Exploiting Process Variation for Retention Induced Refresh Minimization on Flash Memory

Yejia Di*, Liang Shi**, Kaijie Wu* and Chun Jason Xue†
*College of Computer Science, Chongqing University, Chongqing, China
† Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong

***Abstract*—Solid state drives (SSDs) are becoming the default storage medium with the cost dropping of NAND flash memory. However, the cost dropping driven by the density improvement and technology scaling would bring in new challenges. One challenge is the overwhelmingly decreasing retention time. The duration of time for which the data written in flash memory cells can be read reliably is called retention time. To deal with the decreasing retention time, refresh has been highly recommended. However, refresh will seriously hurt the performance and lifetime, especially at the end life of flash memory. The second challenge is the process variation (PV). Significant PV has been observed in flash memory, which introduces large variations in the endurance of flash blocks. Blocks with high-endurance can provide long retention time, while the retention time is short for low-endurance blocks.**

Considering these two challenges, a novel refresh minimization scheme is proposed for lifetime and performance improvement. The main idea of the proposed approach is to allocate high-endurance blocks to the data with long retention time requirement in priority. In this way, the refresh operations can be minimized. Implementation and analysis show that the overhead of the proposed work is negligible. Simulation results show that both the lifetime and performance are significantly improved over the state-of-the-art scheme.

I. Introduction

Flash memory has been widely used in embedded systems, personal computers, and data centers due to the benefit of cost dropping from technology scaling and density improvement. Currently, the technology has been scaled to 16nm [1][2] and the number of bits per cell has been advanced to 6 [3]. However, the technology scaling and density improvement bring in new challenges. The first one is the overwhelmingly decreasing retention time [4][5]. The duration of time for which the data written in flash memory cells can be read reliably is called retention time. If the retention time requirement of the data is larger than the supported retention time of the to-be programmed block, the data would be lost before they are updated. The second one is the process variation (PV) [6][7][8]. PV is a common feature of transistors, which has been identified in flash memory with significant impact on the endurance of flash blocks.

There have been many works devoted to the challenges. For the decreasing retention time, refresh is a straightforward scheme [9], which has been widely studied in the design of flash memory storage systems [10][11][12]. However, refresh brings in additional operations, which would impact the performance and lifetime of flash memory. Currently, refresh optimization techniques have been studied to reduce the system impact, similar to the research works in DRAM, Phase Change Memory (PCM) and many other storage memories [13][14][15]. Existing techniques can be classified into three types: The first type proposed to use different refresh frequencies for storage blocks with considering their specific endurance capability [10][15][14]. The second type proposed to adopt stronger reliability protection mechanisms to extend the endurance of blocks and eventually reduce the refresh frequency [11]. The last type proposed to recharge the storage blocks with a small voltage to recover the lost retention time [12]. However, there are no works considering the specific retention time requirement of data.

PV has been identified as the main factor in determining the endurance of flash blocks, which has a direct impact on the supported retention time. Currently, many researches proposed to understand the characteristics of the PV and exploit it for performance and lifetime improvement [8][6][7][16][17]. Jimenez *et al.* [8] found that the bit error rates (BER) of flash pages grow at different speeds, which are determined by the PV. Pan *et al.* [6] measured that different flash blocks have different BER under the same P/E cycles. In order to exactly identify the endurance of each block, Woo *et al.* [7] explored many attributions of blocks under the PV, including the P/E cycles, BER, program, read and erase latencies. Meza *et al.* [16] presented many measured cases in flash memory using the concepts of strong and weak blocks. These works show that the PV of flash memory can introduce significant endurance variations among flash blocks. Considering the above two challenges, if data with high retention time requirement are stored in the low-endurance blocks, frequent refresh operations would be required to keep the data reliably.

The objective of this work is to solve the above issue by taking both the specific endurance features under the PV and data retention time requirement into consideration. A novel refresh minimization (RM) scheme is proposed. The main idea is straightforward, which proposes to allocate high-endurance blocks for the data with high retention time requirement in priority. In order to realize RM scheme, two approaches are proposed: First, a faulty bit based PV identification approach is proposed to acquire the endurance of each flash block; Second, based on the detected endurance, a block allocation scheme is proposed to allocate high-endurance blocks to the data with high retention time requirement in priority. With this approach, the number of refresh operations can be minimized during the lifetime of flash memory, which would bring in significant improved performance and lifetime. In order to validate and analyze the effects of the proposed scheme, a set of simulations are carried out using several workload traces and the solid state drive (SSD) simulator [18].

The major contributions of this work are as follows:

- Proposed a refresh minimization (RM) scheme by exploiting the PV of flash memory with considering the retention time requirement of the data;

*Liang Shi is the corresponding author: shi.liang.hk@gmail.com.

978-3-9815370-6-2/DATE16/

- Proposed a PV identification approach to acquire the endurance of each flash block;
- Proposed a block allocation scheme, which allocates high-endurance blocks for the data with high retention time requirement in priority;
- Implemented a refresh queue in flash memory controller for refresh operation management.

The rest of the paper is organized as follows. In the next section, we describe the related works and the motivation of the proposed work. In Section III, we present the RM design in details. Section IV is the experiment and analysis. We conclude the paper in Section V.

II. Related Work and Motivation

A. Related Work

1) Refresh On Flash Memory: Refresh is a straightforward mechanism for the decreasing retention time [9] of flash memory. However, the key issue for the refresh mechanism is that it would bring in additional operations, which has drawback in performance and lifetime of flash memory. Cai *et al.* [12] observed that the retention errors are the dominant errors in flash memory. They proposed Flash Correct-and-Refresh (FCR) approach to reduce the number of refresh operations. Two techniques are proposed in FCR: The first one is hybrid FCR, which selectively applies remapping and reprogramming to reduce the number of refresh operations. The second one is adaptive FCR, which applies a low refresh frequency for the blocks with smaller retention errors. Park *et al.* [11] presented that strong protection schemes are able to reduce the refresh frequency, but enormous ECC space may be needed to cope with the worst case. The worse case is that the data with long retention time requirement are stored in the blocks with the high P/E cycles. It is expensive in performance and efficiency to apply strong protection schemes. Thus, they proposed a dynamic vertical stripping method to replace the strong ECC, which is low-overhead with reduced refresh frequency. Luo and Cai *et al.* [10] proposed WARM, a write-hotness aware retention management policy for flash memory. WARM is designed to identify and physically group write-hot data together within the flash memory. With this approach, the flash controller is allowed to selectively perform retention time relaxation with little cost.

2) Process Variation: PV is a natural characteristic of semiconductors. Currently, many works for the PV of flash memory are devoted to identify the specific characteristics on the endurance of flash blocks. Yeong *et al.* [7] proposed a new wear index, which took more diverse properties into account including erase counts and program latency, forming a linear model for measuring the wearing of flash blocks. Measured results show that the P/E cycles of the failed blocks are far beyond the guaranteed P/E cycles provided by the manufactories, and are not affected by the locations of the flash blocks. Besides, the actual values of P/E cycles differ from flash blocks to flash blocks. Yang *et al.* [6] presented a dynamic BER-based greedy wear-leveling algorithm that used BER statistics as the wear-out pace for flash blocks, and guided dynamic data swapping among flash blocks to fully maximize the efficiency. Jimenez *et al.* [8] observed that the BERs of pages in one block are significantly varied. This motivates them to reduce the stress on the weakest pages for endurance enhancement. What's more, Onur *et al.* [16] presented the first large-scale study of flash-based SSD reliability in this field.

Even though there are works on the PV and refresh reduction schemes respectively, the combination of them are still not considered. This work is the first time to combine the PV and retention induced refresh into consideration for refresh minimization. The impact on the supported retention time of flash blocks from the PV can be exploited to reduce the number of refresh operations.

B. Motivation

Motivation of the work comes from at least two aspects:

- It is worthy to do refresh reduction for lifetime and performance improvement;
- PV brings in significant endurance variations to the flash blocks. Allocating blocks based on the endurance of blocks is worthy for data with various retention time requirement.

1) Refresh is frequent and costly: Cai *et al.* [12] measured that when the ECC strength and the accepted raw BER are fixed, the lifetime can be extended by relaxing the required retention time. If the required retention time is 3 years, the lifetime provided by a 512-bit BCH code is only ∼3k P/E cycles. While if the required retention time is 3 days, the lifetime provided by the same code will be extended to ∼150k P/E cycles. 50-time lifetime improvement can be achieved with retention time relaxation. However, the refresh frequency at the end life of the flash memory will be much frequent. The supported retention time of each flash block in different periods is presented in Table I, which are collected from [12]. When the flash block has been erased with no more than $\sim 10^3$ P/E cycles, the retention time can be as long as several years. However, when the flash block has been erased with $\sim 10^5$ P/E cycles, the retention time will decrease to days.

TABLE I
Refresh frequency of each period of flash block.

$Periods(P/Ecycles)$	$\sim 10^3$	$\sim 10^4$	$\sim 10^5$
$Refresh\ Frequency$	$year$	$month$	day

Assume that, the same amount of data are updated everyday with updating probability α. Then, the utilization of flash memory after *t* days will be

$$u = (1-\alpha)^t \tag{1}$$

If the refresh frequency is *t*-day refresh, then the refresh rate per day will be:

$$Ref_{rate} = \frac{(1-\alpha)^t}{t} \tag{2}$$

When the update rate of the data and the refresh frequency are both low, the refresh rate will be horrible. For example, *t* has been identified that it is set to 3 at the end life of flash memory [12]. In this case, refresh operations would become the drawback of the lifetime and performance.

2) The endurance of flash blocks are significantly varied: Yang *et al.* [6] worked out that the P/E cycles of all the flash blocks fall into the range of [15000, 24600]. Woo *et al.* [7] also measured the P/E cycles of blocks. They presented that the lowest-endurance blocks can sustain 4999 cycles, and the average cycles are 8524 with the standard deviation of 1318. All the measured results show the following conclusions: First, the highest-endurance blocks can sustain ten-times thousands

P/E cycles. Second, the P/E cycles of the blocks with the lowest endurance are only thousands. Thus, the endurance among blocks is significantly varied.

With understanding the above basic characteristics, the proposed work is motivated. In order to reduce the number of refresh operations, we prefer to allocate the highest-endurance blocks for the coming data. However, it is not worthy to allocate the high-endurance blocks for the data with frequent updates. The reason is that the retention time requirement of these data are usually short, e.g. shorter than 1 hour commonly [19]. Furthermore, if the low-endurance blocks are allocated for the data with long retention time requirement, refresh will be activated frequently, especially at the end life of flash memory. High-endurance blocks can provide longer retention time, which is able to satisfy the data with long retention time requirement. We can vividly describe the relations between the retention time requirement of data and the endurance of blocks with the behavior of magnetic polarities, shown in Figure 1.

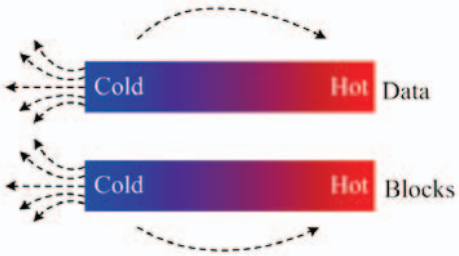


Fig. 1. **The retention time requirement of data and the endurance of blocks perform as magnetic polarity.** No matter in performance or reliability, blocks with higher endurance are always the attractive candidates for data [17][20]. We can regard blocks with higher endurance as hot blocks and vice versa. The data which require short retention time will be updated frequently called hot data, and the data with long retention time requirement are called cold data.

III. Exploiting PV for Retention Induced Refresh Minimization

A. Overview

Figure 2 shows the organization of the proposed scheme. There are three components implemented in the flash controller: (1) PV Identifier (PVI), (2) Block Allocator (BA), (3) Refresh Queueing (RQ). In addition to these components, the unidirectional and bidirectional arrows represent the relationships among different components. Besides, there is a time trigger for RQ, which is represented by a red button in the figure. First, PVI is designed to identify the endurance of each flash block. Second, BA is designed to recognize the retention time requirement of data and then allocate high-endurance blocks for the data with long retention time requirement in priority. Finally, RQ is designed to record the refresh information to manage the refresh in the flash memory.

B. PV Identifier

Exploiting the PV of flash memory is the key for the proposed work. The endurance of each flash block has direct impact to the decisions in the RQ and BA. In the following, the supported retention time of each flash block is first detected. Then, based on the detected retention time, an endurance queue is constructed, which will be used in the BA. Note that a high-endurance block is always able to support a longer retention time and vice-versa. Endurance and retention time will be used interchangeably in the following sections.

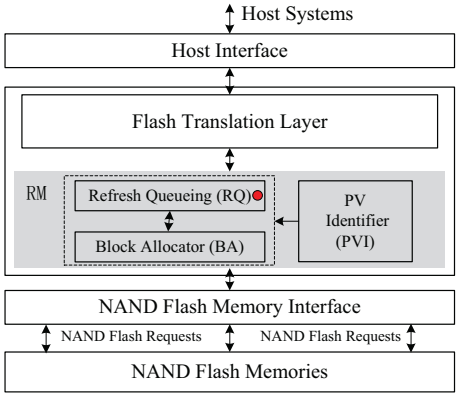


Fig. 2. Implementation of the RM scheme in the flash memory controller.

1) Supported Retention Time Detection Under the PV: The relationship among raw bit error rate (RBER), P/E cycles and retention time will be used to get the supported retention time of each flash block. In flash memory, RBER is the error rate corrected by Error Correcting Code (ECC) when the data are saved in the flash memory over a period of time under a specific number of P/E cycles. Cai *et al.* [4][12] presented that RBER = $f(c, d)$, where c is the number of P/E cycles and d is the retention time. Besides, Park *et al.* [11] presented the relationship function RBER(c, d) as follows:

$$RBER(c, d) = d_r(c) \cdot d \tag{3}$$

where $d_r(c)$ is the error deterioration rate per day when P/E cycles are c, and retention time is d. Based on Equation 3, the supported retention time of a block under a specific number of P/E cycles can be computed.

$$d = \frac{RBER_{th}}{d_r(c)} \tag{4}$$

where $RBER_{th}$ represents the largest RBER which ECC can correct up to.

Under the above basic model, the specific retention time of a block under the PV is detected as follows. The main idea is to check the number of faulty bits in the data pages. If the maximum number of faulty bits of all data pages in the block is within the capability of ECC, the retention time is supported and vice-versa. Let's assume that the most faulty bits that ECC can correct up to are H. In order to make sure that the number of faulty bits induced is always smaller than ECC capability, a faulty bit limitation threshold h is set, where $h < H$. The detection procedure is shown in Figure 3. Once the predefined retention time is reached, the pages of the block will be read out and then the maximum number of faulty bits can be gotten and recorded. Assume that the number of faulty bits of the page is k. If $k \leq h$, the retention time will be maintained. Otherwise, the retention time of the block will be recalculated by Equation 4, and the maximum number of faulty bits will be reset to 0.

The supported retention time of each block is recorded at the out-of-band (OOB) of the first page.

2) Endurance Queue: As shown in Figure 4, after detecting the retention time, the location information of the block will be recorded in one of the ranks in an endurance queue. The endurance queue is organized into ranks according to the retention time of blocks. The blocks in the rank 1 have the

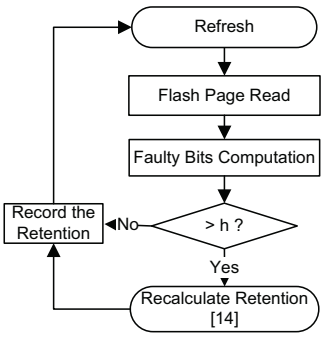


Fig. 3. The procedure of retention time detection for flash blocks.

highest endurance and rank N have the smallest endurance. Considering the endurance of flash blocks, refresh can be minimized.

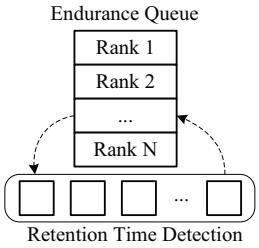


Fig. 4. Retention time detection and endurance queue in PVI.

C. Block Allocator

The main function of BA is to allocate blocks with high endurance to the data with long retention time requirement in priority. In this way, refresh can be further minimized.

In order to realize the BA, the retention time requirement of data should be firstly recognized. Retention time requirement of data can be represented by the hotness of the data. A long retention time requirement always means that the data are cold and not updated for a long time. We will use cold to represent the long retention requirement and hot for short retention requirement. There have been many mature works on identifying hotness of data. In this work, for simplicity, size of request data is used as the metric [21]. Small size of data requested is identified as hot data and large size of data is identified as cold data. This knowledge is consistent with the common access characteristics of systems.

Under the hotness of data, BA will allocate blocks for the data according to the endurance queue of PVI. Once the data are recognized as cold, blocks are allocated from rank 1 of endurance queue, which can support the longest retention time. Then, the blocks with high endurance will be allocated to cold data first. For the hot data, BA will leave alone blocks with high endurance and remain them for cold data as enough as possible. Here, we set a threshold t to divide blocks for hot and cold data allocation, where rank 1 to t are blocks with high endurance for cold data, and rank t+1 to N are blocks with low endurance for hot data.

There is a shortage for RM that putting hot data to blocks with low endurance will speed up the wearing of the blocks, so that the overall lifetime of flash memory may be decreased. While RM gives a benefit to blocks with low endurance is reduced refresh frequency for cold data. Thus, if the reduced refresh frequency is larger than the update frequency of hot data, RM is worthy. Besides, with technology scaling and density improvement, flash memory is declined to the low-cost storage, performance is more important.

D. Refresh Queueing

In this section, a detailed implementation for refresh queue in flash memory controller is presented. The retention time requirement of the data stored in the flash blocks can be larger than the supported retention time. In order to solve this issue, refresh queue is required to refresh the data before they are lost. Two records are added in the RQ to realize a correct refresh, including the data location and the time to refresh, called expired time.

Once the data are programmed into flash memory, the corresponding information are recorded. In this work, the proposed work adopts to record the refresh information in the unit of flash block. With regard to the updates of RQ, there are three states: insert into RQ, stay at RQ, and refresh out of RQ.

1) Insert into RQ: Data will be allocated to blocks decided by BA, as shown in prior section. Before they are programmed, we can get the supported retention time of the blocks from the OOB of the first page. Then, the expired time of the page can be gotten and it will be recorded in the OOB of the page. Once there is a valid page in a block, the refresh information of the block will be put into RQ immediately, as shown in Figure 5. The expired time of the block is the expired time of the first valid page. RQ is a time-increasing-order queue. The time trigger will be set by the expired time of the first refresh information in RQ.

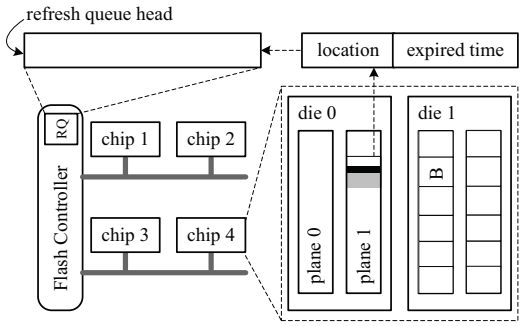


Fig. 5. **Refresh queue implementation in flash memory controller.** The architecture of SSD includes six levels: channels, chips, dies, planes, blocks, and pages. B represents a block in plane. For the block in plane 0 of die 0, the gray part represents invalid pages and the black part represents the first valid page.

2) Stay at RQ: Before a block is refreshed, data in the valid pages of the block may be updated, especially for the hot data. If the updated data are at the first valid page of the block, the expired time of the block will be updated to the expired time of the second valid page, and RQ will be updated for its time-increasing order. It is possible that no valid pages are in the block any more, then the refresh information of the block will be evicted from RQ. Otherwise, if the updated data are at any valid page instead of the first valid page, there are nothing to do for RQ.

3) Refresh out of RQ: When time trigger is activated, refresh will start from the head of RQ. All valid pages in the refresh block will be refreshed at one time, which is reasonable

for in-order programming principle of blocks and the adjacent expired time of pages.

E. Overhead Analysis

The memory and storage overheads of the proposed scheme are analyzed as follows. According to the detailed descriptions of components above, there are only memory and storage overhead for RQ and PVI components. In RQ, a refresh queue is implemented, which stores the refresh information for each flash block. Refresh information includes the block location and the expired time. Assume that there are 8 channels, 8 chips per channel, 4 dies per chip, 4 planes per die, and 2048 blocks per plane in flash memory. Thus, 21 bits are needed to represent the location of a block. For the expired time, 64-bit is enough. In total, each refresh node needs 85-bit storage space, and there are at most 21MB storage space for the refresh information of the whole storage systems. In fact, this implementation is the common configuration for flash memory. The space overhead should not be the overhead of the proposed work. For PVI, endurance queue holds the location of all blocks. As described above, there is 21-bit storage space for the location record of a block. Thus there are only 5MB storage space for endurance queue, which is negligible.

IV. Experiments and Analysis

A. Methodology

In this section, we use a trace driven simulator, SSDsim [18], to verify the effects of the proposed work. The proposed work is evaluated with 9 workload traces on the simulator. These traces have been widely used in many prior works [17][11][10] and all of them are random workloads, mixed with hot and cold data. And the operation latencies are set according to Liu *et al.* [22]. We add the PVI, BA and RQ to the simulator. In order to simulate the PV of flash memory, our work uses a simple normal distribution of the P/E cycles of blocks, ranging at [15000, 24000], similar to Shi *et al.* [17]. Retention time is detected by the value of P/E cycles of flash blocks and endurance queue rank is also decided by the range of P/E cycles, which is set to 12 here. To divide blocks for hot and cold data allocation, we simply adopt the middle rank. The refresh queue is implemented as described in Section III-D.

The workload traces are all from MSR-Cambridge [23], and the tracing time is 7 days. In order to trigger refresh operations, we use linear extrapolation to shorten the retention time of blocks according to Table I. Besides, to measure the effects on lifetime, we assume that the trace is repeated until the flash drive is worn-out. Two important results will presented and analyzed in this section:

1) How many will the number of refresh operations be decreased with the proposed work. How much will the performance be improved;
2) What effects does the proposed work have on the lifetime of flash memory.

The experiment platforms respectively are most recent work FCR [12], PVI only identifying PV, and RM the proposed work.

B. Experiment Results

1) Effects on Performance: The proposed work is proposed to minimize the number of refresh operations. A key to the proposed work is to classify the data into hot and cold data. Chang *et al.* [21] proved that the distribution of writes with respect to their size is bimodal: Most of the writes are either large or small. Small writes are prone to hot, and large writes seldom touch a piece of data more than once. Here, we take workload HM_0 trace as an example. The distribution of writes are measured as shown in Figure 6. For small writes, 16-sector requests are the dominate part. 72-sector requests are large writes for HM_0. In this way, hot data and cold data can be classified for all workloads.

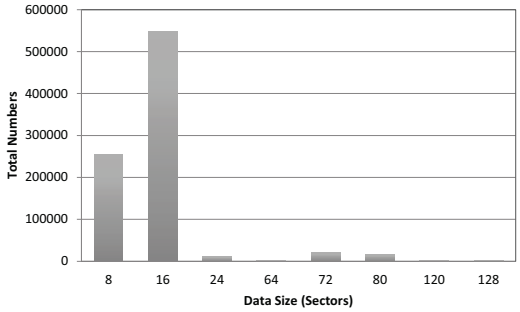


Fig. 6. Distribution of total writes for each data size.

The number of refresh operations is evaluated to show the effectiveness of the proposed scheme. As shown in 7, the number of refresh operations is decreased up to 70%. The significant improvement comes from two benefits: First, cold data will be put into blocks with high endurance so that they can be stored in place as long as possible, the number of refresh operations will naturally be reduced; Second, the data owing similar updating frequency will be put together into one block, which is also a benefit as introduced in Luo *et al.* [10].

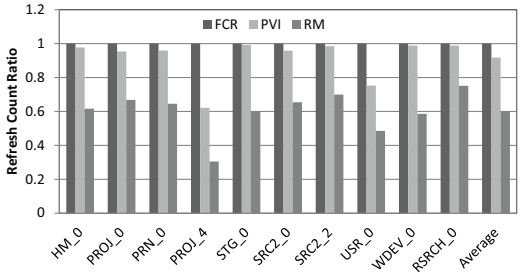
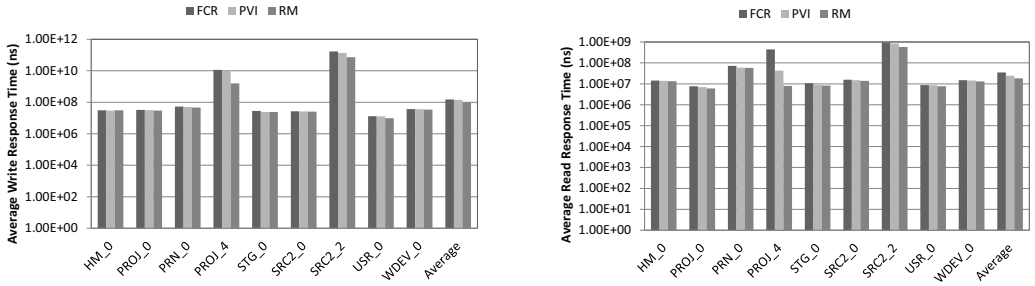


Fig. 7. **The number of refresh comparison among FCR, PVI and RM schemes under different workload traces.** FCR is the baseline, and the values of y-axis represent the ratio.

One of the goal of reducing the number of refresh operations is to improve the performance. The average write and read response time are also measured, and the effects on improving response time are impressive. As shown in Figure 8, the average response time is reduced significantly compared to FCR scheme, 27.9% and 25.4% improvement for write and read performance.

2) Effects on Lifetime: We measured the effects on lifetime at the end life of flash memory. The lifetime is reflected by the time when any block is worn-out. Figure 9 shows the effects on lifetime comparing among FCR, PVI and RM. As shown in Figure 9, we can find that the lifetime has been improved by several times. RM scheme will not only improve the performance of flash memory, but also bring in benefits for lifetime at the end life of flash memory. The reason comes from several aspects: First, exploiting PV can bring in new space on wearing. Second, the reduction of refresh operations



(a) Average write response time. (b) Average read response time.

Fig. 8. **Average write and read response time comparison among FCR, PVI and RM schemes.**

is the main reason to the lifetime improvement. Third, as described above, the data will be divided into respective blocks according to update frequency, which is also a benefit.

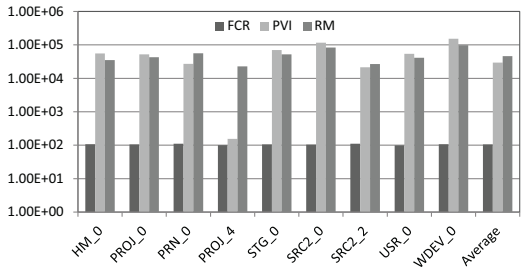


Fig. 9. **Lifetime comparison among FCR, PVI and RM schemes under different workload traces.** The values of y-axis represents the ratio, FCR is the baseline. For worn-out measurement, the experiment at the end life of flash memory is evaluated.

V. Conclusion

With technique scaling and density increasing, the decreasing retention time and significant PV are becoming more and more serious. Refresh is a straightforward strategy for decreasing retention time, but the accompanying additional operations will hurt the performance and lifetime of flash memory. Our work proposes the RM scheme to reduce the retention induced refresh with exploiting the PV. In this work, we first identified PV of flash memory so that we can get the endurance degree and retention time of blocks. Then, blocks with high endurance are allocated to the cold data in priority. Besides, we implemented a refresh queue in flash memory controller. Implementation analysis and experiment results show that the overhead is negligible. Series of simulations were evaluated and from the experiment results, performance and lifetime of flash memory all be significantly improved.

VI. Acknowledgement

This work is partially supported by the Fundamental Research Funds for the Central Universities (106112014CDJZR185502), NSFC 61402059, NSFC 61472052, NSFC 61173014, National 863 Program 2013AA013202, 2015AA015304, Chongqing High-Tech Research Program cstc2012ggC40005, cstc2014yykfB40007.

References

[1] M. Helm, J.-K. Park, A. Ghalam, J. GUO, C. wan Ha, C. Hu, H. Kim, K. Kavalipurapu, E. Lee, A. Mohammadzadeh, D. Nguyen, V. Patel, T. Pekny, B. Saiki, D. Song, J. Tsai, V. Viajedor, L. Vu, T. Wong, J. H. Yun, R. Ghodsi, A. d'Alessandro, D. Di Cicco, and V. Moschiano, "19.1 a 128gb mlc nand-flash device using 16nm planar cell," in *ISSCC '14*, Feb 2014, pp. 326–327.

[2] S. Choi, D. Kim, S. Choi, B. Kim, S. Jung, K. Chun, N. Kim, W. Lee, T. Shin, H. Jin, H. Cho, S. Ahn, Y. Hong, I. Yang, B. Kim, P. Yoo, Y. Jung, J. Lee, J. Shin, T. Kim, K. Park, and J. Kim, "19.2 a 93.4mm2 64gb mlc nand-flash memory with 16nm cmos technology," in *ISSCC '14*, Feb 2014, pp. 328–329.

[3] K.-C. Ho, P.-C. Fang, H.-P. Li, C.-Y. Wang, and H.-C. Chang, "A 45nm 6b/cell charge-trapping flash memory using ldpc-based ecc and drift-immune soft-sensing engine," in *ISSCC '13*, Feb 2013, pp. 222–223.

[4] Y. Cai, E. Haratsch, O. Mutlu, and K. Mai, "Error patterns in mlc nand flash memory: Measurement, characterization, and analysis," in *DATE '12*, March 2012, pp. 521–526.

[5] Y. Cai, Y. Luo, E. Haratsch, K. Mai, and O. Mutlu, "Data retention in mlc nand flash memory: Characterization, optimization, and recovery," in *HPCA '15*, Feb 2015, pp. 551–563.

[6] Y. Pan, G. Dong, and T. Zhang, "Error rate-based wear-leveling for nand flash memory at highly scaled technology nodes," *TVLSI '13*, vol. 21, no. 7, pp. 1350–1354, July 2013.

[7] Y.-J. Woo and J.-S. Kim, "Diversifying wear index for mlc nand flash memory to extend the lifetime of ssds," in *EMSOFT '13*, Sept 2013, pp. 1–10.

[8] X. Jimenez, D. Novo, and P. Ienne, "Wear unleveling: Improving nand flash lifetime by balancing page endurance," in *FAST '14*, Santa Clara, CA, 2014, pp. 47–59.

[9] H. So and S. Wong, "Multi-bit-per-cell flash eeprom memory with refresh," Nov. 21 2000, uS Patent 6,151,246.

[10] Y. Luo, Y. Cai, S. Ghose, J. Choi, and O. Mutlu, "Warm: Improving nand flash memory lifetime with write-hotness aware retention management," in *MSST '15*, May 2015, pp. 1–14.

[11] H. Park, J. Kim, J. Choi, D. Lee, and S. H. Noh, "Incremental redundancy to reduce data retention errors in flash-based ssds," in *MSST '15*, May 2015, pp. 1–13.

[12] Y. Cai, G. Yalcin, O. Mutlu, E. Haratsch, A. Cristal, O. Unsal, and K. Mai, "Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime," in *ICCD '12*, Sept 2012, pp. 94–101.

[13] T. J. Schwarz, Q. Xin, E. L. Miller, D. D. Long, A. Hospodor, and S. Ng, "Disk scrubbing in large archival storage systems," in *MASCOTS '04*. IEEE, 2004, pp. 409–418.

[14] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian, and V. Srinivasan, "Efficient scrub mechanisms for error-prone emerging memories," in *HPCA '12*, Feb 2012, pp. 1–12.

[15] I. Bhati, Z. Chishti, S. Lu, and B. Jacob, "Flexible auto-refresh: enabling scalable and energy-efficient DRAM refresh reductions," in *ISCA '15*, 2015, pp. 235–246.

[16] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "A large-scale study of flash memory failures in the field," *SIGMETRICS '15*, vol. 43, no. 1, pp. 177–190, Jun. 2015.

[17] L. Shi, Y. Di, M. Zhao, C. Xue, K. Wu, and E.-. Sha, "Exploiting process variation for write performance improvement on nand flash memory storage systems," *TVLSI '15*, vol. PP, no. 99, pp. 1–1, 2015.

[18] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, "Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity," in *ICS '11*, 2011, pp. 96–107.

[19] L. Shi, K. Wu, M. Zhao, D. Liu, J. Xue, and E. Sha, "Retention trimming for lifetime improvement of flash memory storage systems," *TCAD '15*, vol. PP, no. 99, pp. 1–1, 2015.

[20] 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 *HPCA '12*, Feb 2012, pp. 1–10.

[21] L.-P. Chang, "Hybrid solid-state disks: combining heterogeneous nand flash in large ssds," in *ASPDAC '08*. IEEE, 2008, pp. 428–433.

[22] R.-S. Liu, C.-L. Yang, and W. Wu, "Optimizing nand flash-based ssds via retention relaxation," *Target*, vol. 11, no. 10, p. 00, 2012.

[23] "Snia: Iotta reponsitory, http://iotta.snia.org/tracetypes/3."