EFM: Elastic Flash Management to Enhance Performance and Lifetime of Flash Memory

Abstract

NAND-based flash memory has become a prevalent storage media in many storage systems for supporting big data applications due to its fast access time and higher performance. There are trade-offs between lifetime and access latencies in NAND-based flash memory. By setting up different incremental step pulse programming (ISPP) values and threshold voltages, these trade-offs can be exploited. Recently, researchers use these trade-offs to improve either the performance or the lifetime of flash memory. However, these existing studies mostly based on simple heuristics.

In this paper, we first develop a comprehensive model of different physical regions according to their ISPP values and threshold voltages. Then a new Elastic Flash Management scheme, called EFM, is proposed to allocate and migrate data in these regions to improve the overall performance and prolong the lifetime of flash memory at the same time. A Long-Term Classifier (LT-Classifier) and a Short-Term Classifier (ST-Classifier) are proposed to decide which region to allocate/migrate the incoming requests based on their data access patterns in the past and read/write latencies of each region. Moreover, as flash memory wearing out after certain usage, the EFM scheme can be adjusted based on the changed read/write latencies to further improve the performance. To prolong the lifetime of flash memory, we propose a reduced effective wearing management by scheduling write-intensive workloads to the region with a reduced threshold voltage. In our experimental results, the EFM scheme can reduce the average read/write latencies by 53.9% - 2.96x when compared to those of the previous studies.

1 Introduction

NAND-based flash memory is playing an important role in today's storage systems from mobile devices to large-scale data-centers. Compared to magnetic recording based drives, it offers the advantages of high performance and light weight. The current trend of the flash memory is to increase its density to achieve high capacity by introducing more levels (holding several bits) in a cell including multi-level cell (MLC), triple-level cell (TLC), and quad-level cell (QLC). However, with the increasing bit-density, the reliability of flash memory is decreased [1–4]. As a result, the lifetime of flash memory is shortened due to a decreased maximum Program-Erase (PE) cycle. On the other hand, researchers use Error Correction Codes (ECC) such as Low-Density Parity-Check codes (LDPC) to make data accessible with higher Raw Bit Error Rates (RBERs) thus prolong the lifetime of flash memory. Those error correction processes need a long latency to decode data, thus resulting in degraded flash access performance.

There are two main trade-offs in NAND-based flash memory. One is between the performance of read and write. For example, different incremental step pulse programming (ISPP) values [5–8] can impact both read and write latencies. With a larger ISPP value, the program process (write operation) has less iterations and thus achieves a lower write latency. However, a larger ISPP value increases the RBERs of programmed memory cells and thus results in a higher read latency. The other trade-off is between the lifetime of NAND-based flash memory and its access latencies. With a low program threshold voltage, the RBERs of memory cells are generally increased and the maximum number of PE cycles of those cells is also increased. Therefore, a lower threshold voltage causes a higher read latency but longer lifetime of flash memory. By setting up different ISPP values and threshold voltages in several physical regions, it creates a more flexible environment for flash memory to manage data since each region has different effective wearing, and read/write latencies. Therefore, these trade-offs can be used to further improve the performance and lifetime of flash memory by allocating and accessing data from different regions.

According to these two trade-offs, several previous studies [8–12] used different ISPP values and different threshold voltages in flash memory to improve either access performance or lifetime of flash memory. For example, Li *et al.* [10] physically split flash memory into three regions. That is, a region with a lower write latency and a higher read latency, a region with medium read/write latencies, and another with

a lower read latency, but a higher write latency. They used a simple heuristic by classifying I/O requests with their most recent consecutive access patterns like WW, WR, RW, and RR (W: write; R: read) to place them into the three regions to improve the performance and lifetime of flash memory. Pan *et al.* [8] explored a device model of flash memory and re-setting ISPP values of flash memory to improve its write performance. Luo *et al.* [12] proposed the WARM scheme to improve flash memory lifetime by separating data that written many times (write-hot data) from none or fewer times (write-cold data) into two separate queues within a monitoring window. The write-hot data does not need to be retained for a longer period such that the lifetime of flash memory can be lengthened. Another class of studies [13–16] generally focuses on the classification of data access patterns. For example, HOTIS [16] was proposed to classify data into hot and cold clusters based on the access frequency and the time interval of adjacent write requests. Therefore, the performance of flash memory can be improved by reducing the overhead of garbage collection.

However, these classification schemes are not suitable for accurately allocating data into different regions of flash memory. In order to obtain higher performance by allocating data into physical regions with different ISPP values and threshold voltages, we have to consider the differences between write/read latencies of each region and data access patterns. Moreover, with memory wearing out, the read latency of flash memory is continuously increased. A static classification based on the initial situation may not be able to correctly allocate data to the region with the current lowest access cost due to the changed latency.

In this paper, the proposed Elastic Flash Management scheme, called EFM, targets to improve the overall performance by allocating data to several physical regions with different write/read latencies. A Long-Term Classifier (LT-Classifier) is proposed to separate incoming requests based on accumulated read/write sizes and the latencies of different physical regions. Then, a Short-Term Classifier (ST-Classifier) calibrates the LT-Classifier based on short-term access patterns. The results of both classifications are considered to decide where to allocate a given data. Consequently, the EFM scheme is capable of better predicting where to allocate/migrate incoming requests. A migration checker is also developed to reduce migration overhead by filtering out unnecessary migrations of data. Moreover, as the NAND-based flash memory wears out, the LT-Classifier of the EFM scheme will be adaptively updated to adapt to the changed read/write latencies. Additionally, a reduced effective wearing management is used to improve the lifetime of the NAND-based flash memory by scheduling write-intensive workloads to the region with a reduced threshold voltage.

The rest of the paper is organized as follows. Section 2 describes the backgrounds of flash memory. The discussion of several design factors is introduced in Section 3. Section 4 discusses the structure and algorithm of the proposed EFM scheme. Section 5 shows the experimental results of EFM compared to those of previous studies. Section 6 reviews some related work. Finally, some conclusions are presented in Section 7.

Figure 1: One example of voltage distribution for 3-bit memory cell.

2 Background of Flash Memory

NAND-based flash memory basically stores N-bits in a cell by injecting electrons into the memory cell. As shown in Figure 1, N-bit data in the NAND-based flash memory are presented by 2^N voltage states. Each voltage state follows a wide Gaussian-like distribution [8, 17] and can be approximately modeled as shown in Eq. (1).

$$P_e(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{-\frac{(x-\mu_e)^2}{2\sigma_e^2}} \quad (1)$$

where μ_e and σ_e are the mean and standard deviations of the erased state threshold voltage respectively. To realize a tight voltage distribution, the Incremental Step Pulse Program (ISPP) is used [5, 18]. Figure 1 indicates an ideal voltage distribution. Practically, the voltage distribution is also affected by other factors such as PE cycles and cell-to-cell interference [6, 19, 20]. To write a data, the ISPP technique is used to inject certain amount of charge into a gate. The voltage state of a cell can be sensed to read data. As the PE cycle of flash memory increases, any two adjacent voltage states will have smaller separation margins or even be overlapped resulting in low reliability. To increase the reliability and lifetime of memory cells, error correction codes such as Bose–Chaudhuri–Hocquenghem (BCH) [21, 22] and LDPC [23, 24] are used. Therefore, according to the factors of ISPP, PE cycle, etc., there are trade-offs in flash memory between read/write latencies and its lifetime. In the following subsections, the details of these trade-offs are discussed.

2.1 Performance Trade-off in Flash Memory

To make a cell to a target voltage state, a flash memory controller will program memory cells by the ISPP value (ΔV_{pp}) iteratively [19, 25]. The relationship between ISPP and program latency can be found in Eq. (2) [26].

$$t_p \propto \gamma \times \frac{1}{\Delta V_{pp}} \quad (2)$$

where t_p is the program latency, γ is a constant value, and ΔV_{pp} is the ISPP value. Thus, we can find that the program latency is proportional to ΔV_{pp}. A larger ΔV_{pp} can reduce the program latency but introduce a narrow margin between two adjacent voltage states, thus, resulting in a higher RBER. For

a read, it is the process to sense the voltage states of memory cells. The latency of a read is based on the error rate of the memory cell and the speed of ECC decoding. During a read, an ECC scheme like BCH [27, 28] or LDPC [23, 29, 30] code is needed to correct the sensed data. With the same correction capacity of a ECC scheme, if a cell has a higher RBER, the ECC scheme takes a longer time to correct the data. The RBER is related to ΔV_{pp}, PE cycles, cell-to-cell interference, etc. [6, 8, 31], and the error model [32] is shown in Eq. (3).

$$RBER = \sum_{k} \left(\int_{-\infty}^{V_p^{(k)}} p(x)^{(k)} dx + \int_{V_p^{(k+1)}}^{+\infty} p(x)^{(k)} dx \right) \quad (3)$$

where $p^{(k)}$ is the voltage density distribution of k^{th} state [**?**] with a random telegraph noise (RTN) [31, 33, 34] and cell-to-cell interference [19]. As shown in Eq. (1), $p^{(k)}$ is a function of PE cycle and ΔV_{pp}. Thus, as the PE cycle of flash memory increases, the RBER will be increased and ECC decoders need to take more iterations to read correct data out [6]. Consequently, the read latency is increased. A higher ΔV_{pp} can shorten the program latency but increase the RBER. Therefore, by following the constraints of the required retention time (one year) [35], NAND-based flash memory needs to take more iterations to decode data and it results in a longer read latency.

2.2 Lifetime of Flash Memory

During a program process (write), flash cells are charged and the charges are evicted from the gates during an erase process (i.e., the PE cycle increases by one). Consequently, the oxide layers of flash gates will be gradually damaged by injecting and evicting charges. With the accumulated damage of gates, data in the cells become vulnerable and the reliability of the cells becomes lower and lower. Finally, data cannot be read out and the flash memory reaches its maximum number of PE cycles and its lifetime ends. According to previous studies [6, 11, 36], the maximum number of PE cycles of flash memory is proportional to the maximum threshold voltage (V_p). The effective wearing w_e [10] is shown in Eq. (4).

$$w_e = C^k \times V_p^k \quad (4)$$

We can find that by reducing the threshold voltage V_p the lifetime of flash memory is increased. However, if we keep $\frac{V_p}{\Delta V_{pp}}$ as a constant value, that is, the write latency keeps the same. Then, the margin between two adjacent voltage states becomes narrower as shown in Figure 2. Finally, the read latency is increased due to the increased RBERs. Thus, there is a trade-off between read performance and the lifetime of flash memory.

3 Design Factors

In this section, we introduce the potential performance and lifetime improvement of flash memory by using several design factors. Moreover, a dynamic scheme is introduced with changed latencies of flash memory due to wearing-out.

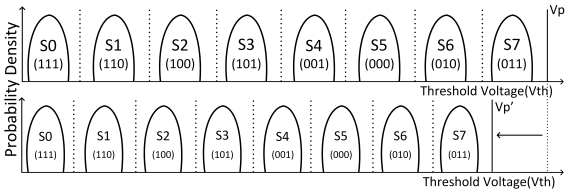


Figure 2: Reduced maximum threshold voltage.

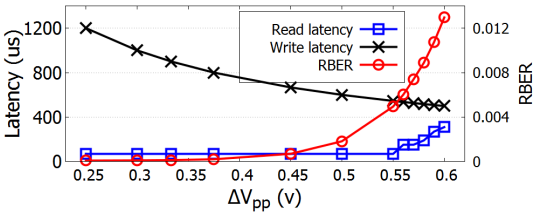


Figure 3: Simulated latencies and RBER varying ΔV_{pp} at the PE cycle of 3000.

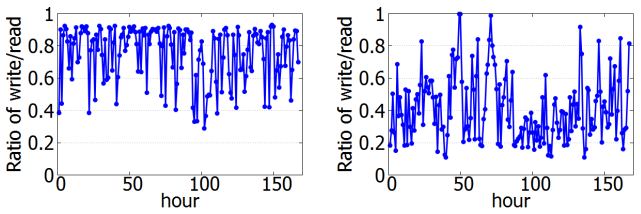
Table 1: An example of different read and write latency pairs

	ΔV_{pp1}	ΔV_{pp2}	ΔV_{pp3}
Read (us)	150	110	70
Write (us)	450	600	800

3.1 Effect of Read and Write Latencies

One important trade-off in NAND flash memory is the latencies between read and write [6, 8, 10, 31, 37]. For example, by increasing ΔV_{pp}, the write latency (program latency) is proportionally decreased while the RBERs of flash cells are increased resulting in longer read latency as seen in Figure 3. To improve the flash memory performance by using this trade-off, some existing studies [10, 38] categorized requests into different categories to improve either read or write performance. The basic idea of these studies is to categorize requests into read-hot or write-hot according to their access patterns (e.g., a pattern of a write following by another write is regarded as write only [10]) or recency [38] (e.g., the most recently updated requests are regarded as write-hot). Then, they simply stored write-intensive requests or write-hot data to a region with a low-write latency and migrated read-intensive (or read-hot) data to another region with a low-read latency. A request between hot and cold is allocated to a region with medium access latencies if a third region exists.

However, the existing studies do not consider the precise latency difference between read and write. By considering different read and write latencies, the request allocation for shortening latency may be different. For example, there is one group of latency combinations based on different ΔV_{pp} as seen in Table 1. Assume a sequence of requests for a logical page is W,R,W,R,W...(W: write; R: read). By using the method in [10], the access pattern is categorized as an interleaved access. Thus, no matter what are the read and write latencies, the scheme [10] always allocates requests in the region with ΔV_{pp2}. However, if considering the ratio of read and write latencies, based on Table 1 we can conclude that ΔV_{pp1} produces the shortest latency.



(a) hm_0 (b) prn_1

(c) proj_0 (d) ts_0

Figure 4: Ratios of read and write in one continuous logic space (100MB) for four sample traces.

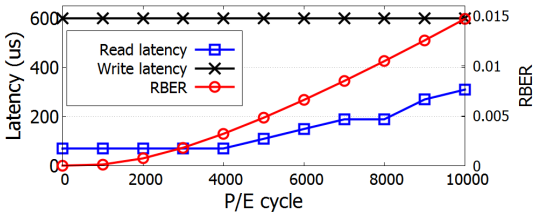


Figure 5: Simulated latencies and RBER changes as increasing PE cycles with the same ΔV_{pp}.

As the ratio of reads and writes in a real workload may change over time as shown in Figure 4, the ratio between accumulated sizes of reads and writes can also be changed dramatically in different workloads. Even in the same workload, the ratio goes up and down at different phases. Therefore, the allocation of data to different regions should be dynamically changed according to their current access patterns. Moreover, the overall performance is also related to the current latencies of read and write. Therefore, the relationship between read and write latencies and data access patterns should be considered together to gain better performance while deciding which region a data to be allocated. Section 4 describes the proposed method to address this issue.

3.2 Effect of Flash Wear-out

As the flash memory wearing out following Eq. (2) and Eq. (3), the read latency is increased as the PE cycle of NAND-based flash memory increases. The program latency is proportional to the ISPP [9,39] since the program procedure will stop when the number of program pulses reaches its limit. To reach the same threshold voltage level under the same program speed (ΔV_{pp}) and the same maximum threshold voltage (V_p), the program latency can be the same, however, this results in a higher RBER. Thus, the read latency is increased due to more iterations of ECC decoder soft-decision are needed [26].

As discussed previously, the read and write latencies and current access patterns need to be considered for a better performance. In Figure 5, with flash memory continuously wearing out, the read latency keeps low in the early stage, and then it starts to increase from the middle stage of the lifetime of flash memory. With the change of the read latency, a static allocation scheme may not deliver good performance at all times. Therefore, a data classification scheme needs to be adjusted based on the current wearing-out stage of NAND flash memory. In Section 4 we introduce a dynamically changed classifier to adapt to the wear-out process of flash memory.

3.3 Lifetime Improvement

The maximum number of PE cycles (flash memory lifetime) depends on the maximum threshold voltage. As indicated in Eq. (4), by decreasing the maximum threshold voltage, the maximum number of PE cycles is increased. Therefore, the lifetime of flash memory is lengthened. However, reducing the maximum threshold voltage potentially has an effect on access latency. As discussed in Section 2.2, if keeping $\frac{V_p}{\Delta V_{pp}}$ as a constant value, the write latency is not changed but the narrow voltage distribution increases the possibility of overlapping two contiguous voltage states. As a result, the RBER is increased and thus the read latency is increased. This clearly indicates a trade-off between read latency and the lifetime of flash memory.

To improve the lifetime of flash memory, the most relevant existing study [10] simply applied a reduced threshold voltage to the middle region (e.g., ΔV_{pp2} in Table 1). Since the data stored in the middle region may read many times, the increased read latency will cause a degradation of the overall performance. In addition, there may be fewer writes to the data allocated to this region based on the decisions of a classifier. Thus, it may also result in less lifetime improvement of flash memory. To minimize the degradation of read performance and to efficiently prolong the lifetime of flash memory, we can intentionally assign data with more writes and less reads to the region with a reduced threshold voltage V_p. By doing so, there are two benefits: 1) less reads accessing this reduced V_p region can mitigate the induced read performance degradation; 2) more writes in the region with a reduced threshold voltage can achieve a longer lifetime for the same workloads. Moreover, the classification for write-intensive workloads is already integrated in the allocation scheme without inducing any additional overhead. In Section 4, we combine these two factors and efficiently improve the performance and lifetime of NAND-based flash memory.

4 Design and Implementation

In this section, we introduce the overall design and components of the proposed EFM scheme. The flash memory can contain N physical regions with different read/write latencies

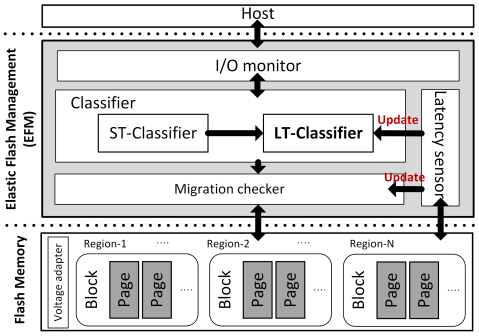


Figure 6: The overall architecture of the EFM scheme.

by using different ISPP values and threshold voltages. In this paper, we simplify the discussion by taking N=3, the same as in the most relevant work [10]. For other N values, the algorithm will follow a similar process. A further discussion of different N values can be found in Section 5.5.

4.1 Overall Architecture

In this paper, the access latencies of three physical regions and workload access patterns are used to classify data and to decide which region the data are stored. Moreover, an adaptive scheme is proposed to adapt to changed read/write latencies due to flash memory wearing. The overall architecture of the EFM scheme is shown in Figure 6. There are four basic components in the EFM scheme. An I/O monitor collects I/O information which will be used for building a classifier. The classifier consists of two different types of classifiers. One is called Long-Term Classifier (**LT-Classifier**) as a major classifier to focus on the long-term data access patterns. The other classifier called Short-Term Classifier (**ST-Classifier**) assists and calibrates the LT-Classifier to obtain a more accurate classification. A migration checker is responsible for filtering out unnecessary migration to reduce the migration overhead. A latency sensor is to check the flash memory wearing and senses the changed latencies of read and write. The observed latencies will be used to adjust the LT-Classifier to make it wearing-out aware.

4.2 EFM Scheduling Algorithm

In this section, we introduce the proposed EFM scheme. As discussed in Section 2 and Section 3, previous schemes do not fully consider the differences between read and write latencies in different physical regions and the changes of them due to wearing. In this paper, both read and write latencies and their request sizes are used as major factors to classify data. According to the access latencies in each region and the accumulated read and write sizes in the recent past, data will be allocated/migrated to a chosen region.

In the design, we assume three different regions. Region-1 is low-cost for writes and high-cost for reads, Region-3 is high-cost for writes and low-cost for reads, and Region-2 is mid-cost for both reads and writes. The read and write latencies for each region are known. First, we investigate a simple scenario to allocate data into two regions (Region-1 (good for writes) and Region-3 (good for reads)). These two physical regions have different read and write latencies due to different physical configurations such as different ΔV_{pp} and V_p. We assume that one (Region-1) has t_p and t_r for write and read latencies, respectively. The other (Region-3) has t'_p and t'_r for write and read latencies, respectively. A data in a workload consisting of x reads and y writes will be allocated into one of these two regions. If we want to achieve that the data allocated to Region-1 has shorter latencies than that in Region-3, it needs to satisfy Eq. (5).

$$Y \times t_p + X \times t_r < Y \times t'_p + X \times t'_r$$
$$=> \begin{cases} \frac{Y}{X} < \frac{t'_r - t_r}{t_p - t'_p}, \text{if } t_p > t'_p \\ \frac{Y}{X} > \frac{t'_r - t_r}{t_p - t'_p}, \text{otherwise} \end{cases} \quad (5)$$

where X and Y are the accumulated read and write sizes of the data in the workload for an observed duration. Therefore, if the data access patterns follow Eq. (5), the data should be assigned to Region-1 to obtain a lower overall execution time. Otherwise, the data should be allocated to Region-3.

According to the above example, with the awareness of

Algorithm 1 EFM Scheduling Algorithm with N=3

```
1:  procedure ADAPTIVE CLASSIFICATION
2:      Record T_starttime
3:      N=3 /*N is the number of physical regions*/
4:      while t < T do
5:          Compute logical block number i of Req_k
6:          if Req_k is Read then
7:              Read_TBL[i] = Read_TBL[i] + Req_k.size
8:              MRO[i] = MRO[i] << 1|1
9:          else
10:             Write_TBL[i] = Write_TBL[i] + Req_k.size
11:             MRO[i] = MRO[i] << 1|0
12:         if |LT-Classifier() - ST-Classifier()| < 2 then
13:             classification = LT-Classifier()
14:         else
15:             classification = 2
16:         if Reg_mapping[i] != 3 and classification = 3 then
17:             migration[i] = migration[i] + 1
18:         if currOp == 0 then /* current operation is write*/
19:             Write Req_k: Region[classification] ← Req_k
20:             Reg_mapping[i] = classification
21:             migration[i] = 0
22:         else
23:             Read Req_k
24:         if migration[i] >= mig_threshold then
25:             Migrate the block i to Region[classification]
26:             Reg_mapping[i] = classification
27:             migration[i] = 0
28:     if t == T then
29:         while for all i do
30:             Read_TBL[i] = Read_TBL[i] * 0.2
31:             Write_TBL[i] = Write_TBL[i] * 0.2
32:             t = 0
33: end
```

Algorithm 2 LT-Classifier()

Input: $Write\_TBL[i]$, $Read\_TBL[i]$
Output: RegionN /* Region number*/

1: **if** $\frac{Write\_TBL[i]}{Read\_TBL[i]} < \frac{t_{r2}-t_{r1}}{t_{p1}-t_{p2}}$ **then**
2: RegionN = 1
3: **else if** $\frac{Write\_TBL[i]}{Read\_TBL[i]} < \frac{t_{r3}-t_{r2}}{t_{p2}-t_{p3}}$ **then**
4: RegionN = 2
5: **else**
6: RegionN = 3
7: return RegionN

Algorithm 3 ST-Classifier()

Input: $MRO[i]$
Output: RegionN /* Region number*/

1: **if** $MRO[i].count(\text{"1"}) >= 5$ **then** /*Read-intensive*/
2: RegionN = 3
3: **else if** $MRO[i].count(\text{"1"}) <= 2$ **then** /*Write-intensive*/
4: RegionN = 1
5: **else**
6: RegionN = 2
7: return RegionN

the read and write latencies and their accumulated sizes, the LT-Classifier follows Eq. (5). The details of the EFM scheme is shown in Algorithm 1. We use a flash memory block as a logic block of I/O monitoring. First, the EFM scheme keeps accumulating the access patterns of the workload and store the read and write sizes of each logical block in **Read_TBL** and **Write_TBL**, respectively. Meanwhile, the Most Recent Operations (**MRO**) on each logical block are stored in the MRO table. Seven bits are used to store the most recent seven requests on each logical block. "0" indicates an operation is write (W) and "1" is read (R) as seen at Lines 6-11 of Algorithm 1. For example, "110" indicates the sequence of requests is "R,R,W". The LT-Classifier in Algorithm 2 follows Eq. (5) to determine which region a request for a logical block is supposed to be allocated. The latencies of the three regions associated with the accumulated read and write sizes are used as parameters to classify access patterns. As shown in Algorithm 3, the ST-Classifier uses the most recent seven operations to identify the short-term access patterns of the workload on each logical block. More than or equal to four writes or four reads out of seven requests are regarded as short-term write-intensive and read-intensive logical block, respectively. Otherwise, the requests for the block will be classified as an interleaved read and write by ST-Classifier.

The LT-Classifier as a major classifier determines the region of allocation (Lines 14-15). The ST-Classifier assists the LT-Classifier when the results of two classifiers have the difference of region values by 2 as seen at Lines 12-13. This means that the classifications of long-term patterns and short-term patterns have a big contradiction with each other and thus the decision is made for the middle region (e.g., Region-2). If a read request for one logical block locates at high-cost read regions (such as Region-1 or Region-2) and the EFM scheme classifies the logic block to the low-cost read region (Region-3), the migration checker will check whether multiple reads happened on this logical block based on the $migration[i]$ (at Lines 24-26). If so, all logical pages in this block will be migrated to Region-3 to achieve a lower read latency in the future. Meanwhile, the region mapping table (Reg_mapping) of the block is also updated to indicate which region the logical block is classified to. Otherwise, the read request is a normal read operation and no extra action is taken. For each period T, $Read\_TBL$ and $Write\_TBL$ will be updated (at Lines 27-30). The coefficient 0.2 indicates the weight of the current period of information. By doing that, parts of workload information at previous periods keep in the two tables and can improve the accuracy of collecting the blocks of the workload.

Moreover, the monitoring function is responsible for periodically (e.g., for each 500 PE cycles) updating the NAND flash memory information such as the read and write latencies of different regions. As indicated in Figure 5, at the early stages of flash memory, the read latency grows up slowly. Therefore, we set a large duration to update the latencies of flash memory to LT-Classifier. As flash memory wearing out, the increase of read latency becomes quicker. A smaller duration of updating latencies can be used to update the changed read latencies in time. To improve the lifetime of flash memory, the reduced threshold voltage will be applied to Region-1 which has the shortest write latency. By doing so, it not only can potentially gain the benefits of low write latency, but also can maximize the lifetime of flash memory by reducing the effective wearing of a large number of writes.

Figure 7 indicates an example of the EFM scheme with three regions. According to Algorithm 1, the three regions are split by three ISPP values (ΔV_{pp1}, ΔV_{pp2}, and ΔV_{pp3}) and two threshold voltages (V_p and $V_p^{'}$). The two blue lines in Figure 7 are drawn by the LT-Classifier according to Eq. (5). Their slopes are based on the access latencies of different regions and request sizes. The red curves are based on the ST-Classifier which calibrates the LT-Classifier from Region-1 or Region-3 to Region-2 when facing a conflict decisions between short-term and long-term classifiers. Moreover, based on the wear-out aware management, the slopes of LT-Classifiers will be adjusted based on the latencies updated by the monitoring function. In Figure 7, one example of a logical block is classified into Region-3 based on its accumulated access patterns of workloads. Therefore, write requests on this logical block will be scheduled to Region-3. For read requests, if a migration of this logical block occurs as indicated in Algorithm 1, all read requests will be read from Region-3. Otherwise, read requests will happen on their original regions.

4.3 Discussion of Physical Region Allocation, Garbage Collection, and Wear-leveling

Three physical regions are initially set up by different physical configurations like different ISPP values and threshold

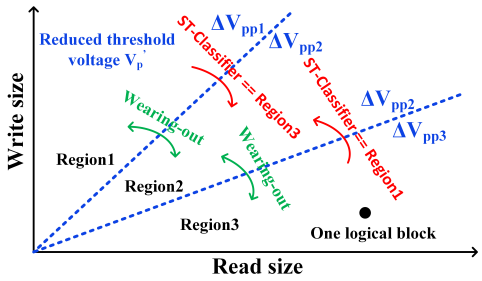


Figure 7: One example of the classification of the EFM scheme with three physical regions (N=3).

voltages. We assume that the physical configurations (e.g., ISPP, V_p and ΔV_{pp}) keep consistent during the whole lifetime of flash memory. When one physical region is full, before releasing free space of the full physical region, an incoming request will be re-scheduled to its neighbor physical region. As a result, this may influence the performance of flash memory when a physical region reaches its full utilization. To avoid it, one potential alternative is that we can estimate the needed capacity of each physical region for applications and set up the required capacity of each region at the beginning. Additionally, some technologies [11, 41–43] make adaptively adjusting capacities of each region possible. Thus, the capacity of regions can be adaptively changed according to the needs of these applications. The dynamically resizing regions can improve the overall performance but may hurt the lifetime of the device and increase tail latency since it potentially needs some amount of data migration. In this paper, we assume each region has an adequate capacity. The problem of capacity allocation for each physical region will be left for future work.

Garbage collection [40, 44–47] is used to release free space by reclaiming invalid pages in flash memory due to different granularities of write and erase operations [48]. A typical greedy garbage collection [44] selects the blocks with the minimum number of valid pages and the largest number of invalid pages to reduce the migration overhead of the garbage collection and maximize the number of additional free space. In the EFM scheme, the greedy garbage collection is applied by default. If there are multiple candidate blocks with the same condition, the block with the least migration latency (due to different latencies in the regions) will be first selected. In doing so, the garbage collection latency can be potentially further reduced.

Wear-leveling [49–52] targets on prolonging the lifetime of flash memory with balancing the PE cycles across all blocks. In previous weal-leveling algorithms [50, 53], cold data (rarely updated data) are stored in hot blocks (i.e., blocks that bear more erases) and hot data (frequently updated data) are stored in cold blocks (i.e., blocks with fewer erase counts). The hot and cold blocks will be swapped based on their PE cycles. Different from the traditional wear-leveling schemes, the wear-leveling scheme in the EFM scheme will also depend on the effectiveness of wearing. When evaluating the PE cycles of flash memory, the PE cycles of the flash memory in the region with the reduced threshold voltage will be multiplied by an effective wearing coefficient (0.8 by default in this paper). Thus, based on the effective wearing PE cycles, the cold and hot data will be swapped. Moreover, since the physical regions store different access patterns resulting in unbalanced PE cycles of flash blocks. Some existing studies [11, 41–43, 54] provide possibilities to adjust threshold voltages and ISPP values in flash memory. Therefore, the EFM scheme can achieve wear-leveling by swapping the configurations of physical regions (e.g., swapping ΔV_{pp1} and ΔV_{pp3}). However, the configuration swapping needs to physically swap the contents of physical regions and thus it may involve a large amount of data migration.

4.4 Overhead Discussion

The overhead of the EFM scheme is mainly from three aspects. One is space overhead from recording block information including $Write\_TBL$, $Read\_TBL$, MRO, and $migration$ tables. Assume that 250GB flash memory is with 4KB page size and each block contains 128 pages. The recording granularity of those tables is one block. The sizes of the tables $Write\_TBL$ and $Read\_TBL$ depend on the maximum accumulated size for each block. If we set the maximum accumulated size for one period 400MB. Then, the storage capacity overhead of all those tables is about 3MB. Compared to the page-level mapping table size (≈512MB) located in the Flash Memory Layer (FTL), the storage capacity overhead of those tables is really small and acceptable.

The second overhead is from classifying and updating tables as indicated in Algorithm 1. The main operations involved in the classification are addition, multiplication and division. We investigated the overhead in a system with Intel(R) Xeon(R) CPU E5-2620 v3 2.4GHz processors. The result indicates that the classification for each operation only needs about 19ns. For each period T, the time for table updates needs about 85ns. Compared to the read/write latencies, the computing overhead is much smaller. Moreover, modern SSDs [55, 56] contain more computing resources. Therefore, the proposed EFM only consumes a small amount of computing resource and running the proposed scheme in SSDs will not be a practical issue.

The third overhead is from hardware implementation for multiple threshold voltages, ISPP values, and latency monitoring. This overhead has been already investigated and verified by previous studies [8, 9, 29, 43, 54, 57–59]. So, the hardware overhead is tolerable.

5 Experimental Results

5.1 Environmental Setup

We evaluated different algorithms based on the SSDsim simulator [60] with the extension of different read and write

Table 2: Configurations of traces

	Number of IOs (Millions)		Total request size (GB)	
	Write	Read	Write	Read
mds_1	0.12	1.52	1.54	87.17
web2	5.14	0.04	0.78	262.82
usr_1	3.86	41.43	56.13	2079.23
usr_2	1.99	8.58	26.47	415.28
proj_1	2.50	21.14	25.58	750.36
ts_0	1.49	0.32	11.34	4.13
hm_0	2.58	1.42	20.48	9.96
prxy_0	12.14	0.38	53.80	3.05
syn1	3.93	1.31	15.00	5.00
syn2	4.10	1.02	16.00	4.00
OLTP1	4.10	1.24	14.57	2.65
OLTP2	0.65	3.05	1.82	6.62

Table 3: Latencies of read and write with different reduced effective wearing for different PE cycles

		Region-1	Region-2	Region-3
Regular	Read (us)	270	170	70
	Write (us)	450	600	800
Effective wearing (reduced 0.8)	Read (us)	310	210	-
	Write (us)	450	600	-

latencies of 3 physical regions. The latencies are obtained from the device model in [8, 10, 30, 32, 38]. The flash memory in the simulation has 256GB capacity with a page size of 4KB. Each block contains 128 pages. The access latencies of flash memory in different regions are indicated in Table 3. The traces used in the experiments are the MSR Cambridge traces [61], two synthetic traces (syn1 and syn2), and two OLTP application traces [62] as shown in Table 2. Three existing schemes, i.e., WARM [12], HOTIS [16], and TOS18 [10] are used as baselines to make comparisons with the proposed EFM scheme.

5.2 Overall Performance

First, we make a comparison between four schemes with all 12 traces. The latencies of three regions used the regular configuration are shown in Table 3. The overall performance comparison is shown in Figure 8. We can find that the EFM scheme obtains the lowest average latencies compared to the other three baselines. On average, EFM delivers 53.9%, 87%, and 2.96x latency reductions compared to HOTIS, WARM, and TOS18, respectively. There are two major reasons that the EFM scheme outperforms the others. One is that the previous schemes misclassify the dat access patterns so that data are allocated into the regions which have longer read or write latencies. The other reason is that the baseline schemes face a large overhead of migrations. For these schemes, the performance gain of migrating data to a region with a lower write/read latency is less than the migration overhead itself, thus resulting in lower overall performance.

Moreover, two typical examples of breakdown analysis are demonstrated in Figure 9. We did not present the results of other traces since they have similar conclusions. "Optimal R+W" indicates that read and write operations access the

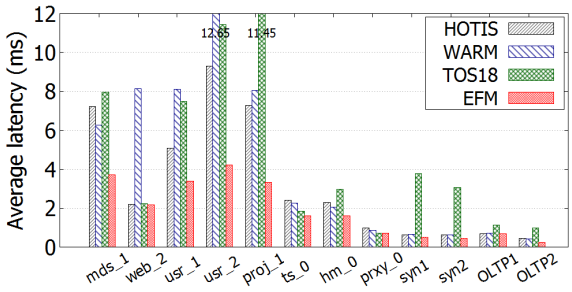
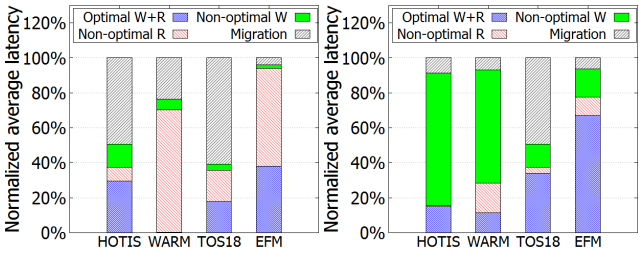


Figure 8: Overall performance comparison between four schemes.



(a) usr_1 (b) hm_0

Figure 9: Latency breakdown analysis.

regions with the lowest read or write latency. For example, a write located in Region-1 having the lowest write latency is denoted as "Optimal W". "Non-optimal R" and "Non-optimal W" indicate that data are not located in their best region when the requests arrive. "Migration" indicates the overhead of migration contributing to the overall latency. In Figure 9, the EFM scheme contains the largest "Optimal R+W" portion for these two traces. For the WARM scheme, it faces a large overhead of "Non-optimal W" and "Non-optimal R" due to its misclassification. The HOTIS and TOS18 schemes have large migration overheads because they pre-allocate data to the low-read latency region but do not have many read requests arrived in the near future. As a consequence, the large migration overhead does not bring too much benefit of low read latency and results in a large overall performance degradation.

5.3 Lifetime Improvement

In this subsection, we investigate the lifetime improvement of the flash memory by reducing the threshold voltage as shown in Figure 2. In order to reduce the write/wearing effect on the flash memory, write-intensive workloads should be scheduled into the region with the reduced threshold voltage. We make a comparison with three baselines. As indicated in [10], the TOS18 scheme stores the wearing effective write requests in Region-2. The EFM, WARM, and HOTIS schemes use the reduced threshold voltage for Region-1. The effective wearing coefficient is set to 0.8. The latency configurations are shown as the effective wearing in Table 3.

As shown in Figure 10, the EFM scheme obtains the lifetime improvement about 17.7% on average, which achieves

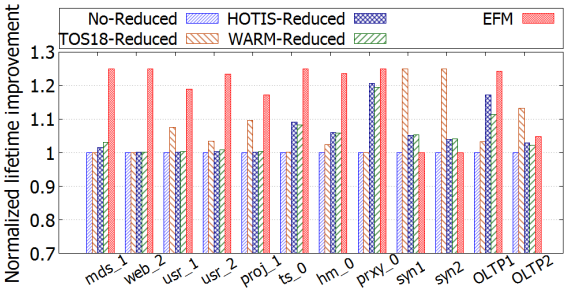


Figure 10: Normalized lifetime improvement.

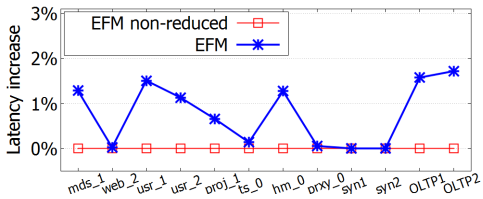


Figure 11: Latency comparison between EFM scheme and EFM scheme without reduced threshold voltage.

a longer lifetime than those of the three baselines – TOS18 (7.5%), HOTIS (5.6%), and WARM (5.1%). The reason is that the lifetime improvement management of the EFM scheme cooperates with the classifiers and allocates write-intensive workloads into Region-1. Therefore, the EFM scheme puts more writes in the effective wearing region than the TOS18 scheme. Another reason is that the EFM scheme accurately schedules write-intensive workloads to Region-1 and thus absorbs more writes in Region-1 than WARM and HOTIS do. As a result, the EFM scheme improves the lifetime of flash memory more efficiently than others. The EFM scheme shows less lifetime improvement on three exception traces – syn1, syn2, and OLTP2. The reason is that these traces contain interleaved writes and reads and are allocated in Region-1 to obtain shorter overall latency. Consequently, the EFM scheme achieves much better overall performance while obtains a slightly lower lifetime improvement than others in these three traces.

Moreover, another advantage of the EFM scheme is that the read performance degradation induced by effective wearing can be mitigated by the large number of writes with the lowest write latency. As indicated in Figure 11, the average latency of the EFM scheme with the reduced threshold voltage is only increased at most 2%. In summary, the EFM scheme is capable of improving the lifetime of the flash memory while its performance slightly decreases. To further improve the flash memory lifetime, we can set a smaller wearing effective coefficient while it will further sacrifice a bit performance.

5.4 Wear-out Aware Management

As discussed in previous sections, with flash memory continuously wearing out, the read performance will be degraded due to higher RBERs and longer decoding latency. In this

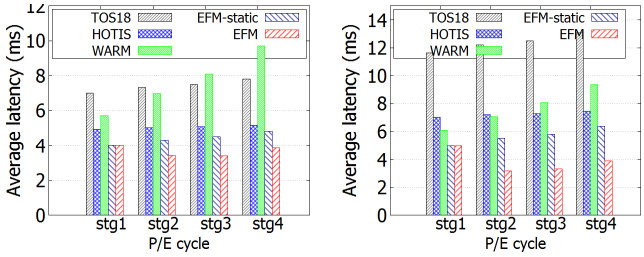
Table 4: Latencies of read and write with different reduced effective wearing for different PE cycles

	PE cycle:	Stage-1	Stage-2	Stage-3	Stage-4
Region-1	Read (us)	150	210	270	350
	Write (us)		450		
Region-2	Read (us)	110	150	170	210
	Write (us)		600		
Region-3	Read (us)	70	70	70	70
	Write (us)		800		

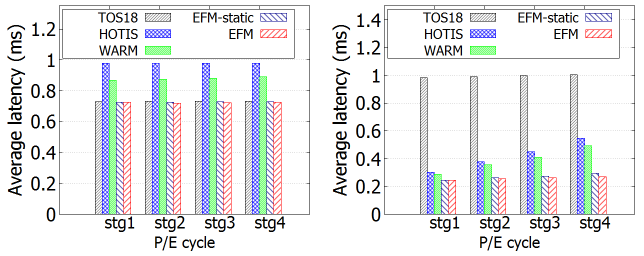
subsection, we investigate the performance of different strategies at different PE cycles. As seen in Table 4, according to the device model [8, 38, 63], four stages are used for demonstrating the flash memory wearing-out process. From Stage-1 (the smallest PE cycles) to Stage-4 (the largest PE cycles), the read latencies of two regions (Region-1 and Region-2) are decreased and the read latency in Region-3 keeps the same since Region-3 uses a smaller ΔV_{pp} which resulting in low RBERs. The write latencies remain the same due to using the same ΔV_{pp} in the ISPP process as discussed in Section 3.2.

Four baselines are used to make comparisons with the EFM scheme. The EFM-static scheme uses the initial setup at Stage-1 and its LT-classifier will not be updated. These four baselines keep the same configurations of the classifiers during their wearing-out process. The EFM scheme will adjust the LT-classifier and the migration checker accordingly as indicated in Algorithm 1. We select four representative traces and the conclusions of other traces are similar to one of the four traces. As shown in Figure 12, the average latencies of the four baselines are increased as flash memory wears out. This is because under the static allocation, the read latencies are increased resulting in larger overall latencies. However, these four schemes obtain different rates of latency increase for different traces. For example, TOS18 only has 0.8% - 2.2% latency increases for OLTP2 trace, but HOTIS, WARM, and EFM-static obtain 25.1% - 81.7%, 23.3% - 70.9%, and 7.5% - 21.7% latency increases, respectively. The reason is that the physical regions have different performance degradation. So, if one scheme allocates more read requests in Region-3, it will obtain less performance degradation.

For the EFM scheme, the wear-out aware management can help decrease the average latency when the flash memory wears out. The reason is that during the process of updating the LT-classifier, the EFM scheme can classify some specific access patterns more accurately than before and thus achieves lower average latencies as shown in Figure 12a and Figure 12b. In some cases, such as prxy_0 and OLTP2 traces, the EFM obtains a similar performance as flash memory wearing out. There are two reasons for this. One reason is if a trace is a write-intensive workload such as prxy_0, flash memory wearing has little influence on performance. The other reason is that the EFM scheme already accurately allocates most of the I/O requests. Thus, although the classifier and migration checker are updated, most of the requests are still classified in the same region and suffer little performance degradation. In



(a) usr_1 (b) proj_1



(c) prxy_0 (d) OLTP2

Figure 12: Performance comparison with flash memory wearing out for different schemes.

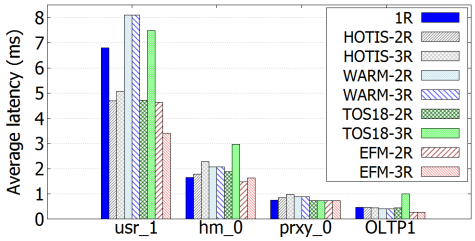


Figure 13: Performance comparison between four schemes with one to three physical regions.

summary, using the wear-out management, the EFM scheme can further improve the flash memory performance by 17.3% on average.

5.5 Different Numbers of Physical Regions

In this subsection, we investigate the influence of number of physical regions on the performance of flash memory. Under the same maximum threshold voltage, the difference of latencies between some regions becomes smaller as increasing the number of regions. So, the results of more than three regions might be similar to the case of three regions. Thus, we only consider one to three regions in this subsection. As seen in Figure 13, four representative traces are used for comparison. All schemes will obtain the same result for one region case (denoted by "1R"). According to the results, the traces can be roughly categorized into three groups. One is that the EFM scheme with three regions (denoted by "EFM-3R") has a better performance than that with one and two regions (denoted by "EFM-2R") like trace usr_1. For this type of traces, the scenario with three regions provides more fine-grained management than that of one and two regions since those traces contain many blocks with a small difference of access patterns. Therefore, the fine-grained management can distinguish the small difference of access patterns and schedule them into the low-cost region which can obtain a shorter latency. For the second group of traces like prxy_0 and OLTP1, the EFM schemes with two and three regions have a similar performance since these traces have obvious classification and the scheme with the coarse-grained physical region partition is good enough to classify those traces. The third category is that using less regions can obtain a better performance like hm_0. The reason is that the migration overhead dominates the overall performance. Using more regions potentially induces higher migration overhead.

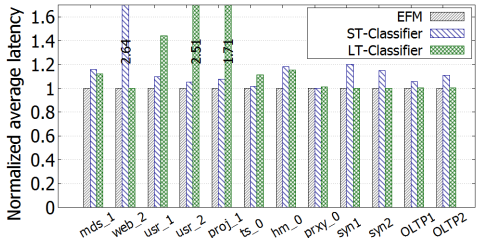
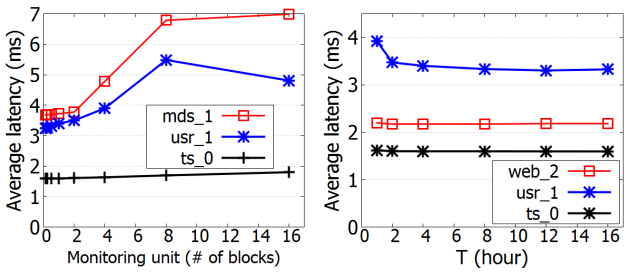


Figure 14: Latency comparison between EFM scheme, ST-Classifier, and LT-Classifier.

For other schemes, we obtain a similar conclusion as above. However, those schemes deliver a massive difference for some traces. For example, TOS18-2R and TOS18-3R have a similar situation with the EFM scheme for the trace prxy_0. While TOS18-3R has much worse performance than TOS18-2R for usr_1 and hm_0. A similar situation can be found for HOTIS-2R and HOTIS-3R for usr_1. The reason for this is that these two schemes suffer much more migration overheads due to their misclassification and the results in Figure 9 validate this conclusion. In summary, the EFM scheme can obtain a better performance than other schemes in the cases of different numbers of regions. Moreover, to gain the best performance, we should determine the number of physical regions in flash memory according to their applications.

5.6 Effect of Individual Classifier

To explore the performance improvement of EFM scheme, the performance of individual ST-Classifier and LT-Classifier is simulated as shown in Figure 14. As seen from the results, a single LT-Classifier can achieve similar performance with EFM scheme in some cases such as in prxy_0 and OLTP1. However, for the traces like usr_1, usr_2, and proj_1, a single LT-Classifier has more than 70% performance degradation than that of EFM scheme. The reason is that the LT-Classifier only considers long-term access patterns. If a change of the access pattern happens, the LT-Classifier cannot immediately react and will mis-classify the access patterns resulting in a longer access latency. Similarly, a single ST-Classifier obtains 22% worse performance than that of EFM scheme since the ST-classifier fails to predict the access patterns accurately in a long term. Therefore, with the assistance of both the



(a) monitoring unit size (b) Period

Figure 15: Sensitivity of different design parameters.

ST-Classifier and the LT-Classifier, not only we can have a long-term view of access patterns following Eq. (5), but also can quickly react to the change of access patterns. In summary, by combining both classifiers, the EFM scheme can further improve flash memory performance more than 20% on average when compared to the single-classifier cases.

5.7 Sensitivity of Design Parameters

In this subsection, we investigate the influence of two design parameters, **Monitoring unit size** and **Period** T, on the performance.

Monitoring unit size: the default monitoring unit size is one block. We varied the monitoring size from $1/8$ to 16 blocks. As seen in Figure 15a, there are two types of workloads. One (e.g., ts_0) has a similar average latency with varying monitoring size. For this type of workloads, the monitoring size has little effect on the performance because few migration happens. The other one (e.g., mds_1 and usr_1) has a longer average latency as the monitoring size increases. The reason is that this type of workloads faces a large amount of migration. Thus, as the monitoring size increases, the migration overhead will be increased as well.

Period T: the period T varies from 1 hour to 16 hours. Three results are provided in Figure 15b. The EFM scheme with different periods obtains similar access latencies. When the period is 1 hour, the performance is a little worse than those with larger periods for usr_1. The reason is that the scheme in a short period cannot accumulate enough information to make accurate classifications. In summary, the period has less effect on the performance, and we can use a little long period to obtain a little better performance.

6 Related Work

Two main research directions are investigated by previous studies due to the trade-off between performance and lifetime. For the first research direction on improving read or write performance, several existing works [28, 30, 37, 64–66] focused on improving read performance by optimizing ECC decoding performance. For example, Dong *et al.* [28] used different stronger-than-normal ECC and reduced soft-decision sensing scheme to improve read performance. In [64], an error aware LDPC decoding scheme called REAL is proposed to improve read performance. LALDPC [30] was proposed to avoid unnecessary read-retries by storing the high read level pages in the cache as long as possible to improve the overall read performance. Choi *et al.* [65] enhanced the performance of flash memories by using a new code technique called Invalid Data-Aware coding to shorten the latency of upper bits of a cell close to lower bit read latency of the cell. To improve write performance, some works [9, 57] tuned the program step size in ISPP to relax retention time requirement. In contrast, Wu *et al.* [67] improved the read performance as satisfying the write performance. However, all those studies only simply improved read or write performance and did not adequately consider the latency difference between read and write.

The other research direction focuses on the lifetime improvement of flash memories [11, 68–70]. In [11], the authors proposed a dynamic program and erase scaling scheme for improving the lifetime of NAND flash memory by lowering the erase voltages. Tang *et al.* [68] showed the Restricted Insertion Priority Queue (RIPQ) framework to achieve a long device lifespan by grouping similar content together and lazily updating. Li *et al.* [69] proposed a container-based flash cache to store the write-intensive pages in the cache and thus extended the lifespan of flash memories. Cheng *et al.* [70] explored offline algorithms and further improved the lifetime of flash memories by exploiting a new trade-off between the NAND endurance and erase voltage. The WARM scheme [12] proposed to improve NAND flash memory lifetime by separating the write hot and write cold data. However, these studies mainly targeted on lifetime extension with less consideration of performance improvement. In this work, we mainly emphasize on improving overall performance based on the different latencies of read and write of three physical regions. At the same time, the lifetime of NAND flash memories is further optimized while maintaining similar overall performance.

7 Conclusion

In this paper, a newly proposed elastic flash management scheme called EFM targets on improving the overall performance and lifetime of flash memory by allocating data into the physical regions with different write/read latencies. Two types of classifiers are used in the EFM to achieve a more accurate allocation for incoming requests. Moreover, as flash memory wears out, the LT-Classifier of the EFM will be adaptively updated to adapt to the performance changes of read and write. Additionally, a reduced effective wearing management is used to improve the lifetime of flash memory by scheduling the write-intensive workloads to the region with the reduced threshold voltage. Finally, the experimental results indicate that the EFM scheme can improve the overall performance 53.9% - 2.96x compared to previous works.

References

[1] O. Mutlu, "Error analysis and management for mlc nand flash memory," *Flash Memory Summit*, 2014.

[2] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in mlc nand flash memory: Measurement, characterization, and analysis," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 521–526, EDA Consortium, 2012.

[3] Y. Cai, O. Mutlu, E. F. Haratsch, and K. Mai, "Program interference in mlc nand flash memory: Characterization, modeling, and mitigation," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pp. 123–130, IEEE, 2013.

[4] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in mlc nand flash memory: Characterization, analysis, and modeling," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 1285–1290, EDA Consortium, 2013.

[5] W. Wang, T. Xie, and D. Zhou, "Understanding the impact of threshold voltage on mlc flash memory performance and reliability," in *Proceedings of the 28th ACM international conference on Supercomputing*, pp. 201–210, ACM, 2014.

[6] W. Liu, F. Wu, M. Zhang, Y. Wang, Z. Lu, X. Lu, and C. Xie, "Characterizing the reliability and threshold voltage shifting of 3d charge trap nand flash," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 312–315, 2019.

[7] K.-D. Suh, B.-H. Suh, Y.-H. Lim, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, *et al.*, "A 3.3 v 32 mb nand flash memory with incremental step pulse programming scheme," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, 1995.

[8] Y. Pan, G. Dong, and T. Zhang, "Exploiting memory device wear-out dynamics to improve nand flash memory system performance," in *Proceedings of the 9th USENIX Conference on File and Stroage Technologies (FAST 11)*, pp. 1–14, USENIX Association, 2011.

[9] Y. Pan, G. Dong, Q. Wu, and T. Zhang, "Quasi-nonvolatile ssd: Trading flash memory nonvolatility to improve storage system performance for enterprise applications," in *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pp. 1–10, IEEE, 2012.

[10] Q. Li, L. Shi, C. Gao, Y. Di, and C. J. Xue, "Access characteristic guided read and write regulation on flash based storage systems," *IEEE Transactions on Computers*, 2018.

[11] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Lifetime improvement of nand flash-based storage systems using dynamic program and erase scaling," in *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)*, pp. 61–74, 2014.

[12] Y. Luo, Y. Cai, S. Ghose, J. Choi, and O. Mutlu, "Warm: Improving nand flash memory lifetime with write-hotness aware retention management," in *Mass Storage Systems and Technologies (MSST), 2015 31st Symposium on*, pp. 1–14, IEEE, 2015.

[13] F. Chen, D. A. Koufaty, and X. Zhang, "Hystor: making the best use of solid state drives in high performance storage systems," in *Proceedings of the international conference on Supercomputing*, pp. 22–32, ACM, 2011.

[14] D. Park and D. H. Du, "Hot data identification for flash-based storage systems using multiple bloom filters," in *2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–11, IEEE, 2011.

[15] B. Li, C. Deng, J. Yang, D. Lilja, B. Yuan, and D. Du, "Haml-ssd: A hardware accelerated hotness-aware machine learning based ssd management," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, IEEE, 2019.

[16] J. Gu, C. Wu, and J. Li, "Hotis: A hot data identification scheme to optimize garbage collection of ssds," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, pp. 331–3317, IEEE, 2017.

[17] K. Takeuchi, T. Tanaka, and H. Nakamura, "A double-leve1-v th select gate array architecture for multilevel nand flash memories," *IEICE Transactions on Electronics*, vol. 79, no. 7, pp. 1013–1020, 1996.

[18] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 489–502, 2003.

[19] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on nand flash memory cell operation," *IEEE Electron Device Letters*, vol. 23, no. 5, pp. 264–266, 2002.

[20] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, O. Unsal, A. Cristal, and K. Mai, "Neighbor-cell assisted error correction for mlc nand flash memories," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, pp. 491–504, ACM, 2014.

[21] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and control*, vol. 3, no. 1, pp. 68–79, 1960.

[22] T. Kgil, D. Roberts, and T. Mudge, "Improving nand flash based disk caches," in *ACM SIGARCH Computer Architecture News*, vol. 36, pp. 327–338, IEEE Computer Society, 2008.

[23] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang, "Ldpc-in-ssd: Making advanced error correction codes work effectively in solid state drives," in *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)*, pp. 243–256, 2013.

[24] D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[25] S. M. Amoroso, A. Maconi, A. Mauri, C. M. Compagnoni, A. S. Spinelli, and A. L. Lacaita, "Three-dimensional simulation of charge-trap memory programming—part i: Average behavior," *IEEE Transactions on Electron Devices*, vol. 58, no. 7, pp. 1864–1871, 2011.

[26] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Errors in flash-memory-based solid-state drives: Analysis, mitigation, and recovery," *arXiv preprint arXiv:1711.11427*, 2017.

[27] H. Choi, W. Liu, and W. Sung, "Vlsi implementation of bch error correction for multilevel cell nand flash memory," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 5, pp. 843–847, 2009.

[28] G. Dong, N. Xie, and T. Zhang, "Enabling nand flash memory use soft-decision error correction codes at minimal read latency overhead," *IEEE Transactions on Circuits and Systems I: regular papers*, vol. 60, no. 9, pp. 2412–2421, 2013.

[29] M. Zhang, F. Wu, Y. Du, C. Yang, C. Xie, and J. Wan, "Cooecc: A cooperative error correction scheme to reduce ldpc decoding latency in nand flash," in *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 657–664, IEEE, 2017.

[30] Y. Du, D. Zou, Q. Li, L. Shi, H. Jin, and C. J. Xue, "Laldpc: Latency-aware ldpc for read performance improvement of solid state drives," in *2017 33rd International Conference on Massive Storage Systems and Technology (MSST)*, pp. 1–13, 2017.

[31] K. Wang, G. Du, Z. Lun, W. Chen, and X. Liu, "Modeling of program vth distribution for 3-d tlc nand flash memory," *Science China Information Sciences*, vol. 62, no. 4, p. 42401, 2019.

[32] L. Shi, K. Wu, M. Zhao, C. J. Xue, and H. Edwin, "Retention trimming for wear reduction of flash memory storage systems," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2014.

[33] C. M. Compagnoni, M. Ghidotti, A. L. Lacaita, A. S. Spinelli, and A. Visconti, "Random telegraph noise effect on the programmed threshold-voltage distribution of flash memories," *IEEE electron device letters*, vol. 30, no. 9, pp. 984–986, 2009.

[34] K. Fukuda, Y. Shimizu, K. Amemiya, M. Kamoshida, and C. Hu, "Random telegraph noise in flash memories-model and technology scaling," in *2007 IEEE International Electron Devices Meeting*, pp. 169–172, IEEE, 2007.

[35] H. Park, J. Kim, J. Choi, D. Lee, and S. H. Noh, "Incremental redundancy to reduce data retention errors in flash-based ssds," in *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–13, IEEE, 2015.

[36] B. Peleato and R. Agarwal, "Maximizing mlc nand lifetime and reliability in the presence of write noise," in *2012 IEEE International Conference on Communications (ICC)*, pp. 3752–3756, IEEE, 2012.

[37] S.-H. Chen, M.-C. Yang, and Y.-H. Chang, "The best of both worlds: On exploiting bit-alterable nand flash for lifetime and read performance optimization," in *Proceedings of the 56th Annual Design Automation Conference 2019*, p. 212, ACM, 2019.

[38] Q. Li, L. Shi, C. Gao, K. Wu, C. J. Xue, Q. Zhuge, and E. H.-M. Sha, "Maximizing io performance via conflict reduction for flash memory storage systems," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pp. 904–907, EDA Consortium, 2015.

[39] Q. Xiong, F. Wu, Z. Lu, Y. Zhu, Y. Zhou, Y. Chu, C. Xie, and P. Huang, "Characterizing 3d floating gate nand flash: Observations, analyses, and implications," *ACM Transactions on Storage (TOS)*, vol. 14, no. 2, p. 16, 2018.

[40] "Micron technical note: Design and use considerations for nand flash memory." https://www.micron.com/~/media/documents/products/technical-note/nand-flash/tn2917.pdf.

[41] S. Hachiya, K. Johguchi, K. Miyaji, and K. Takeuchi, "Tlc/mlc nand flash mix-and-match design with exchangeable storage array," in *International Conference on Solid State Devices and Materials*, pp. 894–895, 2013.

[42] Y. S. Kang, "Flash memory device and method of controlling program voltage," July 21 2009. US Patent 7,564,714.

[43] S. K. Kim, "Method for setting programming start bias for flash memory device and programming method using the same," June 16 2009. US Patent 7,548,464.

[44] R. Jones and R. Lins, *Garbage collection: algorithms for automatic dynamic memory management*, vol. 208. Wiley Chichester, 1996.

[45] R. Subramani, H. Swapnil, N. Thakur, B. Radhakrishnan, and K. Puttaiah, "Garbage collection algorithms for nand flash memory devices–an overview," in *Modelling Symposium (EMS), 2013 European*, pp. 81–86, IEEE, 2013.

[46] S. Lee, D. Shin, and J. Kim, "Bagc: Buffer-aware garbage collection for flash-based storage systems," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2141–2154, 2012.

[47] W.-H. Lin and L.-P. Chang, "Dual greedy: Adaptive garbage collection for page-mapping solid-state disks," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 117–122, EDA Consortium, 2012.

[48] P. Desnoyers, "Analytic modeling of ssd write performance," in *Proceedings of the 5th Annual International Systems and Storage Conference*, p. 12, ACM, 2012.

[49] L.-P. Chang, "On efficient wear leveling for large-scale flash-memory storage systems," in *Proceedings of the 2007 ACM symposium on Applied computing*, pp. 1126–1130, ACM, 2007.

[50] M. Murugan and D. H. Du, "Rejuvenator: A static wear leveling algorithm for nand flash memory with minimized overhead," in *2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–12, IEEE, 2011.

[51] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo, "Improving flash wear-leveling by proactively moving static data," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 53–65, 2009.

[52] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo, "Endurance enhancement of flash-memory storage, systems: An efficient static wear leveling design," in *2007 44th ACM/IEEE Design Automation Conference*, pp. 212–217, IEEE, 2007.

[53] J. Huang, A. Badam, L. Caulfield, S. Nath, S. Sengupta, B. Sharma, and M. K. Qureshi, "Flashblox: Achieving both performance isolation and uniform lifetime for virtualized ssds," in *15th USENIX Conference on File and Storage Technologies (FAST 17)*, pp. 375–390, 2017.

[54] T. Tanzawa, A. Umezawa, T. Taura, H. Shiga, T. Hara, Y. Takano, T. Miyaba, N. Tokiwa, K. Watanabe, H. Watanabe, K. Masuda, K. Naruke, H. Kato, and S. Atsumi, "A 44-mm/sup 2/ four-bank eight-word page-read 64-mb flash memory with flexible block redundancy and fast accurate word-line voltage controller," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1485–1492, 2002.

[55] "Samsung key value ssd enables high performance scaling." https://www.samsung.com/semiconductor/global.semi.static/Samsung_Key_Value_SSD_enables_High_Performance_Scaling-0.pdf.

[56] "Key value ssd explained, concept, device, system and standard." https://www.snia.org/sites/default/files/SDC/2017/presentations/Object_ObjectDriveStorage/Ki_Yang_Seok_Key_Value_SSD_Explained_Concept_Device_System_and_Standard.pdf.

[57] R.-S. Liu, C.-L. Yang, and W. Wu, "Optimizing nand flash-based ssds via retention relaxation," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST12)*, pp. 1–11, USENIX Association, 2012.

[58] S. Lee and J. Kim, "Effective lifetime-aware dynamic throttling for nand flash-based ssds," *IEEE Transactions on Computers*, vol. 65, no. 4, pp. 1075–1089, 2014.

[59] J. Jeong, Y. Song, S. S. Hahn, S. Lee, and J. Kim, "Dynamic erase voltage and time scaling for extending lifetime of nand flash-based ssds," *IEEE Transactions on Computers*, vol. 66, no. 4, pp. 616–630, 2016.

[60] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren, "Exploring and exploiting the multilevel parallelism inside ssds for improved performance and endurance," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1141–1155, 2013.

[61] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: Practical power management for enterprise storage," *ACM Transactions on Storage (TOS)*, vol. 4, no. 3, p. 10, 2008.

[62] "Umass." http://traces.cs.umass.edu/index.php/Storage/Storage.

[63] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Improving nand endurance by dynamic program and erase scaling," in *Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems (HotStrage 13)*, 2013.

[64] M. Zhang, F. Wu, X. He, P. Huang, S. Wang, and C. Xie, "Real: A retention error aware ldpc decoding scheme to improve nand flash read performance," in *2016 32nd Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–13, IEEE, 2016.

[65] W. Choi, M. Jung, and M. Kandemir, "Invalid data-aware coding to enhance the read performance of high-density flash memories," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 482–493, IEEE, 2018.

[66] Q. Li, M. Ye, Y. Cui, L. Shi, X. Li, and C. J. Xue, "Sentinel cells enabled fast read for nand flash," in *11th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 19)*, 2019.

[67] G. Wu, X. He, N. Xie, and T. Zhang, "Exploiting workload dynamics to improve ssd read latency via differentiated error correction codes," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 4, p. 55, 2013.

[68] L. Tang, Q. Huang, W. Lloyd, S. Kumar, and K. Li, "Ripq: Advanced photo caching on flash for facebook," in *13th USENIX Conference on File and Storage Technologies (FAST 15)*, pp. 373–386, 2015.

[69] C. Li, P. Shilane, F. Douglis, and G. Wallace, "Pannier: Design and analysis of a container-based flash cache for compound objects," *ACM Transactions on Storage (TOS)*, vol. 13, no. 3, p. 24, 2017.

[70] Y. Cheng, F. Douglis, P. Shilane, G. Wallace, P. Desnoyers, and K. Li, "Erasing belady's limitations: In search of flash cache offline optimality," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pp. 379–392, 2016.