IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 27, 2021, accepted May 6, 2021, date of publication May 10, 2021, date of current version May 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3078720

Layer-Aware Request Scheduling for 3D Flash-Based SSDs

JINMING XU[1], YAJUAN DU[1], AND CONG DING[2]
[1]School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China
[2]School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China

Corresponding author: Yajuan Du (dyj@whut.edu.cn)

This work was supported by self-determined and innovative research funds of WUT under Grant 2020-JSJ-A1-01.

ABSTRACT Recent development on 3D flash memories largely promotes the wide application of Solid-State Drives (SSDs) by providing larger capacity from vertically-stacked layers. However, there exist speed variations across different layers because of manufacturing process variations or physical designs, which induces new challenges to fully explore the advantages of existing SSDs, e.g. the parallel structure. This paper investigates the effect of speed variation on the parallel performance of SSDs. To balance chips' workloads, traditional method selects chips in a round-robin way. As chip queue time can be estimated by some main factors, the chip also can be chosen in a greedy way. However, because of the layer speed variation, queue time estimation model should be modified. This paper first establishes a new queue time estimation model with the awareness of the flash layer information. Then the model is used to estimate the chip queue time and to direct write requests into the chip with the least queue time. The key idea is to largely reduce queue time of each write, thus reducing the average SSD response time. Finally, this new request redirection method is evaluated on SSDsim with real world workloads. Experimental results show that our model can estimate the queue time more accurately and our new request redirection method can improve 8.3% of write queue time on average under the situation of 4 times speed variation among 16 layers.

INDEX TERMS 3D solid-state drives, flash chip parallelism, layer speed variations, queue time estimation.

I. INTRODUCTION

Multi-level parallelism of Solid State Drives (SSDs) plays an important role in SSD performance advantages [1], [2]. With the rapid development of 3D layer-stacked flash technology, 3D SSDs gradually become the main stream in SSD market because of largely increased capacity [3], [4]. However, the advantages of 3D SSDs have not been fully studied in two aspects. 3D flash shows special programming/reading speed variation characteristics from layer to layer [5]–[7]. A large number of traditional schemes e.g. data placement, request scheduling/direction and garbage collection that are designed for 2D flash cannot be aware of this variation, which induces suboptimal performance in 3D SSDs. On the other hand, because a flash block contains data in multiple layers, the block size increases significantly. Meanwhile, layer variations would occur inside one flash block, which complicates data management in 3D SSDs [8]–[10]. We study the effect of the first aspect on parallel performance that has been widely studied in existing SSD designs.

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

Existing works have promoted SSD performance by optimizing parallel designs from various views. Elyasi *et al.* [11] exploited the slack time among sub-requests and proposed a new scheduling approach to re-order sub-requests for overall request access time improvement. By estimating the slack time, the proposed algorithm can insert sub-requests into these slacks occasionally. Guo *et al.* [12] proposed a request dispatching method to exploit channel resources and to avoid the effect of garbage collection. Mao *et al.* [13] proposed a garbage collection aware request dispatching method to avoid GC conflicts. Chen *et al.* [14] established a delay model to dispatch and schedule read/write requests separately according to the obtained delay value. Cui *et al.* [15] proposed to schedule read requests according to retention time and write requests according to hotness, in order to be aware of device process variations. Liu *et al.* [16] pointed that 3D flash memory suffered from reduced utilization of chip-level parallelism when the layer information is added, inducing sub-optimal parallel performance. From these works, it can be found that delay is an important metric for SSD parallel performance and many of these works use a method to estimate delay values. Existing study has uncovered that

there exists significant layer-to-layer access speed variations in 3D flash [5], [6]. Thus, above methods may become not applicable for 3D flash because of this significant variation.

By exploiting the layer speed variation characteristic of 3D flash memory, this paper first establishes a layer-speed aware access time estimation method, and then use this model to provide more accurate chip queue time judgment in advance, i.e. computing the sum of estimated access time for requests in each chip. In order to verify the effectiveness of the proposed estimation method, a greedy request dispatching algorithm is proposed to distribute read/write requests into the chip with a short-est estimated queue time. At last, we evaluate the dis-patching method on SSDsim using real-world workloads. Evaluation results show that our method can estimate more accurate chip queue time and achieve improved 8.3% write response time and 6.5% of SSD response time on average.

Contributions of this paper are concluded as follows:

- A layer-speed aware estimation model is established to provide more accurate chip access time of 3D flash memories;
- In order to verify the proposed model, a greedy request assignment algorithm is implemented to dis-tribute sub-requests into the chip with a shortest queue time, which can verify the accuracy of this estimation method;
- The effectiveness of the proposed estimation method has been evaluated on SSDsim with real work-loads and experimental results show that 3D SSD response time can be improved.

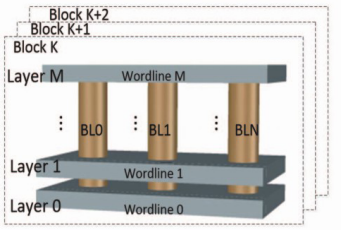
The rest of paper is organized as follows. Section II introduces the background of 3D flash memory and the motivation of exploiting layer speed variations. Section III presents our proposed model on queue time estimation and Section IV presents our boosting method that exploits this model. Section V evaluates our proposed model and method with real-world workloads. Section VI introduces some related work about 3D flash memory. Section VII concludes our work.

II. BACKGROUND AND MOTIVATION

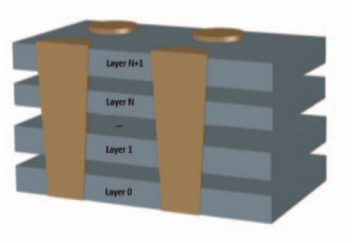
In this section, some necessary background is first presented about the basics of 3D flash memory and the characteristics of layer speed variation. Then, the parallel structure of 3D SSDs and request dispatching methods are introduced. At last, our motivation to propose layer aware access time estimation for 3D SSDs is illustrated.

A. 3D FLASH BASICS

This part introduces the charge trap (CT) based flash memory, a special type of 3D flash memory widely used in 3D SSDs, which utilizes an effective way to construct a vertical flash structure. There are multiple gate stack layers and vertical cylinder channels in 3D CT flash [10], [17], as shown in Figure 1(a). A special chemical liquid is used to erode the stacked layers. The physical properties of this liquid cause the upper layer to have a larger opening than lower layers as shown in Figure 1(b), which leads to asymmetric feature process size



(a) Physical organization.



(b) Layer speed variation.

FIGURE 1. 3D flash memory basics. Multiple vertical layers are stacked to provide a much larger capacity than planar flash (Fig. (a)) but show significant access speed variations (Fig. (b)).

across the stacked layers. The electric field strength of each layer is different, and for the larger opening, the electric field strength would be high, which induces a slower access speed. Thus, access speed on lower layers is faster than that on upper layers. This phenomenon is called the layer speed variations, as shown in Figure 1(b).

This physical phenomenon results in different strength of electric field at different layers. The access speed at the top layer could be multiple times faster than the bottom layer. Consequently, pages within the same block can have different access speeds for the sake of the asymmetric feature process size of channels. Existing works have already studied this phenomenon. For example, Chen *et al.* [5] discussed the potential drawbacks bought by this asymmetric access speed feature of 3D charge trap flash memories, and presented a fine-grained hot/cold data separation method to place data into suitable layers. Our work would exploit this phenomenon in the parallel structure to boost 3D SSD performance.

B. PARALLEL STRUCTURE

In 3D flash-based SSDs, multiple flash memory chips are partitioned and connected into multiple channels and ways. Different constituent operational units can operate in parallel, as shown in Figure 2, in which an SSD consists of multiple flash chips. Consequently, the request data can be simultaneously loaded/stored from/into the flash memories, thus providing the potential to achieve high parallel performance.The command queue is equipped for enabling the asynchronous execution of multiple I/O requests. When the number of the requests from workloads is larger than the size of command queue, I/O requests have to wait in the queue of the parallel units. Chip-level parallelism has been widely studied because that it provides flexible data storage. Thus, how to schedule the queue requests and how to dispatch a request

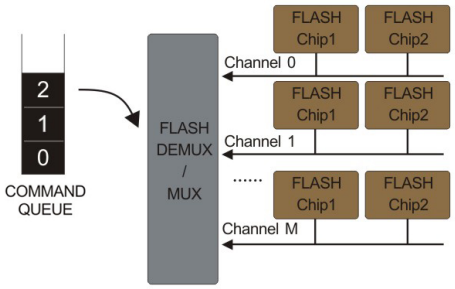


FIGURE 2. The internal parallelism of a typical SSD consisting of multiple flash channels, each of which contains multiple chips.

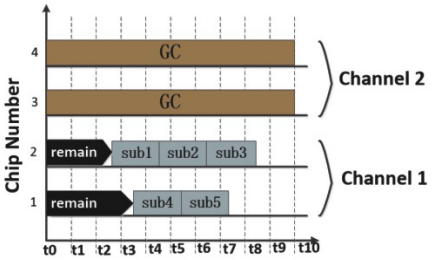
from command queue into chips would largely affect SSD performance.Nima *et al.* [18] proposed a method which can exploit intra-request slack to improve SSD performance by directing sub-request to a slack chip; Jie *et al.* [11] improved SSD performance via balanced redirected read request, they balance the workload on each chip by redirecting incoming read requests.Zhang *et al.* exploit parallelism of 3D SLC-TLC Hybrid SSD to improve write performance [19], Zhu *et al.* improve the SSD performance by parallel garbage collection [20].

During the process to exploit parallelism, write requests would be more flexible to choose different chips while the addresses of requested read data are often fixed and can't easily change. There are two main schemes to direct a write into chips, the static method and the dynamic method. The static method would allocate each address into corresponding chips during manufacture. Thus, the chip would be fixed for each write request. The dynamic method can direct requests according to current situation, which would result in more flexibility. This work belongs to a dynamic chip selection method. How to effectively dispatch requests into a suitable chip would directly influence the queue time of the current request and further the overall response time.

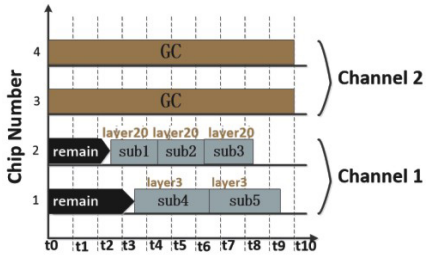
C. A MOTIVATIONAL EXAMPLE

To maximize I/O throughput of SSDs, some existing works have been proposed to schedule requests during a chip or redirect requests into new chips. These methods all rely on an estimated request access time to guide the algorithm.However, in 3D flash memory, the situation becomes different and complicated. As illustrated above, the layer of 3D flash has access speed variations, which would directly induce inaccurate access estimation and suboptimal performance. For chip redirection, the chip queue time has to be estimated. Traditional methods would decide the queue length using the overall requested data size of waiting requests or the number of these requests. However, when layer speed variation occurs, this method may be invalid.

In practice, the layer speed variation values would exist across multiple layers. Here we only give a simple example in two layers for the sake of simplifying this problem. This example is shown in Figure 3, in which the brown part represents the time for GC operation, the black part represents



(a) Traditional queue time estimation.



(b) Layer-aware time estimation.

FIGURE 3. An example to show chip dispatching schemes with and without the awareness of layer speed variations. It can be seen that two sub-figures indicate different decisions and the new model would be more accurate to guide chip directions for write requests.

the time remaining for the performing request and the gray one represents the latency time of sub-requests that are being queued. When a channel is performing GC operation, any chip under that channel can't deal with request, thus only Channel 1 is considered for requests. In the situation of Figure 3(a), the new request will be assigned to chip 1 by traditional method, because it has the shortest queue length on the surface. However, access speed variations across layers within the same block have to be considered. Assuming that the requests in Chip 2 are executed at the bottom layer, e.g. the 3rd layer, as shown in Figure 1(b), while the requests in Chip 1 are at the upper layer, e.g. the 20th layer with two times slower access speed than that in the 3rd layer, actually the overall execution time in Chip 1 can be longer than that in Chip 2, as shown in Figure 3(b). By layer-aware estimation, the queue time of chips is obviously more accurate and it proved that traditional method makes fault judgement on chip latency. Thus, the layer speed variation has an impact on calculating queue time and chip selection.

In order to solve the above problem, we propose a layer speed variation aware access time estimation model to combine the layer speed difference together with the request length, thus providing more accurate chip queue time judgement in 3D SSDs. Furthermore, in order to verify the accuracy of this estimation method, a greedy request assignment algorithm is implemented to greedily dispatch sub-requests into the chip with the shortest queue time according to estimated queue time of each chip.

III. THEORETICAL MODEL

This section presents the detailed layer speed aware model for queue time estimation. When requests come to SSDs,

they would be first decomposed into several sub-requests, sub-requests often have the size of one page. To accomplish re-direction of sub-requests, it is of utmost importance to accurately estimate the queue time of each chip. According to the actual states of SSDs, the queue time of a chip, T^{Queue}, is composed of 1) overall response time of waiting read/write sub-requests in the chip queue, $T^{Response}$; 2) the time for GC operation, T^{GC}; and 3) the remaining time of the operation to complete on the current chip, T^{Remain}, which is also shown in the example of Figure 3. Thus, Equation 1 can be first obtained to compute the overall queue time.

$$T^{Queue} = T^{Response} + T^{GC} + T^{Remain} \tag{1}$$

When there does not have enough valid pages in an SSD, the GC operation will be performed. If a channel is busy with GC operation, this channel bus is unavailable for request. When GC operation starts, T^{GC} will be a fixed large value, otherwise its value is zero.

Most SSDs are designed with two priority rules: 1) Read First (RF); 2) First Come First Served (FCFS). In the first rule, read requests have priority over write requests. In the second rule, requests that arrive earlier have higher priority over those that arrive later. In commonly used scheduling schemes, the control unit will process all read sub-requests ahead of the write sub-requests in the channel. In other words, the coming write sub-requests have to wait for all the read sub-requests within the same channel and other write sub-requests in their individual queues. According to the features above, for each chip, the queue time, $T^{Response}$ consists of both service time of read sub-requests and write sub-requests. Therefore, the response time in Equation 1 can be computed in Equations 2-4, shown as follows, in which n refers to the number of chips in a channel while $T^{Response}_{r\_sub}$ and $T^{Response}_{\omega\_sub}$ represent the response time of read sub-requests and write sub-requests, separately.

$$T^{Response} = T^{Response}_{RD} + T^{Response}_{WR} \tag{2}$$

$$T^{Response}_{RD} = \sum_{chip=0}^{n} \sum T^{Response}_{r\_sub} \tag{3}$$

$$T^{Response}_{WR} = \sum_{chip=0}^{n} \sum T^{Response}_{\omega\_sub} \tag{4}$$

In the following, the estimation model for the time of a single request is established by exploiting the 3D flash characteristic of layer speed variation. For read sub-requests, its service time consist of three phases: command transfer time, medium access time and data transfer time. Thus, the time can be estimated as Equation 5. The first phase is consumed by transferring command address, is often small and can be neglectful.

$$T^{Response}_{r\_sub} = T^{command}_{r\_sub} + T^{data}_{r\_sub} + T^{trans}_{r\_sub} \tag{5}$$

The second phase is consumed by reading data from flash to the registers of its plane, which takes varied time for different requests according to their size. However, in the real-world 3D SSDs, the variations from layer to layer is not exactly the same. In order to simply estimate the speed relation of each layer, we suppose to set variations among layers increasing by a certain percentage. Specially, because of layer access speed variations, the model should multiply a parameter to indicate speed times on the basic latency value R. For example, the access time for pages stored at the top layer will be faster than that in the bottom layer. Thus, the data access time is estimated as Equation 6 where $Speed_{layer}$ is determined by current layer as Equation 7. To simplify the model calculation, we set k a fixed value in the Equation 7, it means the variations between neighbouring layers are considered as the same.

$$T^{data}_{r\_sub} = R \times Speed_{layer} \tag{6}$$

$$Speed_{layer} = k \times Cur_{layer} \tag{7}$$

The last phase is consumed by transferring data from registers to the controller and further to the host through channel buses, which is only related to requested data size. Thus, this part of time is estimated as Equation 8, in which Tc represents the latency to transfer one KB data and $Size_{r\_sub}$ is the size of transferred data in the unit of KB.

$$T^{trans}_{r\_sub} = Size_{r\_sub} \times T_c \tag{8}$$

For write sub-requests, its service time is estimated almost the same as read sub-request, but moving the third phase in reads into the first phase. Thus, the access time for write sub-requests can be estimated as Equation 9, in which W represents the basic write latency for one page a $Size_{\omega\_sub}$ represents the size of data to write.

$$T^{Response}_{\omega\_sub} = W \times Speed_{layer} + Size_{\omega\_sub} \times T_c \tag{9}$$

Until now, all parameters in the queue time estimation model shown in Equation 1 are computed with known values. After all the queue time of chips are estimated, the target chip would be chosen as the one with the least estimated queue time, details of which are presented in Section IV.

IV. METHOD

In order to verify the proposed chip queue time estimation model, this section presents the detailed design of our layer aware chip re-direction method, named as LaCR, which exploits the new model for more effective request dispatching and improved response time. The overview of LaCR is first presented. An important module in LaCR, the layer information storage module is then illustrated. Finally, the workflow of chip re-direction is introduced.

A. OVERVIEW

LaCR makes layer-aware request dispatching at the Flash Translation Layer (FTL) firmware. The system architecture with LaCR is presented in Figure 4. When an I/O request is picked up by device interface, it will be divided into several page-sized sub-requests and then distributed into multiple

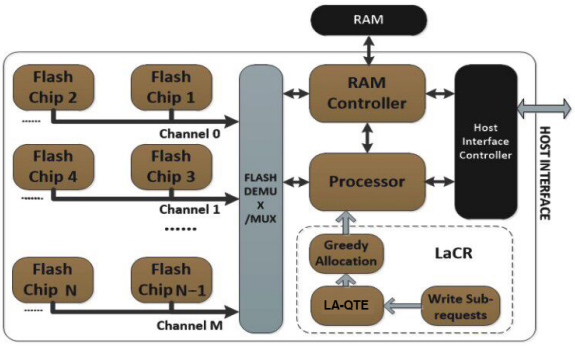


FIGURE 4. The overview of LaCR, our layer-aware chip re-direction method. LA-QTE:Layer Aware- Queue Time Estimation.

flash chips. In order to exploit its parallelism in a more effective way, we suggest to allocate write (not read) sub-requests to more suitable chips in a greedy algorithm since there is no Write-in-Place in flash. The mechanism of this allocation method is explained in Section IV-C. To decide which chip is the best choice, we conduct chip queue waiting time estimation, which will be talked in the next section.

B. OBTAINING LAYER SPEED INFORMATION

A key in our proposed LaCR method is to exploit the layer information and provide speed parameters. Firstly, for obtaining layer information, some advisable ways are provided to figure out the layer information. One direct method is to directly compute the current layer of requested address in the situation of fixed address organization. This method does not consume extra storage and is easily implemented. Thus, it is used in our work. However, it would be invalid in the situation of dynamic address mapping. Thus, the other way is to establish a relationship with extra storage, which should be carefully designed in implementation. Secondly, for the speed parameters, as existing works do not report detailed value, it is set in a progressive way. For example, accesses in the bottom layer are set to be two times slower than that in the top layer, speed variations across multiple middle layers would be increased progressively based on the speed in the top layer.

C. THE WORKFLOW OF CHIP REDIRECTION

This section introduces the workflow of our LaCR meth-od that consists of two steps: 1) computing the estimated queue time of each chip; 2) choosing the chip with the least queue time. When a write sub-request arrives, LaCR will estimate queue time of each chip exploiting the layer information of sub-requests presented in Section III. After all of the chip queue time are calculated, a greedy algorithm is used to allocate the coming write sub-request into chips, as shown in Algorithm 1. More specifically, most SSDS follow the rule of First Read-First Come First Served (FR-FCFS), whereby it is difficult to make a global optimal choice for the unpre-dictable behavior of the future read requests that can get prioritized over writes. Our greedy

Algorithm 1 Greedy Chip Selection Strategy

Input: $Write\_Sub_i$
Output: $targetChannel_i$, $targetChip_i$

```
perform queue time estimation;
chip in channel means how many chips in a channel;
Shortest_time=∞;
Token++;
for i = 0; i < chipNum; i++ do
  if Queuetime[i] < Shortestctime then
    Shortest_time = Queue_time[i];
    clear multi chip[ ] array;
    multi_chip[0] = i;
    multi_chipnum = 1;
  end if
  if Queuetime[i] == Shortest_time then
    multi_chip[multi_chip_num] = i;
    multi_chip_num++;
  end if
end for
chipID = multi_chip[Token%multi_chip_num];
targetChannel = chipID/chips in channel;
targetChip = chipID%chips in channel;
determine die and plane using static strategy;
```

algorithm chooses the best chip for the current estimated queue time and our experimental result also proved its effectiveness. When there does not have enough valid pages in an SSD, the GC operation will be performed to move lots of pages. As is known, GC operation will keep the bus busy for a long time and all the chips in this channel would not serve any coming requests. Considering the great time consumption of GC, if GC operation is performed on a channel, we will neglect its chips and choose other free chips. When the chip number and channel number are determined by our method, the die number and plane number are obtained by simple modulo calculations. It may be in the situation that there may be several chips with the same queue time. In order to deal with this issue, we keep a pointer to select a different one next time. Until now, our proposed access time estimation model and the method is fully illustrated. In the next section, the effectiveness of LaCR would be evaluated.

D. COMPLEXITY AND OVERHEAD ANALYSIS

Our proposed estimation model incorporates the layer information, which only requires negligible computational overhead as the layer information can be directly computed with simple modulo calculations. Our proposed queue time estimation and greedy chip selection method only is a linear scheduling algorithm, the complexity of which is also linear and also induces negligible overhead.

V. EVALUATION

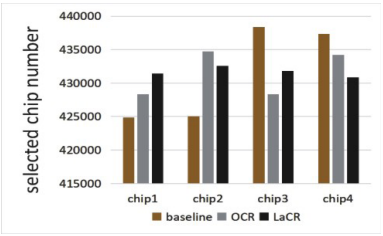
This section first presents the experiment settings to evaluate LaCR. Then, SSD performance results are presented and

TABLE 1. Ssd Configurations.

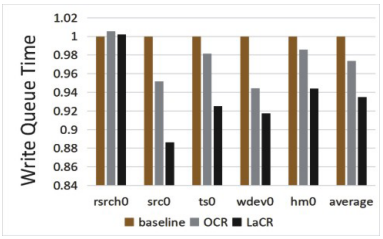
Flashsize:16GBs	Pagesperblock:64
Pagesize:16KBs	Blockerasetime:4ms
Dimension:2x2chips	Pagewritelatency:600 µs
Channelwidth:1Byte	Pagereadlatency:49µs
Transferrate:533Mbps	Interface:333MT/s(ONFI3.1)

TABLE 2. Workload Characteristics.

Workload	Requestnumber	ReadRatio
rsrch0	207587	65.69%
hm0	3993316	35.50%
src0	1557814	11.34%
ts0	1801488	17.58%
wdev0	1143261	20.08%



(a) Selected chip number.



(b) Write queue time.

FIGURE 5. Comparison of the three methods on the number distribution of selected chips. It can be seen that our proposed method indeed changes the selection method of chips and achieves write queue time improvement.

analyzed for three compared methods. At last, sensitivity study results are shown and analyzed to verify the effects of two parameters: the speed variation and the layer number.

A. EXPERIMENTAL SETUP

Our evaluation experiments are developed on an event-driven SSD simulator, SSDSim [21], which shows several page allocation strategies, thus is suitable to make changes and comparisons. Specifications and latency parameters of our modeled SSD are showed in Table 1. Five workloads with obvious differences in read/write request ratio are chosen to verify the effectiveness of our proposed method LaCR, as shown in Table 2. Three chip selection methods have been implemented and compared: 1) Baseline is a static method and the way of striping logical addresses across the channels, chips, dies and planes is in the order of Channel-first, Chip-second, Die-third and Plane-fourth. 2) Original chip redirection (OCR) is the method to allocate write sub-requests according to queue time estimation without the consideration of layer speed variations. 3) LaCR is our proposed method which estimates queue time with the awareness of layer speed variation.

Besides the parameter settings of the above basic experiment, a sensitivity study has been performed on two parameters to further verify the effectiveness of LaCR. In the basic experiment, the layer speed variation value and the layer number are set as 2 and 16, respectively. Note that the variation value is the speed times in the bottom layer over the top layer, the middle layer speed would progressively increase with the same rate. Extra values for both parameters are set to study the sensitivity and the detailed results would be discussed in Section V-C.

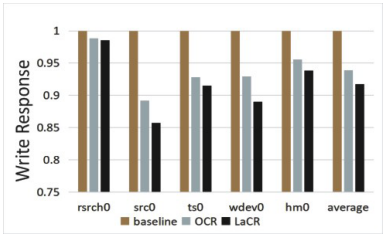
B. SSD PERFORMANCE

The experimental results of SSD request response time are shown in Figure 5 with the results of selected chip number and write queue time and Figure 6 with the results of write response time and overall response time. According to these results, four observations can be obtained. Firstly, we can see that, compared with the basic static method, the other two methods OCR and LaCR both improve parallelism greatly and can reduce IO request response time by 3.8% and 6.5% on average respectively. The improvements for write request are more obvious with 6.2% and 8.3% on average respectively, shown in Figure 5(b). This is because they both take the queue time of chips into consideration.

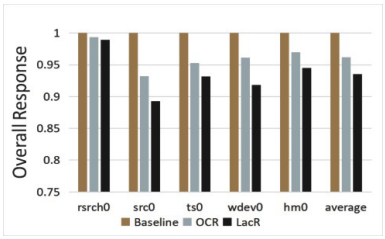
Secondly, LaCR works efficiently and achieve the best parallelism compared with other methods. It is because of significant layer-to-layer access speed variations in 3D flash. LaCR exploits this characteristic, which can provide more accurate chip queue time judgment. Chip selection results are indeed changed, as shown in Figure 5(a). Thirdly, write queue time and overall queue time are both decreased, as shown in Figure 6(a) and 6(b), which indicates that our distribution of write sub-requests can not only improve write performance, but also do no harm on overall SSD performance. At last, LaCR performs better for write intensive workloads than read intensive workloads e.g. src0 and ts0, their read ratio are less than 20%. Write response time improved by 2% within read intensive workloads while by 10% within write intensive workloads. This is because our LaCR is applied to write requests. The performance of LaCR is positively related to the proportion of write requests. These results verify the effectiveness of LaCR.

C. SENSITIVITY STUDY

This section presents the sensitivity study results of LaCR on two parameters, the range of layer speed variation and layer number. Under the condition of the maximal layer speed variation equal to 4, when layer number increases, LaCR

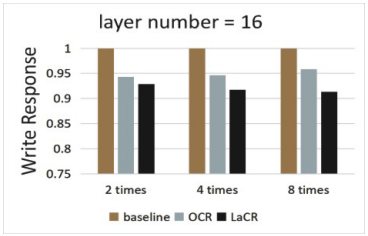


(a) Write response time.

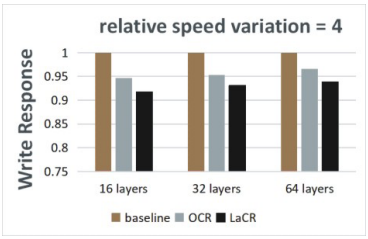


(b) Overall response time.

FIGURE 6. SSD performance of three compared methods.



(a) The speed variation.



(b) The layer number.

FIGURE 7. Sensitivity study results of LaCR on two parameters, each of which are set as two extra values. (Result data is the average value of five workloads.)

and OCR still performs better than the baseline method and LaCR always performs the best. Write performance improvements are reducing when layer numbers are increasing with the speed variation fixed. That is because the access speed variation between layers is less obvious and the queue time estimated is more similar between LaCR and OCR. Note that under the condition of a fixed layer number of 16, the performance of OCR is getting worse while LaCR is getting better when the maximal layer speed variation increases. This makes sense because the accuracy of queue time estimation in OCR is decreasing when layer speed difference aggravates. However, our LaCR method still performs well under these conditions. These sensitivity study results show that our LaCR method can improve SSD response time under various parameter settings.

VI. RELATED WORK

In the research community, many studies have been con-ducted on characteristics and performances of flash memories, especially in parallelism. Existing related works can be classified to three aspects. The first aspect aims to assign data into paralleled chips by considering current queue length. Related works in this aspect have been introduced in Section I.

The second aspect aims to minimize the interference between reads, writes and GC operations. Chen *et al.* [22] concluded that strong interference between reads and writes may have a negative effect on I/O parallelism on SSDs. Missimer *et al.* [23] presented a flash translation layer design that partitions read and write requests into different flash chips. Sun *et al.* [24] designed a buffer management strategy, called CalmWPC, to effectively reduce write performance cliff caused by GC, and shorten user response time.

The third aspect aims to ameliorate the relevant operations about address translations. Xie *et al.* [25] proposed an enhancement to the state-of-the-art DFTL approach that reduces its address translation overhead by decoupling from data access. Jin *et al.* [26] proposed a dynamic die binding policy to maximize the parallel processing capability of SSDs. Compared with these works that focus on 2D flash, this paper is the first to optimize parallel performance of 3D SSDs by exploiting access speed variations among layers.

VII. CONCLUSION

This paper studied the parallel performance of SSDs with 3D flash memories that have significant layer-to-layer speed variation. In order to improve SSD performance, we proposed to a new chip queue time estimation model by exploiting this speed variation and applied this new model into chip re-direction. Experimental results show that our method can explore the parallel structure more efficiently.

REFERENCES

[1] C. Gao, L. Shi, C. Ji, Y. Di, K. Wu, C. J. Xue, and E. H.-M. Sha, "Exploiting parallelism for access conflict minimization in flash-based solid state drives," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 168–181, Jan. 2018.

[2] M. Kim, W. Jung, H.-J. Lee, and E.-Y. Chung, "A novel NAND flash memory architecture for maximally exploiting plane-level parallelism," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1957–1961, Aug. 2019.

[3] (2018). *MICRON*. [Online]. Available: https://www.micron.com/about/blog/2018/march/the-evolution-of-nand-and-what-it -means-for-you

[4] *A. M. Research*. Accessed: Apr. 2021. [Online]. Available: https://www.alliedmarketresearch.com/3D-NAND-flash-memory-market

[5] S.-H. Chen, Y.-T. Chen, H.-W. Wei, and W.-K. Shih, "Boosting the performance of 3D charge trap NAND flash with asymmetric feature process size characteristic," in *Proc. 54th Annu. Design Autom. Conf.*, Jun. 2017, pp. 1–6.

[6] C.-H. Hung, M.-F. Chang, Y.-S. Yang, Y.-J. Kuo, T.-N. Lai, S.-J. Shen, J.-Y. Hsu, S.-N. Hung, H.-T. Lue, Y.-H. Shih, S.-L. Huang, T.-W. Chen, T. S. Chen, C. K. Chen, C.-Y. Hung, and C.-Y. Lu, "Layer-aware Program-and-Read schemes for 3D stackable vertical-gate BE-SONOS NAND flash against cross-layer process variations," *IEEE J. Solid-State Circuits*, vol. 50, no. 6, pp. 1491–1501, Jun. 2015.

[7] Y. Di, L. Shi, S.-H. Chen, C. J. Xue, and E. H.-M. Sha, "1+1>2: Variation-aware lifetime enhancement for embedded 3D NAND flash systems," in *Proc. 20th ACM SIGPLAN/SIGBED Int. Conf. Lang., Compil., Tools Embedded Syst.*, 2019, pp. 45–56.
[8] S.-H. Chen, C.-W. Tsao, and Y.-H. Chang, "Beyond address mapping: A user-oriented multiregional space management design for 3-D NAND flash memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 6, pp. 1286–1299, Jun. 2020.
[9] S.-H. Chen, Y.-T. Chen, Y.-H. Chang, H.-W. Wei, and W.-K. Shih, "A progressive performance boosting strategy for 3-D charge-trap NAND flash," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 11, pp. 2322–2334, Nov. 2018.
[10] Y. Du, Y. Zhou, M. Zhang, W. Liu, and S. Xiong, "Adapting layer RBERs variations of 3D flash memories via multi-granularity progressive LDPC reading," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.
[11] N. Elyasi, M. Arjomand, A. Sivasubramaniam, M. T. Kandemir, C. R. Das, and M. Jung, "Exploiting intra-request slack to improve SSD performance," in *Proc. 22nd Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2017, pp. 375–388.
[12] J. Guo, Y. Hu, B. Mao, and S. Wu, "Parallelism and garbage collection aware I/O scheduler with improved SSD performance," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2017, pp. 1184–1193.
[13] B. Mao and S. Wu, "Exploiting request characteristics and internal parallelism to improve SSD performance," in *Proc. 33rd IEEE Int. Conf. Comput. Design (ICCD)*, Oct. 2015, pp. 447–450.
[14] R. Chen, Q. Guan, C. Ma, and Z. Feng, "Delay-based I/O request scheduling in SSDs," *J. Syst. Archit.*, vol. 98, pp. 434–442, Sep. 2019.
[15] J. Cui, Y. Zhang, W. Wu, J. Yang, Y. Wang, and J. Huang, "DLV: Exploiting device level latency variations for performance improvement on flash memory storage systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1546–1559, Aug. 2018.
[16] C.-Y. Liu, J. B. Kotra, M. Jung, M. T. Kandemir, and C. R. Das, "SOML read: Rethinking the read operation granularity of 3D NAND SSDs," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2019, pp. 955–969.
[17] Y. Kim, R. Mateescu, S.-H. Song, Z. Bandic, and B. V. K. Vijaya Kumar, "Coding scheme for 3D vertical flash memory," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 264–270.
[18] J. Liang, Y. Xu, D. Sun, and S. Wu, "Improving read performance of SSDs via balanced redirected read," in *Proc. IEEE Int. Conf. Netw., Archit. Storage (NAS)*, Aug. 2016, pp. 1–10.
[19] W. Zhang, Q. Cao, H. Jiang, J. Yao, Y. Dong, and P. Yang, "SPA-SSD: Exploit heterogeneity and parallelism of 3D SLC-TLC hybrid SSD to improve write performance," in *Proc. IEEE 37th Int. Conf. Comput. Design (ICCD)*, Nov. 2019, pp. 613–621.
[20] G. Zhu, J. Han, and Y. Son, "A preliminary study: Towards parallel garbage collection for NAND flash-based SSDs," *IEEE Access*, vol. 8, pp. 223574–223587, 2020.
[21] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, "Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity," in *Proc. Int. Conf. Supercomput. (ICS)*, 2011, pp. 96–107.
[22] F. Chen, R. Lee, and X. Zhang, "Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, Feb. 2011, pp. 266–277.
[23] K. Missimer and R. West, "Partitioned real-time NAND flash storage," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2018, pp. 185–195.
[24] H. Sun, G. Chen, J. Huang, X. Qin, and W. Shi, "CalmWPC: A buffer management to calm down write performance cliff for NAND flash-based storage systems," *Future Gener. Comput. Syst.*, vol. 90, pp. 461–476, Jan. 2019.
[25] W. Xie, Y. Chen, and P. C. Roth, "Parallel-DFTL: A flash translation layer that exploits internal parallelism in solid state drives," in *Proc. IEEE Int. Conf. Netw., Archit. Storage (NAS)*, Aug. 2016, pp. 1–10.
[26] S. Jin and I. Shin, "State-based die binding for enhancing SSD internal parallelism," *Int. J. Appl. Eng. Res.*, vol. 12, no. 15, pp. 4825–4829, 2017.

JINMING XU was born in Wuhan, Hubei, China, in 1999. He is currently pursuing the bachelor's degree in computer software engineering with the Wuhan University of Technology. Since the beginning of 2019, he has been involved in the research of solid-state hard drives in teacher Yajuan Du's Project Group.

YAJUAN DU received the joint Ph.D. degrees from the City University of Hong Kong and the Huazhong University of Science and Technology, in December 2017 and February 2018, respectively. She is currently an Assistant Professor with the School of Computer Science and Technology, Wuhan University of Technology. Her research interests include optimizing access performance, data reliability, and persistency of flash memories and non-volatile memories.

CONG DING was born in Guiping, Guangxi Autonomous Region, China, in 1999. He is currently pursuing the degree with the Wuhan University of Technology. He research interests include computer hardware and started flash memory research in 2018. He also has a keen interest in computers and future education.