Program Error Rate-based Wear Leveling for NAND Flash Memory

Xin Shi†, Fei Wu†*, Shunzhuo Wang†, Changsheng Xie†, Zhonghai Lu‡
†Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, China
‡KTH Royal Institute of Technology, Sweden
*Corresponding author: {Fei Wu, wufei@hust.edu.cn}
{shixin, wangshunzhuo, cs_xie }@hust.edu.cn, zhonghai@kth.se

***Abstract*—Wear leveling scheme has became a fundamental issue in the design of Solid State Disk (SSD) based on NAND Flash memory. Existing schemes aim to equalize the number of programming/erase (P/E) cycles and memory raw bit error rates (BER) among all the flash blocks. However, due to fabrication process variation, different blocks of the same flash chip usually have largely different endurance in terms of BER and program error rate (PER). Such conventional design cannot obtain the wear status of flash blocks precisely. This paper proposes PER-WL, an efficient PER-based wear leveling scheme that uses PER statistics as the measurement of flash block wear-out pace, and performs block data swapping to improve the wear leveling efficiency. In our evaluation with four realistic workloads, PER-based wear leveling scheme can achieve 17% and 9% variance of program error rate reduction, 8% and 3% program error rate reduction with 5% and 2% system performance degradation when compared to two state-of-the-art wear leveling schemes on average.**

I. Introduction

NAND Flash memory has been widely used as storage devices from embedded systems to laptops, desktops, and data centers because of its low access latency advantages, non-volatility, high density, low cost and shock resistance [1]. As it continues to become cheaper with increasing density, the use of NAND flash-based storage devices keeps rising over the next decade [2].

One of the major challenges of NAND Flash memory is the limited endurance (i.e., the maximal endurable P/E cycles) of flash blocks. Typically, NAND Flash memory can withstand 100,000 P/E cycles for single-level cell (SLC) type, 10,000 for multi-level cell (MLC) type or 3,000 for triple-level cell (TLC) type [3][4]. The dielectric barrier in the flash cells is increasingly damaged due to the program and erase operation. When the dielectric barrier is breakdown, the flash cell can no longer store bit information reliably. Error Correcting Codes (ECC) is widely used by the flash memory controller to provide error tolerance ability, but it can only verify bit errors in a limited range. Some blocks could be worn out earlier than the others due to the excessive program and erase operations. As a consequence, the write endurance of NAND Flash memory adversely impacts the lifespan of flash storage devices. In order to resolve the endurance issue, wear leveling techniques are proposed to improve the endurance of NAND Flash memory by equalizing wear-out pace among all the blocks as much as possible [5][6].

There are two types of wear leveling scheme: dynamic wear leveling (DWL) and static wear leveling (SWL) [7]. The dynamic wear leveling reuses blocks according to the erase count. This wear leveling scheme results in unevenness wear among all the blocks. In contrast to dynamic wear leveling, the static wear leveling considers the difference between hot data and cold data. It moves cold data to the high wear blocks, but introduces extra overhead.

However, conventional wear leveling schemes attempt to employ programming/erase (P/E) cycles or memory raw bit error rates (BER) as the equalization target during the NAND Flash memory's lifespan [8][9][10][11][12][13][14][15]. Basically, different blocks of the same flash chip usually have largely different endurance when they have experienced the same number of P/E cycles [16]. On the other hand, BER in a block includes multiple types of errors (program error, retention error and disturb error) which also have significant different impact on the wear out of flash cell [17][18][19][20][21]. Consequently, the above two equalization targets are not the best choice for the design of wear leveling scheme. Thus, the existing techniques undoubtedly have negative influence on the endurance of NAND Flash memory. Such an observation motivates us to consider other factors (such as program error rates) in the wear leveling design so as to estimate the endurance of each block more precisely and to improve the endurance of NAND Flash memory.

In contrast to existing wear leveling designs, this paper proposes PER-WL, an efficient program error rate (PER)-based wear leveling scheme to estimate the wear-out pace of each block more precisely. Besides explicitly using PER as equalization target, PER-WL applies PER-based block data swapping to further equalize the flash block wear-out pace among all the flash blocks. The major contributions of this paper are as follows:

- We propose an efficient PER-based wear leveling scheme (PER-WL), explicitly using PER information of flash blocks as equalization target and performs block data swapping to improve the wear leveling efficiency .
- We evaluate PER-WL with various realistic workloads. Experimental results show the advantages of such PER-based wear leveling over two state-of-the-art wear leveling schemes.

978-3-9819263-0-9/DATE18/©2018 EDAA

The rest of this paper is organized as follows. Section II provides the background and discusses related work. Section III presents the motivation for this research. A detailed description of PER-WL is given in Section IV. Section V presents the experimental results. Finally, Section VI concludes this paper.

II. Background and Related Work

NAND Flash memory resources are managed by a software layer called flash translation layer (FTL) [22]. A typical FTL contains three main functions: address translation, garbage collection, wear leveling. The objective of wear leveling is to evenly distribute the wear-out pace among all the blocks and avoid wearing out blocks more quickly than others, which can maximize the overall NAND Flash memory endurance and lifetime. Currently, wear leveling schemes can be divided into two categories: dynamic wear leveling (DWL) and static wear leveling (SWL). DWL designs recycle blocks according to the number of P/E cycles. Very different from DWL, SWL designs take data access characteristics into consideration. There are several wear leveling techniques for NAND Flash memory in the literature. Gal et al. [5] and Yang et al. [6] surveyed many published wear leveling algorithms and data structures for NAND Flash memory. Ben-Aroya et al. [23] evaluate different wear-leveling algorithms, including both on-line and off-line algorithms. Although the wear leveling for NAND Flash memory has been well studied, most prior work on wear leveling techniques are carried out mainly from P/E cycle and BER perspective. In the following, we briefly describe several existing P/E cycle and BER-based wear leveling scheme.

1) P/E Cycle: Most existing wear leveling techniques attempt to use P/E cycle as the design metric, to even out erase operations among all the blocks. For example, Chang et al. [8] propose a wear leveling algorithm, which allocates free flash blocks to the write requests according to the number of P/E cycle. Chang et al. [9] propose a non-realtime wear-leveler, which sequentially scans a given number of blocks to see if any block has a relatively small erase count. Chang et al. [10] presents a cyclic-queue-based scanning procedure for static wear leveling mechanism, to improve the endurance of flash memory with little overhead. Murugan et al. [11] propose a static wear leveling algorithm, allocating hot or cold data to young or old blocks respectively in a proactive way. It groups blocks that have the same erase count in a list. Liao et al. [12] propose an adaptive wear leveling scheme by leveraging a difference state and several varied bound thresholds. In this design, P/E cycle is the wear metric.

2) BER: Some prior works have attempted to estimate wear status of NAND Flash memory using other factors. Pan et al. [13] take the block endurance variance into consideration and propose a dynamic BER-based greedy wear-leveling algorithm that uses BER as the measurement of flash block wear-out pace. Peleato et al. [14] classify the blocks based on the BER and attempt to wear all the blocks evenly so that they all suffer the same BER. Yang et al. [15] propose a reliability-aware wear leveling scheme to distribute block erases based on the BER of blocks.

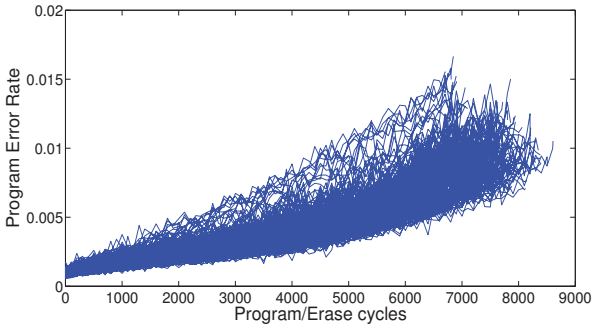

Fig. 1: Program error rates per 9MB block, based on measurements among 237 blocks of a 3D TLC NAND Flash memory device (TOSHIBA TH58TFT1T22BA8H) under the P/E cycles of 7 K-9 K

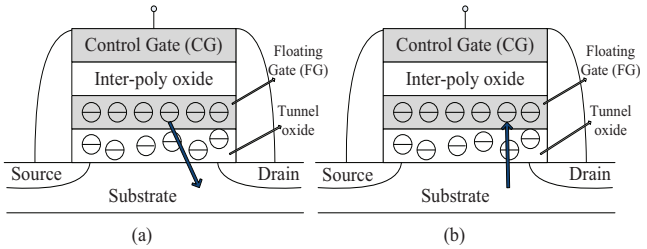

Fig. 2: (a) The cross-sectional view of a flash cell when retention error occurs; (b) The cross-sectional view of a flash cell when disturb error occurs; The blue arrow in (a) denotes the electrons present in the FG leak away; The blue arrow in (b) denotes the electrons inject to the FG;

III. Motivation

The unique characteristic of NAND Flash memory introduces the design of wear leveling scheme. We draw two critical observations as follows:

- **Observation 1**: Due to fabrication process variation [16], different blocks of the same flash chip usually have largely different endurance in terms of bit error rates (BER) and program error rates (PER). To quantitatively demonstrate such phenomenon, we conduct extensive experiments on real 3D TLC NAND Flash memory device. Figure 1 shows the program error rates per 9MB block based on measurements among 237 blocks of a 3D TLC NAND Flash memory device (TOSHIBA TH58TFT1T22BA8H) under the P/E cycles of 7K-9K. Each block has 576 16kB logical pages. Our experimental results indicate that some blocks will become unreliable earlier or later than others when they have experienced the same number of P/E cycles.
- **Observation 2**: Each bit in the flash cell is stored in a transistor called floating gate (FG). Bit errors can occur due to the deterioration of FG over repeated P/E cycles [24]. BER in flash chips mainly have three types of errors: program error, retention error and disturb error (read disturb error, write disturb error). Program errors are caused by over-programming to flash cells or fast

erasing cells when a page is programmed [17]. As show in Figure 1, our experimental results show that the number of program error bits is increased when the block endured more P/E cycles. Figure 2 shows the cross-sectional view of a flash cell when retention error and disturb error occur. The main reason of retention error is that the charge programmed in the FG may loss gradually [18], as shown in Figure 2(a). Retention errors can be reduced by periodically reading, correcting, and reprogramming the flash memory before the number of errors accumulated exceed the error correction ability of ECC [19]. This implies the retention error is not permanent. Disturb error happens when the data stored in a page changes while a neighbouring page is being read or programmed [20][21], as shown in Figure 2(b). Similar to the retention error, disturb error is also not permanent. As a result, only program error in a page has a significant adverse influence on the wear-out of FG. The remaining two types of error can be reset after the page's residing block is erased, which have little influence on the wear-out of FG.

Existing wear leveling schemes, whether DWL or SWL, consider the number of P/E cycles and BER as the measurement for the wear-out pace that a block has suffered, and ignore the phenomenon we observed. However, based on **Observation 1**, the wear status of the block is clearly correlated with the number of P/E cycles. But under the same P/E cycling, all blocks in a NAND Flash memory suffer the different wear-out pace. Using P/E cycles as the measurement of block wear-out pace cannot estimate the wear status of blocks precisely. Based on **Observation 2**, NAND Flash memory has multiple error types. This suggests that the BER is insufficient to accurately measure how much a block is worn out. Thus, the wear status of the block is largely determined by the PER. The conventional wear leveling schemes based on the BER are imprecise and hard to maximize the overall NAND Flash memory endurance and lifetime. The two observations can be applicable for 2D and 3D NAND Flash memory.

The above discussion suggests that we should estimate the wear-out pace of blocks using other factors. Since the wear status of each flash block is reflected as its PER, wear leveling scheme should use PER as the equalization target and ideally aim to equalize the PER among all the blocks.

IV. Design

In this section, a program error rate-based wear leveling scheme (referred to as PER-WL) is proposed to perform wear leveling. We will first describe the whole architecture of PER-WL. Then, the procedures of our proposed design will be presented.

A. Overview

Figure 3 gives an overview of our proposed PER-WL design. As shown in Figure 3, the PER-WL is integrated to the FTL. The key idea behind PER-WL is to explicitly use the PER statistics of each flash block as the equalization target. It consists of two major components: program error handler and block manager. The program error handler intercepts write operations from the allocator and garbage collector, responsible for the PER statistics of each block during the write process. The block manager manages valid blocks according to its PER. When the write request arrives, the block manager is invoked by the allocator and allocates the valid blocks with the minimal PER to the write request. Furthermore, the block manager performs the block data swapping to further improve the wear leveling efficiency.

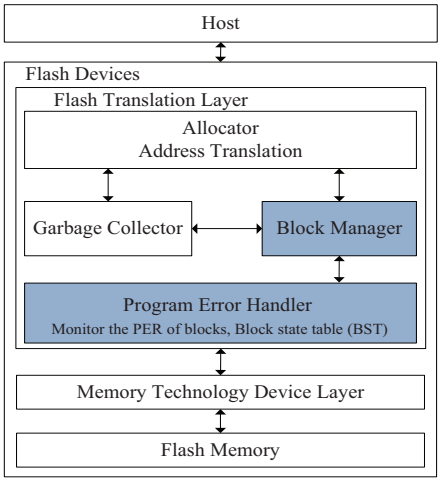

Fig. 3: Architecture of our proposed PER-WL design

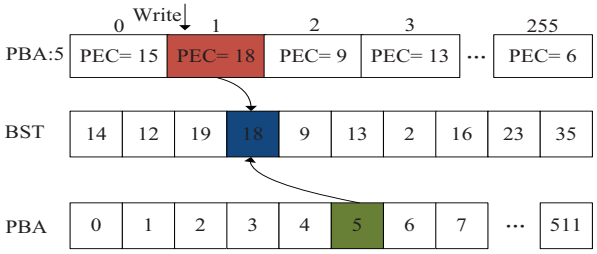

Fig. 4: The update process of block state table (BST)

B. Block State Table

The block state table (BST) is used to record the program error of all the blocks, which means the wear-out pace of blocks. Figure 4 shows the update process of BST. It is a counter array and each counter records the program error information of block. If a block has one worn-out page only, it cannot be used any further. As a result, the largest page program error bit count is regarded as the program error bit count of its residing block, which is recorded in the BST. For example, as shown in Figure 4, we assume that there are 512 blocks. Each block consists of 256 pages. The green part denotes the block whose physical block address (PBA) is 5. When the write request has arrived, the red part denotes the accessed page in the block (PBA: 5). The page program error bit count is 18 (PEC=18), which is the largest program error bit count in the block (PBA: 5). Thus, BST records the 18 as the block program error bit count (PBA: 5, the blue part).

C. Program Error Handler

The objective of program error handler is to intercept write operations and monitor the PER of blocks. When the allocator or garbage collector issues a write operation to the MTD driver, the program error handler is invoked. Algorithm 1 shows the algorithm of program error handler.

Algorithm 1 Program Error Handler

Require:

/*ppa is the address of the accessed page.

/*pba is the block address, which corresponds to this accessed page.

/*$data$ is the internal buffer of NAND Flash system.

/*BST is the block state table.

Ensure:

1: Write(ppa, $data$);
2: Read(ppa, $data$);
3: $per_{count} \leftarrow$ Program_error_Compare($data$);
4: **if** $per_{count} >$ BST[pba] **then**
5: BST[pba] $\leftarrow$ per_{count};
6: **end if**

When a write operation is received, the $data$ are directly written to the allocated pages (ppa) (Lines 1). After the pages are programmed, the written data are immediately read back from the pages to the internal buffer of NAND Flash system (Lines 2). This method can minimize the disturb error and retention error. Then, the program error handler starts to check the number of program error bits in the designated pages, compare the written data with the read data bit by bit, and obtain the program error bits of each page (Lines 3). If the per_{count} is larger than the per_{count} in the BST, the corresponding block pba is replaced with per_{count} (Lines 4-5).

D. Block Manager

Figure 5 shows the flow diagram of block manager. As shown in Figure 5, S denotes all the free blocks to be allocated, H denotes the hot block set and D denotes the dirty block set. When the allocator needs a free block, the block manager is invoked and allocates the valid block with lowest PER to the allocator. In order to further equalize the PER among all the blocks, the block manager performs block data swapping. If the allocated block in S is cold data, we swap the data of the block in S (cold data) with the data of the block in H (hot data) with high PER. If there is no enough free blocks, the block manager will choose a dirty block with minimal PER from D and erase this block. We use this method to equalize the wear-out pace of blocks.

V. Evaluation

A. Experimental Setup

To evaluate the efficacy of the proposed program error based wear leveling scheme (referred to as PER-WL), we developed an event-driven simulator using C programming language. To focus on emerging technology, we use 3D TLC NAND Flash

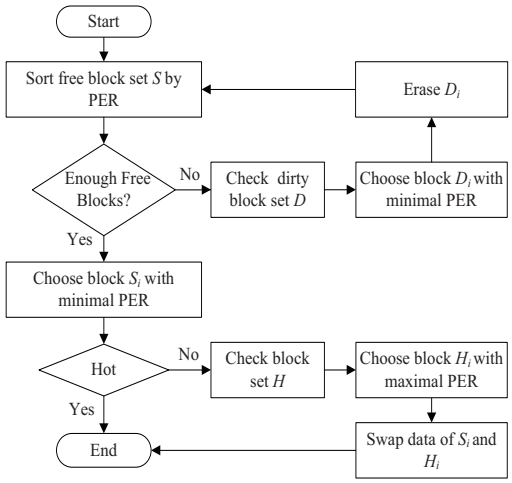

Fig. 5: The operational flow diagram of block manager

TABLE I: Simulation parameters

Flash Page Size	16KB
Flash Block Size	9MB (576 pages)
Page Read Latency	69us
Page Write Latency	2351us
Block Erase Latency	4.2ms
Over-provision Space	20%

memory "TOSHIBA TH58TFT1T22BA8H", whose parameters are shown as Table I. It is worth noting that the latency parameters are obtained from actual testing on real TOSHIBA TH58TFT1T22BA8H 3D TLC NAND Flash memory device. The program mechanism of 3D TLC NAND Flash memory is one shot program, which means program 3 pages at a time. In this experiment, we adopt page mapping scheme. The efficiency of PER-WL is compared with two state-of-the-art wear leveling schemes, DWL and SWL [8][10], which are designed to even the number of P/E cycles and BER among all the blocks, called DWL-P/E and SWL-BER.

We choose four realistic enterprise workloads. Financial1 and Financial2 were collected from OLTP applications running at two large financial institutions [25]. MSR-hm0 and MSR-proj0 are 1-week block I/O traces collected from enterprise servers at Microsoft Research Cambridge [26]. Table II presents the features of the workloads. Different traces have different sizes of logical address space. To maintain the original feature of workloads, we configure the user-visible capacity

TABLE II: Characteristics of workloads

Workload	Write Ratio	Avg. Req. Size	Address Space
Financial1	77.9%	3.5KB	512MB
Financial2	18%	2.4KB	512MB
MSR-hm0	64.49%	7.99KB	16GB
MSR-proj0	87.42%	38.04KB	16GB

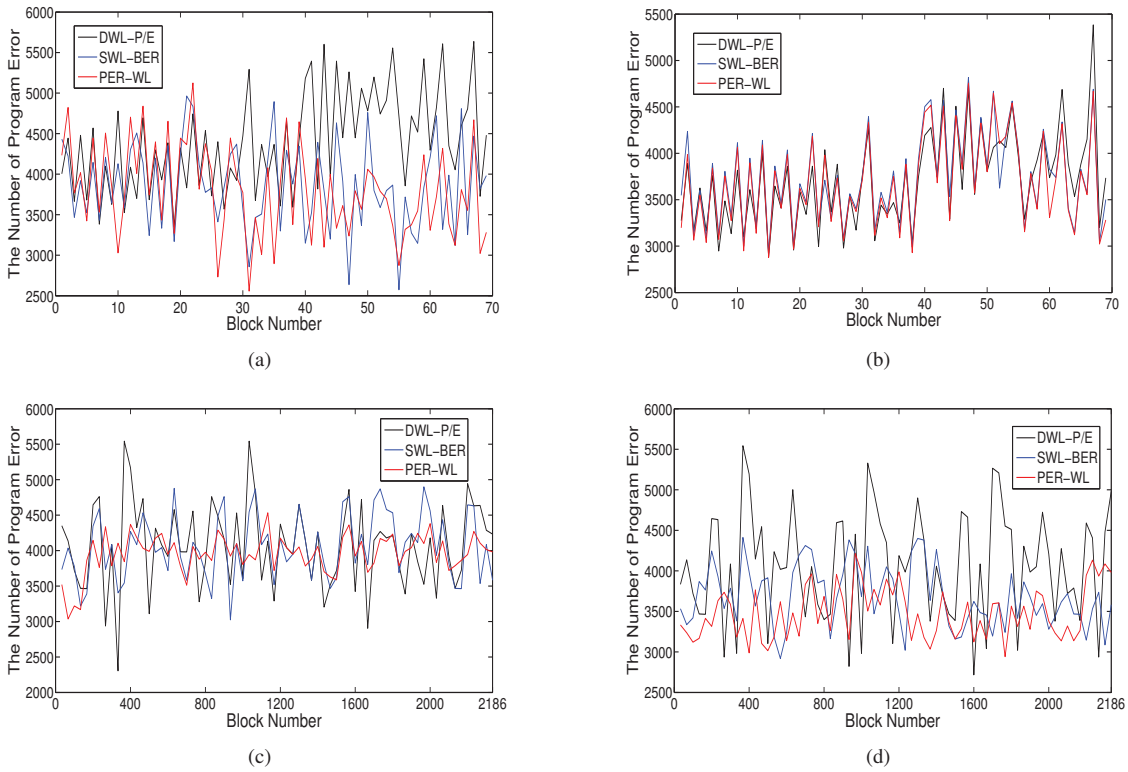

Fig. 6: (a). Financial1; (b). Financial2; (c). MSR-hm0; (d). MSR-proj0

of 3D TLC NAND Flash memory with 512MB in Financial workloads and 16GB in MSR workloads, respectively.

B. Experimental Results

1) Wear Leveling Efficiency: According to section III, the wear-out pace of NAND Flash memory is closely correlated with PER of blocks. Consequently, the efficiency of wear leveling scheme should be evaluated in terms of block PER instead of the number of P/E cycle and BER. Figure 6, Figure 7 and Figure 8 show the number of program error, normalized variance of PER among all the blocks and average program error rate when different wear leveling schemes are used, respectively. As shown in Figure 6, we can clearly observe the variety of each block PER. The variance of block PER is normalized against the case of DWL-P/E. The wear leveling scheme, which results in less variance of program error rate and average program error rate of all blocks, has a better efficiency. As show in Figure 7 and 8, PER-based wear leveling scheme can achieve 17% and 9% variance of program error rate reduction, 8% and 3% program error rate reduction when compared with DWL-P/E and SWL-BER wear leveling schemes on average. Furthermore, PER-WL can achieve 29% and 15% variance of program error rate reduction, 18% and 11% program error rate reduction when compared with DWL-P/E and SWL-BER in MSR-proj0 workload, which is the maximum improvement. Since Financial1, MSR-hm0 and MSR-proj0 workloads are write intensive, the effect of PER-based wear leveling is evident as shown in Figure 7 and 8. The results show that the proposed PER-based wear leveling scheme outperforms the DWL-P/E and SWL-BER. This is a reasonable result since PER-WL uses block PER as the design metric during wear leveling.

2) Performance Comparison: Figure 9 shows the normalized average system response time. The proposed PER-WL introduces more performance overheads than the DWL-P/E, around 5% system performance degradation on average. This is because PER-WL requires an extra read operations to verify program error bits after each write operation. Since Financial2 is read-intensive workload, the average system response time in Financial2 is much smaller than other workloads. Compared with SWL-BER, PER-WL only suffers around 2% system performance degradation on average. The results indicate that PER-WL can efficiently improve the wear leveling efficiency at low performance overhead. This PER-based wear leveling scheme can be a practical choice for storage system designers.

3) Memory Overhead for BST: The amount of memory required is proportional to the capacity of the NAND Flash memory system and the size of the counter stored for each physical block in the BST. For instance, the capacity of SSD based on NAND Flash memory is 512GB (each block is 9MB) and the number of physical blocks is approximately 58255, the memory required for the BST would be approximately 228KB (each counter = 4Bytes). In the industry, the capacity of SSD internal buffer is usually configured 1‰ of the SSD capacity. Thus, the capacity of SSD internal buffer is usually configured to 512M. Consequently, the memory overhead for BST is fairly

small for SSD internal buffer.

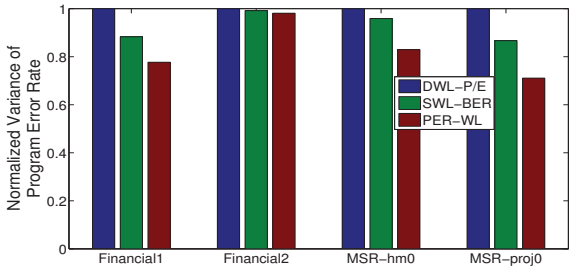

Fig. 7: Normalized variance of program error rate when different wear leveling schemes are used

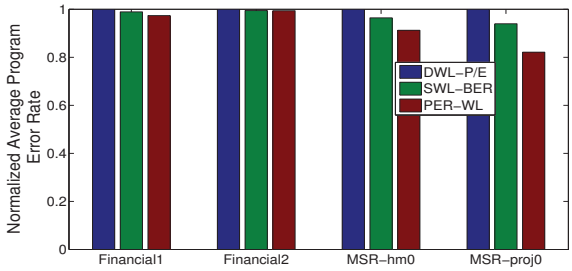

Fig. 8: Normalized average program error rate

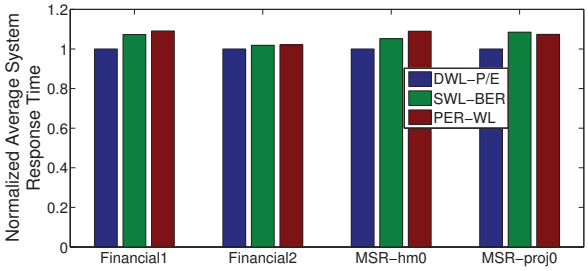

Fig. 9: Normalized average system response time

VI. Conclusion

Conventional NAND Flash memory wear leveling schemes explicitly use the P/E cycle and BER as the optimization target to even the wear among all the blocks. However, our analysis shows that these two design metrics are not precise. To efficiently improve the endurance of NAND Flash memory, this paper proposes PER-WL, a PER-based wear leveling scheme, to estimate the wear-out pace of each block more precisely and equalize the PER among all the blocks. PER-WL explicitly uses PER information of blocks as equalization target and performs block data swapping to further improve the wear leveling efficiency. Since the PER of blocks reflects wear-out pace of NAND Flash memory, wear leveling efficiency should be evaluated in terms of block PER. Based on a series of simulation experiments, the results show that PER-WL can reduce the variance of program error rate and program error rate by 17%, 9% and 8%, 3%, on average, respectively, when compared with two state-of-the-art wear leveling schemes. The average system performance degradation of PER-WL is 5% and 2% when compared with DWL-P/E and SWL-BER, respectively. In addition, the memory overhead for PER-WL is negligible.

Acknowledgment

This work is sponsored by the National Natural Science Foundation of China under Grant No. 61300047, No 61472152, No. 61572209, No. 61370063, and Wuhan Science and Technology Project No. 2017010201010108, and Shenzhen basic research project No. ICYJ20170307160135308, and the Fundamental Research Funds for the Central Universities No.2016YXMS019, and the 111 Project (No. B07038). This work is also supported by Key Laboratory of Data Storage System, Ministry of Education.

References

[1] J. Gray and B. Fitzgerald, "Flash disk opportunity for server applications," Queue, 6(4):18-23, 2008.
[2] Y. Lu, et al., "Extending the lifetime of flashbased storage through reducing write amplification from file systems," in Proc. of FAST, 2013.
[3] Y. Hu and D. Moore, "MLC vs. SLC NAND flash in embedded systems," Technical report, 2009.
[4] SpecTek, "64Gib TLC NAND Flash Features FNNB74A," 2011.
[5] E. Gal and S. Toledo, "Algorithms and data structures for Flash memories, " ACM Comput. Surv., vol. 37, no. 2, pp. 138-163, 2005.
[6] M.-C. Yang, et al., "Garbage collection and wear leveling for flash memory: Past and future," in Proc. of SMARTCOMP, 2014.
[7] Wu C H, et al., "Rethink the Design of Flash Translation Layers in a Component-based View," IEEE Access, vol.5, pp. 12895-12912, 2017.
[8] L.-P. Chang, et al., "An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems," in Proc. of RTAS, 2002.
[9] L.-P. Chang, et al., " Real-time garbage collection for Flash-memory storage systems of real-time embedded systems. ACM Trans," Embed. Comput. Syst., vol. 3, no. 4, pp. 837-863, 2004.
[10] Chang Y H, et al., "Endurance Enhancement of Flash-Memory Storage, Systems: An Efficient Static Wear Leveling Design," in Proc. of DAC, 2007.
[11] M. Murugan and David. H. C. Du, "Rejuvenator: A Static Wear Leveling Algorithm for NAND Flash Memory with Minimized Overhead," in Proc. of MSST, 2011.
[12] Liao J, et al., "Adaptive Wear-leveling in Flash-based Memory," IEEE Computer Architecture Letters, 14(1): 1-4, 2015.
[13] Pan Y et al., "Error Rate-Based Wear-Leveling for nand Flash Memory at Highly Scaled Technology Nodes," IEEE Transactions on Very Large Scale Integration Systems, 21(7): 1350-1354, 2013.
[14] Peleato B, et al., "BER-based wear leveling and bad block management for NAND flash," in Proc. of ICC, 2015.
[15] Yang M C, et al., "New ERA: New efficient reliability-aware wear leveling for endurance enhancement of flash storage devices," in Proc. of DAC, 2013.
[16] M. Dietrich and J. H. (Eds.), "Process Variations and Probabilistic Integrated Circuit Design," Springer, 2012.
[17] P. Cappelletti, et al., "Flash Memories," Kluwer Academic Publishers, 1999.
[18] Cai Y, et al., "Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," in Proc. of DATE, 2012.
[19] Cai Y, et al., "Data retention in MLC NAND flash memory: Characterization, optimization, and recovery," in Proc. of HPCA, 2015.
[20] Cai Y, et al., "Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation," in Proc. of ICCD, 2013.
[21] Cai Y, et al., "Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery," in Proc. of DSN, 2015.
[22] T.-S. Chung, et al., "A survey of flash translation layer," Journal of Systems Architecture, vol. 55, no. 5-6, pp. 332-343, 2009.
[23] A. Ben-Aroya and S. Toledo, "Competitive Analysis of Flash-Memory Algorithms," in Proc. of the Annual European Symposium, 2006.
[24] A. Olson and D. Langlois, "Solid state drives data reliability and lifetime," Imation White Paper, 2008.
[25] "UMass Trace Repository," http://traces.cs.umass.edu/index.php /Storage/Storage.
[26] "MSR Cambridge Traces," http://iotta.snia.org/traces/388.