Layer RBER Variation Aware Read Performance Optimization for 3D Flash Memories

Shiqiang Nie
Xi'an Jiaotong University
Xi'an, China
nsqiang@gmail.com

Youtao Zhang
University of Pittsburgh
Pittsburgh, USA
zhangyt@cs.pitt.edu

Weiguo Wu
Xi'an Jiaotong University
Xi'an, China
wgwu@xjtu.edu.cn

Jun Yang
University of Pittsburgh
Pittsburgh, USA
juy9@pitt.edu

***Abstract*—3D NAND flash enables the construction of large capacity Solid-State Drives (SSDs) for modern computer systems. While effectively reducing per bit cost, 3D NAND flash exhibits non-negligible process variations and thus RBER (raw bit error rate) difference across layers, which leads to sub-optimal read performance for applications with either small or large I/O requests. In this paper, we propose LRR, Layer RBER variation aware Read optimization schemes, to address the challenge. LRR consists of two schemes — LRR subpage read scheduling (SRS) and LRR fullpage allocation (FPA). SRS groups small read requests from the layers with similar RBERs to reduce the average read latency of subpage sized read requests. FPA distributes the data of a large write to multiple layers, which improves the read latency when reading from layers with large RBERs. Our experimental results show that our proposed scheme LRR reduces 46% read latency on average over the state-of-the-art.**

***Index Terms*—3D NAND flash, read performance, unbalanced bit error rate, parallel sub-page read operation**

I. INTRODUCTION

3D NAND flash is one of the most promising technologies to increase the package density and meet the increasing demands for large capacity storage of modern computer systems. While it effectively reduces per bit manufacturing cost, 3D NAND flash exhibits many characteristics that are different from planar flash. 3D NAND flash module often uses smaller number of chips, resulting in reduced chip level parallelism. 3D NAND flash exhibits non-negligible process variations and thus RBER (raw bit error rate) difference across layers. In addition, 3D NAND flash often uses 16KB flash page while the file system may still use 4KB logic page size. In this paper, we use **subpage** and **fullpage** to refer to 4KB and 16KB pages, respectively.

To address reduced parallelism and page size disparity between logic and device pages, Kang *et al.* proposed to merge subpage write requests to fullpage write requests in the write buffer to reduce the number of NAND writes and read-modify-write (RMW) operations for improving the write performance [1]. Kim *et al.* utilized the subpage programming (SP) technique to mitigate writing FTL metadata for prolonged chip lifetime [2]. Kim *et al.* proposed erase-free subpage programming (ESP) to enable programming the same page multiple times for small writes, which reduces the overhead of garbage collection [3]. Liu *et al.* proposed SOML read, which groups subpage sized read requests from different layers to improve the operation parallelism and thus the read performance [4].

Given current flash chips widely adopt LDPC ECC (Error Correction Code) [5], larger RBER leads to more read retries and thus significantly longer read latency. For example, recent studies showed that, with the same retention time and P/E cycles, a page from the most reliable layer may succeed in one read retry while the one from the least reliable layer may need six or seven retries. To address this issue, Du *et al.* proposed multi-granularity LDPC to adapt to speed difference of each layer for read performance improvement [6]. Shim *et al.* proposed to utilize process similarity for both read and write performance improvement by retrenching the incremental step pulse programming and keeping track of the read level.

Unfortunately, the layer-to-layer RBER variation remains a major concern for 3D NAND flash. In particular, the requests from modern applications exhibit a wide mix of small (subpage size) and large (fullpage or bigger sized) I/O requests. Existing schemes lack the ability to improve both types of requests. In this paper, we take proactive designs to improve read performance for both types of requests. We summarize our contributions as follows.

- We propose subpage read scheduling (SRS) scheme that exploits layer RBER variation to improve the read performance of small I/O requests. SRS groups small read requests from the layers with similar RBERs to reduce the average read latency of subpage sized read requests.
- We propose fullpage allocation (FPA) scheme that exploits layer RBER variation to improve the read performance of large I/O requests. FPA distributes the subpages of a fullpage write to different layers, which improves the read latency when reading from layers with large RBERs.
- We evaluate the proposed schemes with widely adopted workload traces. Our experimental results show that, on average, our scheme reduces 46% read latency over the state-of-the-art.

This work is supported in part by the National Key Research and Development Program of China under Grant 2016YFB1000303, in part by the National Science Foundation of China under Grant 61972311, in part by NSF under Grant CCF-1718080, CCF-1910413, CCF-1725657 and CCF-1617071. The work of S. Nie is supported by the Chinese Scholarship Council under Grant 201806280273.

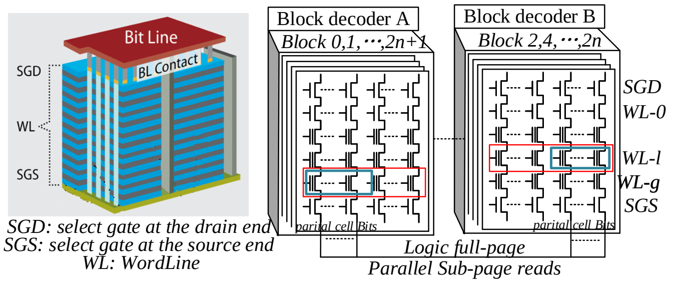


Fig. 1. The structure of 3D NAND flash and parallel sub-page read circuit.

In the rest of the paper, Section II discusses the 3D NAND flash background and motivates our design. Section III presents the detailed scheme. Section IV describes the experiment methodology and analyzes the results. Section V concludes the paper.

II. BACKGROUND AND MOTIVATION

In this section, we breifly discuss the 3D NAND flash architecture and the parallel sub-page read operation. We then motivate our designs with RBER variations across different layers in 3D NAND flash.

A. 3D NAND flash Organization and SOML Read

3D NAND flash boosts storage capacity by stacking memory cells vertically. There are several cell types. In this paper, we adopt Charge-Trap (CT) 3D NAND flash. As shown in Figure 1, a 3D NAND flash chip consists of multiple layers with each layer consists of multiple block segments, and each block segment consists of multiple wordlines (WLs). One wordline, also referred to as a page, is at least 16KB in 3D NAND flash. The aligned block segments from all layers form a block. A read operation reads one flash page, i.e., 16KB, from one block. To simplify discussion, we use **subpage** and **fullpage** to refer to 4KB and 16KB pages, respectively. A fullpage consists of four subpages.

Given modern OSes widely adopt 4KB logic page size, the I/O requests from the file system exhibit a mix of small (4KB or 8KB) and large (16KB or bigger) requests. Liu *et al.* found that the majority requests of many workloads are small requests [4] while the widely used sever traces from Microsoft [7] contain many large requests. Based on the observation that reading 16KB flash pages to service small I/O requests is unnecessary and tends to result in low throughput, Liu *et al.* proposed SOML (single-operation-parallel-read) to enable the parallel read of multiple subpages.

In Figure 1, each red box represents a page (fullpage) while each blue box represent a subpage. Instead of reading one fullpage, with simple hardware enhancement, SOML reads four subpages from different blocks and/or from different layers. Due to hardware constraints, the subpages forming one SOML read should not share either the same bitlines or the block decoders. SOML speeds up read performance by providing subpage level parallelism.

B. RBER and LDPC ECC

3D NAND flash exhibits non-negligible layer-to-layer processing variations. Figure 2(a) illustrates the normalized RBERs (raw bit error rates) for a 48-layer flash chip [8]. From the figure, the variations vary with P/E cycles and retention time. For flash pages at 2K P/E cycles and one year retention, the RBER of the worst layer is about 2× that of the best layer.

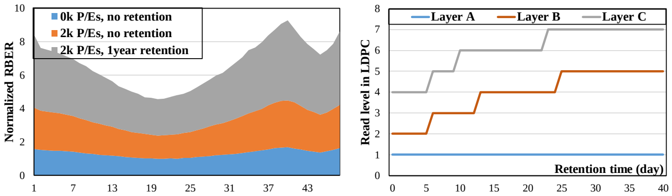


Fig. 2. Variation of RBER across layers and the LDPC read level of three typical layers with increasing retention time[6, 8]

To ensure data reliability, FTL (Flash Translation Layer) often allocates part of Out Of Band (OOB) space as ECC area. Modern flash chips widely adopt LDPC ECC (error correction code) to improve data reliability. The LDPC ECC can be applied at different granularity, e.g., 16kB fullpage, 4KB subpage, or 2KB chunk. In this paper, we assume each 4KB subpage has an individual LDPC ECC (as it is widely adopted in mainstream SSD chips [5]).

LDPC reads the flash page one or multiple times (up to seven times) if there are errors in the page. Each read is referred to as one retry. The more there are errors, the more retries the read operation needs. Thus, reading pages from layers with larger RBERs tends to suffer from more read retires and thus longer read latency. Figure 2(b) illustrates a snapshot of three layers (from [6]). While layer A can always finish read operation in one retry, layers B and C need more retries. With 15-day retention time, layer C needs six retries while layer B needs four. To simplify the discussion, the pages residing in layers with small and large RBERs are referred to as **strong** and **weak** pages, respectively.

In this paper, we focus on the large RBER variations and propose optimization schemes to improve the read performance for applications with either large or small I/O requests.

C. Motivation

We study the impact of layer RBER variation on read requests from popular workloads. We have two key observations as follows.

(1) RBER variation introduces intra-operation idleness. Given reading flash pages with larger RBER needs more retries and thus suffers from longer read latency, reading subpages from different layers simultaneously cannot finish at the same time. The subpage read that finishes early has to wait for the others. For example, assume we have four subpage requests A, B, C, and D. As shown in Figure 3, A and B are strong subpages while C and D are weak subpages. Assume SOML can read two subpages simultaneously. If we group A and C, then A shall finish early so that while A finishes, it has to wait for C. While waiting, no other requests can be scheduled to start. Similarly, B shall wait if we schedule B and D together.

As a comparison, if we schedule A and B together, and C and D together, the subpage reads within one SOML read

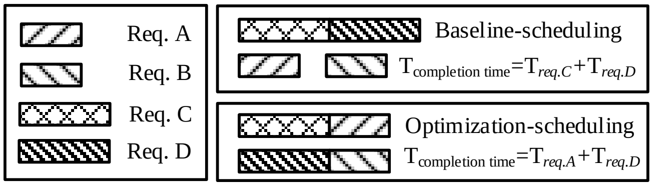

Req. A
Req. B
Req. C
Req. D
Baseline-scheduling
$T_{\text{completion time}}=T_{req.C}+T_{req.D}$
Optimization-scheduling
$T_{\text{completion time}}=T_{req.A}+T_{req.D}$

Fig. 3. Layer RBER aware scheduling is beneficial.

can finish around the same time such that the intra-operation wait time can be minimized, which improves the overall read performance, as shown in Figure 3.

(2) RBER variation introduces large read latency deviations. While different layers exhibit significant RBER difference, existing FTL allocates the same ECC space to subpages at different layers. For one subpage (4KB) that has an individual LDPC ECC, the larger the ECC space is, the stronger capability the ECC has. In particular, at a fixed RBER, the one that has larger ECC space needs fewer read retries. For example, each of two subpages has x errors, the one that has y-byte ECC cannot correct it after one read retry while the other that has z-byte ($z>y$) has stronger error correction capability and thus can correct all the errors with one read retry. The former has to go through more retries while the latter can finish the operation. This observation advocates the allocation large ECC space to protect weak pages. However, given OOB space is a precious resource, it is not always preferable to allocate more space as it restricts its use for other optimizations.

On the other hand, the strong pages accumulate few errors, for example, Figure 2(b) shows that the page from the best layer needs just one retry to finish the read operation, indicating the under-utilization of its allocated ECC space. A recent study [9] shows that the required ECC sizes for different pages can differ by 1.3× if providing the same correction capability. Intuitively, it would be beneficial if we allocate a small portion of the ECC space from strong subpages to weak subpages. That is, if we have a strong subpage A and a weak subpage B. We may shrink A's ECC space such that B's ECC spread in the OOB space of both pages. Given B has larger ECC space, it has stronger error correction capability and thus reduces the number of read retries.

However, such an allocation tends to increase the number of subpage reads and writes. In particular, reading and wrting B needs to access the OOB space of both pages, which may offset the benefits that we gain from the reduction of read retries. The problem, referred to as read and write amplification, has to be properly handled at runtime.

Previous studies on planar SSD have explored the RBER difference across different bits in one MLC or TLC cell. The bit interleaving scheme improve the read performance at average via interleaving the data from each logic page into the different types of the physical page, which amortizes the RBERs at the page level[5, 10–12]. Unfortunately, it introduces large read amplification; the overlong ECCs could increase the lifetime and reliability as well as read performance via expanding the capacity of ECC beyond the OOB. However, it shrinks the available capacity of SSD[12–14].

TABLE I
STATISTICS OF WORKLOADS.

trace	r/w ratio	avg. read	avg. write	size <=4k	size <=8k	size <=16k	size >16k
24HRS8	0.26	85.78	12.41	0.20	0.42	0.16	0.23
BS78	0.90	4.53	35.27	0.96	0.01	0.01	0.02
casa2	0.00	5.42	4.00	1.00	0.00	0.00	0.00
HM_1	0.97	18.15	22.86	0.00	0.01	0.87	0.12
mds_1	0.93	60.09	13.84	0.10	0.01	0.01	0.89
stg_1	0.64	59.54	7.88	0.32	0.04	0.02	0.62
web_1	0.54	45.90	9.22	0.48	0.05	0.03	0.45
webr9	0.00	4.00	4.00	1.00	0.00	0.00	0.00
webu8	0.11	5.31	4.00	0.99	0.00	0.00	0.00
USR_0	0.59	47.42	13.55	0.04	0.47	0.10	0.39
PROJ_1	0.91	43.43	22.23	0.00	0.34	0.04	0.62
PROJ_3	0.90	15.03	30.14	0.00	0.81	0.04	0.16
HM_0	0.25	11.61	11.21	0.25	0.55	0.07	0.12
RSRCH	0.10	15.70	12.70	0.05	0.65	0.13	0.17
PRN_0	0.11	26.55	13.93	0.32	0.42	0.04	0.21
STG_0	0.23	33.56	12.69	0.11	0.55	0.10	0.24
WDEV_0	0.20	16.57	12.11	0.06	0.68	0.07	0.18
PRN_1	0.69	18.32	13.78	0.00	0.64	0.16	0.20

In summary, the above observations suggest that we devise novel techniques to exploit large layer RBER variations to improve read performance on 3D NAND flash.

III. THE DESIGN

Given the default logic page size is 4KB in modern OSes, the I/O requests from modern applications exhibit a wide mix of different lengths. Table I characterizes the widely adopted workloads from Microsoft research Cambridge and OpenStor[7, 15]. From the table, while some applications show dominant small I/O requests, many others have a large portion of requests that are 16KB and above. Some applications have majority 8KB I/O requests. Thus, it is important to optimize the I/O requests at different granularities.

An overview. In this paper, we propose the layer RBER variation aware (LRR) scheme to optimize the read performance on 3D NAND flash SSDs. It consists of the following two designs to address the problems in the key observations.

- *LRR subpage read scheduling (SRS).* LRR-SRS improves the read performance of small I/O requests. It groups small read requests from the layers with similar RBERs to reduce the average read latency of subpage sized read requests.
- *LRR fullpage allocation (FPA).* LRR-FPA improves the read performance of large I/O requests. It distributes the subpages of a fullpage to different layers such that the ECC space from strong subpages can be allocated to weak subpages for improved error correction capability.

A. The LRR Subpage Read Scheduling Scheme

The LRR subpage read scheduling scheme is designed to enhance SOML read [4] when scheduling subpage sized read requests. SOML adopts a set of scheduling rules when composing a SOML read with subpages from different layers, e.g., no two subpages can share the same bitlines. LRR-SRS optimizes the scheduling with a set of new rules.

Rule 1: Prioritize requests for subpages from layers with the same/similar RBERs. Similar as that in SOML scheduling, we place all I/O requests in one scheduling queue for each channel. To compose a SOML read, we mark the old request as the main request and satisfy its subpage demand first. We then scan the scheduling queue to choose the subpage requests that can be combined with the main request. In choosing the other requests, we prioritize these that ask for subpages from layers with same or similar expected RBERs.

For example, Figure 4 shows a set of subpage sized requests A, B, C, and D. A is the oldest request. Assume B, C, and D ask for the subpages that share the bitlines and thus they cannot be scheduled simultaneously. While it is possible to construct a SOML read from A and one of B, C, and D, we prefer to choose B as A and B are from the same layer such that they share the similar RBER. We would expect the two subpages finish at about the same time.

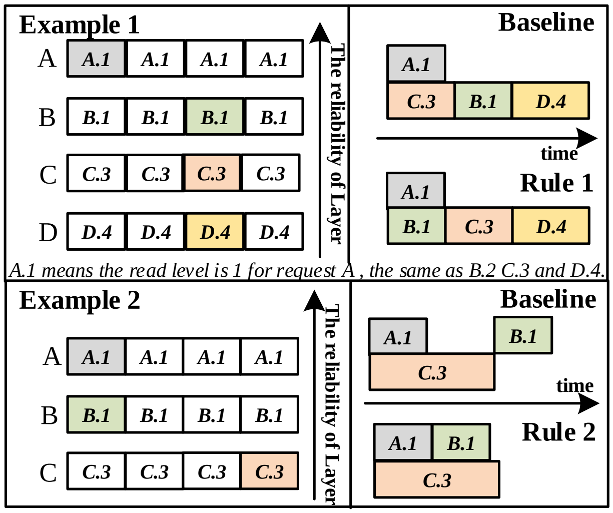
Rule 2: Preemptive read retries. It takes a long latency to read weak subpages as reading such subpages takes many read retries. After each read retry, FTL sets the control register in the flash chip [16] such that each cell is sensed with increased sensing levels. For example, a flash read may use 1, 2, and 3 sensing voltages to differentiate the two neighboring states saved in one flash cell. Given each read retry is controlled individually, we can preempt one SOML read in the middle, construct a new SOML read with the unfinished subpage requests and new subpage requests, and schedule the new SOML read with increased sensing levels.

We use the example #2 in Figure 4 to illustrate how it works. Assume we need to schedule three I/O requests A, B, and C. While A and B read two strong subpages (that can finish in one read retry), C reads a weak subpage (that needs 3 read retries). Assume the subpages requested by A and B conflict and thus cannot be scheduled simultaneously. When we schedule A and C in the first SOML read, A can finish in one read retry but C cannot. Instead of finishing C in three read retries, we preempt the SOML read, and return the subpage that A requested. We then construct a new SOML read with B and C, and schedule the new SOML read starting with 2 sensing voltages to differentiate every two neighboring states. We can then finish B in the next read retry.

Comparing to the LRR oblivious scheduling, we read B with more than necessary sensing levels, and thus slow down its sensing time. However, by combining B with C, we eliminate the long queuing delay that B would otherwise have.

B. The LRR Fullpage Allocation Scheme

While LRR-SRS helps to improve the read performance of subpage sized requests, many requests ask for two or more subpages. As shown in Table I, for some applications, e.g., HM-1, all their I/O requests are large requests (i.e., 16KB or more). Given the page size is 16KB in 3D NAND flash, a natural page allocation strategy would allocate 16KB logic data to a flash page (16KB). If such a page resides in a layer with large RBER, the subsequent read requests to this page tend to suffer from long read latency as it takes more read retries.



A.1 means the read level is 1 for request A , the same as B.2 C.3 and D.4.

Fig. 4. An example of LRR Subpage Read Scheduling Scheme.

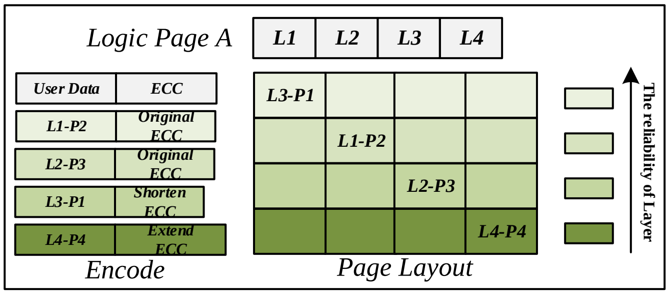


Fig. 5. An example of the LRR Fullpage Allocation Scheme.

To exploit layer RBER variation across different layers, we propose LRR fullpage allocation scheme that works as follows. Given 16KB user data in an I/O request, we split the data to four subpages, and program the four subpages (L1, L2, L3, L4) into four layers that have different RBERs. Assume subpages P1, P2, P3, and P4 are from four layers with increasing RBERs, i.e., P1 is the strongest subpage while P4 is the weakest subpage, as shown in Fig.5. These four subpages do not have hardware conflicts and thus can be scheduled in one SOML read. Our proposed LRR-FPA scheme decides to allocate L1 and L2 to P2 and P3, respectively; and L3 and L4 to P1 and P4, respectively. To achieve layer RBER variation awareness, we shrink the ECC space of P1, and allocate the saved space to increase the ECC capability of P4. That is, P4 has a larger ECC space that has stronger error correction capability. We leave the ECC space for P2 and P3 untouched. The existing work shows that, two pages with 2.7 times RBER difference (i.e., the RBER of both two pages are 0.0078 and 0.0028 respectively) achieve the same decoding efficiency by sharing about 225-bits ECC space from the strong page to the weak page (i.e., the default ECC is 512B per 4KB user data), leading to reduced read latency at average [9]. This ECC rearrangement is proposed as a proof-of-concept. We will devise more strategies in the future studies.

When we have a large I/O request that need to fetch the 16KB data, we construct a SOML read and read all subpages. Given P1 has small RBERs, e.g., it always needs one retry to finish the operation [6]; and P2 and P3's ECC space is not touched, we assume they can return L1, L2, and L3 in one read retry. For P4, its RBER is high, a normal sized ECC would need, e.g., three read retries. However, by expanding P4's ECC to P1 OOB, we achieve a stronger ECC and thus may finish the reading also in one read retry. In this way, we strive to achieve improved read performance for large I/O requests.

However, splitting a fullpage to four subpages may face two issues as follows.

1) Addressing: One issue with LRR-FPA is that enforcing addressing mapping at subpage level could lead to significant increase of FTL mapping table. Given that the table is already large for TeraByte SSDs, we exploit the following design to minimize the size increase of the mapping table. We partition the data from the user application to 16KB size pieces — each 16KB piece is referred to as a fullpage request; those whose sizes are smaller than 16KB before or after partitioning are referred to as small requests. We divide the flash blocks to two groups — those that are for handling fullpage requests and those for small requests. For small requests, we create the mapping the same way as that in the baseline.

For fullpage requests, we allocate them to blocks reserved for fullpage writes. For a 48-layer 3D NAND flash, we divide the layers to four groups and have blocks reserved in all groups. For discussion purpose, we assume layers 12-23 are the strong layers and layers 36-47 are the weak layers; layers 0-11 and 24-35 are two other groups. The ECC of subpages on layer 36-47 expands to subpages on layers 12-23, respectively.

When we write fullpages, the first fullpage chooses subpages from layers 0, 12, 24, 36; while the second fullpage chooses subpages from layers 1, 13, 25, 37. Similar as that we discuss in the example, the logical four subpages L1, L2, L3, L4 are written to layers 0, 24, 12, 36, respectively. The reason that we write L3 instead of L1 to layer 0 is that, we observe that the applications have a large number of read requests that ask for 8KB data. Such requests can be naturally supported by reading the strong and weak layers simultaneously.

2) Read Amplification and read performance degradation: Another issue with LRR-FPA is that we may have an I/O request that asks for one weak or strong subpage only. For example, if we request L4, we would still need to read P1 and P4 as part of P4's ECC is saved in P1's OOB space. This leads to read amplification; if we request L3, since we shrink the P1's ECC capacity, the read performance may degrade sometimes for additional read-retries. However, the impact on read performance can also be neglected according to the experiment result in the experimentation section. That's because, assuming any one of these subpages could be read equiprobably, only about 6% possibility that the L3 is selected alone, leading to read performance degradation, while 94% possibility that the SSD benefit from this design. We study the workloads and summarize our findings in Figure 6. In the experiment, we collect the read amplification if we adopt LRR-FPA allocation scheme. Figure 6 presents the percentage of reads that need to read one more page. From the figure, we observe that, for most applications, the read amplification is modest, i.e., less than 15%, while the read amplification of other schemes range from 1 to 1.7[12].

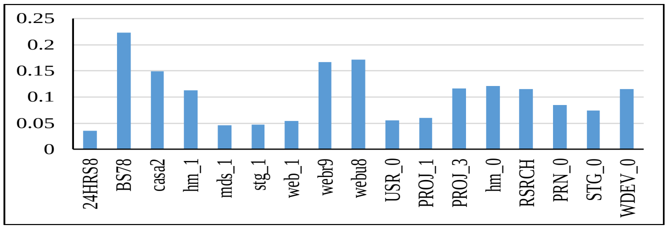


Fig. 6. The read amplification across these workloads.

TABLE II
SSD PARAMETERS.

48-layer 3D NAND chip parameters	
(Channel, Chip, Die)	(4, 8, 8)
(Plane, Block, Page)	(8,1888,1792)
(Page size, Cell density)	(16KB, TLC)
Latency	write 900us, Erase 10ms
Latency	read(90us,120us,180us)
Capacity per page	16KB
Smallest subpage read size	4KB
Max parallel subpage reads	4

IV. EXPERIMENTAL EVALUATION

A. Experimental Setting

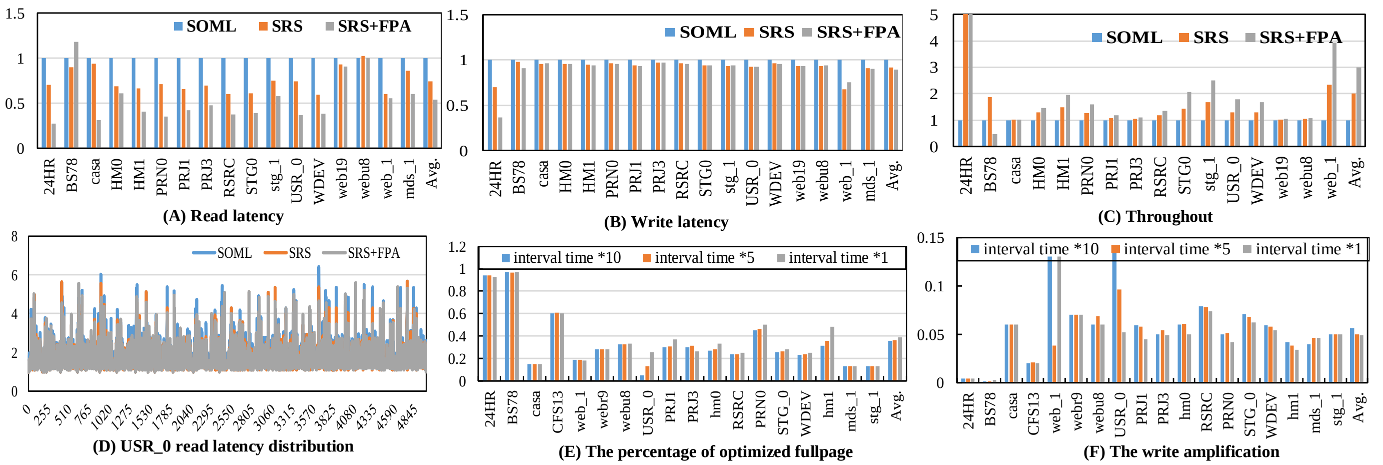
To evaluate the effectiveness of the proposed scheme, we implemented both the proposed scheme and SOML scheme based on SSDsim, which has been validated against hardware platform[17]. In our experiments, the configure parameters of SSD are adopted from [4]. Table II provides the detailed configuration of the 3D NAND flash-based SSD. We used the enterprise servers traces from Microsoft research Cambridge and OpenStor[7, 15] to evaluate the proposed scheme. These workloads are widely used in previous studies. The details of these workloads are given in Table I.

In this section, we compare the following schemes.

- Baseline — We chose the state-of-the-art SOML scheme as the baseline. It support parallel read of multiple subpages from different layers.
- SRS — This scheme implements the LRR subpage read scheduling scheme.
- SRS+FPA — This scheme implements the LRR fullpage allocation scheme on top of SRS.

B. Performance Comparison

Fig. 7 compares the performance among different schemes. The results are normalized to baseline. From the figure #a, SRS and SRS+FPA achieve large performance improvement over the baseline, i.e., we achieved 10% to 73% improvements. For workloads whose majority I/O requests are small requests, SRS and SRS+FPA have similar improvements. For workloads whose majority I/O requests are large ones, SRS+FPA does better. For workloads whose majority read and write I/O requests are small requests, e.g., PROJ_1 and USR_0, SRS shows small or negative improvements over the baseline. The negative results are due to complex request reordering



(A) Read latency

(B) Write latency

(C) Throughout

(D) USR_0 read latency distribution

(E) The percentage of optimized fullpage

(F) The write amplification

Fig. 7. Various comparisons between SOML, SRS and SRS+FPA.

algorithm. By adopting fullpage allocation optimization, we achieve larger flexibility by mitigating the unbalance read latency across different layers. Fig.7 #b compares the write performance between the proposed scheme and the baseline. Although the write performance does not benefit from the proposed scheme directly, the write latency is also reduced significantly. This is because, in SSDs, the writes typically have lower priority than the reads. As a result, the writes need to wait for the completion of the queued reads. Consequently, shortening the overall read latency shortens the write latency as well. Figure7 #c shows the throughout comparison among these three schemes. SRS+FPA achieves higher throughout than the baseline. Figure7 #d also shows that the read latency distribution of SRS and SRS+FPA are much smaller than the baseline for the first 5000 read requests of USR_0.

C. Sensitivity Analysis

Figure 7 shows the percentage of fullpage read optimization comparison with different loads and the lifetime overhead across several workloads. We change the interval time of requests to explore how the loads impact on the proposed scheme. In particular, we increase the interval time between any neighboring requests by 10 times and 5 times. Compared with the original one, the percentage of fullpage write requests, which could utilize LRR-FPA, does not fluctuate significantly. It shows the robust stability of the proposed scheme; then we compare the write amplification overhead of the different loads in terms of average page usage. It is observed that our proposed scheme induces additional page usage. However, since the number of generated page usage is minimal, the impact on worsening reliability is also slight. The evaluated results show that there are only 4% additional page usage on average compared with the baseline.

V. Conclusion

In this paper, we propose two schemes that exploit layer RBER variations to improve read performance for 3D NAND flash. The two proposed schemes help to address the read performance degradation from workloads with wide ranges of different I/O requests. Our experimental results show that read performance can be improved by 46% compared to the state-of-the-art.

References

[1] M. Kang *et al.*, “Subpage-aware solid state drive for improving lifetime and performance,” *TC*, vol. 67, no. 10, 2018.

[2] J.-H. Kim *et al.*, “Subpage programming for extending the lifetime of nand flash memory,” in *DATE*. IEEE, 2015.

[3] M. Kim *et al.*, “Improving performance and lifetime of large-page nand storages using erase-free subpage programming,” in *DAC*. IEEE, 2017.

[4] C.-Y. Liu *et al.*, “Soml read: Rethinking the read operation granularity of 3d nand ssds,” in *ASPLOS*. ACM, 2019.

[5] K. Zhao *et al.*, “Ldpc-in-ssd: Making advanced error correction codes work effectively in solid state drives,” in *FAST*, 2013.

[6] Y. Du *et al.*, “Adapting layer rbers variations of 3d flash memories via multi-granularity progressive ldpc reading,” in *DAC*. ACM, 2019.

[7] N. Dushyanth *et al.*, “Write off-loading: Practical power management for enterprise storage,” *TOS*, vol. 4, no. 3, 2008.

[8] Y. Shim *et al.*, “Exploiting process similarity of 3d flash memory for high performance ssds,” in *Micro*. ACM, 2019.

[9] C. Zambelli *et al.*, “Characterization of tlc 3d-nand flash endurance through machine learning for ldpc code rate optimization,” in *IMW*. IEEE, 2017.

[10] T. Nakamura *et al.*, “AEP-LDPC ECC with error dispersion coding for burst error reduction of 2d and 3d NAND flash memories,” in *IMW*. IEEE, may 2017.

[11] S. Nie *et al.*, “DIR: Dynamic request interleaving for improving the read performance of aged SSDs,” in *NVMSA*. IEEE, 2019.

[12] Y. Zhou *et al.*, “Score: A novel scheme to efficiently cache overlong eccs in nand flash memory,” *TACO*, 2018.

[13] S. Jung *et al.*, “In-page management of error correction code for mlc flash storages,” in *MWSCAS*. IEEE, 2011.

[14] S. Wang *et al.*, “Lifetime adaptive ecc in nand flash page management,” in *DATE*. IEEE, 2017.

[15] M. Kwon *et al.*, “Tracetracker: Hardware/software co-evaluation for large-scale i/o workload reconstruction,” in *IISWC*. IEEE, 2017.

[16] Q. Li *et al.*, “Improving ldpc performance via asymmetric sensing level placement on flash memory,” in *ASP-DAC*. IEEE, 2017.

[17] Y. Hu *et al.*, “Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity,” in *ICS*. ACM Press, 2011.