Management of Next-Generation NAND Flash to Achieve Enterprise-Level Endurance and Latency Targets

ROMAN PLETKA, IOANNIS KOLTSIDAS, NIKOLAS IOANNOU, SAŠA TOMIĆ, NIKOLAOS PAPANDREOU, THOMAS PARNELL, and HARALAMPOS POZIDIS, IBM Research – Zurich
AARON FRY and TIM FISHER, IBM Systems

Despite its widespread use in consumer devices and enterprise storage systems, NAND flash faces a growing number of challenges. While technology advances have helped to increase the storage density and reduce costs, they have also led to reduced endurance and larger block variations, which cannot be compensated solely by stronger ECC or read-retry schemes but have to be addressed holistically.

Our goal is to enable low-cost NAND flash in enterprise storage for cost efficiency. We present novel flash-management approaches that reduce write amplification, achieve better wear leveling, and enhance endurance without sacrificing performance. We introduce block calibration, a technique to determine optimal read-threshold voltage levels that minimize error rates, and novel garbage-collection as well as data-placement schemes that alleviate the effects of block health variability and show how these techniques complement one another and thereby achieve enterprise storage requirements.

By combining the proposed schemes, we improve endurance by up to 15× compared to the baseline endurance of NAND flash without using a stronger ECC scheme. The flash-management algorithms presented herein were designed and implemented in simulators, hardware test platforms, and eventually in the flash controllers of production enterprise all-flash arrays. Their effectiveness has been validated across thousands of customer deployments since 2015.

CCS Concepts: • **Information systems → Flash memory**; • **Hardware → Memory and dense storage**;

Additional Key Words and Phrases: NVM controller design, NVM-based storage, flash memory, level shifting, endurance, wear leveling, data placement

ACM Reference format:
Roman Pletka, Ioannis Koltsidas, Nikolas Ioannou, Saša Tomić, Nikolaos Papandreou, Thomas Parnell, Haralampos Pozidis, Aaron Fry, and Tim Fisher. 2018. Management of Next-Generation NAND Flash to Achieve Enterprise-Level Endurance and Latency Targets. *ACM Trans. Storage* 14, 4, Article 33 (December 2018), 25 pages.
https://doi.org/10.1145/3241060

Authors' addresses: R. Pletka, I. Koltsidas, N. Ioannou, S. Tomić, N. Papandreou, T. Parnell, and H. Pozidis, IBM Research—Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland; emails: {rap, iko, nio, sat, npo, tpa, hap}@zurich.ibm.com; A. Fry and T. Fisher, IBM Systems, 10777 Westheimer Rd, Houston, TX 77042; emails: {aaronfry, fisher}@us.ibm.com.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
2018 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 1553-3077/2018/12-ART33 $15.00
https://doi.org/10.1145/3241060

1 INTRODUCTION

NAND flash technology has seen widespread adoption in enterprise storage systems over the past few years. Recent all-flash storage arrays exhibit excellent I/O throughput, latency, storage density, and energy efficiency. However, as the global NAND flash demand is driven primarily by consumer electronics (e.g., smartphones) flash manufacturers focus their efforts mostly on improving what matters for that market, i.e., cost ($/GB) and density. To reduce the cost and increase the storage density of flash memory, they continuously scale the technology node, increase the number of bits stored per memory cell, and stack cells vertically (3D NAND) [32]. Unfortunately, some of these techniques worsen the applicability of flash in enterprise environments, where high endurance and low error rates on the system level are of paramount importance.

The limited endurance of NAND flash results from permanent damage to the insulating oxide due to cycling-induced charge trapping [3, 16, 26]. Further, read disturb and retention effects can cause additional increases in the raw-bit error rate (RBER) because of wider threshold voltage distributions that increase the overlapping of the levels. These effects have been shown to be more pronounced as the program-erase cycles (PEC) of memory increase during its lifetime [26]. More specifically, 1y nm MLC NAND flash[1] has a specified endurance of only about 3000 PECs and requires an error-correction code (ECC) that uses $\sim$ 40bits per 1024B as well as read-retry operations to achieve the specified endurance. Even though higher endurance can be expected from three-dimensional (3D) NAND, flash manufacturers typically recommend using stronger ECCs capable of correcting an RBER on the order of 10^{-2}.

Figure 1 shows the average block endurance we obtained (as detailed in Section 7.3) from a large-scale block characterization. Endurance and standard deviation are normalized to the 1x nm MLC device and assume that all devices use the same ECC scheme as is recommended for 3D NAND. Scaling down the feature size of 2D flash cells reduces their endurance. The recent switch to the third dimension helped to re-gain in cell endurance owing to the larger cell size, but the complex cell architecture and the high aspect ratio edging still exhibit large variations across blocks and weaker data retention and read disturb characteristics. Increasing the endurance is of paramount importance for two reasons. First, it enables the storage system to achieve at least 5–7 years of lifetime, which is the typical enterprise storage refresh cycle, even under demanding write-intensive enterprise workloads. Second, a high NAND flash endurance means that the system can achieve the required lifetime with higher write amplification [17], which in turn allows us to reduce capacity overprovisioning and, thus, cost.

Traditional approaches to address the decreasing endurance and increasing error rates of flash involve stronger ECC schemes [24, 38], PEC-based wear leveling (WL) [8, 9, 14], the use of ECC schemes based on soft-decision information [45], and simple read-retry schemes [6]. Besides the fact that these approaches cannot take advantage of the available endurance of all blocks, they lead to unpredictable performance and latency spikes [34]. Recently proposed RBER-based WL schemes mitigate high error rates but have been shown to increase the overall write amplification, which results in lower endurance [29, 33, 45].

In this article, we rethink the flash management functions in the flash translation layer (FTL) with the ultimate goal of achieving the best possible endurance and at the same time offering *consistently* high performance. To do so, we depart from traditional approaches towards schemes that operate proactively and in a co-ordinated way in the background without impacting active host I/O operations. Our contributions can be summarized as follows:

[1]We use 1x nm as a term for 19–20 nm and 1y nm for 15–16 nm 2D-NAND flash technologies.

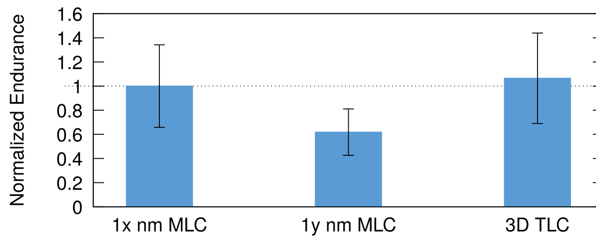


Fig. 1. Average block endurance and standard deviation normalized to 1x nm MLC flash.

- We present the flash block management architecture of an enterprise flash controller and demonstrate how our new flash management functions can be implemented efficiently.
- We introduce novel background algorithms to track optimal read voltage levels.
- We describe a holistic approach to block calibration, data placement, garbage collection (GC), and WL that combines the heat metrics of user accesses to logical data with health information of the physical flash memory to minimize write amplification and maximize system endurance.
- We evaluate the effects of heat segregation, the separation of host and relocation writes, and health binning on the system-level write amplification.
- We experimentally demonstrate that, using our techniques, one can significantly enhance the endurance without sacrificing performance at a system level.
- We acknowledge the importance of skewed workloads as being most representative of real workloads and identify the challenges they introduce with respect to flash management.

The remainder of this article is organized as follows: We first reassess flash management in a holistic way in Section 2 and present the consequences for the block management architecture in the FTL of an enterprise flash controller as well as our evaluation environments in Section 3. Then, we propose a novel calibration mechanism for tracking optimal read voltage (ORV) levels (Section 4), followed by the N-Bin GC policy (Section 5), which, when combined with heat segregation (Section 6) and health binning (Section 7), enables our controller to incur minimal write amplification. Finally, related work is discussed in Section 8, and the conclusions are drawn in Section 9.

2 RETHINKING NAND FLASH MANAGEMENT

We now discuss the limitations of existing flash management functions, which, when applied independently from one another, only achieve limited improvements. We then propose key modifications and enhancements in read voltage shifting, the GC algorithm, and the WL scheme. Figure 2 gives an overview of the effects and dependencies that arise from these changes. As we point out below, our holistic design, based on carefully designed interactions, enables the turning of these effects into tangible synergies, which when applied all together significantly enhance endurance and performance.

2.1 Read Voltage Shifts

With increasing PECs, the programmed threshold voltage distributions widen, therefore increasing the RBER [6], which limits endurance. An effective approach to improve endurance is to adjust the read voltages that represent the values of stored bits. Existing read-retry commands (recommended by manufacturers) that try a sequence of threshold voltages and return the data with the least number of errors are used by many SSD controllers [6, 15]. However, multiple reads result in proportionally higher read latency. Even worse, as block wear increases with increasing PECs, at some point the majority of reads undergo a read-retry, and the observed device latency becomes

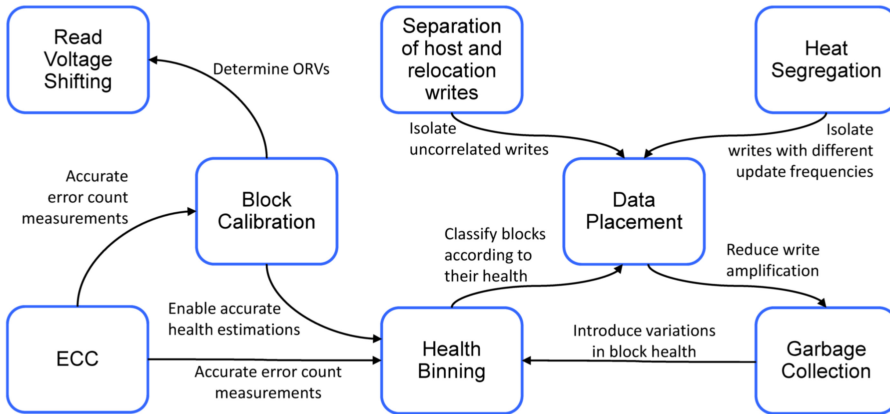


Fig. 2. Synergies and dependencies in a holistic flash management approach.

significantly higher [46]. Unpredictable read latency is extremely undesirable in enterprise storage, as it results in long tails in the application response time distribution [11].

Papandreou et al. [31] have shown that per-flash-block optimal read voltage (ORV) levels significantly reduce the average RBER. However, the ORV levels are not *a priori* known and change over time; thus, they need to be continuously re-adjusted. We refer to the process of determining the ORVs using page-level RBER provided by the ECC as *block calibration*. Because the three main error sources, namely, cycling-induced charge trapping, read disturbs, and retention effects, tend to influence the voltage distributions in different ways, determining ORVs for a page would best be done before the page is read. However, this is not practical: Because the page (or multiple pages in the same block) has to be read several times before the ORVs can be determined, (1) the read overhead may affect host throughput and latency, and (2) any read targeting a block that saw a significant number of PECs or reads, or an extended retention period, since the last calibration (including a combination of all these factors) likely results in a temporarily higher RBER for those pages as will be discussed in Section 4.

We modify the concept of determining ORVs such that the optimal values will *always* be accurate and available *before* a host read operation accesses the data but without the necessity of an immediately preceding calibration. Doing so eliminates the need for read-retry operations and guarantees low-latency reads until the end of device lifetime. The challenges associated with determining ORVs, described in detail in Section 4, are (1) deciding *when* to perform calibration to minimize the impact on latency and (2) managing the calibration metadata.

2.2 Garbage Collection

The FTL maps logical addresses that are being overwritten (i.e., logical pages that are updated with new data) to new different physical locations, invalidating the data at the old locations. The invalidated space cannot be reclaimed immediately as the blocks to which they belong typically hold other, still valid, data. Therefore, GC is required that relocates valid data to new locations so that the block can be reclaimed, erased, and made available for re-use to store new data.

A large body of research has focused on optimizing GC algorithms for flash memory, in most cases evaluating the performance of such algorithms under I/O workloads with uniform access patterns (uniform random, sequential) [4, 5, 27]. Our experience, however, has shown that real-world workloads tend to be skewed in their access-pattern distributions. This was also reported in prior work [13, 44].

Therefore, our design aims to maximize the performance and endurance under these workloads: We make data-placement decisions to enforce multi-dimensional segregation of logical data that separates host and relocation writes and further segregates all writes according to their update frequency. This, combined with our new GC strategy, can significantly reduce write amplification for a certain level of over-provisioning. Or, alternatively, it allows the system to achieve the same write amplification with less over-provisioning, effevctively offering more usable capacity. That said, we still ensure that we can maintain the same performance and endurance levels under workloads with uniform access patterns.

Our design is complementary to other techniques that reduce writes and increase usable storage, such as compression and deduplication. However, the integration of these techniques is beyond the scope of this article.

2.3 Wear Leveling

Each flash block can endure a certain number of PECs before it generates uncorrectable read errors, i.e., too many errors for the ECC to correct. Traditionally, the FTL performs WL in an attempt to equalize the PECs across blocks [9, 14]. Although this works well with uniform random workloads, it is suboptimal with skewed workloads.

Conventional WL comes in two flavors: *Dynamic WL* places data on the block with the lowest PEC count from the pool of erased blocks and hence has low overhead. However, it has limited effectiveness, because blocks with data that are never updated are never moved in this process. *Static WL* occasionally moves data from less-worn blocks (i.e., with a low PEC count) and makes them available for data placement. This process, typically more complex to implement, can result in a significant number of additional writes, because the number of invalidated pages in those blocks may be very low [9, 14, 42]. Nevertheless, it has been shown that static WL on top of dynamic WL can result in enhanced endurance [3, 7].

Recent work has shown that conventional WL can maintain balanced PECs across all blocks, even under highly skewed workloads. However, flash characterization has revealed that, due to process variations, some blocks reach the maximum correctable RBER significantly earlier in their lifetimes than others [29]. This motivated the introduction of a technique called health binning [34].

Blocks exceeding the RBER limit are retired by the FTL, and write amplification increases, because over-provisioning is de facto reduced. Eventually, with conventional WL, the number of retired blocks is too high, causing premature end of life, although a significant number of blocks is still healthy, i.e., have RBERs below the manufacturer-specified threshold. Also, it has been shown that the RBER in early life cannot be used to estimate the block endurance towards the end of life [34]. Therefore, we reconsider WL with the following new objectives in mind:

- We replace dynamic WL, which aims at balancing PECs, with health binning, which periodically evaluates health metrics based on measured RBERs.
- We substitute static WL by timely relocations to address retention and read disturb limitations.
- We use block calibration results to accurately determine the health metrics of blocks independently of their current state.
- We classify blocks in the background by their health and combine the result with multi-dimensional data segregation in the data placement to reduce write amplification and enhance endurance.

The next section introduces the architecture of a typical all-flash array and describes the flash block management performed by the FTL in such an architecture.

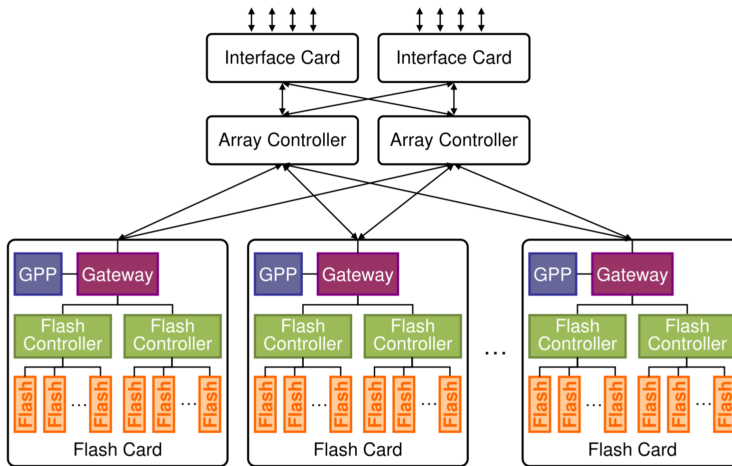


Fig. 3. All-flash array architectural overview.

3 BLOCK MANAGEMENT

A typical all-flash array architecture, illustrated in Figure 3, is motivated by the IBM FlashSystem [20] architecture; however the same design may apply to many other all-flash arrays. It consists of a certain number of flash cards or SSDs storing the actual data, of redundant array controllers that protect against flash-card failures using a log-structured array (LSA) or a RAID-like parity scheme, and of interface cards that handle the I/O requests from hosts using a block storage interface (e.g., Fiber Channel, Infiniband, or iSCSI) or an object interface (e.g., Ceph, Amazon S3).

Flash cards implement the FTL that performs all key flash management functions, which are executed on a general-purpose processor (GPP) and are assisted by several hardware flash controllers, each controlling multiple flash channels. The flash controller's main tasks are the execution of read, write, relocation, and block erase operations, ECC encoding and decoding, as well as encryption and compression. The gateway interfaces to the array controllers.

A prerequisite for high IOPS and low latency is a flash controller design that supports high parallelism on a large number of flash devices. Several flash chips are attached to a flash channel, and chips on different channels can be programmed in parallel. However, that is in fact not sufficient: The risk exists that on a sequence of read operations data will be located in one flash channel or, even worse, in a single flash block. As read commands have to be executed sequentially in this case, the read bandwidth is thus reduced to the maximum bandwidth of that particular channel or chip [19]. We mitigate the exposure to read contention using the following techniques: First, adequate data placement, which entails the use of a log-structured array in the flash card, transforms the write workload into a sequential write stream in which writes are striped over all flash channels and flash chips in small data chunks (typically 4KiB). This requires address translation from host addresses to physical flash pages. We perform this in the flash controller that also maintains the logical-to-physical table (LPT). Therefore, there is no write penalty for small random write updates, and write performance is significantly enhanced. At the same time, rotating parity data can be added in a RAID5-like manner to protect against loss of flash channels or chips and uncorrectable ECC errors. Second, we opt for an in-memory fine-grained LPT at logical page granularity to induce no additional flash reads on address translation operations. Third, the high internal parallelism results in an maximum internal write bandwidth that exceeds the bandwidth

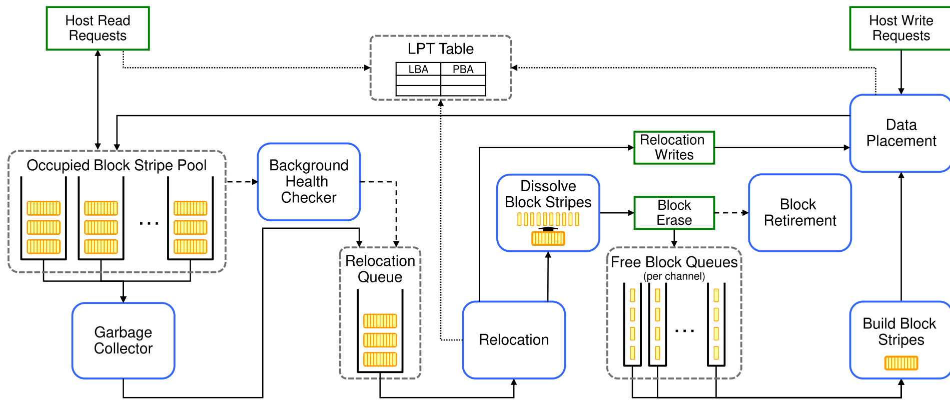


Fig. 4. Block management overview.

characteristics of the flash card interface. Therefore, write latency can be hidden using a reasonably sized destage buffer as stripes are fully destaged in the background before the next stripe is ready. These design choices are a key enabler for outstanding enterprise-level small block size random read and write performance.

The addition of parity data requires the organization of flash blocks into block stripes, in which blocks belonging to one block stripe reside on different physical flash channels. The log-structured data organization ensures that always full RAID stripes are written and no parity updates to flash are needed. To reconstruct data after a page or block failure, all remaining good blocks of the stripe are needed, even if some of them have been fully invalidated. Hence, data can only be garbage-collected in entire block stripes. This is why a block stripe is typically called a logical erase block (LEB).

Next, we outline the typical block-management procedure as shown in Figure 4. Initially, all erased blocks are kept in free block queues (FBQ). FBQs are maintained for each channel in the system. Firmware prepares block stripes from erased blocks in the FBQ in advance and ensures that the flash controller always has a sufficient number of block stripes queued for data placement.

Once all pages in a block stripe have been written, the block stripe is placed in the occupied block stripe pool (OBSP). This pool consists of a set of queues whose sole purpose is to facilitate the GC decision. A detailed description of the organization of our occupied block pool, and the GC process is given in Section 5.

The out-of-place write policy causes logical pages, also known as logical block addresses (LBAs),[2] to be invalidated implicitly when they are overwritten by the host. Hence block stripes in the OBSP will see the number of invalidated LBAs increasing over time. At some point, nearly all blocks are filled with data, and space from invalidated locations needs to be reclaimed to accommodate new writes. GC then chooses the most cost-effective block stripe to be cleaned up. The GC cost typically decreases as the number of invalidated pages increases. In fact, fully invalidated stripes are immediately garbage-collected as they incur no additional relocation cost. The relocation process writes all still valid pages of a block stripe to new locations before the block stripe is dissolved and the blocks are returned to the FBQs.

Besides GC, we use a background health checker (BGHC) that selects block stripes that no longer fulfill the health criteria and queues them for relocation. The BGHC, a critical block management component for sub-20nm and 3D-NAND flash, performs the following tasks:

[2]Note that the logical page size is typically 4KiB and therefore different from the physical flash page size.

- Read scrubbing for early detection of unacceptable increases in the RBER or retention limitations.
- Block calibration to determine the ORVs that minimize the RBER.
- Grading of blocks by gathering block health statistics.

Clearly, these tasks require measuring the RBER and therefore can only be performed while a block is programmed (i.e., holding valid or invalidated data).

3.1 Evaluation Environments

The internals of all-flash arrays and SSDs are typically well-kept secrets by the manufacturers. Because of their large architectural variations, system-level comparisons cannot draw direct conclusions on the performance characteristics of particular flash management functions. Therefore, we use a set of evaluation environments for which we can easily analyze individual performance contributions.

The calibration and characterization results were obtained from a large set of an in-house developed platform [30]. All other results presented below are obtained from a simulator and a real hardware flash card based on a prototype version of the IBM FlashSystem 900 flash module adapted to the needs of those tests [20]. Both environments implement all flash management functions described. Apart from write amplification, evaluating the system-level endurance of all test scenarios would have taken years to complete on the real hardware. Therefore, we exclusively used the simulator for that part, which combined with our flash model and a reduction of the number of channels simulated w.r.t. the real hardware significantly reduced the amount of time needed to run the experiments. Nevertheless, in specific experiments we analyzed the agreement of the two environments and can confirm an extremely high similarity in behavior. Clearly, the simulator also enables us to evaluate scenarios that exceed the capabilities of the hardware environment.

4 BLOCK CALIBRATION

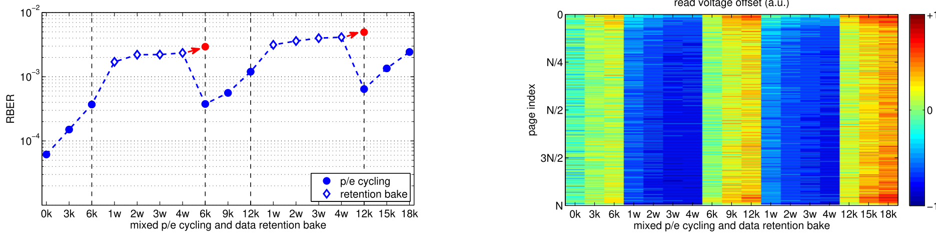
In Section 2.1, we discussed the necessity of block calibration. Here we assess when calibration should take place and how we can achieve minimal impact on throughput and latency of host I/O operations. To do so, we introduce a background calibration process that judiciously pre-computes a set of ORV levels based on a statistical analysis of a certain number of read operations performed with different threshold voltages as well as on current block state information.

4.1 Core Block Calibration Challenges

The following observations outline the key calibration challenges: When calibration is performed immediately after an uncorrectable ECC error on a host read operation, it increases read latency as the calibration process requires multiple consecutive read operations. It therefore entails a similar overhead as read retries. When calibration is executed immediately after writes, it will impact the sustained write performance owing to back-pressure caused by calibration reads. Further, the determined shift values are likely to be outdated at the point in time data are going to be read again.

To better understand the calibration timing tradeoffs, we first discuss the RBER and ORV level changes at specific points in time in a characterization experiment with mixed phases of program-erase (P/E) cycling and retention as illustrated in Figure 5. More specifically, the experiment is based on a 1y nm MLC device, and the block under test is subjected to 18k PECs, where every 6k cycles a retention bake equivalent to one month at 40°C is applied [21].[3]

[3]Retention bake has been defined by JEDEC as a standard test method at elevated temperature to accelerate NAND flash retention evaluations.



(a) Measured RBER of a typical page with ORVs (blue) and when previous ORVs are not adapted upon a transition from a retention to a cycling phase (red).

(b) ORV levels for lower pages in a tested block. Red values correspond to positive (hot), green ones to nominal, and blue ones to negative (cold) offsets.

Fig. 5. RBER and ORV levels with mixed phases of program-erase cycling and retention bake.

In particular, Figure 5(a) shows the evolution of the RBER of a typical page at specific points in time. We observe that the RBER increases with cycling and exhibits a further deterioration during the intermediate data-retention phases. Nevertheless, after each retention phase, the RBER of the page under test returns to the level of the pre-retention cycling state.

Figure 5(b) highlights the variations of the ORV levels for the lower pages in the block under test during the different cycling and retention phases of the experiment.[4] The color code corresponds to positive (red), nominal (green), and negative (blue) offsets, and the offsets are normalized to the maximum range supported by the device. During cycling, the read voltage offsets must be increased progressively to account for the increased charge trap caused by the memory cell degradation and the associated widening of the programmed threshold voltage distributions. In contrast, during retention, a negative offset is required to mitigate the charge loss and the associated shift of the threshold voltage distributions to lower values.

We further observe that the transitions from retention to cycling exhibit an abrupt change in the offset values (e.g., between 4 weeks of retention after 6k cycles and the next P/E cycle, as well as between 4 weeks of retention after 12k cycles and the next P/E cycle). Hence, if the read voltage levels are not immediately adapted on a block erase after an extended retention phase, the RBER continues to increase as shown in Figure 5(a) (red circles).

4.2 The Base-delta Block Calibration Scheme

To handle continuous ORV changes due to cycling, retention, and read disturb as well as abrupt changes caused by the disappearance of temporary effects after a block erase and reprogramming sequence, we propose a remarkably simple scheme that splits the voltage thresholds into two components: (1) A base threshold voltage shift, V_β, which tracks *permanent changes* in the underlying threshold voltage distributions from program-erase cycling, and (2) a delta threshold voltage shift, V_Δ, which adapts to *temporary changes* (w.r.t. to a block's erase count) in the underlying threshold voltage distributions from retention or read disturb errors. This allows a simple resetting of the potentially large V_Δ value after a block has been erased, thereby re-establishing accurate ORVs and effectively avoiding uncorrectable read errors and the use of read-retry commands.

The advantages of separating V_β and V_Δ are illustrated in Figure 6 as a schematic representation for an exemplary page in a given block depicting the calibration effects we observed in NAND flash. The block undergoes three different phases. In the first phase, we simply show the typical

[4]In MLC NAND flash, each cell is programmed to store 2 bits. The LSB/MSB of all the cells in the same word line form the lower/upper page, respectively.

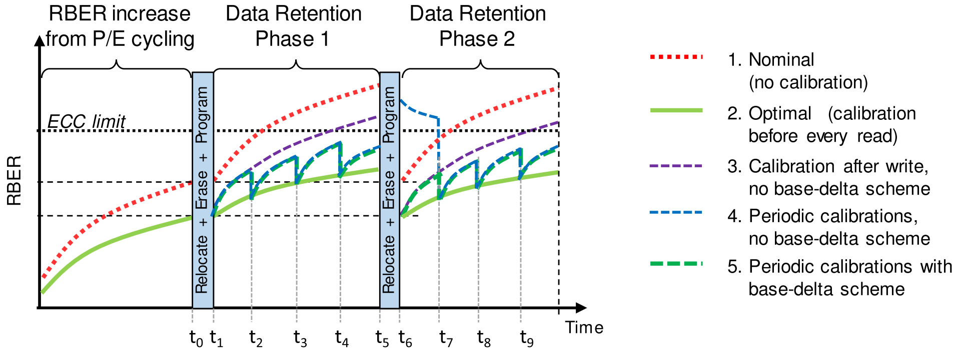


Fig. 6. Illustration of flash calibration effects in 3D-TLC NAND flash.

RBER increase from P/E cycling for nominal read voltages without calibration (Scheme 1) and when ORVs are used (Scheme 2). The cycling phase is followed by two data retention phases. In 3D-TLC NAND flash, we observe that with nominal read voltages (Scheme 1) the RBER keeps increasing during retention.[5] Scheme 2 with ORVs can be implemented by an in-line calibration, which requires three to ten successive reads and therefore causes high latency. Clearly, as long as reads are correctable, no in-line calibrations have to be executed. Here Scheme 2 is used to pinpoint the achievable lower bound RBER using OVRs.

In the two data retention phases, we further distinguish three ORV tracking schemes. Scheme 3 calibrates blocks only after they have been fully written. Even though the number of calibrations performed is reduced compared with Scheme 2, the large number of calibration reads will ultimately throttle the write throughput. In addition, as read voltages are not updated in retention phases, the RBER steadily increases.

In contrast, Scheme 4 performs periodic background calibrations, but does not adapt the read voltages immediately after the block has been re-programmed. Here, the periodic calibrations are done at time instances $t_1, \ldots, t_4$ and $t_7, \ldots, t_9$, but no calibrations are done at the end of the first and the beginning of the second data retention phase (e.g., at t_5 and t_6). As a consequence, a read shortly after time t_6 using Scheme 4 will apply suboptimal read voltages, resulting in an abrupt increase in the RBER. Only when the block gets calibrated at t_7, are ORVs re-established again. Further, with periodic calibrations, the read voltages are only optimal for a limited amount of time after the execution of the calibration. Retention effects gradually increase the RBER, resulting in the saw-tooth behavior depicted. The same applies for read disturbs effects. Except for Scheme 2, all schemes introduced so far exhibit points in time at which the error correction-capability of the ECC is exceeded and therefore in-line re-calibration or other read retry techniques must be applied.

Finally, Scheme 5 performs periodic calibrations and separates V_β and V_Δ values by applying the base-delta scheme. At time t_0 and t_5, the scheme resets the V_Δ values and preserves only the V_β component. Therefore, at t_6, the RBER immediately returns to the low level seen at time t_1. As a result, the undesirable effect of Scheme 4 is eliminated, while keeping the benefits of Scheme 3.

[5]Even though retention and read disturb effects shift the read voltages into the negative direction and may therefore compensate for the positive shifts from P/E cycling, the widening of the read voltage distribution negatively impacts the RBER. The latter is typically more pronounced with increasing number of levels per cell and dominates the RBER in 3D-TLC devices. In contrast, 2D-MLC devices typically have sufficient margin between the levels and therefore may see the RBER decrease as is reported in Reference [31].

Table 1. Calibration Based on Block State Information

Sweep Count	Reads Low	Reads High	
		E_β Low	E_β High
= 0	Update (V_β) and Reset (V_Δ)	Update (V_Δ)	Update (V_β) and Update (V_Δ)
> 0	Update (V_Δ)	Update (V_Δ)	Update (V_Δ)

Legend:
V_β Base threshold voltage shift tracks permanent changes in thresholds voltage distributions.
V_Δ Delta threshold voltage shift tracks temporary changes.
E_β P/E cycle count since last base calibration.

Knowing the importance of the separate tracking of V_β and V_Δ, the actual contributions to each voltage threshold component on calibration are not straightforward to determine. In Table 1, we describe a heuristic method that can be used to decide whether to update V_β or V_Δ on calibration. The method is based on block state information, which includes (1) the retention time, which represents the time since the block was last written; (2) the read count, which corresponds to the number of times pages in a block have been read and hence tracks the contributions to the RBER from read disturbs; and (3) the PEC count since the last base calibration E_β. For wear-leveling purposes, the BGHC traverses the OBSP within a given time interval and increments a sweep counter for each stripe. We leverage the WL sweep count to approximate the retention time. When the sweep count is zero and the read count is low, the block has been written recently, and temporary changes to the RBER from retention or read disturb effects can be neglected. Thus, on calibration, only V_β needs to be adjusted, whereas V_Δ is reset. Note that at this point, V_Δ will already be zero as it was reset after the last block-erase operation and has not been adjusted since then. When the sweep count is larger than zero, the block has not been programmed for at least one BGHC iteration, and all changes in the RBER, since the last calibration are temporary so that only V_Δ has to be updated. When the sweep count is zero, but a high read count is observed, there are permanent and temporary contributions to the changes in the RBER, and, E_β, the PEC count since the last base calibration, is used to better distinguish them: A low E_β indicates negligible permanent RBER changes so that updating V_Δ is sufficient. A high E_β suggests that both V_β and V_Δ need updating. In this case, one can either use models based on statistics obtained from extensive characterizations to determine each contribution (see Section 7.3), or assume that the RBER changes are temporary, and thus update V_Δ only, but mark the block for another calibration in the near future.

4.3 Other Block Calibration Aspects

Having outlined the base-delta calibration scheme, we now address more specific implementation aspects that have to be taken into account in a enterprise-level all-flash array.

Calibration frequency: During calibration, every page in a block must be read with a few candidate threshold voltages to obtain each candidate's RBER, which is then used to determine the ORVs. This generates a non-negligible number of read operations, which might impact the overall performance negatively. The frequency at which the BGHC performs calibrations depends on the sustained maximum number of write operations the system is able to deliver and the frequency at which blocks need to be re-calibrated in the data-retention phase. First, regarding the frequency of V_β updates, the simplest way of updating V_β is when the block only has short retention time and a low number of reads, a situation that is typically encountered directly after writing the page. To avoid throttling the write bandwidth because of calibration reads as it occurs in Scheme 3 discussed above, we propose to incorporate the observation found in Reference [31], where it has been shown that changes of the ORVs due to cycling only can be accurately tracked by calibrating

blocks every few thousand PECs. Hence, V_β updates do not have to be done after every PEC. Second, retention-related periodic re-calibrations result in V_Δ updates. Their frequency is based on the results shown in Reference [31], which suggest that traversing the entire flash in 1–2 weeks is reasonable. With these frequencies, calibration read operations have a very low probability of affecting the latency of user reads even at queue depth (QD) 1. A user read is stalled only if it hits the same flash chip on which a calibration read is outstanding; in fact, for our system, that probability is less than 3×10^{-5}. The effect on user I/O bandwidth is also negligible: the internal I/O bandwidth between the flash controller and the flash chips is higher than the I/O bandwidth exposed to the host; the difference between the two allows enough leeway to perform background operations, such as calibration reads.

Synergies from traditional read scrubbing: Calibration can be performed block by block or stripe by stripe, thereby handling all blocks in the stripe. Data scrubbing as is typically performed in controllers using RAID protection schemes requires the traversal of stored data in a stripe-by-stripe fashion to check data integrity. When a flash controller uses internal RAID, calibration can leverage these scrub reads to reduce background activity by (1) traversing data in the same fashion and (2) using the measured RBER of the scrub reads for the calibration. However, ongoing writes cause new stripes being created (with a new set of blocks), and at the same time, stripes being garbage-collected (resulting in the blocks being returned to the free-block queues). This continuous process of assembling and dissolving stripes may result in a situation where the same block is handled twice in different block stripes or, worse, not calibrated at all during a full iteration through the OBSP. Fortunately, this only affects non-retention-related calibrations (i.e., calibration of V_β) as stripes in retention mode do not change the blocks associated with them. The effect can therefore be neglected.

Reductions in the metadata overhead: Significant reductions in metadata can be obtained from page grouping by reducing the number of read voltage levels maintained per block and the aggregation of read counters. Ideally, each page in a block should have its own set of read voltage levels. However, storing that many read voltage levels becomes prohibitive for large storage systems with hundreds of thousands or millions of blocks. For example, additional DRAM on the order of 100MiB per TiB of storage capacity would be required to access the read voltage levels fast enough on reads for an MLC device.[6] Characterization data showed that pages with similar properties can be combined into static groups of voltage threshold levels that do not vary among blocks of the same flash chip generation of a manufacturer. We group pages based on characterization data. This significantly reduces the amount of metadata to less than 64B per block. However, pages in the same group may not be adjacent. Hence, a small static mapping table (i.e., only a few kilobytes in size) that maps each page in a block to one group must be maintained. On a read operation, the flash controller first determines to which page group the physical page belongs, and then applies the voltage threshold levels for this group found in the block metadata. Regarding the read count, it is too costly to maintain per-page counters. We have found per-block read counters to be an adequate approximation of the real read count. In fact, when reading a page from a selected word line, a bias voltage is applied to all unselected word lines in the block, thereby affecting all pages in the block.

From large-scale characterizations, we found that up to 3× endurance gains can be achieved with ORVs as compared to nominal read voltages for some MLC flash devices. Moreover, from our experiments on real hardware flash cards using 3D TLC flash, we were able to reduce the average read latency by 1.2% and the 99th percentile latencies by up to 13% already at the beginning of the device lifetime because of the prevention of read retry operations. These values were measured

[6]We assume that three ORV values from a small set of discrete levels are maintained per physical 16 KiB page here.

on a properly preconditioned device (sequential full device write followed by 6× random 4KiB full device writes) during a measurement interval of 24 hours using a 4KiB random read workload at queue depth one generated by FIO [1]. All random read and write workloads are uniformly distributed over the entire logical address space.

Our block calibration scheme leverages several synergies from other flash management functions, which operate independently from each others in traditional controllers: First, we use feedback from the ECC decoder, namely, the number of corrected errors, to determine ORVs. Second, the stripe count information maintained by the BGHC for wear-leveling purposes is leveraged to decide whether the base and/or delta components of the ORVs are updated. And third, traditional read scrubbing for data-consistency checks is combined with calibration reads to reduce the overall background read activity.

5 THE N-BIN GARBAGE COLLECTION POLICY

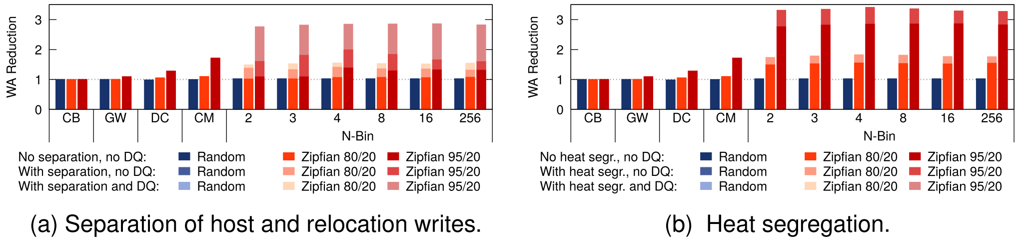
For uniform random write workloads, the greedy GC policy [10] has been shown to be optimal in terms of write amplification [5]. The full greedy algorithm can be trivially implemented with P bins (i.e., sets), where P is equal to the number of pages in the LEB. Each of the P bins holds all LEBs that have the same number of invalid pages. The highest bins hold LEBs with the most and the lowest bins those with the fewest invalid pages. On a page invalidation, the LEB invalid count is incremented, and the LEB is moved into a higher bin. When the GC process wants to reclaim an LEB, it chooses the highest (i.e., the one having the most invalidated pages per block) non-empty bin.

The circular buffer (CB) GC algorithm [35], in contrast, maintains a single FIFO queue with all LEBs in the system, and always chooses the oldest LEB for reclamation.

We generalize the greedy and the CB algorithms into the N-Bin GC algorithm, which augments a greedy policy with an aging factor of the CB and where $N \leq P$. We group LEBs with similar numbers of invalid pages into FIFO queues, speeding up the search for the highest non-empty bin.[7] For example, in a system with 256 pages per LEB and 4 equally distributed bins, bins (0–3) contain LEBs that have (0–63, 64–127, 128–195, 196–256) invalid pages, respectively. We call the fences between the bins thresholds. In this case, the thresholds are $T_1 = 64$, $T_2 = 128$, and $T_3 = 196$. The ranges they define may not overlap, i.e., each LEB belongs to only one bin at any given time. On a page invalidation, the LEB invalid count is incremented, compared with the threshold of the higher bin, and, if found to be larger, the LEB is removed from its current bin and enqueued to the new bin. The GC process will re-claim the oldest LEB in the first non-empty bin with the highest invalidity count, which may counteract WL objectives and cause a wider block health distribution. As shown in Figure 2, these variations can be leveraged by health binning, which will be discussed in more details in Section 7.

The log-structured controller logic transforms writes at the logical space into sequential writes (consisting only of full-block writes) at the physical space. While a non log-structured controller only preserves spacial locality, a log-structured controller additionally has a positive effect on temporal locality: for a workload that exhibits temporal locality in its writes, an LSA controller would effectively transform the temporal locality of writes at the logical space into spatial locality at the physical space. For example, in a OLTP workload, there might be co-related updates to a number of records that repeat over time (update to record A, followed by an update to record C, then record E, etc.). In this example, an LSA controller would transform this temporal locality of repeated updates to spatial locality at the physical space. Consequently, a recurring pattern

[7] Note that at the boundaries of one bin and of the number of bins equal to the pages of the LEB, the N-Bin would degenerate to the CB and the full greedy algorithm, respectively.



(a) Separation of host and relocation writes. (b) Heat segregation.

Fig. 7. Write amplification reduction w.r.t. cyclic buffer (CB) for the Greedy Window (GW), d-Choices (DC), Container Marking (CM), and the N-Bin GC strategy, each of them with and without using the DQ. Numbers are normalized to CB using no separation and no segregation with a uniform random workload.

in a workload's sequence of writes would be demonstrated as consecutive overwrites in a block of pages in the physical space. We exploit this property of LSA controllers to effectively reduce write amplification for workloads that exhibit correlated writes. To this end, we introduce a fixed size FIFO queue that guarantees that each LEB will only be considered for GC (i) after its first logical page invalidation and (ii) after the system has seen a certain number of writes, equal to the estimated maximum length of periodicity. This delay is achieved by the queue having a fixed size: the number of page writes before an LEB is considered for GC is the size of the delay queue times the number of pages in an LEB. We call this delay FIFO queue the delay queue (DQ), and its target fixed size the delay queue length (DQL). The DQ augments the N-Bin algorithm as follows: On the first page invalidation of an LEB, the LEB is placed at the head of the DQ and the DQ size is incremented; if the size of the DQ is greater than DQL, then we dequeue the LEB from the tail of the DQ, decrement its size, and place it in the appropriate bin based on its invalidity. Note that the DQL is always lower than the chosen over-provisioning so that GC does not starve. As we will show below, the DQ has a positive effect on skewed workloads: It will avoid prematurely garbage collecting a "hot" LEB that has seen a large number of invalidates but is potentially expecting many more. Last, the algorithm is further optimized to immediately reclaim any LEB that becomes fully invalid on an overwrite.

6 DATA PLACEMENT

Before re-claiming and erasing an LEB, GC must relocate all valid data from the LEB. Under non-uniform (i.e., skewed) and multi-user workloads, the update frequency of data varies across the logical capacity of the device. Data segregation into LEBs based on their update frequency minimizes the amount of relocated data in skewed workloads [18]. We call each segregation container a data stream. Ideally, the number of data streams should be equal to the update frequencies that the workload exhibits. In practice, however, the supported number of data streams is limited by hardware resources, as each data stream is written to a different LEB, typically maintained in DRAM in the data-placement unit.

In this article, we apply a two-level data segregation scheme: First, we segregate data based on their origin: host writes or GC relocations. Second, we segregate data further based on their *heat* (i.e., update frequency). The results are shown in Figure 7 and compare the N-Bin against existing GC and data-placement algorithms: CB, greedy window (GW) [25], d-Choices (DC) GC algorithm [39] (with d = 10), and Container Marking (CM) [18] with four heat levels. A detailed discussion is given below.

Knowing that real-world workloads are typically skewed in their access pattern distribution, we approximate them with synthetic workloads that generate access patterns following Zipfian distributions of varying skewness [44]. In addition, we utilize traces collected from real-world

enterprise storage arrays in the field that help to better understand the impact on those types of workloads. A description of the traces is given in Section 6.2.

6.1 Separating Host and Relocation Writes

Under our scheme, data placement of host writes is separated from that of relocation writes. We perform this type of segregation to avoid inhibiting correlated (e.g., sequential) host writes from fully invalidating LEBs owing to having mixed host and relocated data. Moreover, there is no correlation between host and relocation writes; thus we expect the update frequency of LBAs that are written by the host to be different from that of LBAs that are being relocated. By separating host and relocation writes, we achieve a significant reduction in write amplification in the common case when the host workload is skewed and differs from the relocation-induced workload, which is usually random.

On Figure 7(a), we can see the benefits from host and relocation write separation under skewed workloads for the N-Bin GC. Separating host and relocation writes is critical in enabling WA reductions. With the N-Bin GC, we measured an average reduction in write amplification of 23% and 28% due to this form of data segregation compared to the N-Bin without separation when running the Zipfian 80/20 and Zipfian 95/20 workloads, respectively. Also, separating host writes from relocates enables the DQ optimization introduced in Section 5 to provide an additional improvement for the skewed workloads: 10% and 38% for Zipfian 80/20 and Zipfian 95/20 , respectively. Without this form of separation, the DQ brings very little benefit. This is expected, since the DQ targets correlated user overwrites: if the user and relocation stream is mixed then recurring patterns at the logical level are not reflected at the physical level. From Figure 7(a) we further see that the N-Bin with the DQ and segregation of host and relocation writes outperforms the best performing algorithm at each synthetic workload: 2% for the uniform random (GW), 29% for the Zipfian 80/20 (CM), and 39% for the Zipfian 95/20 (CM).

6.2 Heat Segregation

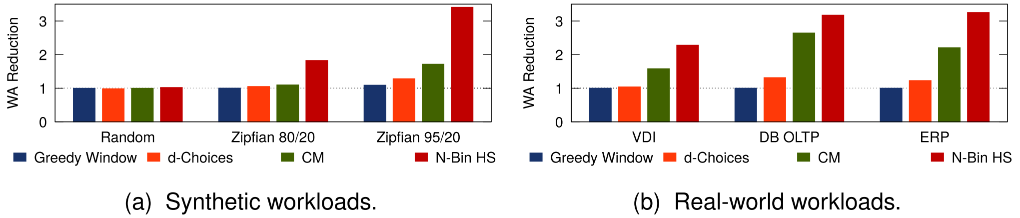
After separating data based on their origin, we perform segregation based on *heat*. Heat here refers to the rate (frequency) at which the data are updated (i.e., overwritten). Hot data tend to get overwritten frequently over time, whereas cold data are rarely overwritten. The heat of an LBA in the context of heat segregation can be determined based on the LBA's update frequency. The granularity at which heat is tracked, the resolution of the heat, the procedure for updating the heat, the resolution at which heat segregation is performed (i.e., the number of streams per heat level), and the mapping of the heat value to the data stream are the design choices in a heat-segregating scheme.

We augment the LPT entry of each LBA with an n-bit saturating counter to track heat. The insight behind our design choice is that the optimal way of measuring heat is using bins with exponentially decreasing update frequency ranges and increasing the number of heat levels quickly leads to diminishing returns [37]. At the same time, maintaining a large number of heat levels is impractical. A 3-bit counter, for example, would result in a resolution of eight different heat values. This is ideal when the LPT entries already have unused bits, for example because of byte-alignment requirements for memory accesses. If tracking heat at an LBA page granularity is prohibitive from an implementation point of view, then it can be done at the flash-block or LEB level: The heat of an LBA can be deduced from the data stream that contains the physical block [18].

Figure 7(b) shows the benefits of heat segregation (with four heat levels) for the N-Bin algorithm on top of host and relocate separation and the DQ: It further reduces write amplification by 14% and 15% for Zipfian 80/20 and Zipfian 95/20 , respectively.

Table 2. Summary of Used I/O Traces

I/O Trace	Duration	# Write IOs	LBA Range
VDI	24 hours	79455871	8.4TiB
DB OLTP	33 hours	155547595	44.1TiB
ERP	24 hours	96682912	21.1TiB



(a) Synthetic workloads. (b) Real-world workloads.

Fig. 8. WA reduction w.r.t. cyclic buffer (CB) from GC and data-placement strategies. N-Bin with heat segregation (HS) uses four heat levels.

We have also evaluated our GC algorithm and data-placement policies under real-world traces. The traces were collected at the block interface from enterprise storage arrays in the field across multiple client environments. Each trace was collected at the host interface layer of the storage array, i.e., it captures all the read and write requests to the storage system across all client hosts before they get processed by the system. Thus, the traces capture the application workload as it is seen by the storage system (before any portion of it is served by caches, etc.). Each trace entry tracks the host where the request originated, the time it reached the storage system, the target volume, offset within the volume and number of sectors that are accessed (either for a read or a write). We replay the traces in a way that reflects the same ordering of requests. Our primary interest at this point is in measuring the effective total write amplification in the storage system under the I/O patterns that the various enterprise applications generate; we are not so much interested in the time it takes the storage system to serve the requests. Therefore, the timing of the requests is largely irrelevant, as is distinguishing between requests from different client hosts: the resulting write amplification is determined solely by the order and the parameters (target volume, offset, size) of the requests. A summary of the characteristics of the traces that include Virtual Desktop Infrastructure (VDI), Online Transaction Processing (OLTP), and Enterprise Resource Planning (ERP) workloads is given in Table 2.

For the synthetic workloads we have verified the write amplification results on the real card with those from our simulator. For the real-world traces, we analyzed the CDFs of the write accesses and confirmed the presence of skew in these workloads. The traces could be simulated using a combination of Zipfian workloads; however, we believe this is out of the scope of the article. Nevertheless, our trace-replay results give good indications of the expected WA and endurance of these type of applications and hence allow for putting them into a better perspective with respect to synthetic workloads.

The bottom line results for synthetic workloads and real-world trace are depicted Figure 8, comparing our combined N-Bin scheme (four bins, DQ, host relocation separation, heat segregation with four levels), against existing GC and data-placement algorithms. The combined N-Bin GC exhibits 39% and 49% less write amplification than the best existing scheme (CM) on the synthetic Zipfian 80/20 and Zipfian 95/20 workloads (Figure 8(a)). Under real workloads (Figure 8(b)), the

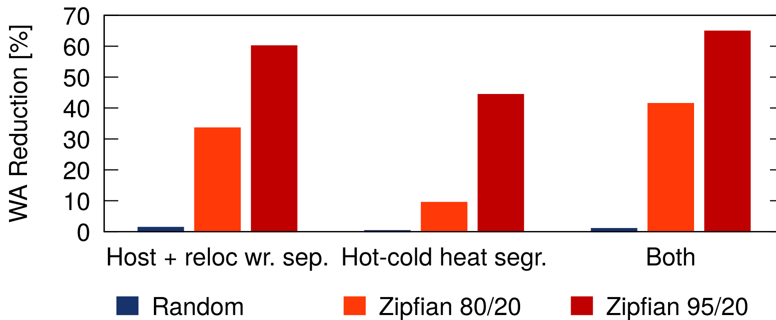


Fig. 9. WA reduction from separation of host and relocation writes, hot-cold heat segregation, and both schemes using the N-Bin GC.

reduction in write amplification is still impressive: 30%, 17%, and 32% reduction for the combined N-Bin scheme compared to CM for the VDI, the OLTP, and the ERP workloads, respectively.

Finally, Figure 9 compares WA reductions for the N-Bin GC w.r.t. a single stream (with neither segregation and separation of host and relocation writes) from separation of host and relocation writes, hot-cold heat segregation [41], and the combination of both schemes. We use four GC bins and limit heat segregation to two streams, emulating the hot-cold frontiers presented in Reference [41]. Hence the combination of both schemes has four independent write streams. Host and relocation write separation is more important than simple hot-cold heat segregation, but both schemes together still further improve WA reduction resulting in 65% for a Zipfian 95/20 workload. We made similar observations for real-world workloads.

7 HEALTH BINNING

In this section, we evaluate and compare endurance gains from WL seen above and below the FTL. To do so we use simulations in conjunction with flash wear models obtained from large-scale characterization data. Therefore, we first introduce our notion of block wear and the flash wear models before presenting our results. We then complete the discussion by addressing the impact of WL on read-retry and uncorrectable read errors.

7.1 Block Wear Definition

Owing to the reduced retention characteristics of newer NAND flash generations, we suggest to use static WL to address only data retention and read disturb limitations. When doing so, most WL work is handled dynamically, thus additional writes from static WL, which may move large amounts of still valid data, becomes negligible. We expect it to be less than 0.07% of the available endurance in the worst case.[8]

Recent work suggests to use the RBER instead of the PEC to estimate block wear [26, 29, 33]. This is reasonable, because the acceptable unrecoverable bit-error rate of an ECC algorithm, typically in the range of 10^{-13} to 10^{-16}, is determined by the RBER and not by the PEC. Therefore, we use health binning [34], which departs from PEC balancing and suggests to use the number of errors corrected by the ECC decoder as the unit of block wear. In addition, we apply the base-delta block calibration scheme in the background to get accurate error counts in our flash cards. As a block must be retired when any codeword c reaches the correction capability of the ECC irrespective in which physical page p the codeword c is located within the block, the wear of a block W_b is hence

[8]This value is obtained from the maximum block retention time, the expected device lifetime, and the average block endurance.

defined as the number of errors R_b found in the worst codeword of the worst page in the block:

$$W_b = \arg\max_{p,c} R_b(p,c). \quad (1)$$

W_b has to be maintained for each block b in the device. Note that W_b can be significantly larger than the average number of errors of all codewords in the block. Further, depending on the ECC scheme and the decoder used, it may not be possible to extract accurate error count information, and more sophisticated ways to measure wear are required—this, however, exceeds the scope of this article.

7.2 Health Binning and Block Grading

Health binning leverages information and properties provided by other flash management functions: Accurate block health is determined from block calibration information by using error count measurements obtained with ORVs. The workload dependent widening of the block health distribution caused by GC allows for distinguishing blocks. In particular, health binning uses block grading to provide the possibility to classify flash blocks in the background into a range of health grades according to their current wear. The health grade of each block is then used in combination with heat segregation to place write-hot data into healthier blocks and write-cold data into less-healthy blocks. Hence, health binning can be combined in a very natural way with data placement introduced in Section 6: Using healthier blocks for write-hot data, increases the likelihood that these blocks are GC-ed earlier than blocks storing write-cold data, because data get invalidated sooner. Hence, healthier blocks will be used for data placement sooner and therefore will experience a faster PEC increase relative to less-healthy blocks.

The current wear of a block is incorporated into the BGHC and determined when error count information is collected during block calibration and read scrubbing. At the end of the BGHC iteration, all blocks are graded according to their current wear: Using the cumulative density function of all block wear values, each block is assigned a particular health grade. The number of health grades corresponds to the number of heat levels being tracked. We have found that having more health grades will only marginally improve performance. The number of blocks associated to a health grade may be approximately equally distributed in a simplified approach or derived from large-scale characterization data. Simulations showed that both approaches will operate well with various skewed workloads.

With health binning, separate sets of FBQs are maintained for each health grade. Once a block stripe has been garbage-collected, each block is inserted into the FBQ corresponding to its health grade and channel.

7.3 Modeling Flash Block Wear

Mohan et al. [28] have modeled the RBER of individual physical flash pages as a function of the PEC. We used their model as a starting point and found from characterization that the wear of a 2D flash block of advanced age can indeed be accurately modeled using the following log-log model

$$\log_{10}(W_b) = x_b + y_b \log_{10}(E(b)), \quad (2)$$

where $E(b)$ denotes the PEC of block b, and x_b and y_b are parameters obtained from large-scale characterization and are distinct for every block. However, a log-lin model is more accurate for 3D flash:

$$\log_{10}(W_b) = x_b + y_b E(b). \quad (3)$$

Table 3 shows the coefficient of determination R^2 that denotes the accuracy of the models as a value in the range $[0, 1]$ to justify our selection. A value closer to 1 denotes a better fit. With a

Table 3. Coefficient of Determination R^2 for Each Model

Model	1x nm MLC	1y nm MLC	3D TLC
log-log	0.975	0.963	0.893
log-lin	0.910	0.942	0.960

large enough characterization dataset, the expectation maximization algorithm can be applied to create a 2D Gaussian mixture model to generate the block wear parameters. At the beginning of the block lifetime, the model can be inaccurate owing to the low error rates. This can be easily mitigated by using $E(b)$ instead of the estimated wear.

In our simulations, we used characterization data of different 1x nm and 1y nm MLC and 3D-TLC flash memories, from which we generated wear parameters for a large number of blocks. Different configurations with distinct wear parameters generated for several 10 to 100 thousands of blocks yielded only marginal variations in our results. We assume that the ECC is designed to tolerate an RBER of 1×10^{-2} and that blocks get retired when the worst codeword reaches this limit.

In the remainder of this section, we use our model created from characterization data using nominal shift values and deliberately neglect the effects from retention and read disturbs. This is reasonable, because the RBER contributions from non-optimal threshold voltages are orthogonal to those from the type of wear leveling we use. We refer to the results presented in Section 4, which show how ORVs combined with the base and delta shift separations efficiently address the effects neglected. In other words, the endurance improvements obtained from ORVs with base and delta separation and those from health binning are independent of each other and add up to the overall endurance.

7.4 Endurance Analysis

The endurance gain from health binning on the physical space only has been studied in Reference [34]. Here we take a more integral approach and extend the analysis to endurance gains on top of the FTL, hence to the logical address space where the effects of GC and over-provisioning are included. The amount of over-provisioning denotes the percentage of physical capacity not available to the host, but used by the FTL to reduce write amplification.

Figure 10(a) illustrates the endurance gains on the physical space using the same Zipfian workloads as done in the heat segregation analysis above. The physical space does not distinguish GC relocations from host writes; hence effects from reduced write amplification are not visible. Therefore, the endurance gains on the physical space are only observable with heat segregation *and* health binning, because for this combination the system endurance is dictated by the average endurance of all blocks, compared with the endurance dictated by a small set of worst blocks otherwise, thereby resulting in a gain of up to 58.8%. Also, the endurance gain is not sensitive to over-provisioning changes when the skew of the workload is high (i.e., Zipfian 95/20). With less skew (i.e., Zipfian 80/20), the hot dataset is larger, resulting in more writes from GC relocations and hence less endurance gain when over-provisioning decreases. Overall, 1y nm flash exhibits higher gains while 3D TLC almost reaches the level of 1x nm flash.

As evident in Figure 10(b), the relative endurance gains on the logical space are generally substantially higher for more skewed workloads due to the smaller relative write amplification reduction of the N-Bin GC. In particular, the endurance gains are increasingly dominated by health binning with higher over-provisioning as the relative write amplification reduction decreases at the same time. Nevertheless, with Zipfian 80/20, heat segregation alone enhances endurance already notably by up to 79.1%.

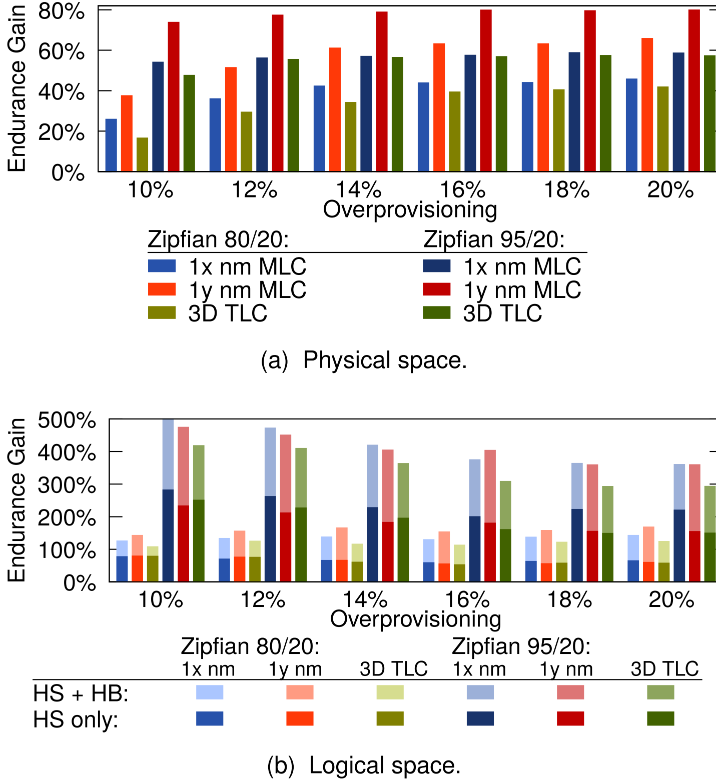


Fig. 10. Endurance gain for varying over-provisioning using heat segregation (HS) only and in combination with health binning (HB).

The Zipfian 95/20 workload, however, exhibits strong endurance gains from both heat segregation and health binning, but with higher over-provisioning the relative gain is again reduced owing to smaller relative write amplification reduction. With moderate skew (Zipfian 80/20), less over-provisioning also reduces endurance, because the segregation efficiency decreases. Further, we observe that endurance gains from health binning alone are higher for devices with smaller feature sizes that points out the increased block variability.

7.5 Performance Analysis

To better understand the impact of RBER variations on the read performance and uncorrectable errors in general as reported in Reference [36], we analyze the behavior of the worst code word in a block during the device lifetime in simulations running a Zipfian 95/20 workload.

Figure 11 compares the block RBER as a function of each block's PEC using PEC-based WL and health binning at different points in the lifetime of the device. Despite the high skew of the workload, the variations of the PEC count among different blocks remain remarkably low with PEC-based WL (Figure 11(a)). However, their RBER is widely spread throughout the device lifetime, and blocks reaching the error-correction capability of the ECC start accumulating shortly after the device has seen a number of writes corresponding to the nominal endurance (i.e., normalized to the manufacturer-specified endurance).

In contrast, with health binning, the PEC distribution keeps widening with increasing number of writes (Figure 11(b)) and the vast majority of blocks remains within a small range of the RBER.

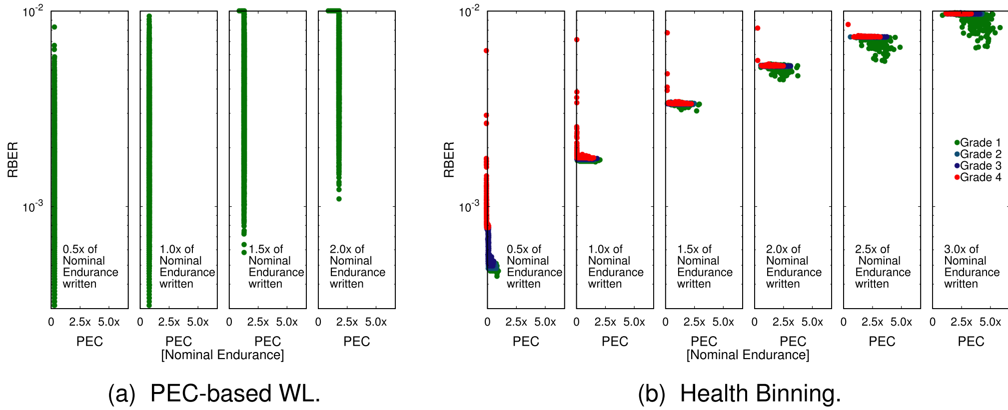


Fig. 11. Block RBER as a function of each block's PEC using PEC-based WL and health binning under a Zipfian 95/20 workload. Despite the heavy skew, PECs only vary marginally for PEC-based WL; however their RBER exhibits huge fluctuations. With health binning, the RBER distribution is significantly narrower. At the same time, healthier grades exhibit lower RBERs and typically have higher PEC counts.

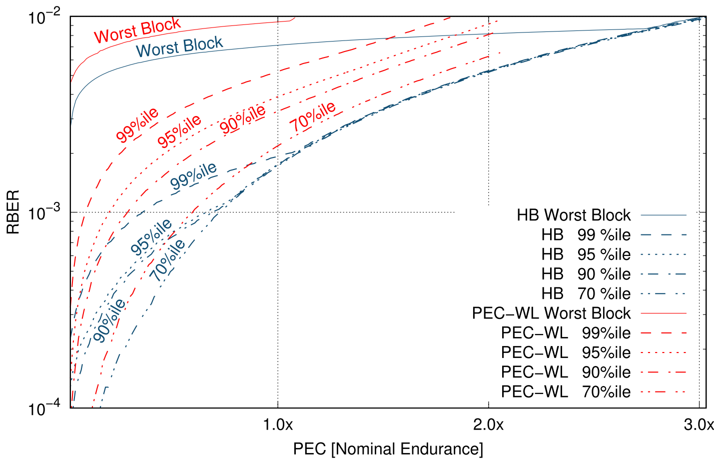


Fig. 12. RBER of worst block and percentiles for PEC-based WL and health binning (HB).

Throughout the device lifetime, healthier blocks have lower health grades and typically also higher PEC counts. Even after the device has seen more writes than 3 times the number of the nominal endurance, the blocks did not exceed the error correction capability of the ECC (although getting very close to it), which demonstrates the benefits of health binning.

Figure 12 shows the number of blocks that is healthier than a given percentile for the entire evaluation. With PEC-based WL, the worst block reaches the error correction capability of the ECC at 1.1× the nominal PEC the device can endure. After that point, spikes in the response time due to read retries as well as uncorrectable errors occur. Note that the PEC-based WL percentiles end shortly after 2× the nominal device endurance, because too many blocks will have been retired.

With health binning, the worst block never reaches the ECC limit and the 99 percentile is between 1.8× and 2.7× lower than that of PEC-based WL and closely approaches the average RBER at about 1.1× the nominal device endurance, thus avoiding unnecessary read retries and uncorrectable read errors.

8 RELATED WORK

NAND flash error characteristics, such as cycling-induced charge trapping, data retention, and read disturb, and their effects on the RBER were analyzed by Mielke et al. [26]. These error sources result in changes in the underlying threshold voltage distributions, which were studied to predict future flash behavior and to design more effective error-correction mechanisms [6, 7]. This led to the introduction of the concept of dynamically adjusted read voltage thresholds to minimize the RBER [31].

Pan et al. [29] describe how blocks can have very different RBER at the same PEC count. Instead of integrating WL with data placement, they perform block swapping and thereby trade off WL against GC efficiency that leads to increased write amplification. Nevertheless, they report noticeably improved WL efficiency. To address retention errors, Cai et al. [8] propose to determine the RBER of flash blocks periodically and perform in-place reprograms where needed. Peleato et al. [33] use the page program time and the PEC besides the RBER to form an error prediction model. If necessary, the healthiest blocks in the pool of coldest blocks will then be exchanged with the most worn blocks available. Again this leads to additional writes that affect write amplification. As we combine data placement with heat segregation and health binning, no blocks have to be swapped, and write amplification does not increase.

In contrast to these RBER-based research activities, a recent large-scale reliability study of particular SSDs in the field found that the RBER and uncorrectable errors are not correlated [36]—an observation that can be explained when aggregate statistics are used instead of collecting the worst-page BER for every page in the system. In addition, the uncorrectable errors they observe could be the result of PEC-based WL and suboptimal or no voltage shifting.

From the flash management point of view, early work showed that garbage-collection and block-erasure policies are best separated from wear leveling, and if considered together, heuristics that predict future request sequences can improve endurance [2]. The greedy and windowed GC algorithms have been widely studied for uniform-random write workloads [5, 10, 17, 40]. However, they do not consider skewed workloads, which are more dominant in real workloads than random ones, and for which the benefits can be significantly higher. Desnoyers [12] extends the analysis and reports on the challenges of simple hot and cold data segregation but does not discuss segregation of host and relocation writes.

Significant endurance improvements from static WL have been reported by Chang et al. [9] using a time based trigger to relocate cold data. They use the effective PEC count, which refers to the number of times a block has been erased, since it has been involved in a swap event. They also found that combining hot and cold data causes an increase in relocations.

Several studies address data placement and heat segregation: Kim et al. [23] introduced a flash management scheme that segregates hot and cold data. Cold data are identified and relocated periodically. As this increases write amplification they must take the relocations into account when evaluating the GC cleaning cost. Hu et al. [18] proposed container marking where the address space is partitioned into heat regions. Pages are promoted to a hotter region on overwrite and demoted to a colder region if they are relocated during GC. Min et al. [27] separate write data into four streams and estimate the heat of a LEB based on the average update frequency of all blocks in the LEB in a predefined period of time. They observe significant improvements in performance and write amplification with precise estimation of heat. As the latter turns out to be expensive, they propose using an approximation instead. Neither of the approaches estimates or considers block health in the approach. Using an object-based model, Kan et al. [22] showed that file-system data and metadata segregation reduces GC overhead.

Using an enlarged write caches [43] or dedicating a small section of the MLC/TLC flash for use as an SLC write cache are other options to address endurance limitations. Last but not least, one can always improve logical endurance by increasing over-provisioning, and thereby reducing write amplification. However, all those options adversely affect the cost per GiB of the device.

9 CONCLUSIONS

New flash technologies mainly focus on cost reduction, but disregard other properties such as endurance and performance. Without sophisticated flash management, it is no longer possible to match the device properties of previous generations. In enterprise storage, where endurance and performance are paramount, these challenges can only be addressed in a holistic fashion.

We presented a complementary set of techniques that enable us to elevate the performance and endurance of next-generation flash to the levels required in enterprise storage. In particular, we introduce novel algorithms for tracking optimal threshold voltage levels to reduce RBER. Focusing on real-world workloads that typically exhibit skewed I/O access patterns, we outline the importance of data placement for both garbage collection and wear leveling, and introduce heat segregation with health binning as an integral approach with low overhead. The combination of the heat metrics of accesses to data with the health metrics of the physical flash memory enables us to enhance endurance and reduce write amplification at the same time. To the best of our knowledge, this is the first time all the above ideas have been addressed holistically in the literature. In particular, this is the first study that (1) combines data-placement and garbage collection aspects with block health variability, (2) quantifies the system-level endurance and write amplification gains, and (3) introduces and evaluates a base-delta block calibration scheme.

Our results demonstrate that the ideas presented can increase the device endurance dramatically without incurring any performance penalty and using only acceptable additional resources. ORVs with base and delta shift separation alone can increase endurance up to 3×, and our data placement with heat segregation reduces write amplification up to 5×, achieving in up to 15× higher overall endurance together. Further endurance gains can be obtained by using a stronger ECC. This allows the scaling to larger all-flash array sizes in the future, making them very practical for use in real-world systems.

Finally, we implemented our techniques in a commercially available all-flash array, but have to emphasize that all results and parameters presented in this article are neither specific to any particular all-flash array nor to any flash memory-product line.

ACKNOWLEDGMENTS

We gratefully acknowledge Tom Griffin and Gary Tressler at IBM Systems for providing the characterization data used in this article.

REFERENCES

[1] Jens Axboe. 2014. FIO—Flexible IO Tester. Retrieved from https://linux.die.net/man/1/fio.

[2] Avraham Ben-Aroya and Sivan Toledo. 2006. Competitive analysis of flash-memory algorithms. In *Proceedings of 14th Annual European Symposium on Algorithms (ESA'06)*. 100–111. DOI: https://doi.org/10.1007/11841036_12

[3] Simona Boboila and Peter Desnoyers. 2010. Write endurance in flash drives: Measurements and analysis. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST'10)*. 115–128.

[4] Luc Bouganim, Björn Por Jonsson, and Philippe Bonnet. 2009. uFLIP: Understanding flash IO patterns. In *Biennial Conference on Innovative Data Systems Research (CIDR'09)*.

[5] Werner Bux and Ilias Iliadis. 2010. Performance of greedy garbage collection in flash-based solid-state drives. *Perform. Eval.* 67, 11 (Nov. 2010), 1172–1186.

[6] Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai. 2013. Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'13)*. 1285–1290.

[7] Yu Cai, O. Mutlu, E. F. Haratsch, and Ken Mai. 2013. Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation. In *Proceedings of the IEEE 31st International Conference on Computer Design (ICCD'13)*. 123–130.

[8] Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman Unsal, and Ken Mai. 2012. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *Proceedings of the IEEE Conference on Computer Design (ICCD'12)*. 94–101. DOI: https://doi.org/ICCD.2012.6378623

[9] Li-Pin Chang. 2007. On efficient wear leveling for large-scale flash-memory storage systems. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC'07)*. 1126–1130.

[10] Li-Pin Chang, Tei-Wei Kuo, and Shi-Wu Lo. 2004. Real-time garbage collection for flash-memory storage systems of real-time embedded systems. *ACM Trans. Embed. Comput. Syst.* 3, 4 (Nov. 2004), 837–863.

[11] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (Feb. 2013), 74–80. DOI: https://doi.org/10.1145/2408776.2408794

[12] Peter Desnoyers. 2012. Analytic modeling of SSD write performance. In *Proceedings of the 5th Annual International Systems and Storage Conference (SYSTOR'12)*. 12:1–12:10. DOI: https://doi.org/10.1145/2367589.2367603

[13] Yoav Etsion and Dror G. Feitelson. 2012. Exploiting core working sets to filter the L1 cache with random sampling. *IEEE Trans. Comput.* 61, 11 (2012), 1535–1550. DOI: https://doi.org/10.1109/TC.2011.197

[14] Eran Gal and Sivan Toledo. 2005. Algorithms and data structures for flash memories. *Comput. Surv.* 37, 2 (Jun. 2005), 138–163. DOI: https://doi.org/10.1145/1089733.1089735

[15] Alessandro Grossi, Lorenzo Zuolo, Franesco Restuccia, and Piero Olivo. 2015. Quality-of-service implications of enhanced program algorithms for charge-trapping NAND in future solid-state drives. *IEEE Trans. Device Mater. Rel.* 15, 3 (Sept. 2015), 363–369. DOI: https://doi.org/10.1109/TDMR.2015.2448108

[16] Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. 2009. Characterizing flash memory: Anomalies, observations, and applications. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42)*. 24–33.

[17] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman Pletka. 2009. Write amplification analysis in flash-based solid state drives. In *Proceedings of the Israeli Experimental Systems Conference (SYSTOR'09)*. 10:1–10:9. DOI: https://doi.org/10.1145/1534530.1534544

[18] Xiao-Yu Hu, Robert Haas, and Evangelos Eleftheriou. 2011. Container marking: Combining data placement, garbage collection and wear levelling for flash. In *Proceedings of the 2011 IEEE 19th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'11)*. 237–247. DOI: https://doi.org/10.1109/MASCOTS.2011.50

[19] Jian Huang, Anirudh Badam, Laura Caulfield, Suman Nath, Sudipta Sengupta, Bikash Sharma, and Moinuddin K. Qureshi. 2017. FlashBlox: Achieving both performance isolation and uniform lifetime for virtualized SSDs. In *Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST'17)*. 375–390.

[20] IBM. 2015. FlashSystem 900. Retrieved from http://www-03.ibm.com/systems/storage/flash/900/.

[21] JEDEC 2017. *Stress-Test-Driven Qualification of Integrated Circuits*. Retrieved from http://jedec.org/.

[22] Yangwook Kang, Jingpei Yang, and Ethan L. Miller. 2010. Efficient storage management for object-based flash memory. In *Proceedings of the 18th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'10)*. 407–409. DOI: https://doi.org/10.1109/MASCOTS.2010.52

[23] Han-Joon Kim and Sang-Goo Lee. 1999. A new flash memory management for flash storage system. In *Proceedings of the 23rd International Computer Software and Applications Conference (COMPSAC'99)*. 284–289. DOI: https://doi.org/10.1109/CMPSAC.1999.812717

[24] Youngjoo Lee, Hoyoung Yoo, Injae Yoo, and In-Cheol Park. 2012. 6.4 Gb/s multi-threaded BCH encoder and decoder for multi-channel SSD controllers. In *Proceedings of the IEEE International Solid-State Circuit Conference (ISSCC'12)*. DOI: https://doi.org/ISSCC.2012.6177075

[25] Jai Menon and Larry Stockmeyer. 1998. An age-threshold algorithm for garbage collection in log-structured arrays and file systems. In *High Performance Computing Systems and Applications*. 119–132. DOI: https://doi.org/10.1007/978-1-4615-5611-4_13

[26] Neal Mielke, Todd Marquart, Ning Wu, Jeff Kessenich, Hanmant P. Belgal, Eric Schares, Falgun Trivedi, Evan Goodness, and Leland R. Nevill. 2008. Bit error rate in NAND flash memories. In *Proceedings of the 46th Annual Int. Reliability Physics Symposium (IRPS'08)*. 9–19. DOI: https://doi.org/10.1109/RELPHY.2008.4558857

[27] Changwoo Min, Kangnyeon Kim, Hyunjin Cho, Sang-Won Lee, and Young Ik Eom. 2012. SFS: Random write considered harmful in solid state drives. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST'12)*. 139–154. http://dl.acm.org/citation.cfm?id=2208461.2208473

[28] Vidyabhushan Mohan, Taniya Siddiqua, Sudhanva Gurumurthi, and Mircea R. Stan. 2010. How I learned to stop worrying and love flash endurance. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'10)*.

[29] Yangyang Pan, Guiqiang Dong, and Tong Zhang. 2013. Error rate-based wear-leveling for NAND flash memory at highly scaled technology nodes. *IEEE Trans. VLSI Syst.* 21, 7 (Jul. 2013), 1350–1354. DOI: https://doi.org/10.1109/TVLSI.2012.2210256

[30] Nikolaos Papandreaou, Theodore Antonakopoulos, Urs Egger, Aspa Palli, Haris Pozidis, and Evangelos S. Eleftheriou. 2013. A versatile platform for characterization of solid-state memory channels. In *Proceedings of the 2013 18th International Conference on Digital Signal Processing (DSP'13).* 1–5. DOI: https://doi.org/ICDSP.2013.6622745

[31] Nikolaos Papandreou, Thomas Parnell, Haris Pozidis, Thomas Mittelholzer, Evangelos S. Eleftheriou, Charles J. Camp, Thomas J. Griffin, Gary A. Tressler, and Andrew A. Walls. 2014. Using adaptive read voltage thresholds to enhance the reliability of MLC NAND flash memory systems. In *Proceedings of the 24th ACM Great Lakes Symp. on VLSI (GLSVLSI'14).* 151–156. DOI: https://doi.org/10.1145/2591513.2591594

[32] Ki-Tae Park, Sangwan Nam, Daehan Kim, Pansuk Kwak, Doosub Lee, Yoon-He Choi, Myung-Hoon Choi, Dong-Hun Kwak, Doo-Hyun Kim, Min-Su Kim, Hyun-Wook Park, Sang-Won Shim, Kyung-Min Kang, Sang-Won Park, Kangbin Lee, Hyun-Jun Yoon, Kuihan Ko, Dong-Kyo Shim, Yang-Lo Ahn, Jinho Ryu, Donghyun Kim, Kyunghwa Yun, Joonsoo Kwon, Seunghoon Shin, Dae-Seok Byeon, Kihwan Choi, Jin-Man Han, Kye-Hyun Kyung, Jeong-Hyuk Choi, and Kinam Kim. 2015. Three-dimensional 128 Gb MLC vertical NAND flash memory with 24-WL stacked layers and 50 MB/s high-speed programming. *IEEE J. Solid-State Circ.* 50, 1 (Jan. 2015), 204–213. DOI: https://doi.org/10.1109/JSSC.2014.2352293

[33] B. Peleato, H. Tabrizi, R. Agarwal, and J. Ferreira. 2015. BER-based wear leveling and bad block management for NAND flash. In *Proceedings of the IEEE International Conference on Communications (ICC'15).* 295–300. DOI: https://doi.org/10.1109/ICC.2015.7248337

[34] Roman A. Pletka and Saša Tomić. 2016. Health-binning: Maximizing the performance and the endurance of consumer-level NAND flash. In *Proceedings of the 9th ACM International Systems and Storage Conference (SYSTOR'16).* Article 4, 10 pages. DOI: https://doi.org/10.1145/2928275.2928279

[35] Mendel Rosenblum and John K. Ousterhout. 1992. The design and implementation of a log-structured file system. *ACM Trans. Comp. Syst.* 10, 1 (Feb. 1992), 26–52. DOI: https://doi.org/10.1145/146941.146943

[36] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. 2016. Flash reliability in production: The expected and the unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST'16).* 67–80. http://dl.acm.org/citation.cfm?id=2930583.2930589

[37] Radu Stoica and Anastasia Ailamaki. 2013. Improving flash write performance by using update frequency. *Proc. VLDB Endow.* 6, 9 (Jul. 2013), 733–744. DOI: http://dx.doi.org/10.14778/2536360.2536372

[38] Fei Sun, Ken Rose, and Tong Zhang. 2006. On the use of strong BCH codes for improving multilevel NAND flash memory storage capacity. In *Proceedings of the IEEE Workshop on Signal Processing Systems: Design and Implementation (SiPS'06).* 241–249. DOI: https://doi.org/10.1049/iet-cds:20060275

[39] Benny Van Houdt. 2013. A mean field model for a class of garbage collection algorithms in flash-based solid state drives. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'13).* ACM, New York, NY, 191–202. DOI: https://doi.org/10.1145/2465529.2465543

[40] Benny Van Houdt. 2013. A mean field model for a class of garbage collection algorithms in flash-based solid state drives. *SIGMETRICS Perform. Eval. Rev.* 41, 1 (Jun. 2013), 191–202. DOI: https://doi.org/10.1145/2494232.2465543

[41] Benny Van Houdt. 2013. Performance of garbage collection algorithms for flash-based solid state drives with hot/cold data. *Perform. Eval.* 70, 10 (Oct. 2013), 692–703. DOI: https://doi.org/10.1016/j.peva.2013.08.010

[42] Steven E. Wells. 1994. Method for Wear Leveling in a Flash EEPROM Memory. U.S. Patent 5 341 339.

[43] Jingpei Yang, Ned Plasson, Greg Gillis, and Nisha Talagala. 2013. HEC: Improving endurance of high performance flash-based cache devices. In *Proceedings of the 6th International Systems and Storage Conference (SYSTOR'13).* 10:1–10:11. DOI: https://doi.org/10.1145/2485732.2485743

[44] Yue Yang and Jianwen Zhu. 2016. Write skew and zipf distribution: Evidence and implications. *ACM Trans. Stor.* 12, 4 (Jun. 2016), 21:1–21:19. DOI: https://doi.org/10.1145/2908557

[45] Kai Zhao, Wenzhe Zhao, Hongbin Sun, Tong Zhang, Xiaodong Zhang, and Nanning Zheng. 2013. LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST'13).* 243–256. http://dl.acm.org/citation.cfm?id=2591272.2591298

[46] Lorenzo Zuolo, Christian Zambelli, Rino Micheloni, Marco Indaco, Stefano Di Carlo, Paolo Prinetto, Davide Pertozzi, and Piero Olivo. 2015. SSDExplorer: A virtual platform for performance/reliability-oriented fine-grained design space exploration of solid state drives. *IEEE Trans. Comput.-Aid. Design Integrat. Circ. Syst.* 34, 10 (Oct. 2015), 1627–1638. DOI: https://doi.org/10.1109/TCAD.2015.2422834

Received February 2017; revised March 2018; accepted July 2018