Aging Capacitor Supported Cache Management Scheme for Solid-State Drives

Congming Gao, Liang Shi, Qiao Li, Kai Liu, Chun Jason Xue, Jun Yang, and Youtao Zhang, *Member, IEEE*

***Abstract*—Solid-state drives (SSDs) have been widely adopted in embedded systems, data centers, and cloud storage due to its well-identified advantages. Inside SSD, random access memory (RAM) is adopted as the built-in cache for achieving better performance. However, due to the volatility characteristic of RAM, data loss may happen when sudden power interrupts. In order to solve this issue, a capacitor has been equipped inside emerging SSDs as an interim power supplier. But due to the capacitor aging issue, which will result in capacitance decreases over time, there still may exist data loss when power interruption occurs. Once the remaining capacitance drops to the threshold value where all dirty pages in the cache can not be written back to flash memory, data loss happens. To solve the above issue, an efficient cache management scheme for capacitor equipped SSDs is proposed in this article. The basic idea of this scheme is to bound the number of dirty pages in a cache within the capability of the equipped capacitor. The proposed scheme includes three steps: 1) a periodical dirty page budget detection (DPBD) scheme is proposed to acquire the maximal number of dirty pages that can be written back within current capability of equipped capacitor; 2) a smart dirty page synchronizing scheme is proposed during normal run time to bound the number of dirty pages in the cache; and 3) when power supply interrupts, an efficient writing back method is applied to further reduce the capacitance consumption of capacitor. The simulation results show that the proposed scheme achieves encouraging improvement on lifetime and performance while power interruption induced data loss is avoided.**

***Index Terms*—Capacitor, data loss, lifetime, performance, solid-state drives (SSDs).**

Manuscript received March 10, 2019; revised June 17, 2019 and August 27, 2019; accepted September 30, 2019. Date of publication October 25, 2019; date of current version September 18, 2020. This work was supported in part by the National Science Foundation (NSF) under Grant 1910413, Grant 1617071, Grant 1725657, and Grant 1718080, in part by the National Natural Science Foundation of China (NSFC) under Grant 61572411, Grant 61772092, and Grant 61872049, and in part by the Frontier Interdisciplinary Research Funds for the Central Universities under Project 2018CDQYJSJ0034. This article was recommended by Associate Editor C.-L. Yang. *(Corresponding author: Liang Shi.)*

C. Gao is with the College of Computer Science, Chongqing University, Chongqing 400044, China, and also with the Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15261 USA.

L. Shi is with the School of Computer Science and Software Engineering, East China Normal University, Shanghai 200241, China (e-mail: shi.liang.hk@gmail.com).

Q. Li and C. J. Xue are with the Department of Computer Science, City University of Hong Kong, Hong Kong.

K. Liu is with the College of Computer Science, Chongqing University, Chongqing 400044, China.

J. Yang and Y. Zhang are with the Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15261 USA.

Digital Object Identifier 10.1109/TCAD.2019.2949541

I. Introduction

SOLID-STATE drives (SSDs) have been widely used in embedded systems, data centers, and cloud storage due to its well-identified advantages, such as shock resistant, high random access performance, low power consumption, and lightweight form factors. In order to further improve the performance of flash-based SSDs, static or dynamic random access memory (RAM), which has tiny access latency, is equipped inside SSD to temporarily store the data. However, due to the volatility characteristic of RAM, there is a risk of data loss when sudden power supply interruption happens, leading to the degradation of the robustness of SSDs [38]. To avoid such a data loss issue, capacitor is equipped within emerging SSD as interim power supplier [3]. However, one of the key issues for the capacitor is the aging problem, which may reduce the existing capacitance of the capacitor, especially for the impact of temperature [5]. Once the remaining capacitance of the capacitor drops to a threshold value where rest capacitance can not write all dirty pages back to flash memory, data loss happens. To avoid such a problem, the most straightforward way is to strictly bound the maximal number of dirty pages in the cache. Currently, although there are several capacitance detection methods, which can accurately get the remaining capacitance of capacitor, they are too complicated to be implemented inside SSDs [17]. In this article, we propose to indirectly detect the current capacitance of capacitor so that the capacitor aging (or low-capacitance capacitor) induced data loss issue can be avoided with minimized performance and lifetime impact.

In this article, a cache management scheme is proposed, of which the basic idea is to bound the maximal number of dirty pages in the cache. There are several challenges for this scheme. First, the maximal number of dirty pages that can be written back to SSDs by existing capacitance is unknown to the cache manager. Second, dirty pages in the cache should be carefully synchronized back to flash memory to avoid introducing server write amplification or performance degradation for SSDs. To solve the above challenges, the proposed scheme is realized in the following steps. First, a periodical dirty page budget detection (DPBD) scheme is proposed, which is activated to constantly synchronize pages back to SSD until the capacitance exhausts. Second, with the dirty page budget, a smart dirty page synchronizing scheme is proposed. In the last step, a capacitance-saving writing method (CSW) is proposed, which is designed to write all dirty pages back to flash memory with less capacitance consumption. To evaluate

the effectiveness of the proposed scheme, a set of analysis and experiments are carried out on an SSD simulator [18]. The experimental results show that the proposed scheme achieves significant improvement on performance and lifetime with negligible overhead when data integrity is enhanced. The major contributions of this article are as follows.

1) Proposed a DPBD, which acquires the maximal number of dirty pages within the capability of a capacitor with negligible overhead.
2) Proposed a smart dirty page synchronizing scheme, which can avoid the negative impact on the performance and lifetime of SSDs while the capacitor aging-induced data loss problem is avoided.
3) Proposed a capacitance-saving method, which is designed for 2-D and 3-D flash memories.
4) Implemented the proposed work in an SSD simulator to evaluate the effectiveness of the proposed cache management scheme.

The rest of this article is organized as follows. Section II presents the background and related works. Section III presents the motivation of this article. In Section IV, efficient cache management is presented. The experiments are presented in Section V. Finally, Section VI summarizes this article.

II. Background and Related Works

A. Capacitor Equipped Solid-State Drives

SSD is constructed with several flash chips, which can be exploited for performance improvement of SSDs [18], [15], [14]. Inside the SSD controller, there exist several controller components. First, for DRAM cache, it is designed to temporarily store data for providing better requests access performance based on its faster access speed. For the flash translation layer (FTL), it is in charge of address translation from a logical address to physical address, determines and records the physical locations of data. For the capacitor, it is equipped inside SSD as an interim power supply, which is activated to provide a stable power supply for guaranting data integrity.

Apart from controller components, NAND flash memories contain different types of technologies (2-D and 3-D NAND flash memories). Compared with traditional 2-D NAND flash memory and 3-D NAND flash memory adopts "vertical channel," which builds a charge trap cell with a cylindrical channel [34]. This physical characteristic of vertical channel results in that pages resided in smaller opening can be written with faster speed than pages in larger opening [25]. For 2-D NAND flash memory, in this article, multilevel cell (MLC) 2-D NAND flash memory is taken into consideration, which contains two types of pages: 1) the least significant bit (LSB) and 2) the most significant bit (MSB) pages, of which the program latency is larger than LSB page [33].

B. Characteristics of Equipped Capacitor

One of the key characteristics of a capacitor is the aging issue, which will make capacitance gradually decrease over time. The reason is that the capacitor aging is visible by the increase of resistance value and the decrease of capacitance value. Normally, the industry suggests that once there is a significant capacitance reduction, the capacitor should be ended its life. In this article, the maximal loss of capacitance is set as 30% according to [5].

Currently, two types of capacitors, namely, super-capacitor [9] and Tantalum capacitor [21], have been widely adopted in advanced SSDs. For super-capacitor, its lifetime is quite sensitive to temperature. Based on the Arrhenius law lifetime model [5], the lifetime of super-capacitor is reduced by half when the temperature is increased by 10 °C [2]. This phenomenon has been verified in [16], which shows that the capacitance degradation of super-capacitors under 60 °C can be as high as 30% within five years. For the Tantalum capacitor, although it can operate with a larger temperature range (e.g., −40 °C to 105 °C) [3], the capacitance will be significantly decreased as well when the temperature falls out of the range. Based on the above discussion of the aging issue of capacitors, one can see that capacitor aging-induced capacitance decrease still is a key problem in SSDs.

C. Related Works

To solve the capacitor aging issue in SSDs, using emerging nonvolatile memory is adopted as one of the effective methods. Guo *et al.* [16] and Kim and Kang [23] proposed to equip emerging nonvolatile memory as cache to store dirty pages so that fewer dirty pages need to be written back with the support of equipped capacitor. However, there still exists an order of magnitude latency difference between nonvolatile memory and RAM. Another possible method is to write dirty pages back to SSDs as soon as possible. Huang *et al.* [19] proposed a smartbackup strategy, which aims to exploit the parallelism and LSB page for providing faster program speed [7]. Since this article does not build the relationship between the number of dirty pages in cache and the remaining capacitance of capacitor, once resided capacitor is highly aged and the number of dirty pages exceeds the capability of remaining capacitance, data loss still occurs.

III. Motivation

A. Capacitor Aging May Induce Data Loss

Based on the work presented in [11], the maximum energy of capacitor, E_{cap}, can be calculated as follows:

$$E_{\text{cap}} = \frac{1}{2} \times C_{\text{cap}} \times \left(V_{\text{cap}}^2 - V_{\text{min}}^2\right) \times m \tag{1}$$

where C_{cap} indicates current capacitance of capacitor. V_{cap} is the charging voltage of the capacitor. V_{min} is the minimum voltage for activating internal circuity normally. m is the energy conversion efficiency, of which the value is set to 0.9 in [11]. In (1), the maximum energy supplied by a capacitor is proportional to C_{cap}. In addition, based on the work in [16], the maximum number of dirty pages within the capability of a capacitor can be calculated according to the following equation:

$$E_{\text{cap}} = P_{\text{writing}} \times \frac{N_{\text{dirty}} \times \text{Page\_Size}}{\text{BW}_{\text{SSD}}} \tag{2}$$

where P_{writing}, Page_Size, and BW_{SSD} represent the power cost of each write operation, the size of dirty page, and the maximum write bandwidth of SSD, respectively. N_{dirty} represents the number of dirty pages that are going to be written to flash memory. Based on (1) and (2), we can find that the maximum number of dirty pages within the capability of a capacitor is proportional to the existing capacitance of a capacitor.

However, the capacitance will decrease with capacitor aging, which is highly related with the temperature [5]. For both, super-capacitor and Tantalum capacitor used in [1], the relationship between lifetime and temperature can be described by the law of Arrhenius [5] If SSDs are operated in an extreme environment (high temperature for super-capacitor or low temperature for tantalum capacitor), the capacitance of the capacitor will decrease sharply.

B. Dirty Page Synchronization Induced Impact

The straightforward method to avoid the capacitor aging-induced data loss is to bound the number of dirty pages in the cache, which can be easily realized by the following two methods. First, punctual synchronization (PS). PS is to synchronize dirty pages back to SSD once the number of dirty pages reaches the capability of the capacitor [22]. Second, greedy synchronization (GS) is proposed which synchronizes dirty pages during idle time of SSDs to avoid the performance impact from the above method.

To understand the impact on performance and lifetime of SSDs, we select WDEV_0, one of MSR Cambridge traces [30], as an example and implement three schemes in SSDsim simulator [18]. In the experiment, we assume the existing capacitance only can write 90% cache capacity of dirty pages. More details are presented at Section V. We evaluate the normalized access latency and write amplification. The results show that, first, PS has smaller write amplification, but introduces a high write access latency. Second, GS has much better access performance. Since most dirty pages can be synchronized back to flash memory in idle time, incoming read and write requests can be directly inserted into the cache while most pages are clean and can be evicted without writing them back to flash memory. Otherwise, if the current cache is full and incoming requests are missing, the required data of incoming requests only can be inserted into the cache when dirty pages have been evicted. But GS also induces a high write amplification.

From the observations, we can find that although synchronization can avoid data loss issues, the timing of synchronizing dirty pages should be carefully selected to reduce the impact on both performance and lifetime.

IV. Efficient Cache Management Scheme

There are three modules added in SSD controller: 1) DPBD; 2) smart synchronizing activation (SSA); and 3) CSW.

A. Dirty Page Budget Detection Module

The basic idea of DPBD is to acquire the dirty page budget by periodically synchronizing dirty pages in the cache back to SSDs via interim power suppliers. During this process, a page counter is used to record the number of synchronized pages until the capacitor is exhausted. If all dirty pages have been synchronized back to flash memory before capacitance is exhausted, clean pages should be synchronized as well. As shown in Fig. 1, dirty pages and clean pages are separated by adding a bit vector first, which is used to indicate dirty and clean pages. Then, dirty pages are synchronized first, and then clean pages are synchronized if necessary. During this process, if there are host requests arriving, they are supposed to be processed in parallel with the DBPD module by directly writing write requests to flash memory at the cost of additional capacitance consumption. Also, the value of the dirty page budget counter is increased.

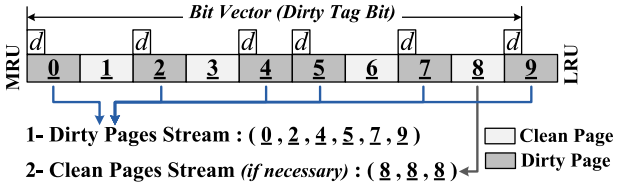


Fig. 1. Process of synchronizing pages from cache to flash memory.

However, if the capacitor is rapidly worn out due to some factors (e.g., strikingly changed temperature), the last detected dirty page budget may exceed the capability of the current capacitor, which only can synchronize a fewer numbers of dirty pages compared with the last detected budget. To avoid this case, the DPBD module should be activated more frequent so that the detected dirty page budget value can be updated in time.

The above process has two additional issues before it becomes practical: 1) the detection period between two continuous DPBD processes should be carefully determined and 2) the dirty page budget redundancy should be well determined.

First, in this article, we propose a conservative method to determine the detection period, which is designed according to the previous work [6]. Brouji *et al.* [6] presented that, after reaching 100 000 charge/discharge cycles, the capacitance of the capacitor will significantly decrease with power cycling. In order to remove the impact from charge/discharge cycles, the detection period is set to 20 days at least in the DPBD module, which is set based on the worst-case presented in [6]. In addition, for supporting the control from users, the DPBD module also can be proactively activated by the command from the host side [3], avoiding data loss in a high-reliability requirement environment. If temperature strikingly drops during two consecutive DBPDs, activating the DPBD module by user can help to detect a new dirty page budget in time so that the data loss problem can be avoided.

Normally, the detection is activated during idle time to minimize the impact on incoming requests. If idle time can not be found immediately, the following method can be used.

1) For the periodically activated DBPD module, we propose to detect idle time one more day after 20 days.

If there is still no idle time, DPBD module is activated immediately with the highest priority.

2) For proactively activated DBPD module, we believe that DPBD is more significant than host I/O requests from the view of users. Hence, the DPBD module is activated immediately.

Second, in order to minimize the impact of capacitance degradation during two continual DPBD processes, a redundant space from the acquired number of pages to the dirty page budget is added. In this article, there are five budgets, including 100%, 90%, 80%, 70%, and 0%. In the meanwhile, we set the redundancy conservatively by maintaining at least 10% redundancy for the detected dirty page budget. For example, assuming that the detected budget is 105%, then the budget is set to 90% with 15% of budget used as budget redundancy. If the data integrity requirement is not such strict, the redundancy also can be set at a finer granularity.

After each DBPD, the capacitor should be recharged again. At this time, we propose to directly write all dirty pages to flash memory without caching during the capacitor recharging process. According to [16] and [36], the worst recharging time cost will not exceed 10 s. In this article, the time cost of the recharging process is set to 10 s.

Overhead Analysis: The overhead in the DPBD module mainly comes from three aspects.

1) For write amplification, the DPBD process may generate additional write operations. However, the DPBD process is activated every 20 days. So, the generated additional write operations are negligible compared with total write requests within 20 days.
2) For time cost, according to [19], the time cost of writing 128-MB data back to SSD only needs less than 30 s. In this article, the maximum time cost of DPBD module is less than 30 s for SSD with 128-MB cache, which is negligible as well compared with 20 days. In order to further reduce the time cost, DPBD is suggested to be activated during idle time.
3) For capacitor cycling cost, in [6], the relationship between the number of cycles and capacitance is proportional. On average, each cycle only reduces 0.001% capacitance to the rated capacitance, which is negligible.

B. Smart Synchronizing Activation Module

In this section, a smart synchronizing scheme is proposed to improve both the performance and lifetime of SSDs, which is activated in two cases.

Idle Time Synchronization: In order to avoid arbitrarily synchronizing dirty pages from the cache, two new synchronization conditions are proposed. The first one is to set a budget threshold, which is used to maintain a budget redundancy so as to prevent frequently synchronizing dirty pages. Once the number of dirty pages reaches the threshold, dirty pages should be synchronized. The second one is that cold dirty pages are synchronized back to SSDs in advance during idle time for minimizing the impact on performance. Therefore, based on the above approaches, the immature synchronization can be minimized and the number of dirty pages can also be reduced. Fig. 2 shows an example for the above two thresholds. Fig. 2(a) presents the first case. Assuming the dirty page budget is 8, the current number of dirty pages is 6, and the budget threshold is 6. When the number of current dirty pages reaches budget threshold, page 9 should be synchronized back. Fig. 2(b) presents the second case. In this case, assuming the coldness threshold is 8, which indicates that pages after the 8th page are identified as cold pages (in the red frame). In this figure, page 8 identified as the cold page is synchronized back although the current number of dirty pages has not reached the threshold. Note that in this article, budget and coldness thresholds are set with 5% of dirty page budget and capacity of cache, respectively.

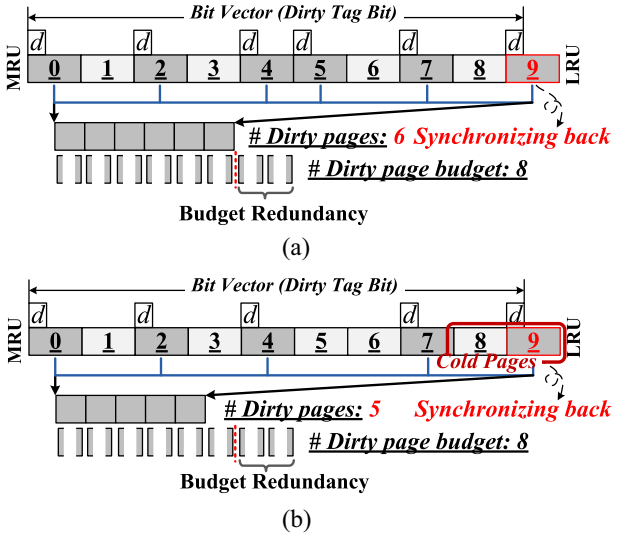


Fig. 2. Two new synchronization timings. (a) Budget threshold. (b) Coldness threshold.

Busy Time Synchronization: During busy time, we propose to evict pages based on LRU policy if the number of dirty pages does not approach the budget. Otherwise, dirty pages are synchronized with a higher priority compared with clean pages. However, during the process of synchronization, if an update request arrives while its original data is being synchronized back to flash memory, this page in the cache would be marked as dirty first. But when the synchronization process completes, this page is going to be marked as clean again. That is, the updated content of this page will loss while this page has been synchronized and marked as the clean page. To avoid this situation, write protection is used to write protect on this page before synchronizing it back to flash memory so that the data consistency can be guaranteed. Under this setting, the arrived update requests has to wait until the completion of the synchronization process. As a result, the performance will be reduced for these blocked update request due to the write protection. But the evaluated results show that the maximal percentage of blocked update requests is 0.0987% and the maximal value of normalized average write latency is 0.8981. That is, the performance impact resulted from write protection is negligible in this article.

C. Capacitance-Saving Writing Method

In this section, a CSW is proposed, which can write dirty pages with less energy cost. To maintain SSD's normal operation, the energy cost can be divided into two parts: maintaining the normal operations of components inside the controller, transferring and writing dirty pages to flash memory. The total energy cost calculation can be divided into two parts, which are presented as follows:

$$E_{\text{total}} = E_{\text{controller}} + E_{\text{program}} \tag{3}$$

$$E_{\text{controller}} = U_{\text{nor}} \times I_{\text{nor}} \times T_{\text{total}} \tag{4}$$

$$E_{\text{program}} = U_r \times I_r \times T_{\text{program}} \tag{5}$$

where E_{total} represents the total energy cost of whole SSD device. $E_{\text{controller}}$ and E_{program} indicate the energy cost of maintaining normal operations of components inside controller and the energy cost of writing dirty pages back to flash memory. U_{nor} and I_{nor} represent the required voltage and current to maintain normal operations of controller. U_r and I_r represent the required voltage and current for regular program operation. T_{total} and T_{program} indicate the total time costs of synchronization process and writing all dirty pages to flash pages, respectively. To minimize the energy cost, there are two approaches being used to achieve a minimal writing time cost. First, CSW exploits the parallelism through distributing dirty pages to all parallel units dynamically, which can reduce T_{total} by overlapping the time cost of multiple program operations. Second, dirty pages are suggested to be written into pages with faster program latency so as to reduce T_{program}. Since 2-D and 3-D flash memories have been widely used in SSD devices in recent years, the proposed CSW is designed for 2-D and 3-D NAND flash memories, respectively.

1) Writing Policy of CSW: In this section, in order to fully exploit the parallelism of SSDs, the priority order of four-level parallelism should be: 1) channel level parallelism; 2) die level parallelism; 3) plane level parallelism; and 4) chip level parallelism [18]. However, for plane level parallelism, it is not considered in the proposed CSW. The reason is that plane level parallelism should be supported by advanced command [18], [32], which has strict limitations for write operations, especially for the intraplane access locations.

Within each die, the space of free blocks is selected as the destination of evicted dirty pages. In order to quantitatively analyze the required number of flash pages per block in free page space, an equation is presented

$$\begin{aligned}\text{Num\_ReqPage} = \lceil \text{Size\_Dirty Page} \div (\text{Num\_Channel} \\ \times \text{Num\_Chip} \times \text{Num\_Die} \\ \times \min\{\text{Num\_FreeBlk}_i | 0 \le i \le \text{Num\_Plane} \\ \times \text{Default\_Num\_Blk}\} \times \text{Size\_Page}) \rceil\end{aligned} \tag{6}$$

where Num_ReqPage indicates the number of required pages for storing dirty pages synchronized from the cache, Size_DirtyPage indicates the size of total evicted dirty pages, Num_Channel, Num_Chip, Num_Die, and Num_Plane represent the number of channel, chips per channel, dies per chip, and planes per die, respectively. Num_FreeBlk_i shows the number of free blocks in die i and Default_Num_Blk represents the default number of free blocks per plane and Size_Page means the size of each flash page. While the first three levels parallelism is exploited, total Num_Channel × Num_Chip × Num_Die dies can be processed in parallel. Since the number of existing free blocks in different dies may be different, the minimal number of free blocks in all dies is determined as the number of blocks which can be used to store synchronized dirty pages for each die. Based on this setting, in each parallel round, all dies in the SSD can be used to process dirty page generated write requests simultaneously. However, if the calculated value of Num_ReqPage is larger than the default number of pages per block, more dirty pages are going to be written to more free blocks after running out of all parallel round dedicated free blocks.

In this article, the threshold of triggering GC is set as 5% and plane level GC is adopted, which is activated while the number of free pages in a plane is lower than the threshold [18], [31], [37]. Assuming that the size of SSD is 128 GB and there exist 7% OPS space. Then, totally, more than 5-GB space can be used in CSW design, which is large enough for storing all data evicted from the cache while the size of the cache is set as 1% of the SSD capacity (e.g., 128 MB for a 128-GB SSD) [24]. In addition, for the internal activities, such as GC, when sudden power supply interrupts, all internal activities are going to be stopped for avoiding consuming additional power of equipped capacitor. During the process of writing dirty pages back to SSDs, the mapping information, from logical page number to physical page number, is written into the out-of-band (OOB) space of each page for reducing the additional time cost of writing mapping information back to SSDs [19]. When the power supply recovers, the mapping information can be rebuilt through scanning the SSD devices.

2) 2-D NAND Flash Memory-Based CSW: For 2-D NAND flash memory, CSW proposes to write dirty pages back to free LSB pages greedily due to the program latency difference between LSB and MSB pages. However, if the required space of dirty pages exceeds free LSB space, free MSB pages should be used for storing more dirty pages. To follow the sequential programming constraint of flash memory, the number of MLCs programmed as SLCs should be determined first. For default, the threshold of triggering GC is set as 5% and OPS space is set as 7% [13]. Assuming that the size of SSD is 128 GB. Then, more than 2.5-GB LSB page space can be used to store dirty pages from cache for MLC SSD.

Based on (6), the number of required flash pages per block, Num_ReqPage, can be used to determine whether MSB pages should be used during the CSW process. If Num_ReqPage is smaller than the half-page number in a block, dirty pages can only be written to LSB pages via skipping MSB pages. Otherwise, some dirty pages are suggested to be written back to MSB pages. But the timing when dirty pages are written to MSB pages should be calculated in advance. The following equation is presented to show the timing when Num_ReqPage is larger than half page number in a block:

$$\begin{aligned}\text{Num\_OnlyLSB} = \text{Num\_Page} - 2 \times (\text{Num\_ReqPage} \\ - (\text{Num\_Page} \div 2))\end{aligned} \tag{7}$$

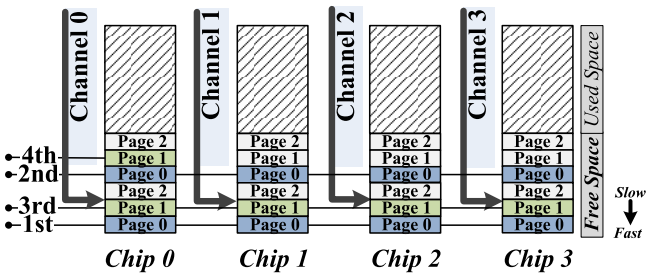


Fig. 3. CSWs for 3-D SSDs.

TABLE I
PARAMETER SETTINGS FOR THE SIMULATED SSD

Parameters	Value	Parameters	Value
Number of Channels	4	T_r (in Page)	75 us
Chips per Channel	4	T_{p_MSB} (in MSB Page)	2600 us
Dies per Chip	4	T_{p_LSB} (in LSB Page)	1300 us
Planes per Die	4	T_e (in Block)	3.8 ms
Blocks per Plane	2048	$T_{transfer}$ (in Byte)	10 ns
Pages per Block	64	OPS Space	7%
Page Size	4 KB	GC Threshold	5%

where Num_OnlyLSB indicates the number of LSB pages per block being written ahead. Num_Page presents the default number of flash pages in a block. Based on this equation, dirty pages will be written back to Num_OnlyLSB flash cells first, where only LSBs are programmed. Then, the rest of the flash cells will be programmed as MLC for writing LSB and MSB pages one by one. For the technology using MLC as SLC, it has been widely studied in previous works [20], [26], [27]. In order to avoid calculation energy cost, (2) and (7) should be calculated during runtime when the power is supplied by the host side.

3) 3-D NAND *Flash Memory-Based CSW:* Differing from 2-D NAND flash memory, 3-D NAND flash memory adopts vertical channel architecture, which builds a charge trap cell with a cylindrical channel. For each channel, it is mapped as one block in SSD. Within each channel, the cells residing on different layers are mapped as flash pages, having various program latencies. The pages in the bottom of vertical channel has the smallest program latency so that dirty pages are suggested to be written back to these flash pages greedily.

Similar to CSW on 2-D NAND flash memory, free space in 3-D NAND flash memory is used to store dirty pages as well. Since the total space of free space is limited, dirty pages may be programmed to multiple pages in each flash block. As shown in Fig. 3, assuming that there are two blocks in free space for each chip and three pages per block. Within the block, Page0 has the lowest program latency and the program latency of Page2 is set as the highest one. In this case, if there are 13 pages evicted from cache, these dirty pages are going to be written into free space in round-robin policy. In the first round, four dirty pages are programmed to pages with the lowest program latency in the first block. After that, the other four dirty pages are programmed to another block in the second round. When all flash pages with the lowest program latency have been used, dirty pages are going to be programmed back to the next page in the block. In the third round, four dirty pages are programmed to Page1s of these four blocks. Finally, the last dirty page is programmed to Page1 of the second block in Chip 0. In the worst case where all used space has been filled with data, similar to 2-D NAND flash memory, the free space is large enough for CSW to store all dirty pages on 3-D NAND flash memory.

V. EXPERIMENT

A. Experimental Setup

1) Simulated SSD Device: In this article, we use a trace-driven simulator, SSDsim [18], to evaluate the proposed framework. For the experiment, a 128-GB 2-D SSD and a 128-GB 3-D SSD are simulated, both of which are configured with four channels with each channel equipped with four chips. Each flash page size is set to 4 KB. The cache capacity is set to 128 MB. In the controller, default management mechanisms are implemented for excellent performance, including page mapping-based FTL [4], static wear leveling scheme [8], dynamic data allocation scheme [18], and greedy-based GC scheme [4], [18]. To simulate an aged SSD device, simulated SSD is warmed up before running each workload for triggering GCs easily. The warming process contains two steps: 1) each plane of the SSD is randomly filled with data from 93% to 95% to trigger GC immediately, of which 80% are valid and 2) the evaluated workload is preprocessed in the SSD to validate read data [12]. The detailed settings of the SSD are listed in Table I, where a typical flash memory is simulated [28]. The evaluated workloads include a set of MSR Cambridge traces [30], two Financial traces [35], and one personal computer traces collected via DiskMon [29]. In addition, for the program latency difference of 3-D SSD, it is set as 2× [10] while the minimal program latency is set as 1300 us. Apart from that, other parameters are the same as values listed in Table I. In order to simulate sudden power supply interruption, each workload is paused after running half way, making the simulated SSD in a stable state. After that, the power supply of the controller is switched from the host side to the equipped capacitor and the CSW module is triggered to synchronize dirty pages back to flash memory.

2) Evaluated Schemes: In the first experiment part, four schemes, including baseline, PS, GS, and the proposed smart dirty page synchronizing scheme, are evaluated. Each scheme except for baseline contains three cases, 90%, 80%, and 70%, which represent different dirty page budget values. All schemes are evaluated on 2-D and 3-D NAND flash memories, respectively. Note that, if the dirty page budget is 100%, the traditional cache management is adopted because current capacitance can write all cached data back to SSDs. If a dirty page budget is smaller than 70%, we set the budget to 0%. In this case, all schemes synchronize dirty pages back to SSDs immediately when dirty pages arrive at the cache. Therefore, when the dirty page budget is set as 0%, all four schemes will achieve the same results so this case is not taken into consideration in this part.

B. Results and Analysis

1) Results of SSA (Latency Analysis for 2-D SSDs): Fig. 4 presents the write latency evaluations of four schemes, where dirty page budgets are set as 90%, 80%, and 70%, respectively.

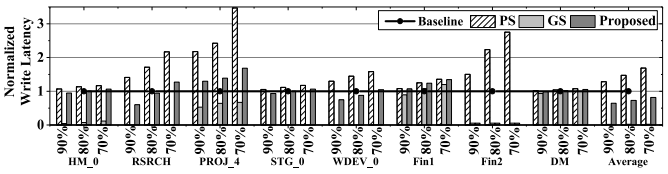


Fig. 4. Normalized write latencies of the four schemes.

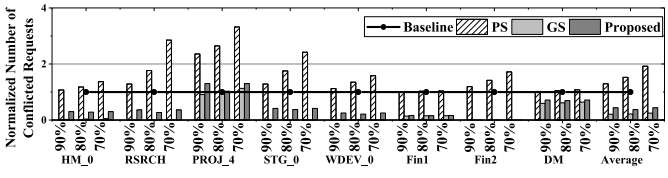


Fig. 5. Normalized number of blocked write requests.

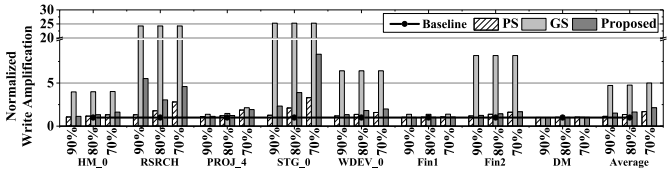


Fig. 6. Normalized write amplifications of the four schemes.

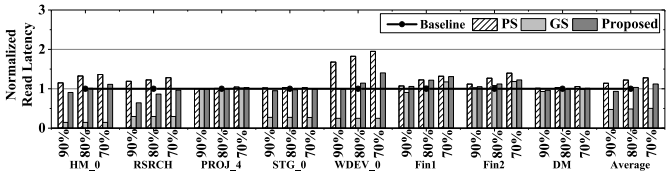


Fig. 7. Normalized read latencies of four schemes.

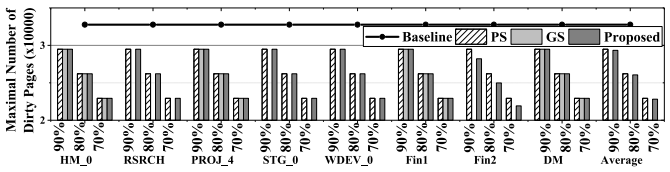


Fig. 8. Maximal number of dirty pages in the cache.

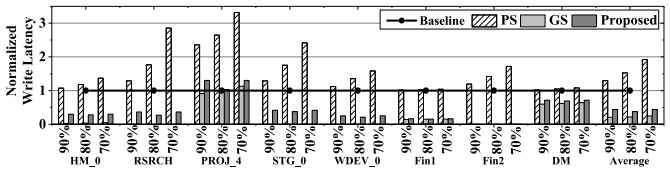


Fig. 9. Normalized write latencies of the four schemes on 3-D SSD.

In this figure, we can find that PS commonly introduces a high write latency, and GS achieves the minimal write latency. Take the scheme with 70% dirty page budget as an example. On average, compared with baseline, PS increases write latency by 70.1%, the write latency of GS is reduced by 97.9% and the write latency of the proposed scheme is reduced by 19.7%.

In Fig. 5, the number of host write requests blocked by synchronization processes of PS is normalized to baseline. We can find that the results in Fig. 5 has a matching pattern with the normalized write latency of PS presented in Fig. 4. That is, the poor write performance of PS mainly comes from the synchronization induced conflict on host requests. In addition, for GS, since dirty pages can be synchronized back to flash memory in idle time, the cache space can be freed to service incoming I/O requests without evicting dirty pages. In Fig. 5, for GS, there are fewer incoming requests being blocked by synchronized dirty pages. For the proposed scheme, there are a few number of write requests being blocked by synchronized dirty pages compared with PS. Therefore, the proposed scheme can achieve better write performance. But the proposed scheme is still far away from the achieved write latency of GS due to its selective synchronization.

Since GS is designed to synchronize dirty pages in idle time, more write operations are generated so that the write amplification will be severe. Fig. 6 presents the normalized write amplification. In this figure, we find that PS achieves similar write amplification to baseline when the budget is set to 90%. But GS has the largest write amplification, of which the worst case is up to five times compared with baseline. Different from these two schemes, the proposed scheme achieves better than GS. The reason is that the number of synchronizing dirty page induced write operations is significantly reduced through smartly activating synchronization. But there also exist some exceptions, such as DM, of which three schemes achieve closely results to the baseline as well. The reason comes from that there exist bursts of accessing requests so that less idle time can be used to synchronize dirty pages.

Fig. 7 presents the normalized read latency. In this figure, several observations can be found from the results: first, in most cases, PS achieves the worst read performance and GS achieves the best read performance among evaluated three schemes. This is because PS may synchronize dirty pages when read requests arrive, causing worse read/write interference [15]. But for GS, it can reduce the impact of read/write interference through synchronizing dirty pages in idle time. For the proposed scheme, since it also synchronizes some dirty pages in idle time, the impact of read/write interference can be reduced as well. Second, when the budget is smaller, the read latency is larger in most cases. This is because more read requests may be blocked by synchronized write operations due to the frequent synchronization operations. Similar results also can be found in Fig. 7, where PS achieves the worst read performance, GS achieves the best read performance and proposed scheme can achieve the tradeoff between PS and GS.

Apart from the above results, the maximal numbers of dirty pages in the cache are collected and presented in Fig. 8 as well. In this figure, PS, GS, and the proposed scheme are confirmed that the number of dirty pages in the cache is smaller than the defined budget.

Latency Analysis for 3-D SSDs: Figs. 9 and 10 show the write and read latencies of evaluated schemes on a 3-D SSD.

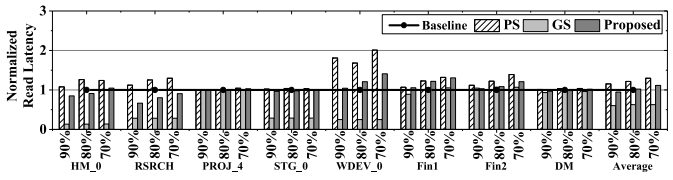


Fig. 10. Normalized read latencies of the four schemes on 3-D SSD.

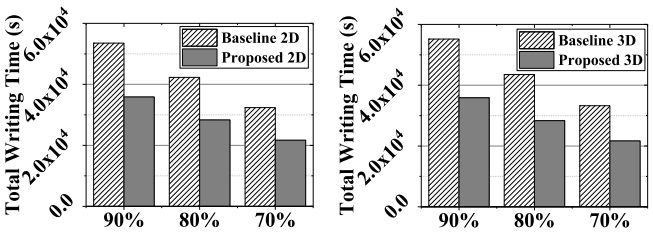


Fig. 11. Total writing time costs of baseline and proposed CSW. Results on (a) 2-D SSDs and (b) 3-D SSDs.

Since the program speeds of flash pages in a block are different for 3-D SSD, the program latency is set to be proportional to page's location [10]. As shown in Figs. 9 and 10, the results have the same trend with the normalized write and read latencies evaluated on 2-D SSD. On the one hand, the achieved performance improvement of the proposed scheme on 3-D SSD is significant compared with PS. On the other hand, compared with GS, although the performance is poorer, the write amplification is highly reduced. The reasons are the same as presented in the latency analysis on 2-D SSD.

2) Results of CSW Method: In this section, we evaluate the total writing time cost to show the efficiency of the proposed CSW on 2-D and 3-D SSDs, respectively. In Fig. 11, baseline and the proposed CSW are evaluated with different dirty page budget on 2-D SSD. The total writing time cost of all schemes is evaluated with 90% dirty page budget. As shown in Fig. 11(a), the total writing time cost is averagely reduced by 67.1% compared with baseline with three different dirty page budgets. This is because that all dirty pages are evenly written to LSB pages so that the time-consuming MSB page write operations are eliminated. In addition, we also find that the smaller the dirty page budget, the less the total writing time cost. This has resulted from that fewer dirty pages are written back to SSDs.

In Fig. 11(b), the total writing time cost is evaluated on 3-D SSDs, which has a similar trend with results presented in Fig. 11(a). First, compared with baseline, the total writing time cost of the proposed CSW can be averagely reduced by 64.9% by reason of writing dirty pages to flash pages with smaller program latency. Second, the total writing time cost is reduced when a dirty page budget decreases. This has resulted from the same reason presented in the above. Based on the evaluated results and (5), the energy costs on writing dirty pages back to flash memory are reduced by 67.1% and 64.9% on 2-D and 3-D SSDs, respectively.

Similarly, the operation time cost of the SSD controller is also evaluated, which indicates the interval of writing

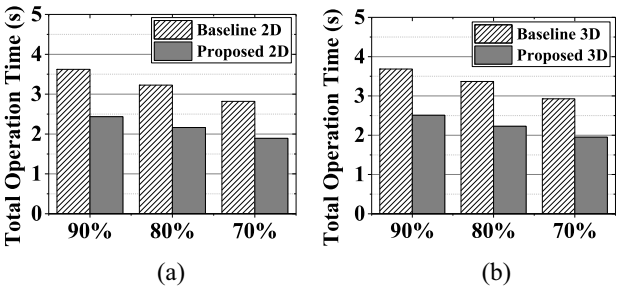


Fig. 12. Operation time costs of baseline and proposed CSW. Results on (a) 2-D SSDs and (b) 3-D SSDs.

all dirty pages back to flash memory. Based on the results presented in Fig. 12(a) and (b), one can see that there are 66.9% and 65.1% time cost reduction on 2-D and 3-D SSDs averagely. Therefore, according to (4), the energy cost, which is used to maintain the normal operations of the SSD controller, can be reduced by 66.9% and 65.1% as well.

Based on the reduction of energy cost, we can see that, if the existing capacitance is further reduced and cannot write all dirty pages from the cache to flash memory in traditional case (without CSW), CSW may still work due to less capacitance requirement. Therefore, CSW is able to reduce the possibility of data loss.

VI. Conclusion

In this article, we take capacitor aging issue into consideration for capacitor equipped SSDs. First, a DPBD scheme is proposed. Then, a smart synchronizing scheme is proposed. Finally, a CSW is proposed to write dirty pages with less capacitance consumption when sudden power supply interrupts. The experiments show that the proposed scheme improves both performance and lifetime while the capacitor aging-induced data loss issue is being avoided.

References

[1] *Tantalum Hybrid Capacitor Life Test*, Evans Capacitor Company, East Providence, RI, USA, 2002.
[2] *Ultracaptm Double Layer Capacitors—A New Energy Storage Device for Peak Power Applications*, EPCOS, Munich, Germany, 2002.
[3] *Power Loss Imminent (PLI) Technology*, Intel Corporat., Santa Clara, CA, USA, 2014.
[4] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for ssd performance," in *Proc. USENIX ATC*, 2008, pp. 57–70.
[5] G. Alcicek, H. Gualous, P. Venet, R. Gallay, and A. Miraoui, "Experimental study of temperature effect on ultracapacitor ageing," in *Proc. IEEE EPE*, 2007, pp. 1–7.
[6] E. H. E. Brouji, O. Briat, J.-M. Vinassa, N. Bertrand, and E. Woirgard, "Impact of calendar life and cycling ageing on supercapacitor performance," *IEEE Trans. Veh. Technol.*, vol. 58, no. 8, pp. 3917–3929, Oct. 2009.
[7] Y.-M. Chang, Y.-H. Chang, T.-W. Kuo, Y.-C. Li, and H.-P. Li, "Achieving SLC performance with MLC flash memory," in *Proc. DAC*, 2015, pp. 1–6.
[8] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo, "Endurance enhancement of flash-memory storage, systems: An efficient static wear leveling design," in *Proc. DAC*, 2007, pp. 212–217.

[9] F. Chen, T. Luo, and X. Zhang, "CAFTL: A content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives," in *Proc. USENIX FAST*, 2011, p. 6.
[10] 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. DAC*, 2017, pp. 1–6.
[11] W. Choi, P. Enjeti, and J. W. Howze, "Fuel cell powered UPS systems: Design considerations," in *Proc. IEEE PESC*, 2003, pp. 385–390.
[12] C. Gao *et al.*, "Exploiting chip idleness for minimizing garbage collection—Induced chip access conflict on SSDs," *Trans. Design Autom. Electron. Syst.*, vol. 23, no. 2, p. 15, 2018.
[13] C. Gao *et al.*, "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.
[14] C. Gao, L. Shi, C. J. Xue, J. Yang, and Y. Zhang, "Parallel all the time: Plane level parallelism exploration for high performance SSDs," in *Proc. IEEE MSST*, 2019, pp. 1–13.
[15] C. Gao, L. Shi, M. Zhao, C. J. Xue, K. Wu, and E. H.-M. Sha, "Exploiting parallelism in I/O scheduling for access conflict minimization in flash-based solid state drives," in *Proc. IEEE MSST*, 2014, pp. 1–11.
[16] J. Guo, J. Yang, Y. Zhang, and Y. Chen, "Low cost power failure protection for MLC NAND flash storage systems with PRAM/DRAM hybrid buffer," in *Proc. DATE*, 2013, pp. 859–864.
[17] J. Hannonen, J. Honkanen, J.-P. Ström, T. Kärkkäinen, S. Räisänen, and P. Silventoinen, "Capacitor aging detection in a DC–DC converter output stage," *IEEE Trans. Ind. Appl.*, vol. 52, no. 4, pp. 3224–3233, Jul./Aug. 2016.
[18] 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. ICS*, 2011, pp. 96–107.
[19] M. Huang, Y. Wang, L. Qiao, D. Liu, and Z. Shao, "SmartBackup: An efficient and reliable backup strategy for solid state drives with backup capacitors," in *Proc. HPCC*, 2015, pp. 746–751.
[20] S. Im and D. Shin, "ComboFTL: Improving performance and lifespan of MLC flash memory using SLC flash buffer," *J. Syst. Archit.*, vol. 56, no. 12, pp. 641–653, 2010.
[21] W.-H. Kang, S.-W. Lee, B. Moon, Y.-S. Kee, and M. Oh, "Durable write cache in flash memory SSD for relational and NoSQL databases," in *Proc. SIGMOD*, 2014, pp. 529–540.
[22] R. Kateja, A. Badam, S. Govindan, B. Sharma, and G. Ganger, "Viyojit: Decoupling battery and DRAM capacities for battery-backed DRAM," in *Proc. ISCA*, 2017, pp. 613–626.
[23] D. Kim and S. Kang, "Dual region write buffering: Making large-scale nonvolatile buffer using small capacitor in SSD," in *Proc. SAC*, 2015, pp. 2039–2046.
[24] H. Kim and S. Ahn, "BPLRU: A buffer management scheme for improving random writes in flash storage," in *Proc. USENIX FAST*, 2008, pp. 239–252.
[25] J. H. Lee, G. S. Lee, S. Cho, J. G. Yun, and B. G. Park, "Investigation of field concentration effects in arch gate silicon–oxide–nitride–oxide–silicon flash memory," *Jpn. J. Appl. Phys.*, vol. 49, no. 11, 2010, Art. no. 114202.
[26] S. Lee, K. Ha, K. Zhang, J. Kim, and J. Kim, "FlexFS: A flexible flash file system for MLC NAND flash memory," in *Proc. USENIX Annu. Tech. Conf.*, 2009, p. 9.
[27] S. Lee and J. Kim, "Improving performance and capacity of flash storage devices by exploiting heterogeneity of MLC flash memory," *IEEE Trans. Comput.*, vol. 63, no. 10, pp. 2445–2458, Oct. 2014.
[28] *NAND Flash Memory MLC Data Sheet, MT29F128G08CFAAB NAND Flash Memory*, Micron Technol. Inc., Boise, ID, USA, 2009.
[29] *DiskMon*, Microsoft, Redmond, WA, USA, 2006.
[30] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, "Migrating server storage to SSDs: Analysis of tradeoffs," in *Proc. EuroSys*, 2009, pp. 145–158.
[31] A. K. Olbrich and D. A. Prins, "Flash memory controller garbage collection operations performed independently in multiple flash memory groups," U.S. Patent 8 533 384, Sep. 10, 2013.
[32] S.-Y. Park, E. Seo, J.-Y. Shin, S. Maeng, and J. Lee, "Exploiting internal parallelism of flash-based SSDs," *IEEE Comput. Archit. Lett.*, vol. 9, no. 1, pp. 9–12, Jan. 2010.
[33] K.-D. Suh *et al.*, "A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, Nov. 1995.
[34] H. Tanaka *et al.*, "Bit cost scalable technology with punch and plug process for ultra high density flash memory," in *Proc. IEEE VLSI*, 2007, pp. 14–15.
[35] *OLTP Application I/O Traces*, UMassTraceRepository, 2009.
[36] *How Does a Supercapacitor Work*, Battery Univ, 2019.
[37] W. Wang and T. Xie, "PCFTL: A plane-centric flash translation layer utilizing copy-back operations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3420–3432, Dec. 2015.
[38] M. Zheng, J. Tucek, F. Qin, and M. Lillibridge, "Understanding the robustness of SSDs under power fault," in *Proc. USENIX FAST*, 2013, pp. 271–284.

Congming Gao received the B.S. degree in computer science and technology from Chongqing University, Chongqing, China, in 2013, where he is currently pursuing the Ph.D. degree with the College of Computer Science.

He is currently a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, USA. His current research interests include embedded and real-time systems, nonvolatile memory, and architecture optimizations.

Liang Shi received the B.S. degree in computer science from the Xi'an University of Post and Telecommunication, Xi'an, China, in July 2008, and the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in June 2013.

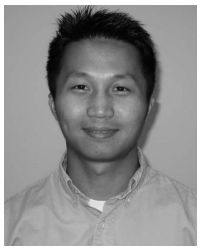
He is currently a Full-Time Teacher with the School of Computer Science and Software Engineering, East China Normal University, Shanghai, China. His current research interests include flash memory, embedded systems, and emerging nonvolatile memory technology.

Qiao Li received the B.S. and M.S. degrees in computer science and technology from Chongqing University, Chongqing, China, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong, Hong Kong.

Her current research interests include NAND flash memory, embedded systems, and computer architecture.

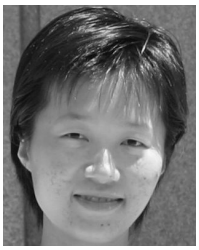
Kai Liu received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2011.

From December 2010 to May 2011, he was a Visiting Scholar with the Department of Computer Science, University of Virginia, Charlottesville, VA, USA. From 2011 to 2014, he was a Post-Doctoral Fellow with Singapore Nanyang Technological University, Singapore; the City University of Hong Kong, Hong Kong; and Hong Kong Baptist University, Hong Kong. He is currently an Assistant Professor with the College of Computer Science, Chongqing University, China. His current research interests include Internet of Vehicles, mobile computing, pervasive computing, and big data.



Chun Jason Xue received the B.S. degree in computer science and engineering from University of Texas at Arlington, Arlington, TX, USA, in May 1997, and the M.S. and Ph.D. degrees in computer Science from the University of Texas at Dallas, Richardson, TX, USA, in December 2002 and May 2007, respectively.

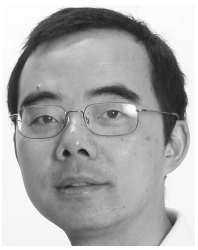
He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include memory and parallelism optimization for embedded systems, software/hardware co-design, real-time systems, and computer security.



Jun Yang received the B.S. degree in computer science from Nanjing University, Nanjing, China, in 1995, and the Ph.D. degree in computer science from the University of Arizona, Tucson, AZ, USA, in 2002.

She is a Professor with the Electrical and Computer Engineering Department, University of Pittsburgh, Pittsburgh, PA, USA. Her current research interests include GPU architecture, secure processor architecture, emerging nonvolatile memory technologies, performance, and reliability of memories.

Prof. Yang was a recipient of U.S. NSF Career Award in 2008 and the Best Paper Awards from ICCD 2007 and ISLPED 2013.



Youtao Zhang (M'17) received the Ph.D. degree in computer science from the University of Arizona, Tucson, AZ, USA, in 2002.

He is currently an Associate Professor of computer science, University of Pittsburgh, Pittsburgh, PA, USA. His current research interests include computer architecture, program analysis, and optimization.

Prof. Zhang was a recipient of the U.S. National Science Foundation Career Award in 2005 and several papers awards for his coauthored papers. He is a member of ACM.