DA-RAID-5: A Disturb Aware Data Protection Technique for NAND Flash Storage Systems

Jie Guo[1], Wujie Wen[1], Yaojun Zhang Li[1], Sicheng Li[2], Hai Li[1], Yiran Chen[1]
[1]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, 15261, USA
[2]Polytechnic Institute of New York University, Brooklyn, NY 11201
$\{jig26, wuw2, yaz24, hal66, yic52\}$@pitt.edu, $sli16$@students.poly.edu

***Abstract*—Program disturb, read disturb and retention time limit are three major reasons accounting for the bit errors in NAND flash memory. The adoption of multi-level cell (MLC) technology and technology scaling further aggravates this reliability issue by narrowing threshold voltage noise margins and introducing larger device variations. Besides implementing error correction code (ECC) in NAND flash modules, RAID-5 are often deployed at system level to protect the data integrity of NAND flash storage systems (NFSS), however, with significant performance degradation. In this work, we propose a technique called "DA-RAID-5" to improve the performance of the enterprise NFSS under RAID-5 protection without harming its reliability (here DA stands for "disturb aware"). Three schemes, namely, unbound-disturb limiting (UDL), PE-aware RAID-5 and Hybrid Caching(HC) are proposed to protect the NFSS at the different stages of its lifetime. The experimental results show that compared to the best prior work, DA-RAID-5 can improve the NFSS response time by 9.7% on average.**

I. Introduction

The success of NAND flash technology in consumer market has been recently extended to the enterprise storage market, thanks to its advantages on cost, power consumption, random access performance, and mechanical robustness. However, the inherent reliability issues, e.g., the component-level bit error, emerge as the biggest concern in such applications. Previous works [1], [2], [3] reveled that cell-level program disturb, read disturb and retention time limit are three major contributors to the NAND flash bit errors. The application of multi-level cell (MLC) technology imposes NAND flash to higher reliability risk due to the scaled threshold voltage noise margin [4].

RAID-5 scheme, combined with Error Correction Coding (ECC) is employed in NAND flash based system (NFSS) to protect the data integrity, i.e., implementing redundant parity blocks to recover the ECC-uncorrectable pages. However, the parity accesses in RAID-5 significantly degrade the system performance,especially for the workloads with intensive random write access. Some works [5], [6] have been proposed to minimize the performance overheads incurred by parity updates. However, their schemes ignore the impacts of cell-level read disturb and retention time limit.

In this work, based on the NAND flash bit error characteristics, we propose DA-RAID-5 technique to improve RAID-5 performance in enterprise NFSS. We first quantitatively study the NAND flash cell reliability degradation trend over its whole lifetime, by considering program disturb, read disturb, retention time limit and their dependency on P/E cycles. Based on the analysis, the lifetime of the NFSS is then divided into three stages. Different schemes are applied to tackle with the bit errors of the NAND flash cells at each stage: At the 1st stage, we apply UDL (unbound-disturb limiting) scheme instead of RAID-5 to handle the data corruptions incurred by read disturb and retention-time-limit beside general ECC protection; At the 2nd stage, PE-aware RAID-5 is introduced to protect the unreliable blocks with high P/E cycles; At the 3rd stage, RAID-5 is applied to all physical blocks and HC (hybrid caching) is adopted to enhance the system response time. By minimize parity access, our techniques can effectively improve performance and extend NFSS lifetime. Our experimental results show that compared to the best prior work [7], DA-RAID-5 technique can improve the average NFSS response time by 9.7%, with very marginal increase of access overhead for very few workloads.

II. Preliminary

A. NAND Flash Bit Error Basics

Program disturb, read disturb and retention time limit are three major factors affecting the bit error rate of NAND flash cells. Program disturb is mainly generated from the cell-to-cell interference and random telegraph noise (RTN) [8]. The disturb magnitude heavily depends on the P/E cycle of the NAND flash cells and the program disturb induced bit error rate rises as the P/E cycle increases [9].

Besides program disturb, read disturb is another major factor responsible for the threshold voltage distortion of the NAND flash cells. Read accesses to a NAND flash cell generate stress-induced leakage current (SILC), incurring threshold voltage gain [3]. Read disturb usually leads to larger threshold voltage shift on the unread pages in the same block than the page being read [10]. Both read count and P/E cycling determine the effect of read disturb. Research in [9], [11] shows that the increases in read count and P/E cycling cause higher read disturb induced bit error rate.

Retention time limit also leads to the threshold voltage distortion of the NAND flash cells. Following the gradual charge loss induced by the interface trap recovery and electron detrapping effect [12], the threshold voltage of the NAND flash cell decreases over time. Besides P/E cycle, data retention time

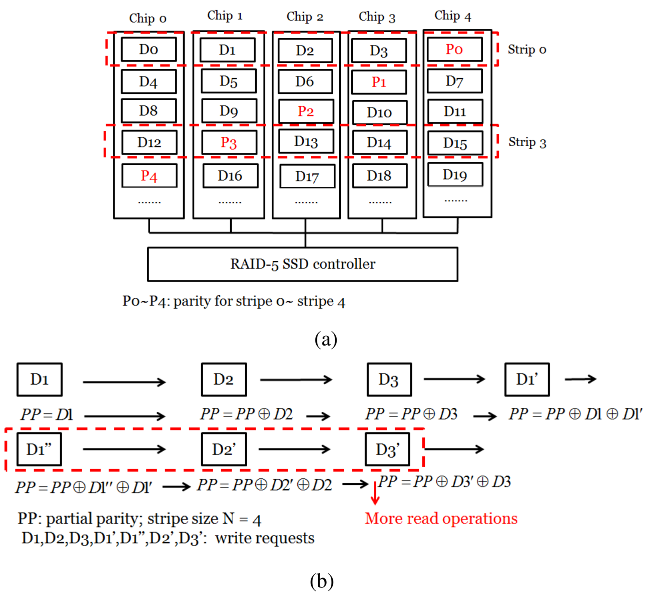


Fig. 1. (a) N+1 RAID-5 architecture; (b) Partial parity generation in PPC.

limit is also affected by read count because threshold voltage increase incurred by read operation can offset the charge loss induced by detrapping effect.

B. FTL and RAID-5 for NFSS

Flash Translation Layer is an essential part of NFSS. It translates the logic page number (LPN) issued by the host to the physical page number (PPN) in NAND flash chips. It is also responsible for garbage collection and wear-leveling due the out-of-place update and limited endurance characteristics of NAND flash [13], [14]. Conventional RAID-5 is based on the logic pages. The NFSS are grouped by stripes. Each stripe includes N sequential logic pages and a parity, which are sequentially stored in N+1 flash chips. A RAID-5 NFSS with 4+1 devices is shown in Fig. 1(a). Updating a logic page requires recalculating its parity page. We assume N is 4 and use $D_0 \sim D_3$ to denote the 4 sequential logic pages in a stripe. D_p is the parity page. When updating page D_0, the new parity page D_p' can be calculated by

$$D_p' = D_p \bigoplus D_0 \bigoplus D_0'. \tag{1}$$

Here, D_0' is the newly updated page of D_0. $\bigoplus$ represents XOR operation. However, if more than half of the pages in a stripe are updated together, the parity page is calculated differently: for example, assume the pages $D_0 \sim D_2$ are respectively updated to $D_0' \sim D_2'$, new parity page D_p' is generated as:

$$D_p' = D_0' \bigoplus D_1' \bigoplus D_2' \bigoplus D_3. \tag{2}$$

As shown in Eq. (1), updating a logic page involves up to two read and two write operations, which degrades the performance of NFSS and accelerates the wear-out of NAND flash cells. In [5], Y. Lee et al. proposed a parity commit delay approach which reschedules the updated parity commits to the NFSS idle time. However, this scheme may fail to recover the corrupted data when its parity is not committed in time. In [6], Lee et al. dynamically adjust the stripe size as the P/E cycle increases to minimize the parity update cost. However, the impacts of read disturb and retention time limit are neglected in this work. In [7], S. Im et al. proposed partial parity cache technique (PPC) to reduce parity update overhead. Parity is generated incrementally and buffered in a cache. When the cache is full or the logic pages of the parity are garbage-recycled, the full parity page is generated from the least-recently-updated (LRU) partial parities and flushed to the NAND flash. An example of partial parity update is shown in Fig. 1(b). Partial parity is generated by using the page being updated and its old version instead of all the pages in the same stripe. By reducing the read accesses to the old pages in the same stripe, PPC can significantly improve the NFSS response time. However, when more than half of the pages in a stripe are being updated together (updating $D1'$, $D2'$ and $D3'$ in Fig. 1(b)), PPC technique generates unnecessary read operations to the old version of the updated pages. Similarly, if a page is updated frequently, its old version will be read repeatedly, causing a longer NFSS response time.

In this work, DA-RAID-5 technique is introduced to improve the performance of the NFSS under RAID-5 protection, with the consideration of all three physical limiting factors of NAND flash cells errors and their P/E cycles.

III. DA-RAID-5 Technique

In this section, we first introduce the reliability analysis on NAND flash cells. We then present an overview of DA-RAID-5, followed by the three schemes to provide the data integrity protection at the different lifetime stages of NFSS.

A. Reliability Analysis

n-bit BCH ECC (m,l,c) is often used in NAND flash module design. The yield of each 512 bytes data with n-bit BCH ECC $Y_{ber}(n)$ can be expressed as:

$$Y_{ber}(n) = \sum_{k=0}^{n} C_m^k p_{epc}^k (1 - p_{epc})^{(m-k)}. \tag{3}$$

Here, p_{epc} is the error rate of single NAND flash cell. As mentioned in Section II-A, $Y_{ber}(n)$ and p_{epc} are affected by program disturb, read disturb and retention time, as well as the P/E cycle. Fig. 2(a) and 2(b) depict our simulated maximum tolerable read count T_{read} and retention time $T_{retention}$ when the P/E cycles $N_{p/e}$ varies, by using the reliability model in [8], [3]. Here we assume 8-bit ECC (4200, 4096,104) is applied to a 512 byte data and the target $Y_{ber}(x)$ is 99.9%. We also exclude the impacts of other factors during the simulation of each factor, e.,g, setting read count to 0 in Fig. 2(b). Our simulation results clearly show the degraded tolerance on T_{read} and $T_{retention}$ follows the increase in P/E cycles.

B. Overview of DA-RAID-5

Based on our reliability analysis, we proposed to use different schemes to protect the data integrity of NFSS at its different lifetime stages, as illustrated in Fig. 3:

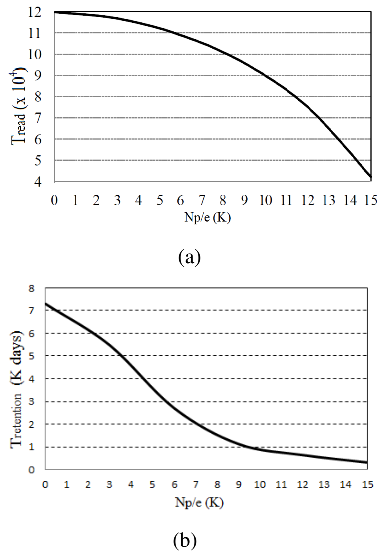


Fig. 2. (a) relationship between $N_{p/e}$ and T_{read}; (b) relationship between $N_{p/e}$ and $T_{retention}$.

Stage 1: Unbound-disturb limiting (UDL) scheme is designed for the 1st lifetime stage of NFSS. The controller monitors the period for which the data stored in the NAND flash block periodically. If the period is close to or longer than $T_{retention}$, the NAND flash block will be recycled. Similarly, we use a read counter to record the number of the NAND flash block being read. If the read count is greater than a certain threshold T_{read}, the block will be also recycled. Obviously, the runtime cost of UDL is significantly lower than that of RAID-5.

Stage 2: The increase of P/E cycle shortens the $T_{retention}$ and T_{read} of NAND flash blocks, causing more frequent block recycling under UDL. If there are any blocks whose P/E cycle exceed a certain number T_p, the whole NFSS enters its 2nd lifetime stage. We start to apply RAID-5 to the stripes whose pages have the P/E cycle higher than T_p while maintain the other stripes still under the protection of UDL. We call this scheme as PE-aware RAID-5. A PAE counter is assigned to each stripe to record the number of the pages with the P/E cycle higher than T_p. Parity calculation is performed to only the stripe with no-zero PAE counter.

Stage 3: When the number of the stripes applied with RAID-5 exceeds some threshold L, keeping the rest of the stripes under UDL protection only gives us very marginal benefit. The NFSS enters its 3rd lifetime stage and the RAID-5 is applied to all the stripes. We propose Hybrid Caching (HC) scheme to reduce the parity update cost in this stage. Two caches – data cache and partial parity cache, are constructed. By buffering the temporary data and parity at the caches, the accesses to the NAND flash blocks are significantly reduced. More details on HC scheme will be given in Section III-E.

C. UDL Scheme

BCH ECC is sufficient to correct the bit errors of the NAND flash block within the retention time of $T_{retention}$ or under the read count of T_{read}. However, when the retention time exceeds $T_{retention}$ or more than T_{read} read operations are performed to a block, UDL scheme is started. To monitor the impact of retention time limit on the NFSS reliability, we introduce a parameter "timestamp" to record the programming time of each NAND flash block's first page and store it in the OOB (out of band) area [15]. The timestamp of each NAND flash book is checked periodically during the 1st stage of the NFSS lifetime. If the difference between the current operation time and the timestamp is longer than $T_{retention}$, the NAND flash block is recycled and its valid data is moved to other blocks. Note that the timestamp checking and block recycling operations can be performed during the system idle time to minimize the performance degradation.

The controller can monitor retention time only at time of power-on. Power-off may lead to uncorrectable bit error rate if the power-down time period exceeds $T_{retention}$. However, as we shall show in Section IV-A, the typical $T_{retention}$ is 381 days, which is much longer than the normal power-down time period in enterprise NFSS applications.

To monitor the impact of read disturb, a read counter is used to record the number of read accesses to each NAND flash block. If the block is accessed for more than T_{read} times, the block will be erased and the valid data will be moved to other blocks. The electron accumulation on the NAND flash cells is removed by the erase operation and the read counter of the block can be safely reset. The read counters are stored in the DRAM buffer of the NFSS to enable the fast real-time access. As we shall show later, the DRAM capacity occupied by read counters is very small, e.g., 1MB for a 256GB NFSS. Again, we perform the block recycling only during the system idle time.

D. PE-aware RAID-5

At the 2nd stage of NFSS lifetime, the P/E cycles of some NAND flash blocks have exceeded the threshold T_p. Excessively short T_{read} and $T_{retention}$ leads to frequent block cycling. PE-aware RAID-5 technique is applied to protect the NFSS at this stage.

Similar to conventional RAID-5, all the logic pages are still grouped into stripes. For every stripe, if there is at least one logic page is written into a physical pages with the P/E cycle

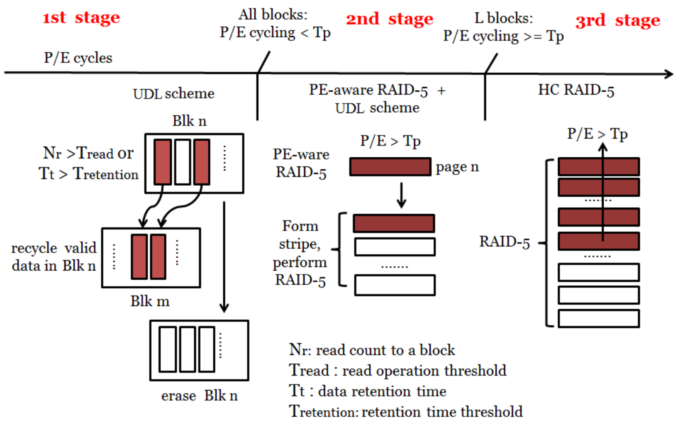

1st stage
All blocks: P/E cycling < Tp
2nd stage
L blocks: P/E cycling >= Tp
3rd stage
P/E cycles
UDL scheme
PE-aware RAID-5 + UDL scheme
HC RAID-5
Blk n
Nr >Tread or Tt > Tretention
recycle valid data in Blk n
Blk m
erase Blk n
PE-ware RAID-5
P/E > Tp
page n
Form stripe, perform RAID-5
P/E > Tp
RAID-5
Nr: read count to a block
Tread : read operation threshold
Tt : data retention time
Tretention: retention time threshold

Fig. 3. DA-RAID-5 overall architecture

greater than T_p, we generate the stripe parity and put the whole stripe under the protection of RAID-5. We introduce PEA counter to record how many physical pages in a stripe have the P/E cycle greater than T_p. Obviously, RAID-5 is applied to the stripe only when the PEA counter is larger than zero. The details of PE-ware RAID-5 is shown in Algorithm 1. Here we assume a stripe includes N logic pages, i.e., $D_{i0} \sim D_{in}$.

Algorithm 1 Basic PE-aware RAID-5

```
Input: Receive a write request of logic number $D_{ij} \in S_i$
Output: Generate and update parity pages
Write $D'_{ij}$ to PPN $M_{ppn}$
**if** P/E cycle of $M_{ppn} > T_p$ **then**
  **if** PEA counter == 0 **then**
    read other logic pages out of NAND flash
    $P'_{Si} = D_{i0} \oplus ...D_{i(j-1)} \oplus D'_{ij} \oplus D_{i(j+1)} ... \oplus D'_{in}$
    write $P'_{Si}$ into NAND flash
  **end if**
  **if** PEA counter != 0 **then**
    $P'_{Si} = D_{ij} \oplus D'_{ij} \oplus P_{Si}$
    write $P'_{Si}$ into NAND flash
  **end if**
  PEA counter = PEA counter + 1
**end if**
**if** P/E cycle of $M_{ppn} \leq T_p$ **then**
  **if** PEA counter != 0 **then**
    PEA counter = PEA counter - 1
    **if** PEA counter == 0 **then**
      Invalidate $P_{Si}$
    **end if**
    **if** PEA counter != 0 **then**
      $P'_{Si} = D_{ij} \oplus D'_{ij} \oplus P_{Si}$
      write $P'_{Si}$ into NAND flash
    **end if**
  **end if**
**end if**
```

In the above algorithm, N-1 read operations and 1 write operation are required to generate the initial parity during the stripe forming. After that, 2 read and 2 write operations are required per update of the parity. We proposed a so called "stripe grouping" scheme to reduce the corresponding parity generation cost: A small buffer is used to group the write requests to the logic pages in the same stripe. The parity is calculated only when the logic pages are evicted from the buffer. Since multiple logic pages in the same stripe may be evicted simultaneously from the buffer, the parity can be updated based on Eq. (2) instead of Eq. (1). The overall parity generation cost is reduced. Here the buffer eviction follows LRU policy.

E. HC Scheme

Following the increase of NFSS operation time, the number of NAND flash blocks that need to be protected under RAID-5 gradually rises. When the number of such blocks exceeds L, we will directly apply RAID-5 across all blocks. However, we propose a Hybrid Caching (HC) scheme to minimize the read operations induced by the parity generation during this NFSS lifetime stage: The logic pages being written are buffered in a write cache where the logic pages from the same stripe are always evicted together. When the logic pages are evicted, the parity is calculated and buffered in the partial parity cache. Different from partial parity generation rule in [7], we adopt two ways to generate the partial parity depending on the updated logic pages: We define a updating threshold N_{update}. If the updated page number in the same stripe exceeds N_{update}, we generate a new partial parity only based on the current updated pages instead of the old logic pages. The algorithm is shown in Algorithm 2. By merging the frequently updated logic pages and eliminating the unnecessary read operations, the overall parity generation cost is reduced. Again, LRU policy is applied to this write cache at stripe level and to the parity cache at page level.

Algorithm 2 parity generation algorithm

```
Input: number of updated pages $n$, updated pages $D'_{ij}$,
$D'_{i(j+1)}$,...,$D'_{i(j+n)}$ and the existing parity number $PP_S$
Output: partial parity $PP'_S$
**if** no $PP_S$ in parity cache **then**
  $PP'_S = D'_{ij} \oplus D'_{i(j+1)} \oplus, ..., \oplus D'_{i(j+n)}$
**end if**
**if** $PP_S$ has existed in parity cache **then**
  **if** $n > N_{update}$ **then**
    $PP'_S = D'_{ij} \oplus D'_{i(j+1)} \oplus, ..., \oplus D'_{i(j+n)}$
  **end if**
  **if** $n \leq N_{update}$ **then**
    $PP'_S$ is generated by the rule in [7]
  **end if**
**end if**
```

An example of HC scheme is depicted in Fig. 4. The stripe size and N_{update} are set to 4 and 2 respectively. The write cache can constrain up to 8 pages. PP_i denotes partial parity. The write request sequence "$D4$, $D1$, $D3$, $D2$, $D22$,..." is also shown in Fig. 4. After logic page $D23$ is written into the write cache, the logic pages in the least frequently updated stripe S0 – $D1$, $D2$, and $D3$, are evicted from the write cache and written into NAND flash. A partial parity PP0 is generated by $D1$, $D2$, and $D3$ and stored in the partial parity cache. Similar process is conducted in the parity calculation of stripe S2. When a new $D2$ is written to the write buffer, a new partial parity $PP0'$ is calculated by only $D1'$, $D2'$ and $D3'$ without incurring any read and write operations to the NAND flash.

The sequential write accesses, e.g., a write request with long length, will degrade the hit rate of the write cache. In our design, we only buffer the write request within a certain length, e.g, 8 pages. For the write requests with longer length, we generate the corresponding full parity immediately and write the logic pages and the parity into NAND flash directly. We also enhanced the stripe group reliability model proposed in [6] by considering data retention time, read count and P/E cycling together. The derived bit error rate of single NAND flash cell can be used to obtain the optimized stripe size N, based on the estimation method in [6]. Our experiments show that the optimal values of the cut-off write request length and stripe size N are 8 and 8, respectively, for the simulated workloads.

IV. Experimental Results

A. Experimental Setup

We selected Disksim-based Flashsim [17] as our simulation platform. We modified the simulator by adding multi-channel NAND flash array and RAID-5 controller. Five disk traces representing different NFSS applications are adopted in the evaluation, as shown in Table I. The specification of MLC NAND flash module from manufacturer M is summarized in Table II. For a MLC page, we assume the program latency of the 1st page is the typical value (1.6ms) while the one of the 2nd page is the maximum value (5ms). We also assume the erase times of the blocks with the P/E cycles less and greater than T_p are typical and maximum values, respectively. The strip size N is set 8 and the PE-ware threshold L is set to 2^{14}. The capacity of parity cache is set to 512KB for both PPC and HC schmes. A 8-bit BCH ECC (4200,4096,104) is applied to every 512-byte data. T_p is set to 15,000. The corresponding $T_{retention}$ and T_{read} are 381 days and 4,6000, respectively, according to Fig. 2(a) and 2(b).

B. Simulation Results

We simulated the average response time, write count and read count of RAID-5 and PPC, as shown in Fig. 5(a)-5(c), respectively. Note that the performance of the NFSS with these two techniques does not change at the different stage of NFSS lifetime. Fig. 5(a)-5(c) also show the evaluations of UDL, PE-aware RAID-5 and HC at their applicable NFSS lifetime stages. At the initialization of UDL simulation, the timestamps of all logic pages are set to zero. The P/E cycle of each physical page gradually raises as the operation time increases. Compared to RAID-5, the average response time of UDL improves by 23.3% across all 5 workloads. The corresponding write count and read count respectively decrease by 20.3% and 65.3%, on average. Even compared to the best prior-work PPC, the improvements of average response time, write count and read count are still 13.2%, 12.1% and 42.1%, respectively. The only exception is the workload *web search*, which is a read-intensive workload and shows significant write count increase.

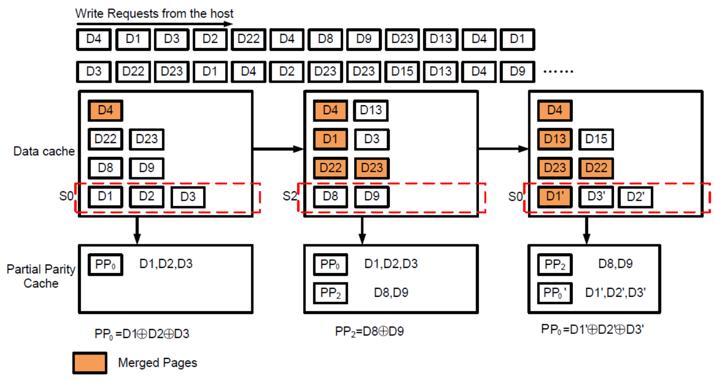


Fig. 4. Stripe formation under PE-aware RAID-5.

This increase is triggered by a large number of read-disturb-induced block recycling in UDL. The maximum performance improvement happens for *TPC-C*, which involves intensive write accesses. The majority of intermediate parity updates is eliminated. The average response time improves by 18.8%.

During PE-aware simulation, we assume the NAND flash blocks that have the P/E cycle greater than T_p randomly show up across the whole NFSS. The number of the NAND flash blocks that need to be protected by RAID-5 keeps increasing during the operation of NFSS. Again, the maximum performance improvement (14.7%) is achieved at *TPC-C*.

In the simulation of HC, N_{update} is set to 6. Compared to RAID-5, HC achieves 20.1% average response time improvement for the 5 workloads, followed by 19.7% and 37.3% reduction on the average write count and read count, respectively. Even compared to the latest PPC technique, HC can still improve the average response time, write count and read count by 9.7%, 9.3% and 6.1%, respectively. Similar to UDL and PE-aware RAID-5, the maximum performance improvement is achieved at *TPC-C*. By merging the frequently-updated write requests in the write cache, the write accesses to the NAND flash and parity calculations are successfully reduced, leading to 11.6% response time improvement. In *web search*, however, the response time is almost unchanged due to the dominant random read access – the write accesses only occupy 0.02% of the accesses to the NAND flash.

The performance improvements of HC under different N_{update} (4, 6, 8) are also evaluated, as shown in Fig. 6(a)-Fig. 6(c). The selection of N_{update} shows phenomenal impact on the workload *WIN 7*: Increasing N_{update} from 4 to 6 reduces the average response time and read count by 7.5% and 8.5%, respectively, while the total write count remains the same. However, further increasing N_{update} to 8 raises the average response time again. N_{update} increase incurs almost invisible performance variations for the other workloads. This is because a large number of overlapped logic pages rarely happens during the normal HC operations.

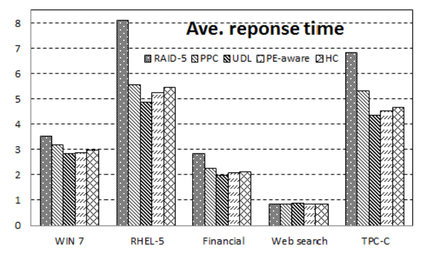
We also evaluate the impact of write cache size on the effectiveness of DA-RAID-5, as shown in Fig. 7. We set N_{update} to 6 and sweep the write cache size from 128KB to

TABLE I
Workload Characteristics

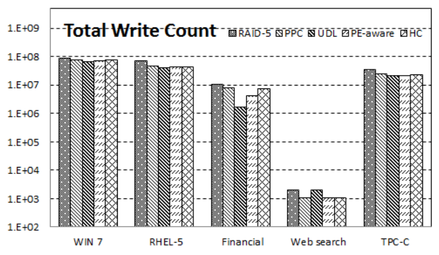
Disk trace	write ratio	seq.wr.	application
Win 7	42%	15.2%	p2p, office and web serfing
RHEL	93%	2.3%	server access
Financial [16]	76%	1.9%	OLTP application
TPC-C	99%	0.9%	OLTP application
web search [16]	0.02%	0	access to search engines

TABLE II
MLC NAND flash Parameters

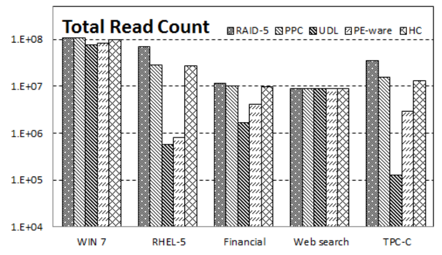
Capacity	Block Size	Block Number	Page Size
	1 MB	8192	8KB
Timing	Program Latency	Read Latency	Erase Latency
	Max. 5 ms	Max. 400μs	Max. 10ms
	Typ. 1.6 ms	–	Type. 1.5ms



(a)

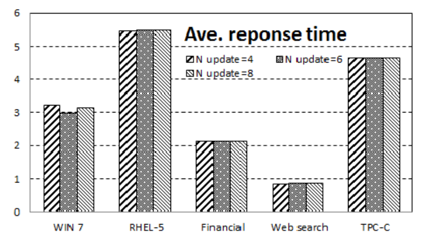


(b)

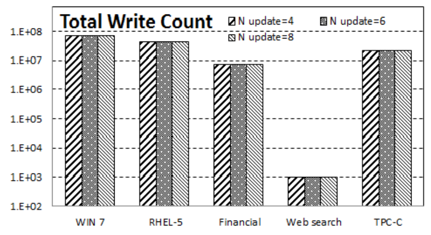


(c)

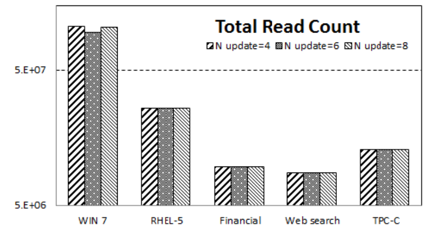
Fig. 5. (a) Average response time under five workloads; (b) Write count under five workloads; (c) Read count under five workloads.



(a)



(b)



(c)

Fig. 6. (a) Average Response Time under various N_{update}; (b) Write count under various N_{update}; (c) Read count under various N_{update}.

2MB. The increase of write cache size results in the response time improvement in all the workloads except for *web search*. The maximum performance improvement is achieved at *TPC-C* again, with an average response time reduction of 12.0% when the write cache size changing from 128KB to 2MB.

V. Conclusion

In this work, we propose Disturb-Aware RAID-5 (DA-RAID-5) scheme to improve the performance of the Enterprise NFSS under RAID-5 data protection. We first quantitatively analyze the impacts of program disturb, read disturb and retention time limit on the reliability of NAND flash cells with different P/E cycles. Based on the analysis, we divide the lifetime of NFSS into three stages with different distributions of P/E cycles across all NAND flash blocks. Three schemes, namely, disturb limiting (UDL), PE-aware RAID-5 and Hybrid Caching (HC), are proposed to protect the data integrity of the NFSS at these three stages, respectively. Our experimental results show that DA-RAID-5 effectively improves the response time of NFSS by 9.7% on average, compared with the best prior work. Only very marginal increases of write and read access are introduced at one simulated workload.

VI. Acknowledgement

This work was supported by the National Science Foundation under grants CNS-1116171 and CCF-1217947.

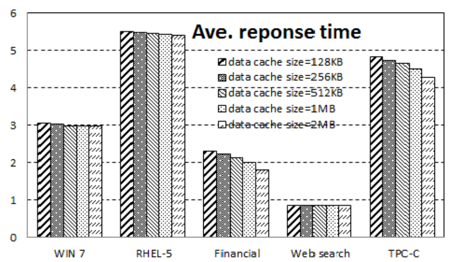


Fig. 7. Average response time under different data cache size

References

[1] T. Jung, Y. Choi, K. Suh, B. Suh, J. Kim, Y. Lim, Y. Koh, J. Park, K. Lee, J. Park *et al.*, "A 3.3 v 128 mb multi-level nand flash memory for mass storage applications," in *42nd ISSCC*. IEEE, 1996, pp. 32–33.
[2] A. Brand, K. Wu, S. Pan, and D. Chin, "Novel read disturb failure mechanism induced by flash cycling," in *31st IRPS*, 1993, pp. 127–132.
[3] T. Wang, W. Tsai, S. Gu, C. Chan, C. Yeh, N. Zous, T. Lu, S. Pan, and C. Lu, "Reliability models of data retention and read-disturb in 2-bit nitride storage flash memory cells," in *IEDM*. IEEE, 2003, pp. 7–4.
[4] C. Xue, Y. Zhang, Y. Chen, G. Sun, J. Yang, and H. Li, "Emerging non-volatile memories: Opportunities and challenges," in *7th CODES+ISSS*. ACM, 2011, pp. 325–334.
[5] Y. Lee, S. Jung, and Y. Song, "Fra: a flash-aware redundancy array of flash storage devices," in *7th CODES+ISSS*, 2009, pp. 163–172.
[6] S. Lee, B. Lee, K. Koh, and H. Bahn, "A lifespan-aware reliability scheme for raid-based flash storage," in *SAC*, 2011, pp. 374–379.
[7] S. Im and D. Shin, "Flash-aware raid techniques for dependable and high-performance flash memory ssd," *IEEE Transactions on Computers*, vol. 60, no. 1, pp. 80–92, 2011.
[8] Y. Pan, G. Dong, and T. Zhang, "Exploiting memory device wear-out dynamics to improve nand flash memory system performance," in *9th FAST*, 2011.
[9] L. Grupp, A. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. Siegel, and J. Wolf, "Characterizing flash memory: anomalies, observations, and applications," in *42nd MICRO*, 2009, pp. 24–33.
[10] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. Nevill, "Bit error rate in nand flash memories," in *46th IRPS*. IEEE, 2008, pp. 9–19.
[11] M. Kang, K. Park, Y. Song, Y. Lim, K. Suh, and H. Shin, "Improving read disturb characteristics by using double common source line and dummy switch architecture in multi level cell nand flash memory with low power consumption," *Japanese JAP*, vol. 50, no. 4, 2011.
[12] J. Lee, J. Choi, D. Park, and K. Kim, "Data retention characteristics of sub-100 nm nand flash memory cells," *EDL*, vol. 24, no. 12, pp. 748–750, 2003.
[13] L. Shi, C. Xue, and X. Zhou, "Cooperating write buffer cache and virtual memory management for flash memory based systems," in *RTAS*. IEEE, 2011, pp. 147–156.
[14] L. Shi, C. Xue, J. Hu, W. Tseng, X. Zhou, and E. Sha, "Write activity reduction on flash main memory via smart victim cache," in *20th GLSVLSI*. ACM, 2010, pp. 91–94.
[15] D. Liu, T. Wang, Y. Wang, Z. Qin, and Z. Shao, "Pcm-ftl: A write-activity-aware nand flash memory management scheme for pcm-based embedded systems," in *32nd RTSS*, 2011, pp. 357–366.
[16] "Oltp application i/o and search engine i/o," http://traces.cs.umass.edu/index.php/storage/storage.
[17] "A simulator for various ftl scheme," http://csl.cse.psu.edu/?q=node/322.