the virtual stream [5]. We classify Lstreams which have similar valid bitmap update count into the same Pstream. Thus, we maintain a bitmap data structure for managing the validity of pages. If the host overwrites the same LPN, the bitmap corresponding to the previous recorded PPN clear, and the valid bitmap update count is added by one. Since only update count is added based on the valid bitmap data structure in FTL, we use less resource consumption than vStream.

The FTL collects valid bitmap update counts when performing a certain amount of write requests, equal to a timestamp. We define one timestamp as the total size of the page in the same block. The timestamp is configurable according to the hardware (i.e., NAND flash and SSD) or workload. As mentioned above, the previously set valid bitmap clear when the host overwrites the LPN for each Lstream. We classify a stream as 'hot' Lstream when there are many update counts, and 'cold' Lstream, if there are a fewer update relatively. We perform k-means clustering based on the update count of Lstream to calculate centroid.

3.2. Adaptive resizing stream: In k-means clustering related research, a method of determining k value has proposed several times. We introduce the technique that bases on adaptive k-means clustering algorithm [11]. This algorithm, which we reference, calculates the distance between each cluster and defines the minimum distance as D_{min}. Also, the nearest cluster is defined as C_{m1}, C_{m2}. The k values will adjust according to D_{min}, C_{m1}, and C_{m2}. These two closest clusters will be merged when the distance of the D_{min} is less than the distance of the element from the nearest cluster. After that, the un-clustered elements are assigned to the empty cluster, creating a new cluster. The algorithm recalculates the D_{min}, C_{m1}, and C_{m2} again. We employ the same method to reduce k-value, but the technique used to increase is different. The DStream technique is as follows:

1. Compute each Element (i.e., Lstream) E_i and the centroid distance of each cluster for every timestamp, and assign it to the closest cluster and update the D_{min}, C_{m1}, C_{m1}, and centroid.
2. If the distance between E_i and the centroid is great than D_{min}, reduce the k value by merging C_{m1} and C_{m2}.
3. If the distance between the E_i and the center is less than D_{min}, increase the k value by setting the mean value of the two hot clusters to the centroid of the empty cluster. It makes hot streams are to classify into more clusters.

In order to reduce the cost of computation, we update cluster at every one timestamp lazily. As we mentioned previously, the number of Pstream equals the number of k. Therefore, we change the memory size in the cache map in DFTL when the number of k increases or decreases. Hence, to manage cache entry, the FTL maintains two lists that manage free pool and cache entry. As the number of k changes, we remove memory from the cache entry or add it to the free pool. We use LRU (Least Recently Used) replacement policy to evict entries.

4. Evaluation

4.1. Experimental Setup: To evaluate DStream, we implement MS-SSD simulator that providing the total capacity of 16GB with a 28% over-provisioning area based on TLC NAND flash memory. The SSD configuration consists of two channels (c = 2), one way (w = 1), two planes (p = 2), two logical pages (l = 2), 8KB page TLC (n=3) with up to 16 physical streams and up to 200 partitioned logical streams. Therefore, the buffer size is 192 KB per stream (c * w * p * l * 8 KB * n). The full-page mapping FTL needs 16MB of memory to cover 16GB capacity. We assume DFTL's cache size as 4MB.

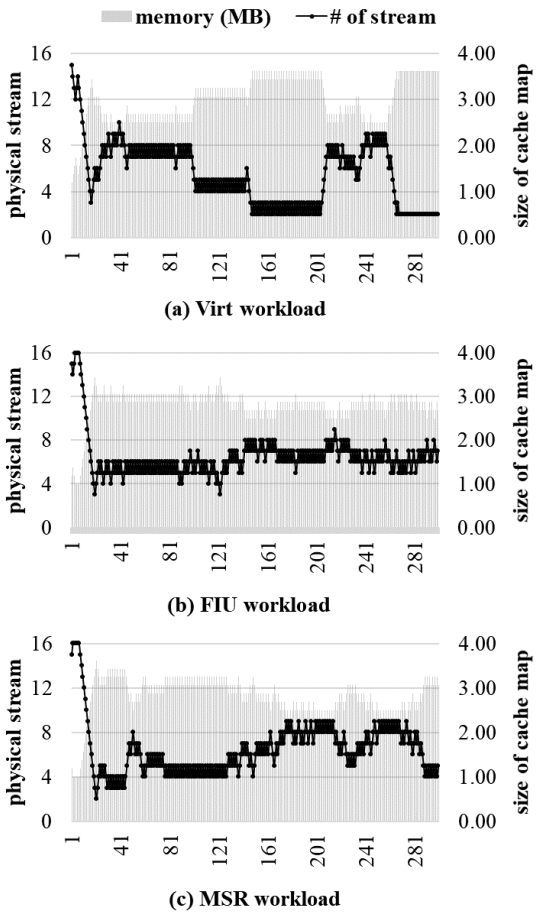

Figure 3. Effect of stream count and cache size.

We observed the impact of DStream on a variety of workload changes during the evaluation. Thus, we run each workload [15] sequentially of MSR (proj_0, rsrch_0, src2_0, stg_0, usr_0) and FIU (online, webmail+online, webresearch, webusers, webmail). Also, we set up the virtual environment running rockDB (append random, overwrite) and Filebench (webserver, oltp) benchmarks at the same time and extract workload by Blktrace [12]. We sequentially write the user address space once before running the target workload to achieve stable results.

4.2. Simulator Result: Figure 3 (a), (b), and (c) shows the change in the number of streams and the memory size while performing various workload with DStream. The X-axis in Figure 3 represents the timestamp. The number of streams shows sharply decrease since it starts with 16 streams at the beginning. We can see that the number of streams and cache size changes adaptively depending on the workload. While FIU and MSR, which have relatively less workload variation than Virt, maintain a stable number of streams. However, Virt experiment results show significant changes with the timestamp increase.

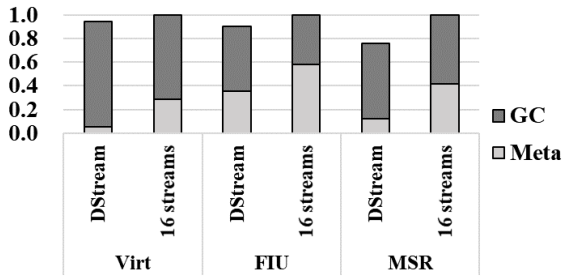

Figure 4. Comparison of the normalized number of flash memory I/Os

Figure 4 illustrates the comparison of flash memory I/Os consisting of GC and metadata I/Os. The results normalized to 16 streams. According to the Virt workload result, metadata I/Os are reduced by 82%, FIU by 39% and MSR by 70% compared to 16 streams. The amount of GC I/Os increased by 25% for Virt, 31% for FIU, and 8% for MSR. Overall, Virt shows a 6% decrease, FIU 10%, and MSR 24% in the number of flash memory I/Os. Table 1 shows the cache miss rate for each workload. We can observe that the cache miss rate reduced by 82%, 36%, and 67% for Virt, FIU, and MSR workload respectively compared to 16 streams.

Table 1: Cache miss rate (%)

	Virt	FIU	MSR
16 streams	0.77	0.05	0.21
DStream	0.13	0.03	0.07

5. Conclusion

In this work, we propose DStream that find the number of streams adaptively in MS-SSDs. The amount of flash memory I/Os are reduced using the resource obtained by changing the stream adaptively. The experiment results show that the number of flash memory I/Os reduce up to 24% and cache miss rates decrease up to 82%, depending on the workloads. As future work, we plan to implement the DStream on a real target SSD.

References

[1] W. Bux and I. Iliadis, “Performance of greedy garbage collection in flash-based solid-state drives,” Performance Evaluation, vol. 67, no. 11, pp. 1172–1186, 2010.

[2] W. Kang, D. Shin, and S. Yoo, “Reinforcement learning-assisted garbage collection to mitigate long-tail latency in ssd,” ACM Transactions on Embedded Computing Systems (TECS), vol. 16, no. 5s, p. 134, 2017.

[3] J.-U. Kang, J. Hyun, H. Maeng, and S. Cho, “The multi-streamed solid-state drive,” In 6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 14), 2014.

[4] J. Yang, R. Pandurangan, C. Choi, and V. Balakrishnan, “AutoStream: automatic stream management for multi-streamed ssds,” In The 10th Annual International Systems and Storage Conference (SYSTOR), p. 3, 2017.

[5] H. Yong, K. Jeong, J. Lee, and J.-S. Kim, “vStream: virtual stream management for multi-streamed ssds,” In 10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18), 2018.

[6] T. Kim, S. S. Hahn, S. Lee, J. Hwang, J. Lee, and J. Kim, “PCstream: automatic stream allocation using program contexts,” In 10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18), 2018.

[7] E. Rho, K. Joshi, S.-U. Shin, N. J. Shetty, J. Hwang, S. Cho, D. D. Lee, and J. Jeong, “FStream: managing flash streams in the file system,” In 16th USENIX Conference on File and Storage Technologies (FAST 18), pp. 257–264, 2018.

[8] A. Gupta, Y. Kim, and B. Urgaonkar, “DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings,” ASPLOS 09, pp. 229–240, 2009.

[9] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, pp. 100–108, 1979.

[10] Y.-J. Choi, K.-D. Suh, Y.-N. Koh, J.-W. Park, K.-J. Lee, Y.-J. Cho, and B.-H. Suh, “A high speed programming scheme for multilevel nand flash memory,” Symp. VLSI Circuits Dig. Tech. Papers, pp. 170–171, 1996.

[11] S. K. Bhatia et al., “Adaptive k-means clustering.” Proceedings of International Florida Artificial Intelligent Research Society Conference (FLAIRS), pp. 695–699, 2004.

[12] Alan D Brunelle. Block i/o layer tracing: blktrace. HP, Gelato-Cupertino, CA, USA, 2006.

[13] T10. SCSI Block Commands-4(SBC-4).

[14] NVM EXPRESS WORKGROUP. NVM Express Revision 1.3.

[15] Storage Networking Industry Association. SNIA Trace repository.