Flash Read Disturb Management Using Adaptive Cell Bit-Density with In-Place Reprogramming

Tai-Chou Wu, Yu-Ping Ma, and Li-Pin Chang
Department of Computer Science, National Chiao-Tung University
Hsinchu, Taiwan, R.O.C.
sm811001.cs04g@g2.nctu.edu.tw, myp497238@gmail.com, lpchang@cs.nctu.edu.tw

***Abstract*—Read disturbance is a circuit-level noise induced by flash read operations. Read refreshing employs data migration to prevent read disturbance from corrupting useful data. However, it costs frequent block erasure under read-intensive workloads. Inspired by software-controlled cell bit-density, we propose to reserve selected threshold voltage levels as guard levels to extend the tolerance of read disturbance. Blocks with guard levels have a low cell bit-density, but they can store frequently read data without frequent read refreshing. We further propose to convert a high-density block into a low-density one using in-place reprogramming to reduce the need for data migration. Our approach reduced the number of blocks erased due to read refreshing by up to 85% and the average read response time by up to 22%.**

I. Introduction

High-level flash cells, such as MLC, TLC, and QLC, are a key technology to boost the storage density per unit cost in NAND-flash based storage devices. High-level cells are prone to various types of noises, including read disturbance, write disturbance, and retention errors. Among these noises, read disturbance is predicted to be a major source of bit errors under read-intensive workloads [2]. In NAND flash, reading a page in a block generates disturbance to all the other pages in the same block. This effect, called *read disturbance*, creates uncorrectable bit errors in a block after many read operations. An SLC block can endure about 1 millions of page reads operations without experiencing uncorrectable bit errors [6].

Because high-level cells use many threshold voltage levels to represent multiple logical bits, their tolerance margin between threshold voltage levels is narrow. Typically, an MLC block endures 100K page read operations, while a TLC endures only 10K [6], [7]. *Read refreshing* is a technique to prevent read disturbance from corrupting useful data. It migrates valid data from a block to another when the residual tolerance of read disturbance of the source block becomes critical [4]. To prepare free space for the data migration, flash management software conducts a lengthy process (called garbage collection), which involves a series of data copying and block erasure. We observed that read refreshing contributed to up to 99% of block erase operations under read-intensive workloads, and prior work also reported that read refreshing increased the worst-case read response time to 66 times as high as the block erasure latency [6].

This work is supported by research grant MOST 104-2221-E-009-011-MY3 from Ministry of Science and Technology, Taiwan, R.O.C.

To reduce the negative impacts of read refreshing, prior studies proposed to store a few frequently-read (read-hot) data exclusively in selected blocks [6], [9]. However, because the space utilization of these selected blocks is low, the storage of read-hot data considerably increases the space requirement. Inspired by software-controlled cell bit-density [8], we propose to reserve selected threshold voltage levels as guard levels. Guard levels are deliberately left unused to extend the tolerance margin. A block with guard levels is a low-density block because it loses the most significant cell bits, but its tolerance of read disturbance is at least ten times as high as that of a block without guard levels (a high-density block).

We propose to identify read-hot data and migrate them to low-density blocks to reduce the need for read refreshing. To further reduce the amount of data migration for read refreshing, based on the physical property that threshold voltages can be reprogrammed toward a higher level, we propose to create guard levels through in-place reprogramming. This way, a high-density block can be converted to a low-density block with reduced data copy operations. The proposed design faces several challenges: Firstly, our approach migrates read-hot data to low-density blocks to fully utilize the extended read tolerance of these blocks. This process requires high-resolution identification of read-hot data with a low space overhead. Secondly, our approach employs an in-place re-programming algorithm to convert a block into a low-density one. The re-programming algorithm must comply with the physical property of NAND flash programming. Thirdly, in-place reprogramming induces extra wear in flash memory, which must be considered by wear leveling in flash memory.

We conducted a series of experiments on the proposed and prior approaches. Compared to prior approaches, our approach reduced the frequency of block erasure due to read refreshing by up to 85%, and it also improved average read response time by up to 22%.

II. Background

A. Flash Read Disturbance

A piece of flash memory is composed by thousands of blocks, each of which consists of a number of word lines. A word line is connected to a set of flash cells. Figure 1(a) shows a typical interconnection among flash cells of a block. Flash reads in terms of word lines. To read from flash, a reference voltage (Vref or Vr) is applied to the word line to

978-3-9819263-0-9/DATE18/©2018 EDAA

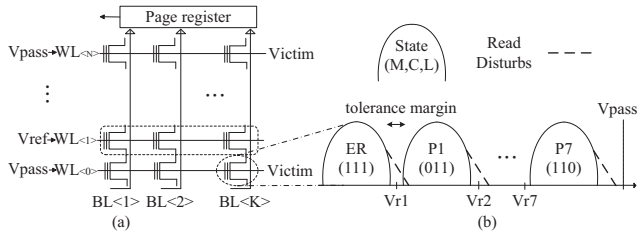

Figure 1. (a) The internal organization of flash cells. (b) Threshold voltage levels and reference voltages of an TLC cell.

be read, and a high read pass voltage (Vpass) is applied to all the other word lines (victims) of the same block. Figure 1(b) shows that a triple-level cell (TLC) uses eight threshold voltage levels to store three logical bits. For the cells on the same word line, the logical bits of the same bit significance form a Most-Significant-Bit (MSB) page, a Center-Significant-Bit (CSB) page, and a Least-Significant Bit (LSB) page. The logical bits of a page can be determined using binary search based on voltage comparison between cell threshold voltages and reference voltages.

Applying a high Vpass to victim word lines may unexpectedly add a shift to cell threshold voltages through electron injection. As Figure 1(b) shows, there is a gap between two adjacent threshold voltage levels, called the tolerance margin. The tolerance margin diminishes as read disturbance amplifies the voltage shift, and eventually the logical bit values cannot be correctly determined by voltage comparison[1]. Read disturbance is a crucial reliability issue of TLC flash because 1) TLC reading requires a high Vpass, which increases the chance of electron injection [5], [7], 2) the tolerance margin of TLC flash is narrow due to many threshold voltage levels, and 3) advanced error-correcting codes such as LDPC exaggerate read disturbance through read retries [11].

Flash storage employs a piece of firmware called Flash Translation Layer (FTL) for address mapping and garbage collection. Read refreshing and host writing both demand free space in flash. Let garbage collection for host writing and that for read refreshing be denoted by wGC and rrGC, respectively. In this study, we assume that a TLC block endures 10K read operations before read disturbance creates uncorrectable bit errors [6], [7]. This number is also called the *read-count limit* of a block. Figure 2 shows the erase counts contributed by wGC and rrGC in a page-mapping FTL [3] under a selection of realistic workloads. Under five out of the nine workloads, rrGC contributed to more than one half of the total erase counts. The results strongly suggest that read refreshing critically impacts on flash lifespan.

B. Related Work

There have been a series of studies on various flash circuit-level noises, including program disturbance [1], read disturbance [1], [2], and retention errors [1]. Among these noises, read disturbance is predicted to critically impact on the reliability of high-level-cell flash memory [5]. Ha et al. [7] observed that read disturbance is proportional to Vpass and the duration of applying Vpass. They proposed to write read-hot data using narrow threshold voltage levels so that read-hot data can be read using a low Vpass within a short time duration. However, this approach requires a high write precision and degrades the write performance.

[1]Low Density Parity Check (LDPC) allows a limited Vref overflow of cell threshold voltages [11]. Conceptually the overflow limit is a part of the tolerance margin.

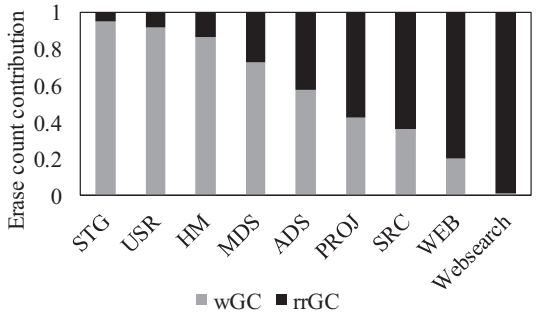

Figure 2. Contributions of erase counts from garbage collection for host writing (wGC) and that for read refreshing (rrGC).

Software-based techniques have also been proposed to prevent read disturbance from corrupting existing data. Ha et al. [6] proposed RedFTL, which reserves a portion of blocks as shadow blocks. RedFTL exclusively stores a few (i.e., 30) read-hot pages in each shadow block. This design improves the long tail read latency caused by data migration for read refreshing. Liu et al. [9] proposed Read Leveling, which also employs shadow blocks. Because unused pages are immune to read disturbance, read-hot pages cycle through all unused pages in a shadow block to virtually increase the block read-count limit. However, both RedFTL and Read Leveling poorly utilize the storage space in shadow blocks, significantly limits their efficacy of response time improvement and read refreshing reduction. Differently, we propose to exploit the non-uniform read-disturbance tolerance provided by adaptive cell bit-density.

III. Read Disturbance Management

A. Overview

Figure 3 shows the three new components for our purpose: A *read-hot data identifier*, which captures read-hot pages so that the FTL can migrate these pages to read-hot blocks, which are an intermediate step before low-density blocks. A *cell bit-density manager*, which decides when to lower or to restore the cell bit-density of a block. When a read-hot block reaches its read-count limit, it is converted to a low-density one through in-place reprogramming. On the other hand, to manage the total capacity of flash storage, it reverts selected low-density blocks to high-density ones. A *block allocator*, which decides the allocation of blocks to read-hot data to manage the extra flash wear induced by in-place reprogramming.

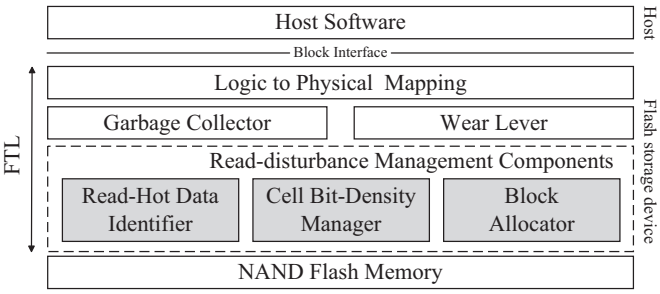

Figure 3. An overview of our read-disturbance management approach.

B. Read-Hot Data Identification

Low-density blocks are dedicated to read-hot pages. Read-hot pages must be identified before they are migrated to low-density blocks. It would be too costly to monitor the read hotness of every page. Instead, we propose a two-level method for low-cost, accurate identification of read-hot pages: Every block is associated with a state, which is either regular, monitor, or read-hot. Initially, all blocks are regular (high-density) ones. Figure 4 shows the migration of page data among these three types of blocks. When a regular block undergoes read refreshing, if its most recent read-refreshing interval is shorter than the average read-refreshing interval (B_{avg}), then the block contains some read-hot pages. The FTL allocates a new block, copies valid page data form the regular block to the new block, and starts to monitor the read count of every page in the new block. The new block is a monitor block. To avoid a large space overhead of storing the page-level read counters, our current design monitors up to 10% of all blocks.

If a monitor block needs read refreshing, then it is time to extract read-hot pages from the block. The FTL calculates the average read count of pages in the block, copies the pages whose read counts are higher than the average (P_{avg}) to a read-hot block, and copies the rest to a regular block. After this, a read-hot block will contain read-hot pages only. There is no quantity limit on read-hot blocks. If a block (regular, monitor, or read-hot) is erased due to garbage collection instead of read refreshing, then it may not contain many read-hot pages. In this case, valid pages are migrated from the block to a regular block, and the source block is erased into a regular block.

C. Cell Bit-Density Management

Figure 5 shows the eight threshold voltage levels of a TLC flash cell. The basic idea of density reduction is to reserve selected levels, i.e., P2, P4, and P6, as guard levels. Guard levels are deliberately left unused, and they become a part of tolerance margin. With guard levels, a flash cells stores 2 bits instead of 3. For example, to program 01 into a cell, the FTL sequentially writes 1 and 0 into the LSB and CSB, respectively. Because P2(001) and P3(101) both represent 01 as their two least-significant bits, the FTL further programs 1 into the MSB to use P3(101) only. This way, P2(001) becomes a guard level between P1(011) and P3(101). A block

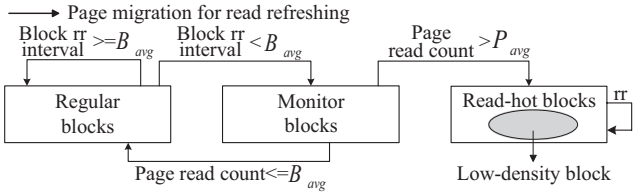

Figure 4. Migration of page data among regular, monitor, and read-hot blocks. "rr" stands for read refreshing.

with guard levels can use existing Vrefs on read. Specifically, voltage comparison against Vr5 determines the LSB of a cell, and further comparison against Vr3 or Vr7 determines the CSB. Notice that a TLC block with guard levels is different from a MLC block because it requires the MSB programming to explicitly create guard levels.

If a read-hot block needs read refreshing, we propose to migrate read-hot pages from the block to new blocks with guard levels. In this study, we assume that the read-count limit of a TLC block is 10K [6]. Based on the measurements reported in [2], we predicted that the read-count limit of a TLC block with guard levels is 100K. A block with guard levels is a low-density one because its MSB pages are not available. By this design, read refreshing on a read-hot block requires up to $\frac{3}{2}$ new blocks.

Inspired by the property that cells can be reprogrammed to increase their threshold voltages, we further propose an in-place reprogramming (IPR) algorithm to create guard levels of a block in an in-place manner. As indicated by the arrows in Figure 5, IPR shifts selected threshold voltages to their right neighboring levels to create guard levels. For example, if the threshold voltage is in P4(100), then IPR reprograms the MSB with 0 to shift the threshold voltage to P5(000)[2].

IPR creates guard levels without affecting LSB and CSB. However, a backup of MSB must be created because it is lost after IPR. Figure 6(a) shows three read-hot blocks, B1, B2, and B3, all requiring read refreshing. For simplicity, let a block contain three pages, i.e., an MSB, a CSB, and an LSB page, and let each page contain one bit only. Firstly, the FTL copies the MSB pages C, F, and I to a new high-density read-hot block B4. Secondly, for each of B1, B2, and B3, the FTL reads its MSB, CSB, and LSB page, computes new page data using the rule in Figure 6(b), and then reprograms (overwrites) its MSB page using the new page data. IPR creates guard levels

[2]The IPR design is based on the basic physical property of NAND flash programming. Vendor-specific optimizations could affect the behavior of in-place reprogramming. Readers are referred to [10] for more details.

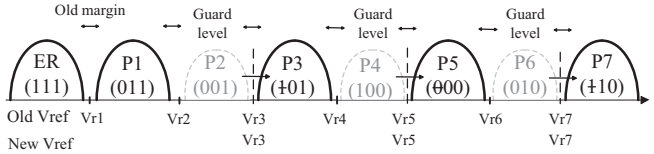

Figure 5. A low-density block, which uses P2, P4, and P6 as guard levels. The leftmost ER needs not be a guard level.

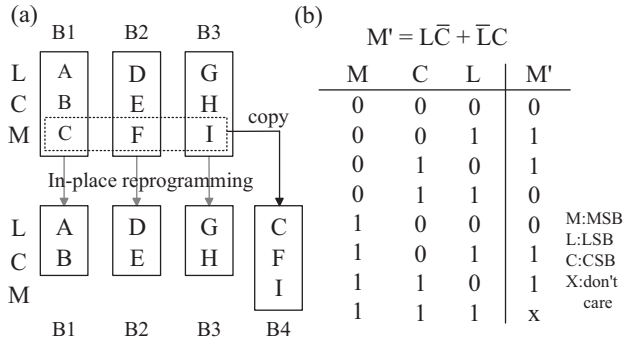

Figure 6. (a) IPR converts three high-density, read-hot blocks into low-density blocks. (b) The rule of producing new data for IPR to reprogram MSB.

in blocks with a reduced amount of data migration. It helps reduce the pressure of read refreshing on garbage collection and save precious flash Program/Erase (P/E) cycles. Our IPR design can involve one read-hot block at a time, and on average it requires only $\frac{1}{3}$ new blocks to refresh a read-hot block.

D. IPR-Aware Block Allocation

Our approach may add an IPR operation to a regular P/E cycle. We are interested in 1) how much extra wear an IPR operation induces and 2) the correlation between the extra wear and how often a block uses IPR. Now, let the IPR rate of a block be how many IPR operations the block undergoes per P/E cycle. If the IPR rate of a block is 0.2, then on average every fifth P/E cycle of the block contains an IPR operation. Let X_0 and X_r be the total number of P/E cycles a block endures without any IPR operation and that of a block with an IPR rate=r, respectively. On a block whose IPR rate is r, the normalize flash wear induced by a program-IPR-erase (P/IPR/E) cycle is calculated by $\frac{X_0-X_r}{X_r r}+1$. Figure 7 shows the normalized flash wear of a P/IPR/E cycle under different block IPR rates. This test was conducted using Samsung NAND flash. Interestingly, the higher the IPR rate is, the less extra flash wear an IPR operation induces. When the block IPR rate is not lower than 0.2, the normalized wear of a P/IPR/E cycle is 1.2. In other words, IPR should be concentrated in selected blocks instead of being dispersed among all blocks.

We propose a new block allocation strategy for IPR operation concentration. As shown in Figure 8, the global pool contains all blocks, and the IPR pool is a physical segment of flash memory that occupies 20% of the highest addresses. To handle write requests, the FTL allocates new blocks from the global pool through garbage collection. To copy read-hot pages for read refreshing, the FTL allocates a new read-hot block from the IPR pool through garbage collection as well, but skipping the following: 1) low-density blocks, 2) read-hot blocks, and 3) blocks recently involved in wear leveling. If this attempt fails, the FTL allocates a read-hot block from the global pool. To increase the IPR rate of blocks, only read-hot blocks in the IPR pool can use IPR. We propose to operate conventional wear leveling on *effective* P/E cycle counts: If a block undergoes a P/E cycle with IPR, then its effective P/E

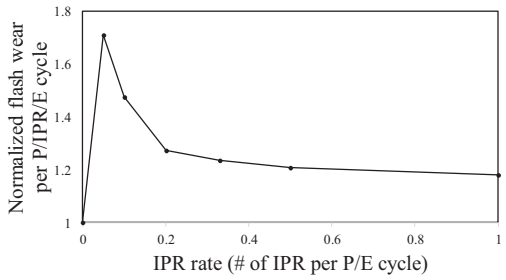

Figure 7. The normalized flash wear per program-IPR-erase (P/IPR/E) cycle on blocks of different IPR rates.

cycle count is incremented by 1.2. Otherwise, it is incremented by 1.

Writing low-density blocks decreases the space utilization of flash memory and increases the pressure on garbage collection. In our current design, the total number of low-density blocks occupy no more than 15% of all blocks, which contribute to three-fourths of the IPR pool size. When this limit is reached, the FTL erases a low-density block into a high-density regular block. The victim block has the smallest reading rate, which is the average number of page reads per unit of time since the last IPR on the block.

E. Overhead Analysis

The IPR approach stores several types of information in the embedded RAM of flash storage. It maintains the following information for every block: a read counter, a timestamp of the most recent read refreshing, and a time stamp of the most recent IPR. The read counter triggers read refreshing on a block, the read-freshing timestamp helps promote page data from regular blocks to monitor blocks, and the IPR timestamp is considered during the reclamation of low-density blocks. All these information are block-level ones, except the page read counters of monitor blocks. However, the amount of monitor blocks is small, no more than 10% of all blocks. Overall, with a 128 GB flash memory, the aforementioned page and block information uses only about 1.37 MB of embedded RAM.

IV. Experimental Results

A. Experimental Setup and Performance Metrics

We conducted a series of experiments on an SSD simulator, which was based on the SSD extension of DiskSim from Microsoft Research. Table I shows our experimental parameters. Our SSD simulator employed Stable Greedy [3] for garbage collection and wear leveling. The experiments involved the following methods: **Baseline**, which simply migrates all valid

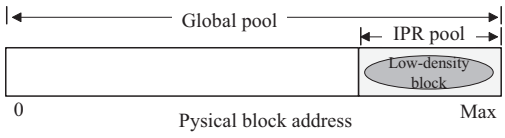

Figure 8. The FTL copies read-hot data to the IPR pool whenever possible. Only blocks in the IPR pool can use IPR for density reduction.

Table I
EXPERIMENTAL PARAMETERS

Flash Setting	Value	FTL Setting	Value
Page Size	32KB	Over-provision	20%
Pages per Block	192	Monitor Blocks	10%
Page Read Latency	100μs	IPR Pool Size	20%
Page Write Latency	1600μs	Low-density Blocks	15%
Block Erasure Latency	5000μs		
High-density R. limit	10000	-RedFTL and RL	
Low-density R. limit	100000	Shadow Blocks	15%

pages from a block to a new block when the source block reaches its read-count limit. **RedFTL** [6] and **RL** (Read Leveling) [9], which have been discussed in Section II-B. **HotLD** (Hot-Low-Density), which is our proposed approach without in-place reprogramming. It copies read-hot pages from a monitor block to new low-density blocks to refresh the monitor block. **IPR**, which is our HotLD plus in-place reprogramming. IPR is the fully-fledged version of our approach.

Table II shows our experimental workloads, which were from the trace repositories of SNIA IOTTA and UMass. All these workloads were read-intensive. Our experiments involved the following metrics: 1) Erase counts contributed by garbage collection for host writing (wGC) and those for read refreshing (rrGC). Our IPR should reduce the erase count of rrGC while not affecting that of wGC. 2) Read response time. Our IPR should prevent read refreshing from incurring long blocking time on read requests. 3) The final distribution of block P/E cycle counts. Our IPR should achieve an even distribution. During experiments, the workloads were replayed until wGC and rrGC contributed to stable fractions of the total erase count.

B. Read-Induced Garbage Collection Overhead

Figure 9 (a) shows the breakdown of wGC and rrGC erase counts under all workloads. The results are normalized to those of Baseline. rrGC contributed to a large portion of the total erase count under the MDS, ADS, PROJ, SRC, WEB and Websearch workloads. This is because under these workloads, read requests were highly focused on a few small disk regions and they quickly accumulated many read counts on a few flash blocks. Our IPR significantly outperformed all the other methods in terms of rrGC erase counts. In particular, compared to Baseline, IPR reduced the rrGC erase count by 85% under the SRC workload, reducing the total erase count and extending the SSD lifespan by 57%. This dramatic improvement achieved by IPR can be explained using Figure 9 (b), which shows the cumulative distribution of used read counts of low-density blocks upon their erasure in IPR. Recall that the read-count limit of low-density blocks is 100K, ten times higher than that of high-density blocks. Under the SRC workload, only 13% of the erased low-density blocks had utilized less than 89% of its extended read-count limit. In other words, low-density blocks successfully absorbed a large number of page reads before being erased.

As previously described in Section III-C, HotLD requires more new blocks to refresh a monitor block than IPR does ($\frac{3}{2}$ vs. $\frac{1}{3}$). Because HotLD imposed a higher pressure on garbage collection, it used more rrGC erase counts than IPR did. The only exception is that HotLD and IPR performed nearly the same under the ADS workload, whose read refreshing was highly concentrated on a very small set of low-density blocks. This can be confirmed in Figure 9 (b) that under the ADS workload, every erased low-density block had fully utilized its extended read-count limit.

Table II
CHARACTERISTICS OF EXPERIMENTAL WORKLOADS

Workload	Source	Description	Volume Size	Read (%)	Read/Write traffic(GB)
STG	MSR	Web staging	107G	81	123 / 28
USR	MSR	Home directories	569G	82	710 / 93
HM	MSR	Hardware monitoring	27G	94	25 / 1.4
MDS	MSR	Media server	509G	96	140 / 5.5
ADS	MSR	Display ads platform	101G	96	88 / 2.2
PROJ	MSR	Project directories	236G	98	354 / 4.2
SRC	MSR	Source control	182G	98	56 / 0.6
WEB	MSR	Web server	182G	99	454 / 2.2
Websearch	UMass	Search engine	17G	$\simeq$100	41 / $\simeq$0

Both RedFTL and RL exclusively stored a few read-hot pages in each shadow block. However, this basic design does not improve the read-count limit of shadow blocks, and therefore RedFTL and Baseline had nearly the same rrGC erase counts. Differently, RL delayed refreshing a shadow block by copying read-hot pages to unused pages in the block, virtually increasing the read-count limit of the block. In our experiments, RL reused a shadow block about five times before erasing it. In other words, however, the average space utilization of a shadow block was only about $\frac{1}{5+1}$=16.7%. Due to the poor space utilization, only limited read-hot pages were admitted to shadow blocks, and RL only moderately improved upon RedFTL and Baseline in terms of rrGC erase counts.

C. Read Response Time and Wear Leveling

Figure 10 shows the distribution of read response times under the Websearch workload. It shows *only* the read response times involving read refreshing. If the response time of a read request fell between 250 and 350 ms, then the request was blocked by conventional read refreshing on a high-density block. Compared to Baseline, IPR significantly decreased the request count in this time interval by 52%. This is because low-density blocks absorbed a large number of page reads with reduced read refreshing. RedFTL successfully shifted many read requests to the response time intervals below 250 ms. This is because RedFTL stored only 30 read-hot pages in a shadow block, and it migrated 30 rather than up to 192 pages to refresh a TLC shadow block. Now, considering all read requests, we observed that IPR improved upon Baseline in terms of the average read response time by 22% (from 261 μs to 206 μs).

As discussed in Section III-D, a program-IPR-erase cycle and a program-erase cycle induce different wear in a block. In our experiments, Stable Greedy [3] operated on the effective P/E cycle counts for wear leveling. Figure 11 use black and gray dots to represent the real and effective P/E cycle counts

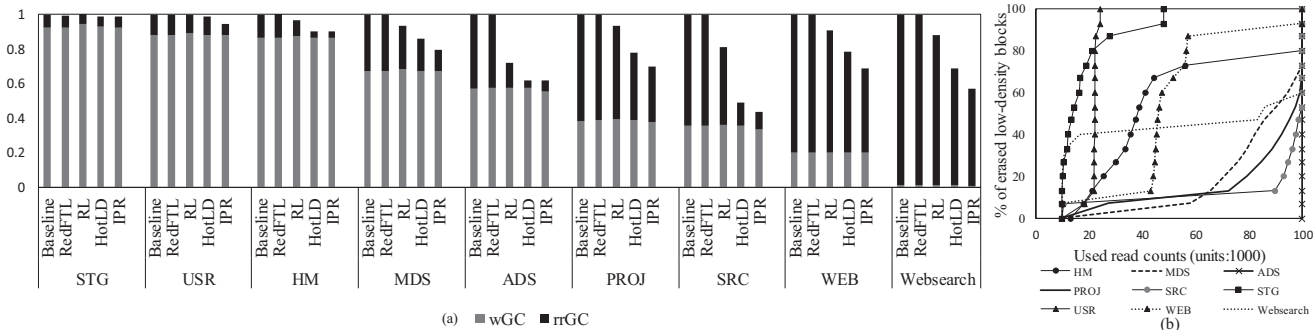

Figure 9. (a) Breakdowns of erase counts contributed by garbage collection for write requests (wGC) and read refreshing (rrGC). (b) Cumulative distribution of used read counts of low-density blocks upon their erasure in IPR.

of blocks under the Websearch workload, respectively. Only blocks in the IPR pool can use IPR so their real and effective P/E cycle counts were different. The effective counts in the IPR pool and the real counts outside of the IPR pool formed a highly even distribution. The extra wear of IPR on a block can be mitigated if the IPR rate of the block is larger than 0.2, and this goal was successfully achieved by our block allocation policy under the rrGC-dominant workloads (0.21, 0.35, 0.34, 0.78, 0.81, and 0.95 under MDS, ADS, PROJ, SRC, WEB, and Websearch, respectively). We also observed that under the rrGC-dominant workloads, the chance that our reprogramming skipped a read-hot block because the block is not in the IPR pool was not higher than 6%. The only exception is the Websearch workload, under which the chance was 60% because the amount of read-hot data was much larger than the IPR pool size.

V. Conclusion

As the cell bit-density increases, read disturbance is becoming a crucial concern of flash reliability. While prior approaches disperse read-hot data among blocks to reduce the negative impacts of read refreshing, we exploit the non-uniform tolerance of read disturbance through adaptive cell bit-density. We propose to reserve selected threshold voltage levels as guard levels to increase the tolerance margin at a cost of reduced storage density. We further propose to downgrade the cell bit-density of a block through in-place reprogramming without corrupting existing data in the block. Experimental results show that, compared to existing designs, our approach reduced the total number of blocks erased for read refreshing by up to 85% and improved the average read response time by up to 22%.

References

[1] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai. Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis. In *DATE*, 2012.

[2] Y. Cai, Y. Luo, S. Ghose, and O. Mutlu. Read disturb errors in MLC NAND flash memory: Characterization, mitigation, and recovery. In *DSN*, 2015.

[3] L.-P. Chang, Y.-S. Liu, and W.-H. Lin. Stable greedy: Adaptive garbage collection for durable page-mapping multichannel SSDs. *ACM Transactions on Embedded Computing Systems (TECS)*, 2016.

[4] H. Frost, C. Camp, T. Fisher, J. Fuxa, and L. Shelton. Efficient reduction of read disturb errors in NAND flash memory, 2010. US Patent 7,818,525.

[5] L. M. Grupp, J. D. Davis, and S. Swanson. The bleak future of NAND flash memory. In *USENIX Conf. on FAST*, 2012.

[6] K. Ha, J. Jeong, and J. Kim. A read-disturb management technique for high-density NAND flash memory. In *Proc. of the APSys*, 2013.

[7] K. Ha, J. Jeong, and J. Kim. An integrated approach for managing read disturbs in high-density NAND flash memory. *IEEE TCAD*, 2016.

[8] X. Jimenez, D. Novo, and P. Ienne. Libra: Software-controlled cell bit-density to balance wear in NANDflash. *ACM Trans. Embed. Comput. Syst. (TECS)*, 2015.

[9] C.-Y. Liu, Y.-M. Chang, and Y.-H. Chang. Read leveling for flash storage systems. In *Proc. ACM Int. Syst. Stor. Conf*, 2015.

[10] F. Margaglia, G. Yadgar, E. Yaakobi, Y. Li, A. Schuster, and A. Brinkmann. The devil is in the details: Implementing flash page reuse with WOM codes. In *USENIX Conf. on FAST*, 2016.

[11] K. Zhao, W. Zhao, H. Sun, T. Zhang, X. Zhang, and N. Zheng. LDPC-in-SSD: making advanced error correction codes work effectively in solid state drives. In *USENIX Conf. on FAST*, 2013.

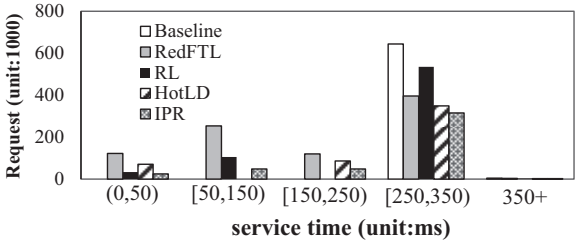

Figure 10. Response time distribution of read requests that triggered read refreshing under the Websearch workload.

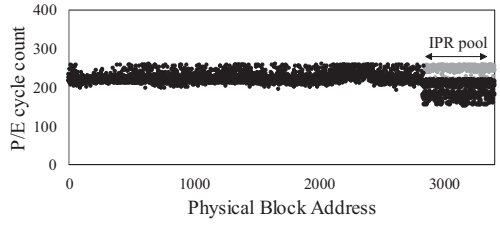

Figure 11. Final distribution of real (black) and effective (gray) P/E cycle counts of blocks under the Websearch workload.