Block Attribute-aware Data Reallocation to Alleviate Read Disturb in SSDs

Mingwang Zhao*, Jun Li*, Zhigang Cai*, Jianwei Liao*, Yuanquan Shi†
*School of Computer and Information Science, Southwest University, Chongqing, China, 400715
†College of Computer Science and Engineering, Huaihua University, Hunan, China 418000
Corresponding author: Zhigang Cai, Yuanquan Shi (*czg@swu.edu.cn*, *syuanquan@163.com*)

***Abstract*—This paper proposes a data reallocation method in RR processes, by taking account of block attributes including P/E cycles and read counts. Specifically, it can distribute the data in the RR block onto a number of available SSD blocks, to allow different blocks keeping their own near optimal (tolerable) read count. Then, it can reduce the number of read fresh operations and the number of raw bit errors. Through a series of simulation experiments based on several realistic disk traces, we demonstrate that the proposed method can decrease the read response time by between 15.11% and 24.96% and the raw bit error rate by 4.14% on average, in contrast to state-of-the-art approaches.**

***Index Terms*—Solid-state drives, Read disturb, Read Refresh, Modeling, Data Reallocation**

I. Introduction

NAND flash-based Solid-State Drives (SSD) have the advantages of fast read/write speed, low power consumption and small size. Then, it has been widely leveraged in digital devices. However, the decrease in endurance and the increase in bit error rates accompanying with the feature size shrinking are now becoming the issues to be reckoned with [1]. Among varied noises that impact the endurance of flash memory, retention noise, program interference noise, program/erase cycling noise, and read disturb [2] are the most important types.

Specially, read disturb is a noise problem at the circuit level. It refers to the fact that when a page is frequently read, the other pages in the same block will be affected [2]. When the affected page is read, it might obtain a piece of corrupted data. Although error correcting code mechanisms (ECCs) can be used to alleviate read disturb by correcting certain read errors, they cannot work when the error rate is out of the correction capacity [3]. The approach of read refresh (RR) can fundamentally cut down the impacts of read disturb [4]. In other words, once the read count of a hot read block reaches a threshold and that block is possible to be affected by read disturb, it thus reads out the block data and reallocates them to another new block for resetting the read count.

Furthermore, Huang et al. [5] have proven that it can yield a minimum read error count in total by evening read accesses across valid blocks. Then, in the process of read refresh, they have proposed to distribute varied parts of the read refresh block (i.e. pages of data) onto other valid blocks, for ensuring all valid blocks can nearly reach the same read count. The experiments verify their approach can indeed cut down the read error rate and the read response time. But, their approach is built on the top of the assumption that all SSD blocks have the same attributes and can afford the same number of read operations. It cannot work in the cases where the blocks have varied attributes, such as the P/E cycle.

To address the issue of data reallocation in read refresh if the valid blocks have varied attributes of P/E cycles and read counts, we propose a reallocation scheme by taking account of block attributes. In summary, it makes following contributions:

- We propose a data distribution scheme by taking the factor of uneven initial attributes of SSD blocks into account, to estimate the optimal read count of varied blocks. Then, it can offer guidance on data reallocation in the process of read refresh.
- We implement the proposal, and offer preliminary evaluation on several disk traces of real-world applications. As measurements indicate, our proposal can afford a better performance improvement on the metrics of average read latency, and read error rate.

The rest of paper is organized as follows: the related work and motivations are shown in Section II. Section III depicts the proposed data reallocation scheme for read refresh. Section IV presents the evaluation methodology and reports experimental results. We conclude the paper in Section V.

II. Background and motivations

A. Background and related work

Error correction codes (ECCs) have been commonly used in SSDs to cover the bit errors caused by many factors, such as read disturb [7]. But, the accumulated read operations on SSD blocks may lead to more read disturb errors, which must affect the reliability of SSDs by suffering from uncorrectable bit errors beyond the capacity of ECCs. To overcome this problem, the mechanism of read refresh (RR) has been introduced to mitigate impacts of read errors [4]. The key idea of read refresh is to read out the valid page data in the disturbed block, and then write them to other available blocks to reset the read count on the data.

To better direct data reallocation in the process of read refresh, Huang et al. [5] aim to balance read accesses among all blocks for optimally limiting negative effects of read disturb (e.g. the raw bit error rate, RBER). Then, they carry out data reallocation in the process of read refresh. At last, it makes all blocks withstand a similar or even read count, so that the read time and the error rate can be noticeably cut down.

978-3-9819263-5-4/DATE21/©2021 EDAA

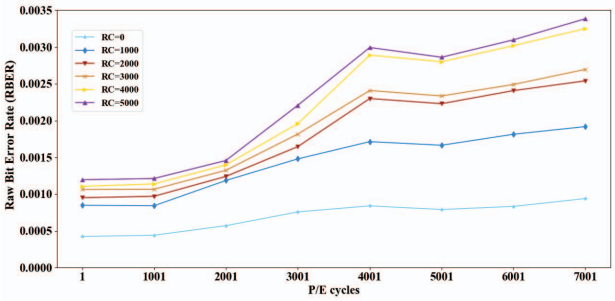

Fig. 1: RBER changes with P/E cycles and read counts [8].

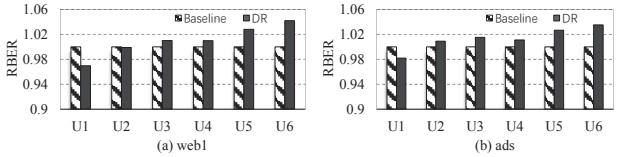

Fig. 2: RBER changes according to varied P/E cycle distribution among all SSD blocks.(Baseline and DR, the short of Data Reallocation, have been shown in Sections IV-A.)

B. Motivations

We have surveyed the factors impacting the read error rate in SSDs resulted by read disturb. Figure 1 illustrates that the blocks of TLC SSDs having different erase counts suffer from read disturb at varied levels. In addition, the accumulated read count on the block is another thing to affect the read error rate.

It is true the wear-leveling mechanism aims to yield a uniform erase distribution in SSDs for better lifetime, but it has been proven that wear-leveling is not necessary in the early lifetime of flash memory [6]. Then, we have further verified the performance changes while adopting the scheme of evenly distributing read accesses among SSD blocks [5], under the conditions of varied P/E cycle distributions.

In the tests, we have introduced `6` kinds of uniform P/E cycle distributions, as shown in Table I to simulate varied P/E cycle distributions among all SSD blocks[1]. Note that the upper limit of variation range of the uniformly distributed P/E cycle is set as `4000`, which is a half of enabled P/E cycle in TLC SSDs. This is because wear-leveling will be triggered in the later lifetime of SSDs, to achieve nearly erasure evenness among all blocks [6].

Figure 2 shows the results of RBER changes when using two read refresh schemes. As seen, the *DR* scheme that evenly distributes read accesses among SSD blocks, can yield the attractive results of reduction on the read error rate while all blocks have no much difference on the P/E cycle (e.g. *U1*). On the other side, when the attribute of P/E cycle among blocks differs from each other greatly, *DR* does even worse than *Baseline* in some cases(e.g. *U3* to *U6*). This illustrates the irrationality of the *DR* scheme that allows blocks with different initial attributes to bear the same number of reads.

[1]Experimental methodology and benchmark specifications can be found in Section IV.

TABLE I: Uniform distributions of block P/E cycles X∼U(a,b)

Variable Name	Value Range	Variable Name	Value Range
U1	`(2000,2000)`	*U2*	`(1980,2020)`
U3	`(1500,2500)`	*U4*	`(1000,3000)`
U5	`(500,3500)`	*U6*	`(0,3980)`

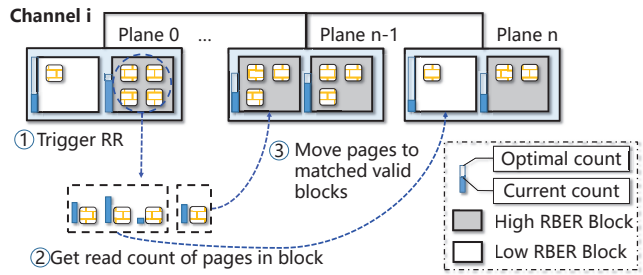

Fig. 3: High-level architectural overview.

Such observations driven us to propose a new data reallocation mechanism to better distribute the data in the process of read refresh. That is to say, we should make use of blocks having high bit error rates to endure fewer reads, and blocks having lower bit error rates to fulfill more reads. Then, the overall bit error rate of SSDs will be confined.

III. Data Reallocation in Read Refresh

A. High-level architecture

The basic idea is to make the low RBER blocks servicing more read requests, and the high RBER blocks for less reads after read refresh operations. Figure 3 illustrates the high level overview on block attribute-aware data distribution in read refresh, and it works with the following steps:

1) It triggers read refresh for migrating data, when the read count of a specific block reaches the predefined limit.
2) It matches the tolerable read count of available move-in blocks (*Optimal read count*−*Current read count*) with the possible reads of data pages in the move-out block.
3) It traverses valid pages in the RR block (i.e. move-in block), and carries out page move operations to distribute the data onto the matched move-out blocks.

Section III-B shows the design used to estimate the optimal read count of each block to classify low RBER and high RBER blocks. After that, it will be shown that algorithm implementation of the proposed block attribute-aware data reallocation in read refresh.

B. Design detail and implementation

The core of obtaining the optimal read count for a block is to allow blocks with higher (lower) error rate characteristics to get fewer (more) read count. First, the sequence numbers of the blocks containing valid pages are sorted in ascending order of read count and in descending order of error rate characteristics (P/E cycle) respectively, and the blocks with low error rate and high read times are matched to obtain the optimal read count. It is displayed by the height of the bar on the left of each block in the Figure 3.

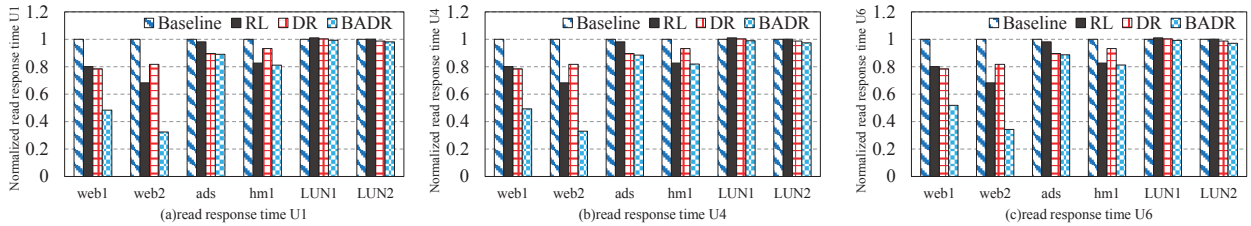

Fig. 4: Read response time under three initial attribute distributions

Algorithm 1: Trigger and execute read reallocation

```
Function RR(blk)
    updateTgtList(tgtList); //update list and R_i
    for page in blk do
        if page.stage == valid then
            if meetCond(page.RCount, tgtList.blk) ==
             TRUE then
                migrate(page, tgtList.blk);
            else
                findNext(page.RCount,tgtList);
                migrate(page, tgtList.blk);
```

TABLE II: Experiment settings of SSDsim (TLC cell)

Parameters	Values	Parameters	Values (ms)
Block number	98304	*Read time*	0.085
Page per block	384	*Extra read-retry time*	0.024
Page size	8*KB*	*Maximum LDPC level*	7
RR threshold	10000	*LSB write time*	0.5
Overprovide	0.25	*CSB write time*	2
FTL scheme	Page	*MSB write time*	5.5
Wear leveling	Static	*Erase time*	15

Algorithm 1 shows the implementation of the proposed data reallocation scheme while performing read refresh. Function `RR()` will be triggered if a block reaches the predefined RR threshold. To be specific, the linked list of valid target block and the optimal read count of each block are supposed to be updated for servicing data migration. Then, Function `meetCond()` is called to check whether the move-out pages can be migrated to the current valid block or not, by matching the read count of move-out page and the remainder read capacity of the block. If not, Function `findNext()` will be evoked to find another matched block. This process repeats until all data in the RR block are reallocated.

IV. Experiments and Evaluation

A. Experiment settings

To verify the effectiveness of our proposed scheme, we make use of a widely used simulator of *SSDSim* to replay the selected real disk access traces. The parameter settings in our tests are shown in Table II and the RR threshold is set as 10000 by referring to [9]. Since the read latencies depend on the level of LDPC soft decision, we set the basic read time as 0.085 ms, and increase the read time by 0.024 ms per read retry after increasing an LDPC level [3], [10].

TABLE III: Specifications on Selected Disk Traces

Trace	# of Req.	Read ratio	Read size	Rpt. addr. ratio
web1	1,055,448	99.9%	15.2 *KB*	99.86%
web2	4,579,809	99.9%	15.0 *KB*	99.97%
ads	1,532,120	90.5%	31.5 *KB*	91.53%
hm1	609,311	95.3%	14.9 *KB*	98.19%
LUN1	1,764,125	76.0%	28.9 *KB*	22.43%
LUN2	1,565,866	82.4%	17.5 *KB*	26.56%

We take advantage of three initial attribute distributions, i.e. *U1*, *U4*, and *U6*, as shown in Table I, to test the performance of the proposed scheme under the different initial attribute distributions of blocks. Besides, we have carried out certain sensitive tests on our platform to find the value of the time window size, and then set each window having 80K requests.

Table III shows the specifications on the selected real disk access traces. The traces *websearch1* and *websearch2* are labeled as *web1* and *web2*. We have also chose two recent I/O traces, i.e. *LUN1* and *LUN2* that are short labels of *additional-03-2016021719-LUN3* and *additional-03-2016021720-LUN4* . Specially, the metric of ***Rpt. addr. ratio*** in the table reflects the concentration level of read access addresses in the traces. It refers to the ratio of repeated read address space to all read address space. Besides, for triggering more RR operations in some traces, we amplify read requests of both *LUN* traces by 10 times, *hm1* and *ads* by 3 times, by referring to [2].

Apart from the proposed *Block Attribute-aware Data Reallocation (BADR)*, the following schemes are selected for comparative tests.

- *Baseline*, which follows the basic principles of read refresh, and unconditionally moves the valid data in the RR block to another new block while RR is triggered.
- *Read leveling (RL)*, which is a software-based mechanism to weaken the side-effects of read disturb [11]. It has been employed as another counterpart in our experiments, and works at Flash Translation Layer of SSDs to distribute hot read data, and then makes use of read refresh to relieve the negative effects of read disturb.
- *Data Reallocation in RR (DR)*, which aims to yield a uniform read accesses among all blocks when carrying out data migration in read refresh processes [5]. We argue that this work is the most related work to ours.

B. Results and discussions

1) Read response time: Figure 4 presents the read latency of replaying the traces by using varied data reallocation schemes in read refresh. As seen, comparing with *Baseline*,

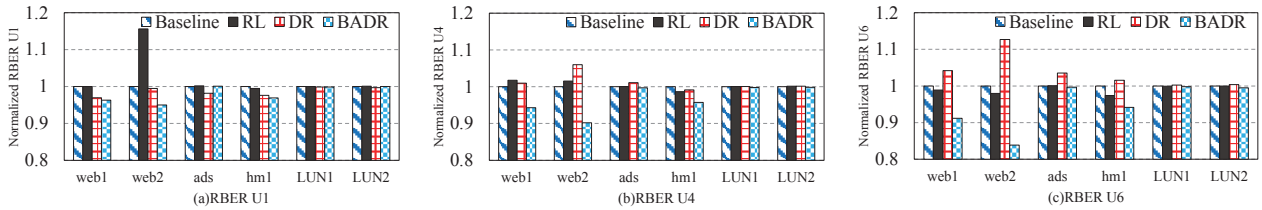

Fig. 5: Raw bit error rate under three initial attribute distributions.

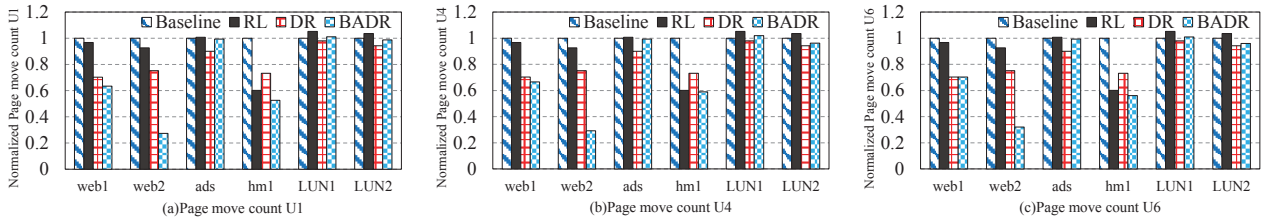

Fig. 6: Page move count under three initial attribute distributions.

RL, and *DR*, the proposed *BADR* method can reduce the read response time by 24.96%, 15.11%, and 16.97% respectively. This is because *BADR* will distribute valid pages of data in the RR block onto different valid blocks for the purpose of reducing Read latency, by matching the predicted read count of move-out pages in the RR block and the optimal read count of the destination blocks.

2) Raw bit error rate: Figure 5 presents the results of raw bit error rate after running the benchmarks. Compared with the *Baseline*, *RL*, and *DR*, the proposed *BADR* method can decrease the raw error rate by an average of 3.57%, 4.20%, and 4.65% respectively. This is because our implementation enables the proposed reallocation model, to distinctively migrate the data onto other valid blocks in the processes of read refresh in SSDs. More importantly, the reduction of RBER becomes more obvious, while the P/E cycle distribution among all blocks becomes more diverse.

3) Page move statistics: In order to further understand the impact of our proposal on read refresh, we have measured the number of page moves (i.e. data migration) caused by read refresh operations. Figure 6 reports the results of page moves while using selected schemes. In contrast to *Baseline*, *RL*, and *DR*, *BADR* can cut down the number of page moves by an average of 25.02% , 19.51%, and 10.21% respectively.

4) Overhead: The main space overhead of the proposed approach is to hold the optimal read limits of active blocks (i.e. *tgtList* in Algorithm 1) is neglectable. The computation overhead is the time required for updating the optimal read count of block (*per RR*) and flushing read count of block pages (*per time window*). The cost time ranges from 0.36 seconds to 4.05 seconds, which only accounts for 0.01% to 0.37% of the total I/O time, even though our tests are conducted on a resource-limited platform that has an ARM Cortex A7 Dual-Core CPU with 800MHz and 128MB of memory.

V. Conclusions

This paper proposes a new scheme for data reallocation issued by read refresh in SSDs, by referring to the factor of P/E cycles and read counts of blocks. It first builds a mathematical model to estimate the optimal read counts of valid blocks by referring to several factors. The experiments verify that our scheme can work well in a variety of P/E cycle distributions, especially while user applications have certain hot read data. In brief, it can noticeably reduce the read response time, and the page move count caused by read refresh, so that both I/O performance and lifetime of SSD can be boosted.

Acknowledgment

This work was partially supported by "National Natural Science Foundation of China (No. 61872299, No. 62032019)" and "Natural Science Foundation Project of CQ CSTC (No. CSTC2018jcyjAX0552)".

References

[1] W. Lee , M. Kang and S. Hong et al.. 2019. Interpage-Based Endurance-Enhancing Lower State Encoding for MLC and TLC Flash Memory Storages. In *TVLSI*, Vol.27(9):2033-2045.

[2] K. Ha, J. Jeong and J. Kim. 2016. An Integrated Approach for Managing Read Disturbs in High-Density NAND Flash Memory. In *TCAD*, Vol.35(7):1079-1091.

[3] Y. Du, Y. Zhou and M. Zhang et al. 2019. Adapting Layer RBERs Variations of 3D Flash Memories via Multi-granularity Progressive LDPC Reading. In *DAC*, pp.37.

[4] Y. Cai , G. Yalcin, and O. Mutlu et al.. 2012. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *ICCD*, pp.94-101.

[5] B. Huang et al.. 2020. Read Disturb-aware Write Scheduling and Data Reallocation in SSDs. In *IEICE Electronics Express*.

[6] J. Liao, F. Zhang and Li Li et al.. 2015. Adaptive Wear-Leveling in Flash-Based Memory. In *CAL*,Vol.14(1):1-4.

[7] O. Mutlu. 2017. The RowHammer problem and other issues we may face as memory becomes denser. In *DATE*, pp.1116-1121.

[8] W. Liu, F. Wu and M. Zhang et al.. 2019. Characterizing the Reliability and Threshold Voltage Shifting of 3D Charge Trap NAND Flash. In *DATE*, pp.312-315.

[9] T. Wu, Y. Ma, and L. Chang. 2018. Flash read disturb management using adaptive cell bit-density with in-place reprogramming. In *DATE*, pp.325-330.

[10] J. Li, B. Huang, and Z. Sha et al.. 2020. Mitigating Negative Impacts of Read Disturb in SSDs. In *ToDAES*, Vol.26(1):1-24.

[11] C. Liu and Y. Chang and Y. Chang. 2015. Read leveling for flash storage systems. In *SYSTOR*, pp.5:1–5:10.