JCST Journal of Computer Science & Technology

Indexed in: SCIE, EI, INSPEC, CBST, DBLP, etc.

Sponsored by: ICT, CAS & CCF

DIR: Dynamic Request Interleaving for Improving the Read Performance of Aged Solid-State Drives

Nie Shi-Qiang, Zhang Chi, Wu Wei-Guo

View online: http://doi.org/10.1007/s11390-023-1601-y

Articles you may be interested in

ROCO: Using a Solid State Drive Cache to Improve the Performance of a Host-Aware Shingled Magnetic Recording Drive

Wen-Guo Liu, Ling-Fang Zeng, Dan Feng, Kenneth B. Kent

Journal of Computer Science and Technology. 2019, 34(1): 61-76 http://doi.org/10.1007/s11390-019-1899-7

Endurable SSD-Based Read Cache for Improving the Performance of Selective Restore from Deduplication Systems

Jian Liu, Yun-Peng Chai, Xiao Qin, Yao-Hong Liu

Journal of Computer Science and Technology. 2018, 33(1): 58-78 http://doi.org/10.1007/s11390-018-1808-5

Extending SSD Lifespan with Comprehensive Non-Volatile Memory-Based Write Buffers

Ziqi Fan, Dongchul Park

Journal of Computer Science and Technology. 2019, 34(1): 113-132 http://doi.org/10.1007/s11390-019-1902-3

A Lookahead Read Cache: Improving Read Performance for Deduplication Backup Storage

Dongchul Park, Ziqi Fan, Young Jin Nam, David H. C. Du

Journal of Computer Science and Technology. 2017, 32(1): 26-40 http://doi.org/10.1007/s11390-017-1680-8

COLIN: A Cache-Conscious Dynamic Learned Index with High Read/Write Performance

Zhou Zhang, Pei-Quan Jin, Xiao-Liang Wang, Yan-Qi Lv, Shou-Hong Wan, Xi-Ke Xie

Journal of Computer Science and Technology. 2021, 36(4): 721-740 http://doi.org/10.1007/s11390-021-1348-2

Hot Data Identification with Multiple Bloom Filters: Block-Level Decision vs I/O Request-Level Decision

Dongchul Park, Weiping He, David H. C. Du

Journal of Computer Science and Technology. 2018, 33(1): 79-97 http://doi.org/10.1007/s11390-018-1809-4

Nie SQ, Zhang C, Wu WG. DIR: Dynamic request interleaving for improving the read performance of aged solid-state drives. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(1): 82-98 Jan. 2024. DOI: 10.1007/s11390-023-1601-y

DIR: Dynamic Request Interleaving for Improving the Read Performance of Aged Solid-State Drives

Shi-Qiang Nie (聂世强), *Member, IEEE*, Chi Zhang (张　驰), and Wei-Guo Wu* (伍卫国), *Member, CCF*

Department of Computer Science and Technology, Xi'an Jiaotong University, Shaanxi 710049, China

E-mail: shiqiang.nie@xjtu.edu.cn; chi.zhang@stu.xjtu.edu.cn; wgwu@xjtu.edu.cn

Received May 28, 2021; accepted December 24, 2023.

Abstract Triple-level cell (TLC) NAND flash is increasingly adopted to build solid-state drives (SSDs) for modern computer systems. While TLC NAND flash effectively improves storage density, it faces severe reliability issues; in particular, the pages exhibit different raw bit error rates (RBERs). Integrating strong low-density parity-check (LDPC) code helps to improve reliability but suffers from prolonged and proportional read latency due to multiple read retries for worse pages. The straightforward idea is that dispersing page-size data across several pages in different types can achieve a lower average RBER and reduce the read latency. However, directly implementing this simple idea into flash translation layer (FTL) induces the read amplification issue as one logic page residing in more than one physical page brings several read operations. In this paper, we propose the Dynamic Request Interleaving (DIR) technology for improving the performance of TLC NAND flash-based SSDs, in particular, the aged ones with large RBERs. DIR exploits the observation that the latency of an I/O request is determined, without considering the queuing time, by the access of the slowest device page, i.e., the page that has the highest RBER. By grouping consecutive logical pages that have high locality and interleaving their encoded data in different types of device pages that have different RBERs, DIR effectively reduces the number of read retries for LDPC with limited read amplification. To meet the requirement of allocating hybrid page types for interleaved data, we also design a page-interleaving friendly page allocation scheme, which splits all the planes into multi-plane regions for storing the interleaved data and single-plane regions for storing the normal data. The pages in the multi-plane region can be read/written in parallel by the proposed multi-plane command and avoid the read amplification issue. Based on the DIR scheme and the proposed page allocation scheme, we build DIR-enable FTL, which integrates the proposed schemes into the FTL with some modifications. Our experimental results show that adopting DIR in aged SSDs exploits nearly 33% locality from I/O requests and, on average, reduces 43% read latency over conventional aged SSDs.

Keywords triple-layer cell solid-state drive (TLC SSD), performance, interleaving data, unbalanced bit error rate

1 Introduction

NAND-based flash has become the primary storage media in modern computer systems, ranging from mobile devices to servers in data centers[1]. High-density NAND flash-based solid-state drives (SSDs) are promising as they meet the capacity demands of modern applications with reduced per-bit cost. Triple-level cell (TLC) SSDs, the widely employed high-density NAND flash, usually have a high raw bit error rate (RBER) as they have a much narrow margin between neighboring voltage levels and thus are more vulnerable to programming noises[2]. While low-density parity-check (LDPC) codes are increasingly adopted for TLC SSDs to improve their reliabilities[3], the extra flash sensing for soft-decision decoding is time-consuming, especially for aged SSDs with high RBERs.

Regular Paper

A preliminary version of the paper was published in the Proceedings of NVMSA 2019.

This work was supported by the National Key Research and Development Project of China under Grant No. 2017YFB1001701, and the National Natural Science Foundation of China under Grant No. 61972311, and in part by Shandong Provincial Natural Science Foundation of China under Grant No. ZR2019LZH007.

*Corresponding Author

The three different bits of TLC flash cells, referred to as least significant bit (LSB), central significant bit (CSB), and most significant bit (MSB), often exhibit very different bit error rates, even when they have the same program/erase (P/E) cycle and retention time. This imbalance often leads to unequal read and decode latency for LDPC-based SSD storage systems. Given a read I/O request consisting of multiple page sub-requests, its completion time is determined by the slowest sub-request, which depends on the sub-request queuing length and the service time of the sub-request. Prior studies optimize queuing latency[4, 5] and reduce the service time[6] to improve SSD performance. However, for aged SSDs that adopt LDPC, reading a device page with a high RBER results in multiple read retries such that page reading remains a significant portion of the I/O request service time. The read latency of the LSB page is only 2/3 of that of the MSB page at the early stage, but the latency gap worsens in the aged SSD[6, 7].

These studies that are close to our design are the bit-level data layout optimization strategies[8–10]. These strategies interleave the data from each logic page into the three types of bits of the same device page. Since the device bits are accessed sequentially, directly integrating these strategies in flash translation layer (FTL) leads to severe read/write amplification and large performance degradation, which is not practical. Compared with [8–10], our work designs a data interleaving-enable FTL, which alleviates the read amplification issue and makes the data interleaving technology available to the SSD design.

This paper is an extended version of our previous work[11]. In the conference paper, we observed that access locality exists in workloads and exploited this observation to interleave the data segment from consecutive sub-requests with limited read/write amplification. Recent research work has depicted the low utilization ratio of plane-level parallelism[12], which motivates us to explore the opportunity to design an enhanced multi-plane command to read the data segment of a logic page within one read cycle. Compared with the previous work[11], this paper makes the following additional contributions: to alleviate the overhead induced by the read amplification issue and assign the specific page type efficiently, we propose a novel page allocation scheme, in which the planes in NAND flash are split into two kinds of regions (i.e., the multi-plane region and the single-plane region). The pages in multi-plane regions are assigned in a page-interleaving-friendly way. Then these pages could be read out by enhanced multi-plane read command, shortening the processing time of additional read requests.

Overall, this paper makes the following contributions.

• We explore page-level access locality from modern applications, and propose exploiting page-level access locality and distributing interleaved data of these pages to device pages of different types at different locations, which amortizes the RBER at the page level[11].

• We design a complementary plane-level organizational scheme. Firstly, we divide an NAND flash chip into two parts: the multi-plane region and the single-plane region. The former maintains the write point, enabling the assignment of a page with any required type for requests whose data is interleaved. The latter serves as the normal plane for normal write requests. Secondly, we redesign the hardware that implements a novel multi-plane command to mitigate the read amplification issue. The scheme in Subsection 3.2 is not proposed in [11].

• We evaluate the proposed Dynamic Request Interleaving (DIR) scheme and compare it with the state-of-the-art. Our experimental results show that adopting DIR in aged SSDs exploits nearly 33% locality from I/O requests and, on average, reduces 43% read latency over conventional aged SSDs.

In the rest of the paper, Section 2 discusses the SSD background and motivates our DIR design. Section 3 presents the detailed DIR scheme. Section 4 describes the experimental methodology and analyzes the results. Section 5 gives related work. Section 6 concludes the paper.

2 Background and Research Motivation

In this section, we discuss the SSD architecture and the execution workflow of I/O requests. We then motivate our design with the uneven bit error rate among different bits of TLC NAND flash.

2.1 SSD Architecture

Fig.1 shows the internal organization of SSD[13], which consists of host interface logic (HIL), FTL, and flash back-end[13]. The function of each component is as follows.

1) HIL receives an I/O request from the host,

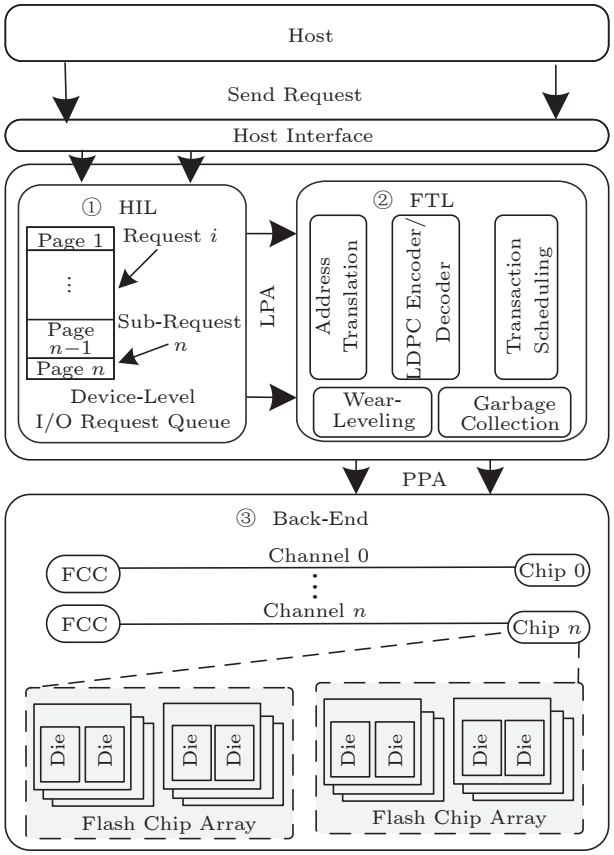


Fig.1. Generic architecture of SSD[13].

splits it into page-sized sub-requests, and then inserts them into device queues for services. Each sub-request has a specific logical page number (LPN)[14].

2) The FTL maintains a mapping table to track the current physical location, i.e., physical page number (PPN), of each LPN. Additional components, such as garbage collection, wear leveling, and the LDPC encoder/decoder engine, are also included in the FTL.

3) The SSD back-end contains multiple channels, which can service I/O sub-requests in parallel. Each channel is connected to one or more chips. Each chip consists of one or more dies, where each die contains one or more planes. Each plane can service an I/O sub-request concurrently with the other planes.

In this paper, we adopt the dynamic mapping scheme such that the channel, chip, die, and plane indices are random for a given LPN. Such an organization provides four levels of parallelism for servicing I/O requests (channel, chip, die, and plane). The front end manages the back-end resources and issues I/O requests to the back-end channels.

2.2 Basic Operations of TLC SSD

In TLC SSD, each cell uses eight states to represent the three bits of data, and each state uses the stored amount of charge, i.e., the threshold voltage, to distinguish itself from the others. Fig.2 illustrates a typical threshold voltage distribution for TLC SSDs[11]. To reduce the raw bit error probability, TLC SSD adopts the gray code so that two neighboring levels differ by one bit—the voltage levels Er, P1, P2, P3, P4, P5, P6, and P7 denote the information bits "111", "011", "001", "101", "100", "000", "010", and "110", respectively.

The program/read operations of LSB, CSB, and MSB pages are different. As shown in Fig.2, the value stored in a TLC cell is determined by the threshold voltage or the amount of charge in the cell. TLC cell programming is often performed by using incremental step-pulse programming (ISPP)[15]. It can be divided into three distinct steps for minimizing the in-

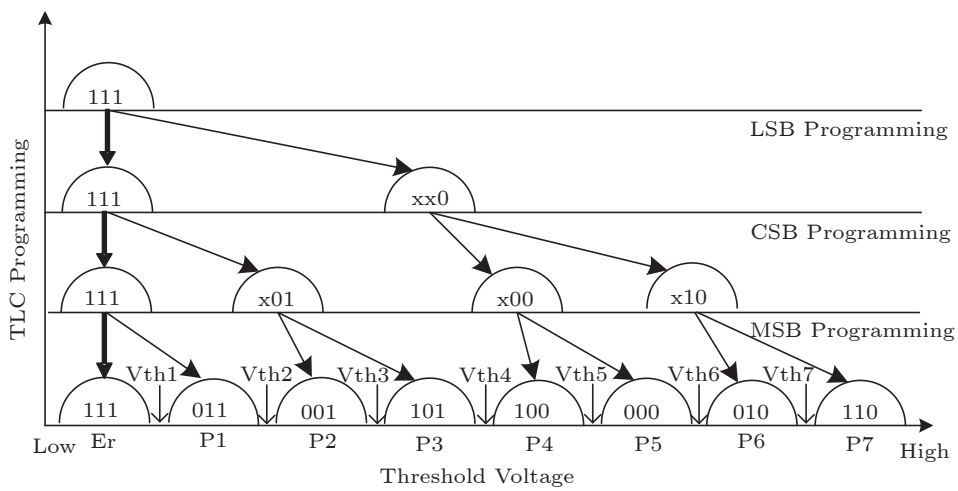


Fig.2. Threshold voltage distribution for TLC NAND flash[11].

ter-page program interference. The LSB page is programmed firstly and quickly to the target threshold voltage range, as shown in Fig.2. If the bit "11" is programmed into the LSB page, the cell threshold voltage is kept in the erased state, marked as Er. If the bit is "0", the cell is charged to transfer the threshold voltage from the Er state to the temporary state. When the CSB page is programmed, if the bit is "1", the voltage threshold does not change, and the cell remains in either the Er state or the next state, depending on the value of LSB, the same program procedure as the MSB page has. When reading LSB page data, during the read operation, if the sensing threshold is lower than Vth1, the cell denotes bit "1"; the cell denotes bit "0" otherwise. For CSB and MSB pages, the flash cell needs to be sampled twice and three times, respectively, by changing the sensing voltage levels. This is referred to as hard decision memory sensing, differentiating one sensing level between two adjacent states.

When adopting LDPC to improve TLC SSD reliability, we may need to differentiate more than one sensing level between two adjacent states, referred to as soft-decision flash sensing. LDPC increases the number of sensing levels so that more errors are likely to be corrected. However, such an approach leads to multiple read tries, significantly increasing flash read latency and degrading the read performance of SSDs, particularly aged SSDs with high RBER.

2.3 Advanced Read/Program Command of NAND Flash

The multi-plane command supports multiple read, program, or erase operations across all planes in the same die. Compared with the basic read, program, or erase operations, it saves operation overhead several times as multiple operations are executed in parallel. However, the host must follow the operation restriction to issue a multi-plane command. That is, a multi-plane read/write operation must have the same chip, die, block, and page addresses. Besides, the blocks executing a multi-plane erase operation must have the same chip, die, and block addresses[16, 17]. However, Gao *et al.* reported that plane-level parallelism was far from well-utilized in a wide range of real workloads due to these strict restrictions (i.e., only about 1%–4% of requests can be written into pages with multi-plane command)[12].

Many researches aim to exploit plane-level parallelism maximally from FTL to the flash hardware design. Gao *et al.* utilized the DRAM cache to evict a multiple of N dirty pages at a time such that these pages can be written by using multi-plane command[12]. A novel NFM architecture enabling a decoupled word-line (WL) selection for the mated planes was proposed to relax the restriction—the WL addresses could be a different value for multi-plane command[18]. An independent plane read scheme was proposed to improve further total system performance, in which two planes can perform read operations independently and asynchronously on any block/page address and combination of QLC/SLC modes[19–21]. In this paper, similar to the above work, we modify the hardware design of NAND flash to permit pages at different positions to operate in parallel at the plane level.

2.4 Problem Statement

Recent studies reveal that different bits of MLC and TLC flash exhibit a significant RBER variation[9, 10]. Fig.3 compares the RBERs of different device pages for TLC SSDs according to [10]. As shown in Fig.3, the RBERs of MSB pages are significantly higher than those of LSB pages. This is because errors come mainly from cells having their voltage levels shifted across neighboring levels. There is only one bit flipping possibility for the LSB page (i.e., 111↔xx0/x01↔x00/P3↔P4), but four possibilities for the MSB page, i.e., Er↔P1, P2↔P3, P4↔P5, and P6↔P7. As another example, the shift between P1 and P2, i.e., P1↔P2, causes CSB bit errors but not LSB and MSB errors. The amount of charges stored in different threshold voltage levels is also different. The charge in the P7 state is more likely to leak.

To address the RBER difference, Zhao *et al.*[9] and Nakamura *et al.*[10] proposed to store data from one logic page to different types of bits in several device

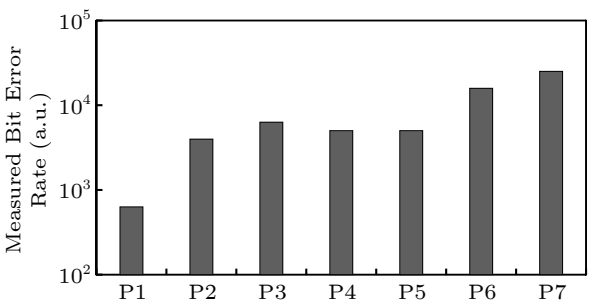


Fig.3. Measured bit error rates of each state[10]. a.u.: arbitrary unit.

pages. The idea of these strategies is that storing data in both the worst page and the strongest page can achieve an average lower RBER and then fewer read-retries for the LDPC decoding procedure. Modern SSD always employs the LDPC engine to recover the corrupted page for its high error-correct capacity. Still, it suffers from severe read latency due to the increasing number of read retries. The page with a high RBER may cost up to 10 times more read latency than that with a low RBER[3]. Let us take an example to illustrate the advantage of this strategy. Assuming both the LSB page and the MSB page are written into the same WL, after a while, we read out the LSB page and the MSB page one by one. If the RBERs of the LSB page and the MSB page are 0.005 and 0.006, respectively, and the number of read-retries is 2 and 3, respectively, then the read procedure costs twice the read latency of the MSB page. If the two page-size data spans the LSB page and the MSB page, the average RBER is 0.005 5, and the corresponding number of read-retry is 2 (e.g., 512 B LDPC coding redundancy per 4 KB user data). However, while these strategies help mitigate the bit error rate at the page level, they face a major challenge—one logic page writes results to more than one device page. Since these writes are done sequentially, these designs face severe read and write amplification and thus extensive performance degradation (i.e., reading one logic page induces several internal read operations).

To summarize, the read latency of reading page-size data can be reduced by data interleaving technology. However, the read procedure induces more internal read requests than those delivered by the host. This data interleaving technology is not practical as directly integrating these strategies in FTL leads to severe read/write amplification and extensive performance degradation[11].

2.5 Motivation

To solve the read amplification issue and page-type-induced read performance deterioration issue, we first study the characterization of I/O requests in modern applications. An I/O read request typically consists of multiple sub-requests for pages spanning different channels, chips, dies, and planes. Without considering the lengths of the request queue, we assume to service these sub-requests at the same time. Due to RBER differences across different pages, their read latency varies dramatically—the data from LSB/-CSB pages tends to be ready much earlier than that from MSB pages. Such scheduling tends to generate sub-optimal results as the I/O latency is throttled by the time servicing the slowest pages.

We have experimented with this read latency variation-induced performance degradation issue, where the experimental parameters are listed in Section 4. We calculate the mean read latency for a given read request with N sub-requests as shown in Fig.4. The result depicts that the mean read latency becomes larger with the request size N greatly. We refer to this issue as the worse page-dominated read. The main reason for the worse page-dominated read is that the default page allocation scheme allocates pages in a round-robin way for the coming write requests, which ignores the page type. Each sub-request has a 1/3 chance of being served by an LSB/CSB/MSB page in TLC SSD, and thus the page-level read latency of the request depends on the page type. If the target pages of sub-requests are all LSB pages, the probability of all sub-requests being served by LSB pages is equal to $(1/3)^n$. If at least one of the sub-requests is issued to an MSB page, the probability is equal to $1-(2/3)^n$. And also, the probability of CSB page-dominated read is equal to $(2/3)^n-(1/3)^n$. According to the math formulation, MSB-dominated read has the largest probability for a given read request. As the RBER of the MSB page increases more quickly than that of the others, a basic idea is to amortize the RBER of the logic page residing in the MSB page to the LSB page or CSB page.

To amortize the RBERs of a worse page to a strong page, we devise to utilize the data interleaving technology to bridge the read latency gap between different pages with suppressed write and read amplification. We conduct an experiment to analyze the

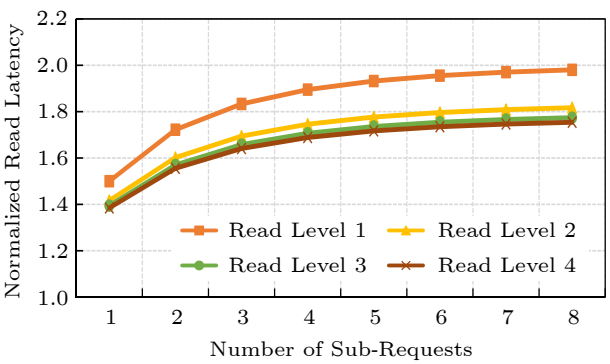


Fig.4. Normalized read latency of a request with the varied number of sub-requests and read levels.

page-level access locality in modern applications, i.e., for pages consecutively written to SSDs, the likeness of accessing them simultaneously later. Fig.5 summarizes the results showing spatial locality in different applications. For example, 80% pages that are written consecutively are accessed (i.e., read or written) simultaneously at a later time. This result motivates our design of page-level data interleaving for mitigating read and write amplification and improving SSD read performance.

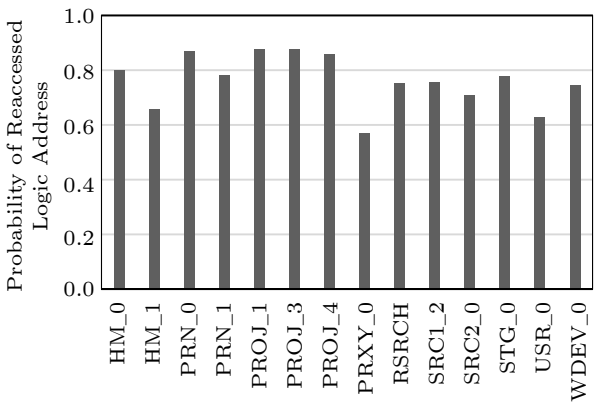


Fig.5. Read locality across workloads[11].

With the above motivational method, this paper aims to resolve these technical difficulties, including 1) how to decide whether the data in the request queue needs to be interleaved or not; 2) how to allocate the page with a specific type and reduce the read and write amplification further. The detail of the proposed strategy is presented in Section 3.

3 Details of DIR-Enable FTL

In this section, we elaborate on the DIR scheme. When an I/O request arrives at the host interfaces, the HIL splits it into multiple page-sized sub-requests sent to the FTL. We assume LDPC is applied to each page to improve data reliability.

DIR is designed to exploit access locality to mitigate the long read latency when reading MSB device pages from aged SSDs. It interleaves the data from any two consecutive pages and writes those pages in two different device pages. These device pages are of different types, i.e., LSB/MSB pages, LSB/CSB pages, or CSB/MSB pages. The interleaving helps mitigate the RBER at the page level such that the number of read retries can be effectively reduced at read time, which greatly improves the read performance of aged SSDs. We also design a novel page interleaving-friendly page allocation scheme to assist the read procedure of the logic page residing in two physical pages, which consists of two strategies: 1) to assign the page with a specific type, we design novel plane organization and employ relaxing program order; 2) to alleviate read amplification, we design an enhanced multi-plane command, which introduces a dedicated peripheral circuit for each block to free the restrictions of conventional multi-plane command. This scheme bases on enhanced multi-plane command to support reading two pages with different page types in parallel. By utilizing this scheme, the read amplification induced by page interleaving is ameliorated further, but the utilization ratio of multi-plane command increases. Fig.6 shows the design architecture of our proposed scheme.

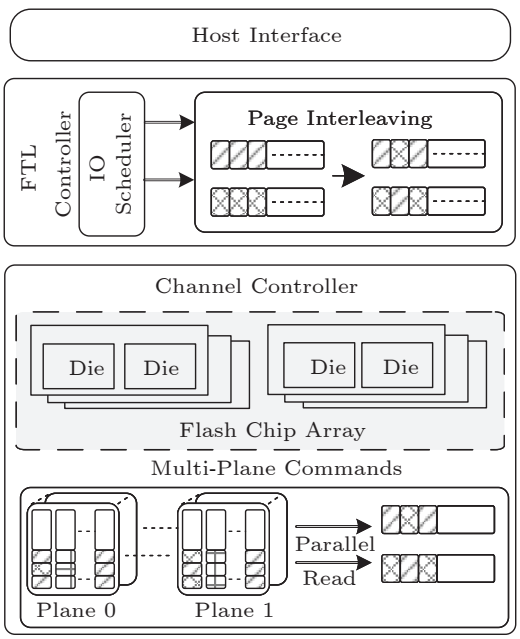


Fig.6. SSD architecture with page interleaving.

The page interleaving module[11] and page assignment module are implemented inside FTL. Before the IO scheduler delivers the requests to the flash backend, DIR exchanges the data of two pages from two consecutive sub-requests segment by segment with the page interleaving module. Besides these two pages are written into two planes in one die with enhanced multi-plane command with the page assignment module. The un-interleaved sub-request is written in the single-plane region with the page assignment module.

3.1 Interleaving Data from Write Requests

When these coming write requests are queued in

the device IO queue for servicing in the next stage, FTL splits each request into page-size sub-requests and decides which sub-requests are pre-processed by the page interleaving technology. The former sub-requests are programmed into the multi-plane regions, where they store the interleaved data. The left sub-requests are programmed into the single-plane regions used as the normal plane in SSD. Note that FTL maintains the logic space of the multi-plane regions. The DIR scheme organizes the data from consecutive write sub-requests in an interleaving way. It consists of two components, i.e., sub-requests grouping and device page assignments. Algorithm 1 depicts the main procedure[11].

Algorithm 1. Interleaving Write Request in DIR Scheme[11]

Require: WQ: the write request delivered by host
Require: $subWQ$: the sub-requests of WQ
1: $subWQ \leftarrow split\_request(WQ)$
2: HWQ: grouped sub-requests
3: NWQ: free sub-requests
4: $HWQ, NWQ \leftarrow group(subWQ)$
5: **for** each $sub \in HWQ$ **do**
6: $assign\_page(sub)$
7: $interleave\_data(sub)$
8: **end for**
9: **for** each $sub \in NWQ$ **do**
10: $assign\_random(sub)$
11: **end for**

Step 1: *Grouping Sub-Requests.* The DIR scheme traverses the sub-requests split from the same I/O request and groups any two sub-requests with adjacent LPNs. The sub-requests in the same group are to be written to two different types of device pages. We adopt the heuristics of using adjacent LPNs while the high-level semantic information may further improve access locality. We always place two sub-requests in a group so that if the number of sub-requests is not a multiple of 2, the remaining one sub-request is left without being placed in any group. The sub-requests in groups and the sub-requests not in any group are referred to as grouped sub-requests and free sub-requests, respectively.

DIR only interleaves the data from grouped sub-requests. Given one group, DIR saves one second of each grouped sub-request on the LSB/MSB/CSB device page and the left one second is saved on the other pages. One device page contains one-second data from each grouped sub-request. As we discuss next, the LSB, CSB, and MSB device pages are from different blocks. By interleaving only grouped sub-requests, DIR avoids write amplification by introducing extra write sub-requests. Writing free sub-request remains the same as that in the baseline (i.e., single-plane regions).

Step 2: *Device Page Assignment.* DIR assigns the interleaving data in one group to two different types of device pages in blocks from two different planes. The detailed page allocation scheme for grouped pages is presented in Subsection 3.2. For free sub-requests, i.e., those not grouped, we first assign LSB or CSB pages so that their response time is short.

3.1.1 *Writing Sub-Requests with Update Operation*

For the page written by free sub-requests, we only invalidate the page and assign another new page to the new coming sub-requests. While for updating the page written by grouped sub-requests, the reference count of the page, which is initialized to 2, is subtracted by 1 each time, and the page is invalidated when the reference count is 0. This scheme does not influence the procedure of wear leveling and garbage collection.

3.1.2 *Generating Dummy Read Sub-Requests*

The page-sized data written by free sub-requests stored on a physical page can be read using the default method. While for the page-sized data written by grouped sub-requests which are distributed on two pages, it is necessary to deliver two sub-requests generated by the host and FTL to read and decode the data. The DIR scheme is host-transparent, and the host is unaware that the data in some LPN is kept on two different pages, and FTL needs to generate another one read sub-request with the request delivered by the host. For simplicity, we refer to the read request generated by FTL as the dummy sub-request. The flag of the LPN in the mapping table is used to indicate whether it requires two read sub-requests. Fig.5 shows the locality among requests in those workloads released by Microsoft[22, 23]. As a result of the locality of the read requests, the dummy read sub-requests may replicate with the other free sub-requests. For example, the host sends a read request to read the data ranging from LPN i to $i+1$. In the best conditions, the data of LPN i and $i+1$ is written by the grouped sub-requests in the same group. FTL generates an extra dummy read sub-request to

read the data of LPN i along with the original sub-request, the same as reading the data of LPN $i+1$. In all, four sub-requests will be generated to read the data from LPN i to $i+1$, but two of the four requests replicate each other. As a result, this method improves read performance with no read amplification.

3.2 Page Interleaving-Friendly Page Allocation Scheme

In Subsection 3.1, we present how to interleave data from write requests, and we discuss a novel page organization scheme in this subsection. This subsection introduces how to depict the building of DIR-enable FTL. We make some modifications ranging from the FTL layer to the hardware layer. Firstly, to meet the requirement of assigning the suitable page type, we split the NAND flash into two parts, namely, multi-plane region and single-plane region. The former maintains the write-point enabling to assign a page with any type for grouped write sub-requests. The latter is used as the normal plane, storing for the free sub-requests. Secondly, we redesign the hardware of implementing multi-plane command, which allows block/page in the paired plane not to be equal. The proposed multi-plane command can read out the interleaved pages in parallel.

3.2.1 *Constructing Multi-Plane Region and Single-Plane Region*

The construction of the multi-plane region aims to assign desired pages for grouped sub-requests as the conventional FTL does not support page type aware allocation. We pick up any two planes in one die to construct the multi-plane regions. The partition of the two regions depends on the workloads. If the workloads are suitable for employing the page interleaving technology, the number of multi-plane regions could be increased dynamically and the number of single-plane regions is decreased accordingly. Additionally, the multi-plane should be distributed on all the die evenly to exploit the parallelism. For large write-dominated workloads, DIR-enable FTL redirects most requests into multi-plane regions, and the multi-plane regions are used up quickly. For small write-dominated workloads, more requests are located in a single plane region. Based on this condition, we split the plane resource dynamically. In the initial stage, only a few planes are set to be multi-plane regions.

3.2.2 *Relaxing Program Sequence*

The strict program order within blocks in the conventional TLC SSD design is necessary to minimize the inter-page (inter-cell) program interference by guaranteeing that a fully programmed word-line is interfered with by only one adjacent page. DIR-enable FTL requires the combination of any two pages in different types to store the interleaved data. However, the conventional program order does not meet the requirement as shown in Fig.7(a). It must follow the following constraints to suppress the program interference, and these pages in each block are programmed in the "Z" mode. To allocate a specific page type, DIR-enable FTL employs a relax program order with ignorable program interference[24] as shown in Fig.7(b). All the LSB/CSB/MSB pages in a block can be sequentially written one by one. We split the usage of a block into three stages, referred to as LSB-block, CSB-block, and MSB-block.

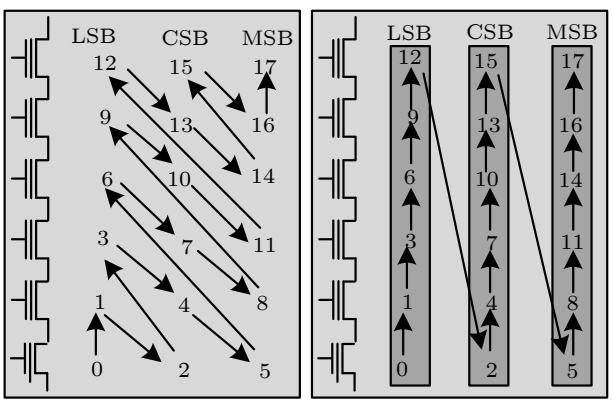


Fig.7. Program order comparison. (a) Default program order. (b) Employed program order.

Noted that only multi-plane regions employ this relax program order. For servicing the write requests, the multi-plane regions maintain several active blocks in each plane. The difference between the two active blocks is that they are in different stages. Let us take an example to illustrate this procedure as shown in Fig.8. At some point, block 0 in plane 0 uses up all the LSB pages and then assigns the CSB page, while block 1 in plane 1 starts to assign its MSB page. Block 0 and block 1 are combined with each other to store the interleaved data from group sub-requests. Therefore, DIR-enable FTL operates block 0 and block 1 in parallel with the help of enhanced multi-plane command.

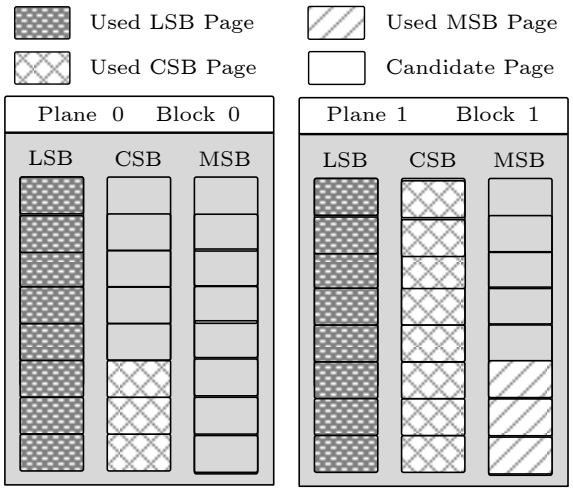


Fig.8. Generating paired pages with different types.

3.2.3 *Implementing Parallel Read in Multi-Plane Regions*

To implement the read/write of different pages in one multi-plane command, the NAND flash requires hardware modifications to decouple block and page selection. Considering the chip area overhead, traditional NAND flash vendors make planes share the row address decoder in one die. Such a decoder-share architecture requires the same page address for multi-plane operations. Fig.9 presents the conventional components in a die. It consists of NAND flash cell arrays, page buffers, and other peripheral circuitry (e.g., command interface, IO logic). The blocks are grouped into two or more planes. Each plane owns a dedicated page buffer; hence, each plane can operate independently. But they share the row address decoder, which denotes the block/page address, limiting the plane-level parallelism. A basic optimization opportunity is to trade off the chip area and performance improvement from plane-level parallelism. In 2D planner NAND flash, the area of peripheral circuitry is limited due to enlarging bit density. Considering the high bit density of 3D NAND flash, which is 1 000 times higher than that of 2D NAND flash[25], it is feasible to design partial dedicated peripheral circuitry for each plane in one die, which contains the row address decoder, the circuitry logic to control the operation timing and input voltage separately, and so on with reasonable area costs. This similar decoupled WL design has presented in many recent studies[18–21]. In their design, the WL of QLC NAND flash can work in SLC/TLC/QLC mode and be read by multi-plane command without any restriction. Additionally, we could put additional dedicated peripheral circuitry underneath the memory cell in the Z direction if it is hard to add hardware in the same X/Y direction to the memory cells[26]. Based on the proposed multi-plane command, the pages in multi-plane regions can be read in one read cycle, eliminating the read amplification issue.

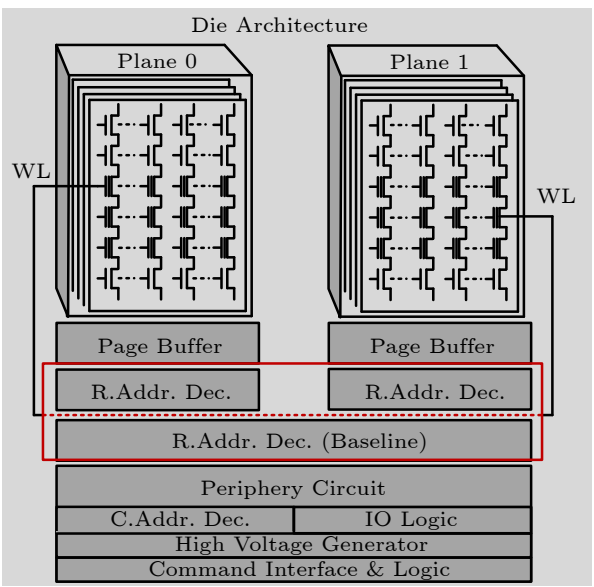


Fig.9. DIR-enable multi-plane operation. C.Addr. Dec.: column address decoder; R.Addr. Dec.: row address decoder.

3.2.4 *Garbage Collection and Wear Levelling*

As the pages with the same offset may be invalided together or invalided separately, the GC procedure in multi-plane regions is different from that in single-plane regions. After the multi-plane regions trigger the GC operation, the pages with the same LPN are read out together; if the two pages are both valid, then both two pages will be written into other active blocks in multi-plane regions; if one page is valid and the other one is invalid, then the data is recovered from two pages and written into single-plane regions. After finishing the GC operation, the FTL mapping table also is modified.

The wear-out speed of the multi-plane region and the single-plane region depends on the workloads. DIR-enable FTL picks up the plane as the multi-plane region in a round-robin way. Therefore the default wear-leveling algorithm still works as well.

3.3 Overhead Analysis

Design Overhead of Enhanced Multi-Plane Com-

mand. The area overhead mainly comes from the block-level selector and the page-level selector (extra 2%–3% area overhead), which could be ignored[18, 26]. The time overhead comes from the GC operation in the multi-plane region, which migrates data from two blocks in two planes once compared with the normal GC operation. The GC overhead comes from reading out valid pages, writing valid pages to another block, and erasing blocks. However, the overhead also can be ignored as the pages in multi-plane regions are read out by the multi-plane read command. Then the pages are programmed by the normal write command or multi-plane command, and then the blocks also are erased by the multi-plane erase operation. In some cases, the GC in multi-plane regions may save GC overhead as it could reclaim more blocks than normal GC and reduces the frequency of GC[12].

Storage Overhead. Some LPNs are associated with two physical pages, i.e., the data of some LPNs is stored in two different physical pages, and thus the mapping table of the FTL needs to trace two PPNs for some specific LPNs. Taking 1 TB SSD with the 4 KB page size as an example, the mapping table size of conventional SSD is (1 TB/4 KB) × (4 B + 4 B) = 2 GB. For SSD with the DIR scheme, the maximum size of the mapping table is (1 TB/4 KB) × (4 B + 4 B + 4 B) = 3 GB. The storage overhead of DIR increases by 50% over the baseline, which only induces an extra 0.9% storage capacity for 1 TB SSD; therefore this storage overhead is negligible. These frequently-used entries in the mapping table could be cached. Other studies[6] show that the performance impact is less than 1%, which can be ignored[6]. Otherwise, for the commercial open-channel SSD, the mapping table of FTL is stored in the server's memory. The memory capacity may be up to hundreds of gigabytes, which is enough to cache all the entries in the mapping table.

Time Overhead. The DIR scheme may introduce time overhead in two folds. Firstly, the time overhead comes from the physical address lookup of the mapping table. It takes only one step for DIR-enable FTL to involve the mapping table and find out one or more physical pages for a given logical address. In our design, the entries in the mapping table include the logic address and the corresponding physical pages; therefore the lookup procedure costs the same time as the original one. The page allocation scheme assigns one or more pages for given sub-requests, and thus it costs no extra time. DIR-enable FTL does not bring extra time overhead compared with the default FTL.

3.4 Feasibility Discussions for High-Density NAND Flash

As high-density NAND flashes (i.e., QLC/PLC) have been designed and popularized into the market by vendors in recent years, the feasibility of the DIR scheme in the high-density NAND flash is studied in this subsection. In Section 3, we take the TLC NAND flash as an example to illustrate how the DIR scheme works. The main idea contains two folds—interleaving data from consecutive logic sub-requests and keeping physical pages containing parts of the page-sized data read in parallel. Compared with the QLC NAND flash or others, the high-density NAND flash has more cell states to store more bits (i.e., four-page types exist in QLC). To employ the DIR scheme in the high-density NAND flash, the flash chip must meet two restrictions. The skewed RBER must exist among pages in one WL; therefore DIR can utilize this characterization to disperse one logic page to more than one physical page and achieve a low RBER on average; otherwise, the high-density NAND flash must support the enhanced multi-plane command in the hardware layer, and the SSD employs DIR-enable FTL to manage the NAND flash resource. To the best of our knowledge, the high-density NAND flash still adapts gray code and the ISPP scheme to program the NAND flash cell, which results in skewed RBER across WL[19]. Otherwise, some vendors have already produced the QLC NAND flash with various multi-plane commands similar to our hardware design[19–21]. Therefore, we argue that DIR-enable FTL is practical to high-density NAND flash.

4 Experimental Evaluation

In this section, we evaluate our proposed scheme against existing schemes in respect of IO performance, overhead, and sensitivity on SSD with varied configurations.

4.1 Experimental Setting

To evaluate the effectiveness of the proposed DIR scheme, we implement the DIR scheme based on SSDsim, which has been validated against the hardware platform[27]. In our experiments, the program and the read latency of LSB, CSB, and MSB pages in TLC SSD are adopted from [10, 28]. Table 1 provides the detailed configuration of the TLC SSD.

Table 1. Configuration of TLC SSD

Parameter	Value
Number of channels	8
Number of chips per channel	2
Number of planes per chip	2
Number of blocks per plane	768
Page size (KB)	4
LSB read latency (μs)	60
CSB read latency (μs)	90
MSB read latency (μs)	120
Flash type	TLC
Transfer latency (ns/byte)	3
Sense latency (μs)	24
Number of pages per block (KB)	4
Erase (ms)	3
LSB write latency (μs)	900
CSB write latency (μs)	1 200
MSB write latency (μs)	1 500

Workloads. We use the enterprise servers traces from Microsoft research Cambridge[22, 23] to evaluate the DIR scheme, as shown in Table 2. These workloads are widely used in previous studies[5, 12].

Table 2. Statistics of Workloads

Trace	Read/Write Ratio	Avg. Read (KB)	Avg. Write (KB)	Interval (ms)
HM_0	0.25	11.61	11.21	194.49
HM_1	0.97	18.15	22.86	513.11
PRN_0	0.11	26.55	13.93	120.29
PROJ_1	0.91	43.43	22.23	8.38
PROJ_3	0.90	15.03	30.14	439.71
PRXY_0	0.03	9.72	6.28	48.33
RSRCH	0.09	15.70	12.70	427.31
SRC2_0	0.14	12.64	11.02	418.94
SRC2_2	0.28	88.26	57.79	146.20
STG_0	0.23	33.56	12.69	273.72
USR_0	0.37	47.42	13.55	275.36
WDEV_0	0.20	16.57	12.11	528.09

In this subsection, we compare the following schemes.

- *NOAC*[27]. This scheme disables the advanced multi-plane command to present the original performance as the baseline.
- *AC*[27]. This scheme enables the advanced multi-plane command to explore the potential opportunity to utilize the plane-level parallelism.
- *Interleaving*[11]. We implement the interleaving technology proposed in our conference paper[11] based on AC. FTL groups any two successive write sub-requests (instead of three successive write sub-requests in the original paper[11] for fairness) in the request queue greedy.
- *DIR.* It is the scheme proposed in this paper adopting the page interleaving strategy and the page interleaving-friendly page allocation strategy in SSDsim.

4.2 Experimental Evaluation

We evaluate the DIR scheme by measuring the average response time, the percentage of pages that benefit from the DIR scheme, the read amplification rate relative to compared schemes, and the utilization of plane-level parallelism, and studying its sensitivity on SSD with varied configurations.

Page-Level Read Latency Comparison. Before measuring the response time of the SSD architecture, we first study our proposed scheme's mean latency compared with the default page. The RBER of LSB/CSB/MSB is referenced from the experimental results[29]. The mean page-level read latency of requests with varied request sizes is shown in Fig.10. We observe that DIR induces increased read latency smoothly when the NAND flash has few errors (i.e., RBER < 0.005, the same LDPC configuration in [3]), as the read latency of interleaved pages is decided by CSB/MSB pages in DIR while the LSB read still has 1/3 chance to be read in the baseline. However, the read latency of the baseline and DIR becomes equal to each other when the request size N increases. When SSD becomes aged, the mean read latency of DIR is reduced by 8%–33% compared with the baseline. This result depicts that DIR achieves significant performance improvement for page-level access.

Read Performance Comparison. Fig.11 shows the normalized read response time among the NOAC, AC, interleaving, and the DIR scheme. In this part, we only compare the read performance of the aged SSD. For aged SSDs, we observe large performance improvement over the baseline—19%–62% improvement could be achieved. Due to the higher locality, workloads such as HM_1, PROJ_0, SRC2_2, USR_0 benefit more from the DIR scheme. We also observe 39%, 41%, 62%, and 61% reduction in read response time, respectively. For SSDs in the early stage, the read response time increases by between 1% and 10% across all the workloads, which is not presented in the paper. This is because the extra dummy read requests cause a decline in read performance. Read requests only require few or no soft sensing at the early stage so that the performance benefits from DIR are negligible.

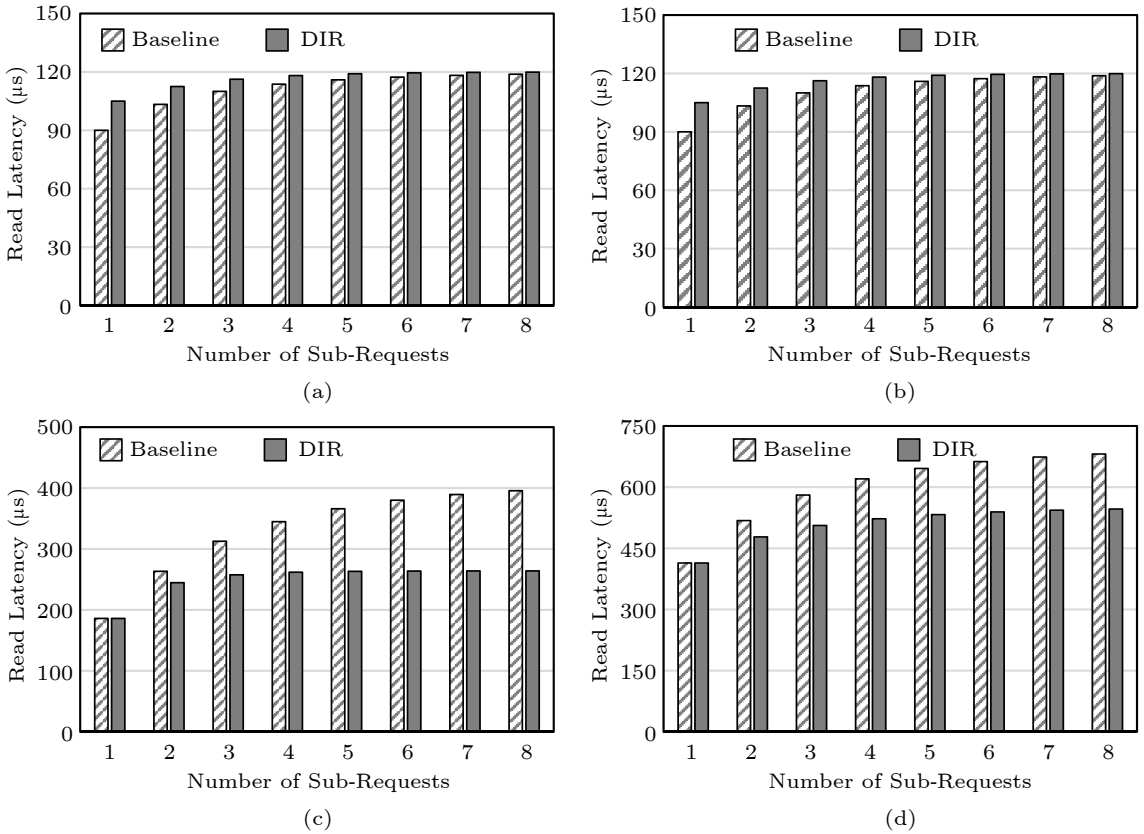


Fig.10. Page-level mean read latency between the baseline and the DIR scheme. Page read latency under (a) RBER < 0.004, (b) RBER < 0.006, (c) RBER < 0.008, and (d) 0.007 < RBER < 0.009.

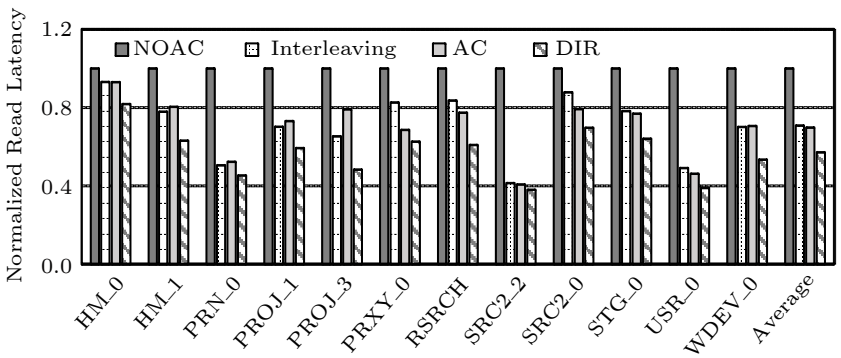


Fig.11. Normalized read latency in aged SSD among these schemes.

Write Performance Comparison. As shown in Fig.12 compared with the AC and interleaving, the write performance in DIR does not fluctuate significantly except HM_1 whose read-write ratio is 97%. When the RBER increases continuously, the decoding time increases as well, and the write performance is sensitive to the processing time of read requests. Accordingly, the read amplification induced by DIR also worsens the write performance. Although the DIR scheme only interleaves the data of sub-requests for a given write request without bringing extra writes, our proposed page allocation scheme allocates two pages with different types once, and the worse pages determine the program latency. For the case that the amount of read requests is larger than that of AC and NOAC, the extra read sub-requests can be serviced between requests. For the workloads with large write ratios, e.g., SRC2_2, we observe large write performance degradation due to many read sub-requests targeting on the same die and holding the channel for a long time (i.e., more pages are interleaved).

Read Amplification Analysis. Next, we evaluate the read amplification in DIR. We compare the num-

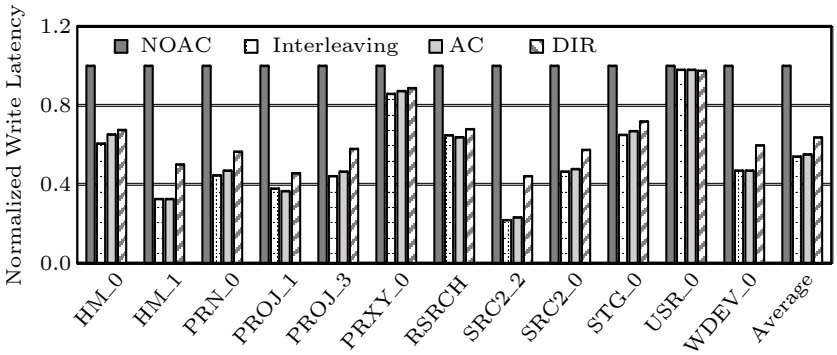


Fig.12. Normalized write latency in aged SSD among these schemes.

ber of read requests introduced by DIR across all the workloads and summarize the results in Fig.13. DIR introduces 3%–20% (14% on average) more read requests. We can see that HM_0 has the most significant increase in the number of read requests, and SRC2_2 has the least read amplification. This is because most read requests in HM_0 range in size from 1 KB to 2 KB. Its read amplification factor is between 1 and 2, and the effectiveness is 0.75 for HM_0 as mentioned above, and thus the read amplification factor is larger than the others. The average read request size of SRC2_2 is 88 KB, such that many read sub-requests replicate each other. In summary, DIR introduces extra reads but its impacts on the lifetime and reliability are negligible.

Multi-Plane Read Analysis. To explore the effectiveness of DIR on multi-plane read, we have statistics for the utilization of multi-plane read. Fig.14 shows the percentage of multi-plane read to all read operations across all the workloads. As more pages are grouped to physical adjacent pages in interleaving, the multi-plane read cannot be utilized totally. For AC, the pages are distributed across all the channels, chips, dies, and planes to exploit parallelism, and thus the multi-plane read is used more frequently compared with interleaving. Our proposed scheme DIR designs the multi-plane friendly page allocation strategy so that its multi-plane utilization could be the highest compared with both interleaving and AC schemes.

Hybrid-Page Read Analysis. We next study the pages being interleaved by counting the number of

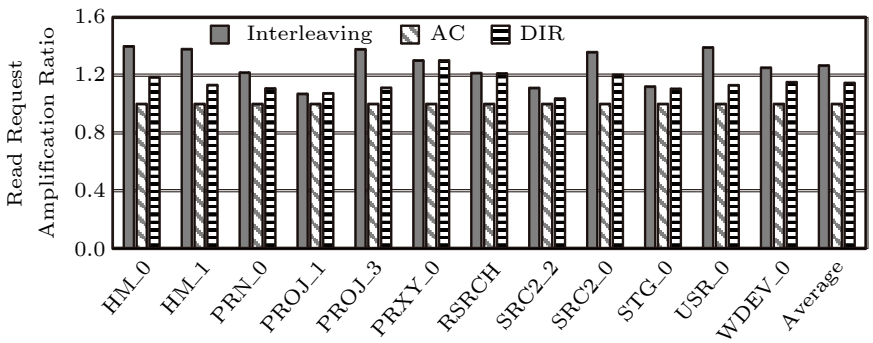


Fig.13. Normalized read amplification rate.

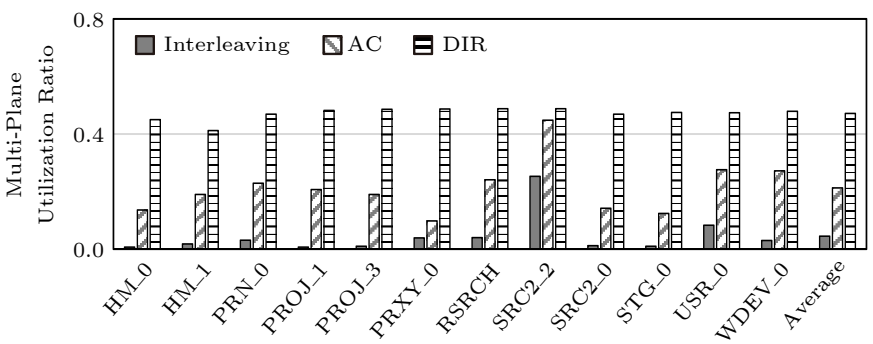


Fig.14. Multi-plane read utilization comparison.

pages with and without interleaving data in the mapping table. Fig.15 illustrates the percentage of the pages with data interleaving. It shows that nearly 33% of SSD pages can benefit from DIR, although interleaving achieves more interleaved pages. HM_1 achieves the highest percentage among all the workloads because the average size of its I/O requests is approximately twice the page size. On the contrary, for SRC2_0, only 19% of pages in SSD can benefit from the DIR scheme because the I/O request size in SRC2_0 is either too small or too large. The efficiency is relatively low with most request sizes being 68 KB or 2 KB.

Sensitivity Analysis. In this part, we study the sensitivity of DIR on SSD with varied parameters. We first modify the number of planes in one die to be 2 and 4, respectively, to observe the performance fluctuation, as shown in Fig.16. Compared with AC, DIR still achieves about 18% performance improvements. We also modify the page size from 4 KB to 8 KB to study the impact of the page size on performance. The result also indicates that DIR still achieves higher performance improvement than AC, as shown in Fig.17.

5 Related Work

Many studies work on optimizing read performance with LDPC in flash-based SSDs. They can be categorized into three groups as follows.

BER Reduction. Zhang *et al.* proposed to dynamically adjust the sensing voltages to reduce read laten-

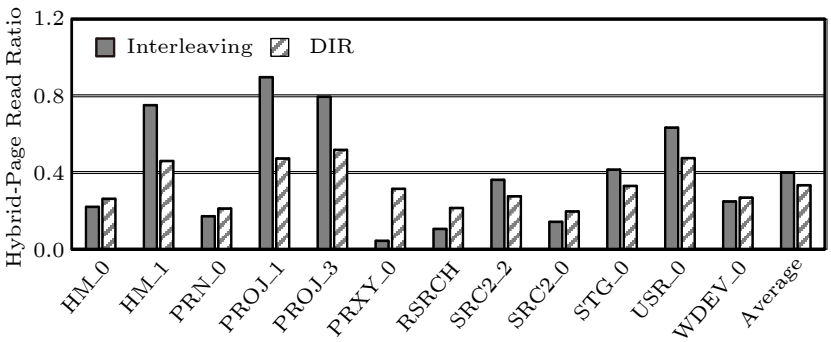


Fig.15. Ratio of hybrid-page read to all reads.

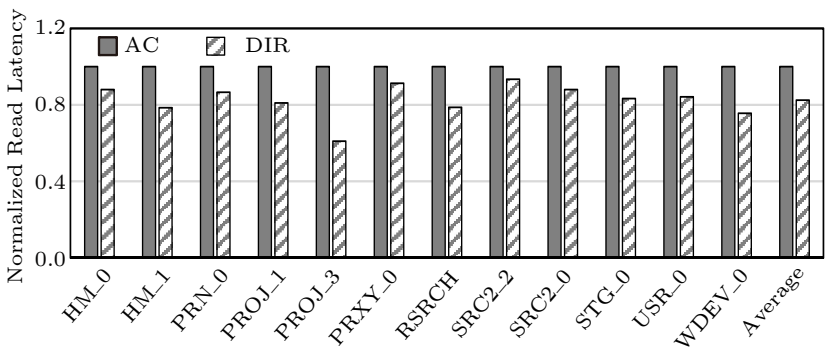


Fig.16. Read latency comparison between two planes per die and four planes per die.

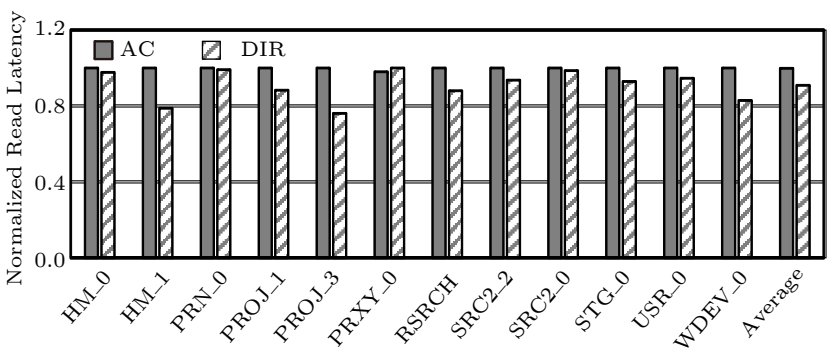


Fig.17. Read latency with 8 KB page size.

cy[30]. Wu *et al.* exploited the error modes of 3D TLC NAND flash to optimize LLR information for further enhancing the decoding performance[31]. Zhang *et al.* proposed to integrate the decoding result of the LSB page into the initial information of LDPC decoding for the MSB page to reduce the LDPC decoding latency of the MSB page in NAND Flash[32]. These refresh methods[33–35] were proposed to periodically correct data with long retention and reprogram the data into new blocks, which can reduce retention-induced errors.

Flash Sensing Optimization. Zhao *et al.*[3] proposed to apply fine-grained levels in LDPC reads progressively. When the read with a lower level fails, the next level with several extra read voltages is applied for flash sensing[3]. Tokutomi *et al.* proposed AEP-LDPC, which considers the effects of program disturb, data retention, and floating-gate capacitive coupling, to reduce the times of decode iterations[36]. Li *et al.* proposed a smart sensing level placement scheme to reduce the LDPC decoding latency[37]. In order to read out correct data with BCH codes, Cai *et al.* proposed to record the optimal threshold voltages of the last programmed page in each block[38]. Peleato *et al.* proposed a mathematical model based on the read voltages in the last read to estimate the appropriate read voltages of the current read adaptively[39].

LDPC Decoding Algorithm Optimization. Zhao *et al.* exploited intra-cell error characteristics to speed up LDPC decoding by reducing overall error probability and decoding latency[9]. REAL incorporates numeric-correlation characteristics of different error patterns in both the MSB page and the LSB page of the MLC flash into the message-passing process of the decoding[40]. Aslam *et al.* proposed a two-round LDPC decoding process by reusing the read-back voltages and the decoded results for flash cells from retention-induced failure, which can further improve read performance[41].

6 Conclusions

This paper proposed the DIR (Dynamic Request Interleaving) scheme that exploits the unbalanced bit error rate among LSB, CSB, and MSB pages in TLC SSD and the locality in requests to improve read performance. A page interleaving-friendly page allocation scheme was also proposed to utilize the plane-level parallelism to speed up read operation to alleviate the read amplification issue. Experimental results showed that DIR can improve read performance by 43% compared with the existing aged SSD. As zoned namespace (ZNS) SSD has become popular in recent studies, it restricts applications to writing data into distinct zones sequentially. We will try to study the DIR scheme on ZNS SSDs and solve the possible potential issues facing the novel write mode of ZNS SSDs.

Conflict of Interest The authors declare that they have no conflict of interest.

References

[1] Min C, Kim K, Cho H, Lee S W, Eom Y I. SFS: Random write considered harmful in solid state drives. In *Proc. the 10th USENIX Conference on File and Storage Technologies*, Feb. 2012, pp.1–16. DOI: 10.5555/2208461.2208473.

[2] Matsui C, Sun C, Takeuchi K. Design of hybrid SSDs with storage class memory and NAND flash memory. *Proceedings of the IEEE*, 2017, 105(9): 1812–1821. DOI: 10.1109/JPROC.2017.2716958.

[3] Zhao K, Zhao W Z, Sun H B, Zhang T, Zhang X D, Zheng N N. LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives. In *Proc. the 11th USENIX Conference on File and Storage Technologies*, Feb. 2013, pp.243–256. DOI: 10.5555/2591272.2591298.

[4] Elyasi N, Arjomand M, Sivasubramaniam A, Kandemir M T, Das C R, Jung M. Exploiting intra-request slack to improve SSD performance. In *Proc. the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems*, Apr. 2017, pp.375–388. DOI: 10.1145/3037697.3037728.

[5] Cui J H, Zhang Y T, Wu W G, Yang J, Wang Y F, Huang J H. DLV: Exploiting device level latency variations for performance improvement on flash memory storage systems. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2018, 37(8): 1546–1559. DOI: 10.1109/TCAD.2017.2766156.

[6] Du Y J, Zou D Q, Li Q, Shi L, Jin H, Xue C J. LaLDPC: Latency-aware LDPC for read performance improvement of solid state drives. In *Proc. the 33rd International Conference on Massive Storage Systems and Technology*, May 2017, pp.1–11.

[7] Wu F, Zhu Y, Xiong Q, Lu Z H, Zhou Y, Kong W Z, Xie C S. Characterizing 3D charge trap NAND flash: Observations, analyses and applications. In *Proc. the 36th International Conference on Computer Design*, Oct. 2018, pp.381–388. DOI: 10.1109/ICCD.2018.00064.

[8] Yaakobi E, Grupp L, Siegel P H, Swanson S, Wolf J K. Characterization and error-correcting codes for TLC flash memories. In *Proc. the 2012 International Conference on Computing, Networking and Communications*, Jan. 2012, pp.486–491. DOI: 10.1109/ICCNC.2012.6167470.

[9] Zhao W Z, Sun H B, Lv M J, Dong G Q, Zheng N N, Zhang T. Improving min-sum LDPC decoding through-

put by exploiting intra-cell bit error characteristic in MLC NAND flash memory. In *Proc. the 30th Symposium on Mass Storage Systems and Technologies*, Jun. 2014, pp.1–6. DOI: 10.1109/MSST.2014.6855550.

[10] Nakamura T, Deguchi Y, Takeuchi K. AEP-LDPC ECC with error dispersion coding for burst error reduction of 2D and 3D NAND flash memories. In *Proc. the 2017 IEEE International Memory Workshop*, May 2017. DOI: 10.1109/IMW.2017.7939070.

[11] Nie S Q, Zhang Y T, Wu W G, Zhang C, Yang J. DIR: Dynamic request interleaving for improving the read performance of aged SSDs. In *Proc. the 2019 IEEE Non-Volatile Memory Systems and Applications Symposium*, Aug. 2019. DOI: 10.1109/NVMSA.2019.8863520.

[12] Gao C M, Shi L, Xue C J, Ji C, Yang J, Zhang Y T. Parallel all the time: Plane level parallelism exploration for high performance SSDs. In *Proc. the 35th Symposium on Mass Storage Systems and Technologies*, May 2019, pp.172–184. DOI: 10.1109/MSST.2019.000-5.

[13] Agrawal N, Prabhakaran V, Wobber T, Davis J D, Manasse M S, Panigrahy R. Design tradeoffs for SSD performance. In *Proc. the 2008 USENIX Annual Technical Conference*, Feb. 2008, pp.57–70.

[14] Tavakkol A, Gómez-Luna J, Sadrosadati M, Ghose S, Mutlu O. MQSim: A framework for enabling realistic studies of modern multi-queue SSD devices. In *Proc. the 16th USENIX Conference on File and Storage Technologies*, Feb. 2018, pp.49–65.

[15] Suh K D, Suh B H, Lim Y H, Kim J K, Choi Y J, Koh Y N, Lee S S, Kwon S C, Choi B S, Yum J S, Choi J H, Kim J R, Lim H K. A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme. *IEEE Journal of Solid-State Circuits*, 1995, 30(11): 1149–1156. DOI: 10.1109/4.475701.

[16] Micheloni R. 3D Flash Memories. Springer, 2016. DOI: 10.1007/978-94-017-7512-0.

[17] Hu Y, Jiang H, Feng D, Tian L, Luo H, Zhang S P. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity. In *Proc. the 2011 International Conference on Supercomputing*, May 2011, pp.96–107. DOI: 10.1145/1995896.1995912.

[18] Kim M, Jung W, Lee H J, Chung E Y. A novel NAND flash memory architecture for maximally exploiting plane-level parallelism. *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, 2019, 27(8): 1957–1961. DOI: 10.1109/TVLSI.2019.2905626.

[19] Khakifirooz A, Balasubrahmanyam S, Fastow R, Gaewsky K H, Ha C W, Haque R, Jungroth O W, Law S, Madraswala A S, Ngo B, Naveen Prabhu V, Rajwade S, Ramamurthi K, Shenoy R S, Snyder J, Sun C, Thimmegowda D, Pathak B M, Kalavade P. 30.2 A 1Tb 4b/cell 144-tier floating-gate 3D-NAND flash memory with 40MB/s program throughput and 13.8Gb/mm^2 bit density. In *Proc. the 2021 IEEE International Solid-State Circuits Conference*, Feb. 2021, pp.424–426. DOI: 10.1109/ISSCC42613.2021.9365777.

[20] Shibata N, Kanda K, Shimizu T, Nakai J, Nagao O, Kobayashi N, Miakashi M, Nagadomi Y, Nakano T, Kawabe T, Shibuya T, Sako M, Yanagidaira K, Hashimoto T, Date H, Sato M, Nakagawa T, Musha J, Minamoto T, Uda M, Nakamura D, Sakurai K, Yamashita T, Zhou J Y, Tachibana R, Takagiwa T, Sugimoto T, Ogawa M, Ochi Y, Kawaguchi K, Kojima M, Ogawa T, Hashiguchi T, Fukuda R, Masuda M, Kawakami K, Someya T, Kajitani Y, Matsumoto Y, Sato J, Raghunathan N, Koh Y L, Chen S, Lee J, Nasu H, Sugawara H, Hosono K, Hisada T, Nakamura H. A 1.33-Tb 4-bit/cell 3-D flash memory on a 96-word-line-layer technology. *IEEE Journal of Solid-State Circuits*, 2020, 55(1): 178–188. DOI: 10.1109/JSSC.2019.2941758.

[21] Kalavade P. 4 bits/cell 96 layer floating gate 3D NAND with CMOS under array technology and SSDs. In *Proc. the 2020 IEEE International Memory Workshop*, May 2020. DOI: 10.1109/IMW48823.2020.9108135.

[22] Narayanan D, Donnelly A, Rowstron A. Write off-loading: Practical power management for enterprise storage. *ACM Trans. Storage*, 2008, 4(3): Article No. 10. DOI: 10.1145/1416944.1416949.

[23] Narayanan D, Thereska E, Donnelly A, Elnikety S, Rowstron A. Migrating server storage to SSDs: Analysis of tradeoffs. In *Proc. the 4th ACM European Conference on Computer Systems*, Apr. 2009, pp.145–158. DOI: 10.1145/1519065.1519081.

[24] Park J, Jeong J, Lee S, Song Y, Kim J. Improving performance and lifetime of NAND storage systems using relaxed program sequence. In *Proc. the 53rd Annual Design Automation Conference*, Jun. 2016, Article No. 63. DOI: 10.1145/2897937.2898032.

[25] Parat K, Goda A. Scaling trends in NAND flash. In *Proc. the 2018 IEEE International Electron Devices Meeting*, Dec. 2018. DOI: 10.1109/IEDM.2018.8614694.

[26] Huh H, Cho W, Lee J, Noh Y, Park Y, Ok S, Kim J, Cho K, Lee H, Kim G, Park K, Kim K, Lee H, Chai S, Kwon C, Cho H, Jeong C, Yang Y J, Goo J, Park J, Lee J, Kirr H, Jo K, Park C, Nam H, Song H, Lee S, Jeong W, Ahn K O, Jung T S. 13.2 A 1Tb 4b/cell 96-stacked-WL 3D NAND flash memory with 30MB/s program throughput using peripheral circuit under memory cell array technique. In *Proc. the 2020 IEEE International Solid-State Circuits Conference*, Feb. 2020, pp.220–221. DOI: 10.1109/ISSCC19947.2020.9063117.

[27] Chen F, Hou B, Lee R. Internal parallelism of flash memory-based solid-state drives. *ACM Transactions on Storage* (*TOS*), 2016, 12(3): 1–39. DOI: 10.1145/2818376.

[28] Jung M, Zhang J, Abulila A, Kwon M, Shahidi N, Shalf J, Kim N S, Kandemir M. SimpleSSD: Modeling solid state drives for holistic system simulation. *IEEE Computer Architecture Letters*, 2018, 17(1): 37–41. DOI: 10.1109/LCA.2017.2750658.

[29] Abe M, Matsui C, Mizushina K, Suzuki S, Takeuchi K. Computational approximate storage with neural network-based error patrol of 3D-TLC NAND flash memory for machine learning applications. In *Proc. the 2020 IEEE In-*

ternational Memory Workshop, May 2020. DOI: 10.1109/IMW48823.2020.9108136.

[30] Zhang M, Wu F, Chen X B, Du Y J, Liu W H, Zhao Y H, Wan J G, Xie C S. RBER aware multi-sensing for improving read performance of 3D MLC NAND flash memory. *IEEE Access*, 2018, 6: 61934–61947. DOI: 10.1109/ACCESS.2018.2873081.

[31] Wu F, Zhang M, Du Y J, Liu W H, Lu Z, Wan J G, Tan Z H, Xie C S. Using error modes aware LDPC to improve decoding performance of 3-D TLC NAND flash. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2020, 39(4): 909–921. DOI: 10.1109/TCAD.2019.2897706.

[32] Zhang M, Wu F, Du Y J, Yang C M, Xie C S, Wan J G. CooECC: A cooperative error correction scheme to reduce LDPC decoding latency in NAND flash. In *Proc. the 2017 IEEE International Conference on Computer Design*, Nov. 2017, pp.657–664. DOI: 10.1109/ICCD.2017.115.

[33] Cai Y, Yalcin G, Mutlu O, Haratsch E F, Cristal A, Unsal O S, Mai K. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *Proc. the 30th IEEE International Conference on Computer Design*, Sept. 2012, pp.94–101. DOI: 10.1109/ICCD.2012.6378623.

[34] Luo Y X, Cai Y, Ghose S, Choi J, Mutlu O. WARM: Improving NAND flash memory lifetime with write-hotness aware retention management. In *Proc. the 31st Symposium on Mass Storage Systems and Technologies*, May 30–Jun. 5, 2015, pp.1–14. DOI: 10.1109/MSST.2015.7208284.

[35] Li Q, Shi L, Gao C M, Wu K J, Xue C J, Zhuge Q F, Sha E H M. Maximizing IO performance via conflict reduction for flash memory storage systems. In *Proc. the 2015 Design, Automation & Test in Europe Conference & Exhibition*, Mar. 2015, pp.904–907. DOI: 10.5555/2755753.2757022.

[36] Tokutomi T, Tanakamaru S, Iwasaki T O, Takeuchi K. Advanced error prediction LDPC for high-speed reliable TLC NAND-based SSDs. In *Proc. the 6th IEEE International Memory Workshop*, May 2014. DOI: 10.1109/IMW.2014.6849375.

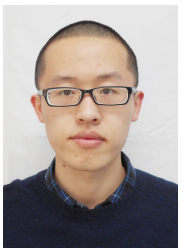
[37] Li Q, Shi L, Xue C J, Zhuge Q F, Sha E H M. Improving LDPC performance via asymmetric sensing level placement on flash memory. In *Proc. the 22nd Asia and South Pacific Design Automation Conference*, Jan. 2017, pp.560–565. DOI: 10.1109/ASPDAC.2017.7858383.

[38] Cai Y, Luo Y X, Haratsch E F, Mai K, Mutlu O. Data retention in MLC NAND flash memory: Characterization, optimization, and recovery. In *Proc. the 21st International Symposium on High Performance Computer Architecture*, Feb. 2015, pp.551–563. DOI: 10.1109/HPCA.2015.7056062.

[39] Peleato B, Agarwal R, Cioffi J M, Qin M H, Siegel P H. Adaptive read thresholds for NAND flash. *IEEE Trans. Communications*, 2015, 63(9): 3069–3081. DOI: 10.1109/TCOMM.2015.2453413.

[40] Zhang M, Wu F, He X B, Huang P, Wang S Z, Xie C S. REAL: A retention error aware LDPC decoding scheme to improve NAND flash read performance. In *Proc. the 32nd Symposium on Mass Storage Systems and Technologies*, May 2016. DOI: 10.1109/MSST.2016.7897085.

[41] Aslam C A, Guan Y L, Cai K. Retention-aware belief-propagation decoding for NAND flash memory. *IEEE Trans. Circuits and Systems II: Express Briefs*, 2017, 64(6): 725–729. DOI: 10.1109/TCSII.2016.2602359.



Shi-Qiang Nie received his Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, in 2021. He is currently a postdoctoral researcher with the Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an. His research interests include non-volatile memory, architecture optimizations, and distributed storage systems.

Chi Zhang is currently pursuing his Ph.D. degree at the Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an. His research interests include performance optimization of SMR drives, garbage collection strategy in the hybrid storage system, and coarse-grained cache replacement algorithm that mitigates the SMR write amplification to optimize read/write cache hit ratio and to reduce replacement overhead.

Wei-Guo Wu received his B.S., M.S., and Ph.D. degrees in computer science from Xi'an Jiaotong University, Xi'an, in 1986, 1993, and 2006, respectively. He is currently with the Department of Computer Science and Technology at Xi'an Jiaotong University as a professor. His research interests include high-performance computer architecture, storage system, cloud computing, and embedded system.