EFM: Elastic Flash Management to Enhance Performance of Hybrid Flash Memory

Bingzhe Li*, Bo Yuan†, and David Du‡
*Oklahoma State University, Stillwater, OK, USA
†Rutgers University, Piscataway, NJ, USA
‡University of Minnesota – Twin Cities, Minneapolis, MN, USA
bingzhe.li@okstate.edu, bo.yuan@soe.rutgers.edu, du@umn.edu

2021 IEEE 39th International Conference on Computer Design (ICCD) | 978-1-6654-3219-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCD53106.2021.00035

***Abstract*—NAND-based flash memory has become a prevalent storage media due to its low access latency and high performance. By setting up different incremental step pulse programming (ISPP) values and threshold voltages, the tradeoffs between lifetime and access latency in NAND-based flash memory can be exploited. The existing studies that exploit the tradeoffs by using heuristic algorithms do not consider the dynamically changed access latency due to wearing-out, resulting in low access performance. In this paper, we proposed a new Elastic Flash Management scheme, called EFM, to manage data in hybrid flash memory, which consists of multiple physical regions with different read/write latencies according to their ISPP values and threshold voltages. EFM includes a Long-Term Classifier (LT-Classifier) and a Short-Term Classifier (ST-Classifier) to accurately track dynamically changed workloads by considering current quantitative differences of read/write latencies and workload access patterns. Moreover, a reduced effective wearing management is proposed to prolong the lifetime of flash memory by scheduling write-intensive workloads to the region with a reduced threshold voltage and the lowest write cost. Experimental results indicate that EFM reduces the average read/write latencies by about 54% - 296% and obtain 17.7% lifetime improvement on average compared to the existing studies.**

I. Introduction

NAND flash memory is playing an important role in today's storage systems from mobile devices to large-scale datacenters. Compared to other types of storage devices [1], [2], it offers the advantages of high performance and lightweight. 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 increased bit-density, the reliability of flash memory is decreased [3]. As a result, the lifetime of flash memory is shortened due to a decreased maximum Program-Erase (PE) cycle. To deal with this, researchers use Error Correction Codes (ECC) such as Low-Density Parity-Check Code (LDPC) to make data accessible with a high Raw Bit Error Rate (RBER) and thus prolong the lifetime of flash memory. However, these error corrections need a longer duration to decode data, resulting in degraded flash access performance.

According to NAND flash memory properties, the tradeoffs between lifetime and access latencies can be exploited by setting different threshold voltages, incremental step pulse programming (ISPP) values, etc. for different flash memory regions. For example, different ISPP values [4], [5] can impact both read and write latencies. With a larger ISPP value, the program process (write operation) has fewer iterations and achieves a lower write latency. However, a larger ISPP value increases the RBER of programmed memory cells and thus results in longer read latency. Another trade-off is between the lifetime of NAND flash memory and its access latencies. With a low program threshold voltage, the RBER of memory cells generally increases and the maximum number of PE cycles of those cells is also increased. Thus, a lower threshold voltage causes a higher read latency but a longer lifetime of flash memory. These trade-offs provide opportunities to improve the performance and lifetime of NAND flash memory.

Many studies [5]–[9] used these trade-offs either to prolong the lifetime of flash memory or to improve its performance. For example, Li *et al.* [7] simply used a simple static heuristic by classifying I/O requests with their most recent consecutive access patterns. Pan *et al.* [5] explored a device model of flash memory and re-setting ISPP values of flash memory to improve its write performance. Luo *et al.* [9] proposed the WARM scheme to improve flash memory lifetime by separating data based on their write frequencies. However, these schemes do not consider the changes of read and write latencies in SSDs due to wearing out. For example, as flash memory wears out, the access latencies of flash memory are continuously changed. Existing allocation schemes based on the initial classification cannot adapt to these changes, resulting in longer access latencies due to less accurate data allocation. Therefore, to obtain high performance and adapt to dynamically changed latencies of flash memory, we have to consider real and dynamically changed write/read latencies under the current PE cycle.

In this paper, the proposed Elastic Flash Management scheme (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 the quantitative difference of current read/write latencies between physical regions and long-term accumulated read/write sizes. Then, a Short-Term Classifier (ST-Classifier) calibrates LT-Classifier based on short-term access patterns. Moreover, as the NAND

978-1-6654-3219-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCD53106.2021.00035

flash memory wears out, the LT-Classifier of EFM will be dynamically adjusted to adapt to the changed access latencies. The outcomes of both classifications are considered to decide where to allocate a given data. Consequently, EFM is capable of more accurately predicting where to allocate/migrate incoming requests. A migration checker is also developed to reduce migration overhead by filtering out unnecessary data migrations. Additionally, reduced effective wearing management is used to improve the lifetime of the NAND flash memory by scheduling write-intensive workloads to the region with a reduced threshold voltage and the lowest write cost.

The rest of the paper is organized as follows. Section II describes the backgrounds of flash memory. The discussion of motivations is introduced in Section III. Section IV discusses the structure and algorithm of the proposed EFM scheme. Section V shows the experimental results of EFM compared to the existing schemes. Finally, some conclusions are presented in Section VI.

II. Background

A. Background of Flash Memory

NAND flash memory stores N-bits in a cell by injecting electrons into the memory cell. As shown in Figure 1, N-bit data in the NAND flash memory are presented by 2^N voltage states. Each voltage state follows a wide Gaussian-like distribution [5] and can be approximately modeled by 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 write data, the Incremental Step Pulse Program (ISPP) programs flash memory cells iteratively following sequential program-and-verify steps until a certain target voltage is achieved [10]. In each step, the step size ΔV_{pp} (ISPP value) determines write latency. To reach the same target threshold voltage, a larger ΔV_{pp} needs a less number of steps resulting in shorter write latency, but introduces a higher error rate due to a narrower noise margin between two adjacent states. Figure 1 indicates an ideal voltage distribution. Practically, the voltage distribution is affected by many factors such as PE cycles and cell-to-cell interference [4], [11]. As flash memory wears out (the PE cycle of flash memory increases), any two adjacent voltage states will have a smaller separation margin or even be overlapped resulting in low reliability.

To read data, the voltage levels of a cell need to be correctly sensed. Due to errors in a cell induced by many factors such as wearing-out and cell-to-cell interference, error correction codes such as Bose–Chaudhuri–Hocquenghem (BCH) [12] and LDPC [13] are used. The decoding process of ECC codes is to sense the probability information of a cell by comparing a series of reference voltages. The number of iterations of comparing reference voltages determines read latency. With a higher error rate/RBER, more sensing iterations are needed resulting in a longer read latency.

Fig. 1: Ideal voltage distribution for 3-bit memory cell.

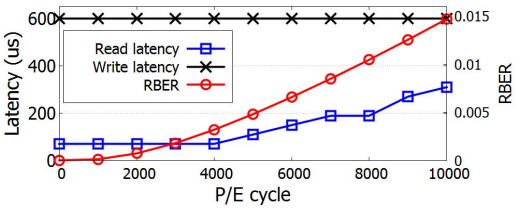


Fig. 2: Simulated latencies and RBER changes as increasing PE cycles with the same ΔV_{pp} based on the flash memory device model [5].

B. Performance Tradeoffs in Flash Memory

There are trade-offs in flash memory between read/write latencies and its lifetime. The relationship between ISPP and program (write) latency can be found in Eq. (2) [14].

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

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 inversely proportional to ΔV_{pp}. A large ΔV_{pp} can reduce the program latency but introduce a narrow margin between two adjacent voltage states, thus, resulting in a high RBER. For a read, the read latency is based on the error rate of the memory cell and the speed of ECC decoding. An ECC scheme such as BCH [12] or LDPC [13], [15] code is needed to correct the sensed data. With the same correction capacity of an ECC scheme, if a cell has a high RBER, the ECC scheme takes a longer time to correct the data. The RBER is related to ΔV_{pp}, PE cycle, cell-to-cell interference, etc. [4], [5], [16], and the error model 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 [5] with a random telegraph noise (RTN) [16] and cell-to-cell interference [11]. So, according to Eq. (1) and Eq. (3), as flash memory wears out (the PE cycle of flash memory increases), the RBER will be increased and thus ECC decoders need to take more iterations to read correct data out [4]. Consequently, the read latency is increased. A high ΔV_{pp} can shorten the program latency but increase the RBER. Therefore, by following the constraints of the required retention time (one year) [17], flash memory needs to take more iterations to decode data and it results in a longer read latency.

III. Design Motivation

As the flash memory wears out following Eq. (2) and Eq. (3) (i.e., the PE cycle of NAND flash memory increases), the read

latency is increased since more ECC decoding iterations are needed as mentioned in Section II. The program (write) latency is proportional to the ISPP [6] 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. We simulated the latencies and RBER changes as PE cycle increases. As shown in Figure 2, 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 access latencies, a static allocation scheme for SSDs cannot deliver good performance at all times. A type of mixed read/write workloads may have the best overall performance in one region at the beginning. However, with the change of read latency, the current workload should be allocated to another region to obtain a better performance. Therefore, a data classification scheme may need to be adjusted based on the current wearing-out stage of NAND flash memory. Moreover, flash memory with a high areal density becomes more and more popular. As the areal density of flash memory increases such as from SLC to QLC, the latency become more sensitive to the PE cycle. In other words, the latency will be changed more significantly with a small PE cycle increase for the flash memory with higher areal densities such as TLC and QLC. Thus, the wear-out aware management is more needed for those high areal density flash memory. In Section IV, we introduce a dynamically changed classifier to adapt to the wear-out process of flash memory for the hybrid SSDs.

IV. 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 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 [7]. For other N values, under the same maximum threshold voltage, the difference of latencies between some regions becomes smaller as the number of regions increases. So, the results of more than three regions might be similar to the case of three regions. Thus, a further discussion of one to three regions can be found in Section V-E.

A. Overall Architecture

In this paper, the access latencies of three physical regions and workload access patterns are used to classify data and to decide in which region the data is stored. Moreover, an adaptive scheme is proposed to adapt to the changing read/write latencies due to flash memory wearing. The overall architecture of EFM is shown in Figure 3. There are four basic components in EFM. An I/O monitor collects I/O information which is used for building a classifier. We use 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 by quantitatively distinguishing read/write

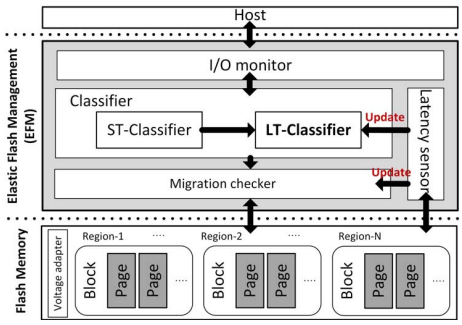


Fig. 3: The overall architecture of EFM.

latencies. The other classifier called Short-Term Classifier (**ST-Classifier**) assists and calibrates LT-Classifier to obtain a more accurate classification. A migration checker is responsible for filtering out unnecessary migration. 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 LT-Classifier to make it wearing-out aware.

B. EFM Scheduling Algorithm

In the design, we assume three physical regions. Region-1 is low-cost (high performance) for writes and high-cost (low performance) 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 to 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. (4).

$$\begin{aligned} & Y \times t_p + X \times t_r < Y \times t'_p + X \times t'_r \\ & => \begin{cases} \frac{Y}{X} < \frac{t'_r - t_r}{t_p - t'_p}, \text{if } t_p > t'_p \\ \frac{Y}{X} > \frac{t'_r - t_r}{t_p - t'_p}, \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

where X and Y are the accumulated read and write sizes of the data in the workload in an observed duration. Therefore, if the data access patterns follow Eq. (4), the data should be assigned to Region-1 to obtain a lower overall execution time. Otherwise, the data should be allocated to Region-3.

Therefore, with the awareness of read/write latencies and their accumulated sizes, LT-Classifier follows Eq. (4). The details of EFM are shown in Algorithm 1. We use a flash memory block as a logical block of I/O monitoring. First, EFM 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

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 Write_TBL[i]/Read_TBL[i] < (t_r2 − t_r1)/(t_p1 − t_p2) then
2:     RegionN = 1
3: else if Write_TBL[i]/Read_TBL[i] < (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("1") >= 5 then /*Read-intensive*/
2:     RegionN = 3
3: else if MRO[i].count("1") <= 2 then /*Write-intensive*/
4:     RegionN = 1
5: else
6:     RegionN = 2
7: return RegionN
```

Most Recent Operations (**MRO**) on each logical block are stored in the MRO table. "0" indicates write (W) and "1" is read (R). For example, "110" indicates the sequence of requests is "R,R,W". LT-Classifier in Algorithm 2 follows Eq. (4) 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 at Lines 6-11 of Algorithm 1, ST-Classifier uses the most recent seven operations to indicate 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 or read-intensive logical block, respectively. Otherwise, the requests for the block will be classified as an interleaved read and write by ST-Classifier.

LT-Classifier as a major classifier determines the region of allocation. The ST-Classifier assists LT-Classifier when the results of two classifiers have the difference of region values by 2 (at Lines 12-13 in Algorithm 1). In other words, if the classifications of long-term patterns and short-term patterns have a big contradiction with each other, the decision is made for the middle region (i.e., Region-2). If a read request for one logical block locates in high-cost read regions (like Region-1 or Region-2) and EFM classifies the logical 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). $mig\_threshold$ is set to $\frac{Cost_{migration}}{2*Gain_{read}}$. This indicates that if the potential gain of migration is bigger than half of the migration cost, we should migrate it. 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 and no extra action is taken. For each period T, $Read\_TBL$ and $Write\_TBL$ will be updated (at Lines 28-32). 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 flash memory information such as the read and write latencies of different regions. As indicated in Figure 2, at the early stages of flash memory, the read latency grows up slowly. Therefore, we set a longer duration to update the latencies of flash memory in LT-Classifier. As flash memory wears out, the increase of read latency becomes quicker. Thus, a shorter 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 can not only 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 4 indicates an example of EFM with three regions. According to Algorithm 1, the two blue lines are drawn by LT-Classifier according to Eq. (4). 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 LT-Classifier from Region-1 or Region-3 to Region-2 when facing conflict decisions between short-term and long-term classifiers. Moreover, the reduced threshold voltage $V_p^{'}$ is applied to Region-1 to prolong the lifetime of flash memory.

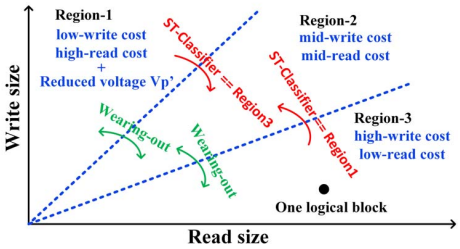


Fig. 4: One example of the classification of EFM with three physical regions (N=3).

Finally, based on the wear-out aware management, the slopes of LT-Classifier will be adjusted based on the latencies updated by the monitoring function (green arrows). In Figure 4, 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 in their original regions.

C. Overhead Discussion

The overhead of EFM is mainly from the following 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 to 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 Translation 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 contain more computing resources (e.g., with multi-core CPU). 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 several previous studies [5], [6], [13]. So, the hardware overhead is tolerable.

TABLE I: 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 II: 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	-

V. Experimental Results

A. Environmental Setup

We evaluated different algorithms based on the SSDsim simulator [18] with the extension of different read and write latencies of 3 physical regions. The latencies are obtained from the device model in [5]. 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 physical regions are indicated in Table II. The traces used in the experiments are the MSR Cambridge traces [19], two synthetic traces (syn1 and syn2), and two OLTP application traces [20] as shown in Table I. Three existing schemes, i.e., WARM [9], HOTIS [21], and TOS18 [7] are used as baselines to make comparisons with the proposed EFM scheme.

B. 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 II. The overall performance comparison is shown in Figure 5. We can find that EFM obtains the lowest average latencies compared to the other three baselines. On average, EFM delivers 53.9%, 87%, and 296% latency reductions compared to HOTIS, WARM, and TOS18, respectively. There are two major reasons that EFM outperforms the others. One is that the previous schemes misclassify the data access patterns such that data are allocated to 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 a lower overall performance.

Two typical examples of breakdown analysis are demonstrated in Figure 6. We did not present the results of other

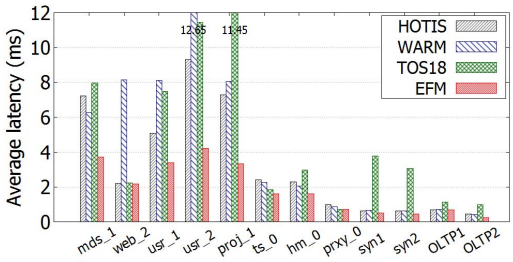
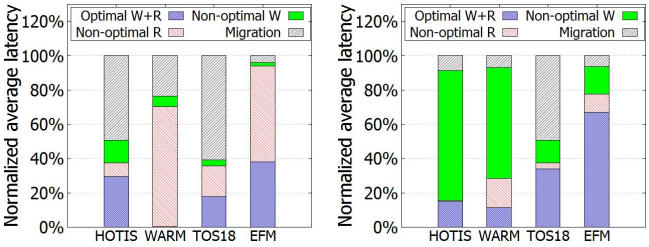


Fig. 5: Overall performance comparison between four schemes.



(a) usr_1 (b) hm_0

Fig. 6: Latency breakdown analysis.

traces since they have similar results. "Optimal R+W" indicates that read and write operations access the regions with the lowest read or write latency respectively. 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 regions when the requests arrive. "Migration" indicates the overhead of migration contributing to the overall latency. In Figure 6, EFM 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 a large migration overhead 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.

C. Wear-out Aware Management

As discussed in previous sections, with flash memory continuously wearing out, the read performance will be degraded due to high RBERs and long decoding latency. In this subsection, we investigate the performance of different strategies at different PE cycles. As seen in Table III, according to the device model [5], [22], [23], 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

TABLE III: 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			

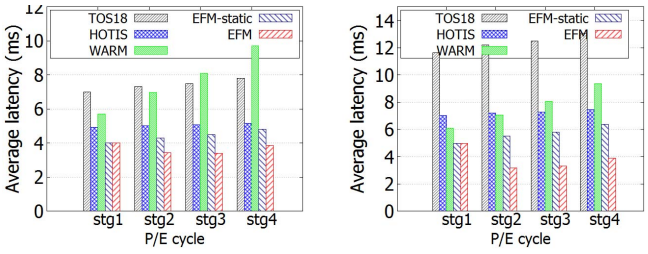
resulting in low RBERs. The write latencies remain the same due to using the same ΔV_{pp} in the ISPP process as discussed in Section III.

Four baselines are used to make comparisons with EFM. EFM-static 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 the wearing-out process. EFM will adjust 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 the four traces. As shown in Figure 7, 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 to Region-3, it will obtain less performance degradation.

The wear-out aware management can help decrease the average latency while the flash memory wears out. The reason is that with the updated LT-classifier, EFM can classify some specific access patterns more accurately than before and thus achieves lower average latencies as shown in Figure 7a and Figure 7b. In some cases, such as prxy_0 and OLTP2 traces, EFM obtains a similar performance as flash memory wears out. There are two reasons for this. One reason is that if a trace is a write-intensive workload like prxy_0, flash memory wearing has little influence on performance. The other reason is that EFM 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 to the same region and suffer little performance degradation. In summary, using the designed wear-out management, EFM can further improve the flash memory performance on average by 17.3%.

D. Lifetime Improvement

The lifetime improvement of flash memory is investigated in this subsection by reducing the threshold voltage. To reduce the wearing effect on flash memory, write-intensive workloads should be scheduled to a region with a reduced threshold voltage. We make a comparison with three baselines. As indicated



(a) usr_1 (b) proj_1

(c) prxy_0 (d) OLTP2

Fig. 7: Performance comparison with flash memory wearing out for different schemes.

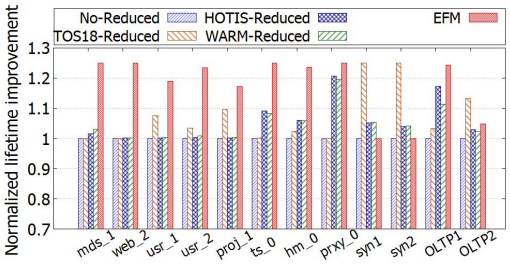


Fig. 8: Normalized lifetime improvement.

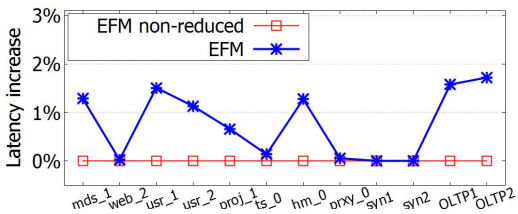


Fig. 9: Latency comparison between EFM and EFM without reduced threshold voltage.

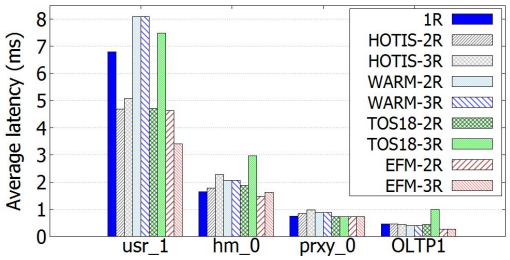


Fig. 10: Performance comparison between four schemes with one to three physical regions.

in [7], the TOS18 scheme stores the wearing effective write requests in Region-2. 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 II.

As shown in Figure 8, EFM obtains an average lifetime improvement about 17.7%. It achieves 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 EFM cooperates with the classifiers and allocates write-intensive workloads to Region-1. Therefore, the EFM scheme puts more writes in the effective wearing region than the TOS18 scheme does. Another reason is that EFM accurately schedules write-intensive workloads to Region-1 and thus absorbs more writes in Region-1 than WARM and HOTIS do. As a result, EFM effectively improves the lifetime of flash memory than others. EFM shows less lifetime improvement with three exceptional traces – syn1, syn2, and OLTP2. The reason is that these traces contain interleaved writes and reads and are allocated to Region-1 to obtain a shorter overall latency. Consequently, EFM achieves a much better overall performance while obtains a slightly lower lifetime improvement than others in these three traces.

Another advantage of EFM 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 9, the average latency of EFM with the reduced threshold voltage is only increased at most 2%. In summary, EFM is capable of improving the lifetime of the flash memory while its performance decreases only slightly. To further improve the flash memory lifetime, we can set a smaller wearing effective coefficient while it may further sacrifice a bit performance.

E. Number of Physical Regions

In this subsection, we investigate the influence of the number of physical regions on the performance of flash memory (one to three regions). In Figure 10, four representative traces are used for comparison. All schemes 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 EFM with three regions (denoted by "EFM-3R") has a better performance than that with one or two regions (denoted by "EFM-2R") like trace usr_1. For this type of traces, the scenario with three regions provides a 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 to the low-cost region with a shorter latency. For the second group of traces including prxy_0 and OLTP1, EFM with two and three regions has 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 including hm_0. The reason is that the migration overhead dominates the overall performance. Using more regions potentially induces a higher migration overhead.

For other schemes, we obtain a similar conclusion as above. However, those schemes deliver a big difference for some traces. For example, TOS18-2R and TOS18-3R have a similar situation with EFM 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 higher migration overheads due to their mis-classification and the results in Figure 6 validate this conclusion. In summary, EFM can obtain a better performance than other schemes with different number of regions. Moreover, to gain the best performance, we should determine the number of physical regions in flash memory according to the I/O behaviors of their applications.

VI. 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 three physical regions with different write/read latencies. Two types of classifiers are used in the EFM to achieve a more accurate allocation to these regions 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% - 296% compared to some previously proposed schemes.

VII. Acknowledgement

This work was partially supported by NSF I/UCRC Center Research in Intelligent Storage and the following NSF awards 1439622, 1812537, and CCF-1854737.

References

[1] F. Wu, B. Li, B. Zhang, Z. Cao, J. Diehl, H. Wen, and D. H. Du, "Tracklace: Data management for interlaced magnetic recording," *IEEE Transactions on Computers*, 2020.

[2] B. Li, L. Ou, and D. Du, "Img-dna: approximate dna storage for images," in *Proceedings of the 14th ACM International Conference on Systems and Storage*, 2021, pp. 1–9.

[3] 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*. EDA Consortium, 2013, pp. 1285–1290.

[4] 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)*, 2019, pp. 312–315.

[5] 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)*. USENIX Association, 2011, pp. 1–14.

[6] 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*. IEEE, 2012, pp. 1–10.

[7] 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.

[8] 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)*, 2014, pp. 61–74.

[9] 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*. IEEE, 2015, pp. 1–14.

[10] 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.

[11] 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.

[12] 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.

[13] 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)*. IEEE, 2017, pp. 657–664.

[14] 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.

[15] 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)*, 2017, pp. 1–13.

[16] 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.

[17] 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)*. IEEE, 2015, pp. 1–13.

[18] 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.

[19] 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.

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

[21] 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)*. IEEE, 2017, pp. 331–3317.

[22] 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.

[23] 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*. EDA Consortium, 2015, pp. 904–907.