GuardedErase: Extending SSD Lifetimes by Protecting Weak Wordlines

Duwon Hong, *Seoul National University;* Myungsuk Kim, *Kyungpook National University;* Geonhee Cho, Dusol Lee, and Jihong Kim, *Seoul National University*

https://www.usenix.org/conference/fast22/presentation/hong

This paper is included in the Proceedings of the
20th USENIX Conference on File and Storage Technologies.

February 22–24, 2022 • Santa Clara, CA, USA

978-1-939133-26-7

Open access to the Proceedings
of the 20th USENIX Conference on
File and Storage Technologies
is sponsored by USENIX.

GuardedErase: Extending SSD Lifetimes by Protecting Weak Wordlines

Duwon Hong*
Seoul National University

Myungsuk Kim*
Kyungpook National University

Geonhee Cho
Seoul National University

Dusol Lee
Seoul National University

Jihong Kim
Seoul National University

Abstract

3D NAND flash memory enables the continuous growth in the flash capacity by vertically stacking wordlines (WLs). However, as the number of WLs in a flash block increases, 3D NAND flash memory exhibits strong process variability among different WLs, which makes it difficult for an SSD to fully utilize the maximum endurance of flash blocks, thus reducing the SSD lifetime. In this paper, we propose a new system-level block erase scheme, called GuardedErase, for extending the lifetime of a 3D flash block. The key feature of GuardedErase is that when a block is erased, a WL of the block can be selectively erased by one of two erase modes, the low-stress erase mode or the normal erase mode. When a WL is erased by the low-stress erase mode, the lifetime of the WL can be significantly extended although it may not store data. By supporting two erase modes at the WL level, GuardedErase enables an FTL to exploit the new endurance-capacity trade-off relationship at the SSD level. We have implemented the GuardedErase-aware FTL, called longFTL, which extends the SSD lifetime with a negligible impact on the overall I/O performance. Experimental results using various workloads show that longFTL can improve the SSD lifetime on average by 21% over an existing FTL with little degradation on the SSD performance.

1 Introduction

3D NAND flash memory has been a key enabling technology for sustaining a continuous capacity increase in flash storage systems [1, 2]. Since the capacity of 3D NAND flash memory depends on the number of vertical wordlines (WLs) in a flash chip, as the capacity of 3D NAND flash memory increases, the number of WLs tends to increase as well. Since the size of a flash block also depends on the number of vertical WLs, one of the key characteristics of 3D NAND flash memory over 2D NAND flash memory is that the block size gets bigger as the capacity of a flash chip increases. Table 1 shows how the capacity of a single block has been changed in recent 3D NAND flash chips. For example, the block size has increased by 5.3 times in 4 years. Since the page size does not change as fast as the block size, larger blocks typically contain more WL[1].

Table 1: Changes in the page size, the number of pages per block, and the block size over time in 3D flash memory.

Year	Total Capacity (Gb)	Block size (KB)	Page size (KB)	No. of pages per block
2014 [3]	128	3072	8	384
2015 [4]	128	6144	16	384
2016 [5]	256	9216	16	576
2017 [6]	512	12288	16	768
2018 [7]	1024	16384	16	1024

Another unique characteristic of 3D NAND flash memory is that there exists strong process variability among different WLs of a block [8, 9]. Process variability occurs because of the manufacturing process of 3D NAND flash memory. Since 3D NAND flash memory is manufactured using a vertically successive etching process from the topmost WL to the bottom WL, the cell structure of WLs may significantly vary along the z-axis, leading to significant WL-to-WL variations. As the number of WLs in a block increases, process variability among different WLs increases as well. Figure 1 illustrates that there exist significant variations in the maximum number of P/E cycles among different WLs. For example, the maximum number of P/E cycles of the weakest WL (i.e., WL *w*) is 46.3% smaller than that of the strongest WL (i.e., WL *b*).

A large reliability variation among different WLs makes it difficult to fully utilize all WLs, for example, up to the maximum endurance of each WL. Since the reliability of weak WLs deteriorates faster than that of strong WLs, the lifetime of a block is decided by the reliability characteristics of the weakest WL in the block. When the weakest WL reaches its maximum bit error rate (BER) value, the whole block becomes a *bad* block. This type of coarse-grained bad block management (BBM) is commonly used in SSDs [10]. Although the block is classified as bad under coarse-grained

*The first two authors contributed equally to this research.

[1]Since a single WL can contain multiple pages, the exact number of pages varies depending on the type of flash memory.

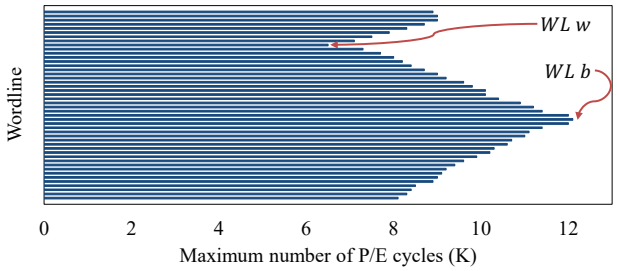


Figure 1: Per-WL variations on the maximum number of P/E cycles.

BBM, all the WLs in the block except for the weakest WL are still usable, thus significantly underutilizing WLs in the block. For example, in Figure 1, when WL *w* reaches its maximum P/E cycles (i.e., about 6,500 P/E cycles), the whole block becomes bad although WL *b* can reliably store data at least for 5,600 more P/E cycles. In order to overcome the shortcomings of coarse-grained BBM, we can manage bad *pages* instead of bad *blocks* so that the good WLs of a bad block can continue to be used. This type of fine-grained scheme is called bad page management (BPM) [11]. Although simple and somewhat effective in extending the SSD lifetime, BPM techniques, which inevitably reduce the total SSD capacity at the end of its lifetime, significantly degrade I/O performance.

A more intelligent solution to mitigate a reliability imbalance between the weakest WL and the rest of WLs would be to protect the weakest WL from degrading its reliability too early. For example, the program relief technique, which was proposed for 2D NAND flash memory [12], tries to reduce the wear stress of weak WLs. When the program relief technique is used, weak WLs are skipped from program operations or are programmed using a less-stressful program method (e.g., SLC programs instead of MLC/TLC programs). However, since erase operations, not program operations, are the main source of NAND flash wear-out [13, 14], the program relief technique is quite limited in extending SSD lifetimes. In our 3D flash characterization study (see Section 3), we confirmed that most flash wear-out comes from an erase operation, not from a program operation in 3D NAND flash memory. In order to effectively erase a large 3D flash block, a higher erase voltage is used for a longer period of time during an erase operation, thus significantly increasing the impact of erase operations on 3D flash over 2D flash wear-out. Our experimental observation strongly suggests a need for a new erase-centric flash stress-relief technique. In this paper, we propose such a stress-relief technique that focuses on erase operations.

As an effective stress-relief solution for 3D NAND flash memory, we propose a new block erase scheme, called GuardedErase (or gErase), that delays weak WLs from reaching their maximum endurance level so that the lifetime of a block can be extended. In order to extend the lifetime of weak WLs, GuardedErase employs two erase modes, normal erase mode and low-stress erase mode, at the WL level. When a WL is erased by the low-stress erase mode, the WL experiences reduced wear stress from a block erase operation, thus effectively increasing its maximum number of P/E cycles. For example, assume that a WL reaches its endurance limit with 10 normal erase operations. In this case, if the low-stress erase mode were used for the remaining erase operations, the WL will take more than 10 P/E cycles before the WL becomes bad. If the wear stress of the low-stress erase mode is just 50% of that of the normal erase mode, it takes 20 more P/E cycles before the WL becomes bad.

On the other hand, when the WL is erased by the low-stress erase mode, the WL is not fully erased so that it cannot reliably store data. That is, when the WL is erased by the low-stress erase mode, the effective capacity of the block is temporarily reduced by the capacity of the WL. Therefore, the key technical challenge of applying GuardedErase at the SSD level is how to efficiently extend the block lifetime using the low-stress erase mode while not degrading I/O performance from the reduced block capacity. More formally, assume that the lifetime L_S of an SSD S is given by the total amount of written data in terabytes, TBW_S, which can be expressed by the following equation:

$$TBW_S = \frac{C_S \times MAX_{P/E}}{WAF}, \quad (1)$$

where C_S is the capacity of the SSD S, $MAX_{P/E}$ is the maximum number of P/E cycles, and WAF is a total write amplification factor of an FTL including garbage collection (GC) and wear leveling. When GuardedErase is used carelessly, it will increase both $MAX_{P/E}$ and WAF, thus limiting its effectiveness in increasing TBW_S.

The low-stress erase mode of GuardedErase focuses on reducing an erase voltage applied to a WL because the erase voltage is the dominant factor affecting the flash wear-out status. In our prototype implementation, we exploited a custom flash test command [15] that allows to change various NAND flash parameters including the erase voltage level of each WL. Using the low-stress erase mode, we performed an extensive characterization study for understanding the reliability impact of the low-stress erase mode. We observed that the wear-stress of a WL under the low-stress erase mode (as implemented in the prototype) was about 35% of that under the normal erase mode. Based on the per-WL low-stress erase mode, we built a NAND endurance model for gErase-enabled flash blocks. Our NAND endurance model supports nine different low-stress block erase modes that differ in their endurance improvement ratios and block capacity reduction factors.

Exploiting the endurance-capacity trade-off of the proposed NAND endurance model, we have implemented the gErase-aware FTL, called longFTL, which dynamically changes the number of WLs erased by the low-stress erase mode while minimizing the I/O performance degradation from the SSD capacity reduction from the low-stress erase mode. LongFTL includes several gErase-specific FTL modules (such as the

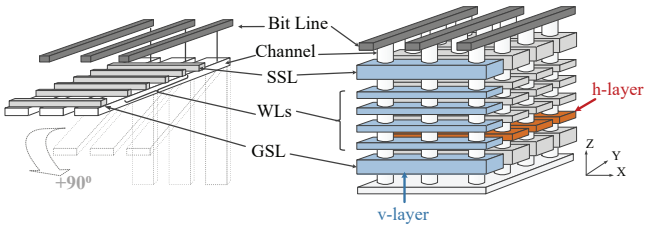


Figure 2: Differences between 2D NAND flash and 3D NAND flash [9].

weak WL monitor and the gErase mode selector) to achieve two conflicting goals, extending SSD lifetimes and maintaining SSD performance. We evaluated the effectiveness of longFTL with the MQSim simulation environment [16] with a custom extension for GuardedErase. Experimental results using various I/O traces show that longFTL can improve the SSD lifetime an average by 21% over an existing gErase-unaware FTL with an average 3% decrease in the overall I/O performance.

The rest of this paper is organized as follows. We review an overview of 3D NAND flash memory and explain how reliability is managed in 3D NAND flash in Section 2. In Section 3, we present key reliability characteristics of 3D NAND flash blocks. The proposed GuardedErase scheme is described in Section 4 and longFTL is covered in Section 5. Sections 6 and 7 describe our evaluation results and related work, respectively. We conclude in Section 8 with a summary and future work.

2 Background

2.1 Overview of 3D NAND Flash Memory

3D NAND flash memory [17] enables the continuous growth in the flash capacity by vertically stacking the memory cell to overcome various technical challenges in scaling 2D NAND flash memory. By exploiting the vertical dimension for higher capacity (instead of focusing on finer process technologies in 2D flash memory), 3D NAND flash memory has been successful in increasing its capacity by 50% annually while avoiding reliability degradation [18].

Figure 2 shows the organizational difference in a flash block[2] between 2D and 3D NAND flash memory. In this example, the 2D NAND flash memory has a matrix structure in which four WLs and three bitlines (BLs) intersect at 90 degrees. On the contrary, the 3D NAND flash memory has a *cube-like* structure. The 3D NAND flash block consists of four vertical layers (v-layers) in the y-axis where each v-layer has four vertically stacked WLs that are separated by select-line (SSL) transistors. As shown in Figure 2, when the 2D NAND flash block is rotated by 90° in a counterclockwise direction using the x-axis as an axis of rotation (i.e., if the WLs are set vertically), it corresponds to a single v-layer. Similarly, the 3D NAND flash block can be described to have four horizontal layers (h-layers) which are stacked along the z-axis, and each horizontal layer consists of four WLs.

In order to increase the capacity of a 3D flash chip, the most effective approach is to increase the number of h-layers in 3D NAND flash memory (i.e., stacking more h-layers along the z-axis). As the number of h-layers increases, the block size increases as well (as summarized in Table 1).

2.2 Reliability Management in NAND Flash Memory

In order to reliably store data in flash cells, various reliability requirements should be satisfied. For example, flash blocks should not be used more than their limited maximum P/E cycles. This is because the NAND cell characteristics in the flash block deteriorate as the number of P/E cycles increases. The main cause of wear-out of a flash block is electrical stress on the tunnel oxide during the program and erase operations. As program/erase operations are repeatedly performed on the block, the amount of charge trapped in the tunnel oxide layer increases. The trapped charges make it difficult for the threshold voltage level (i.e., state) of the erased NAND cells to be located within their intended voltage interval, thus significantly affecting the data retention characteristics of NAND cells in the block. [19]. Therefore, as the P/E cycle increases, the BER characteristics of data stored in the flash block deteriorate.

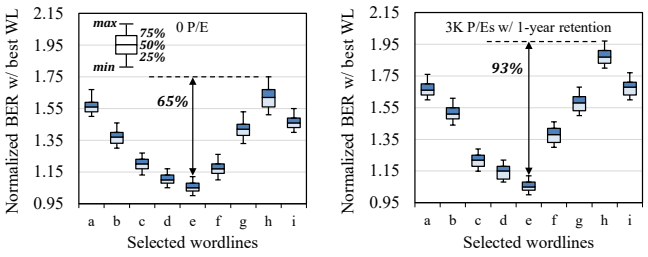
To prevent the number of error bits from surpassing the correction capacity of an error correction code (ECC) engine, NAND manufacturers set the maximum number of P/E cycles allowed for a block, which is called as NAND endurance. If the block continues to be used over the maximum number of P/E cycles, the BER of the block will exceed the ECC capability, which results in data loss (i.e., a read error). In addition to errors due to NAND wear-out, various errors (such as program errors and erase errors) can occur due to process defects during a NAND flash manufacturing procedure [20].

Although NAND operations can fail for different reasons, failed operations are managed in the block granularity within an FTL. For example, if a read operation to a page in a block B fails, the FTL identifies the block B as a bad block and replaces B with a reserved block. After the BBM module of the FTL moves all the valid data from the bad block B to the reserved block, the bad block B is no longer used.

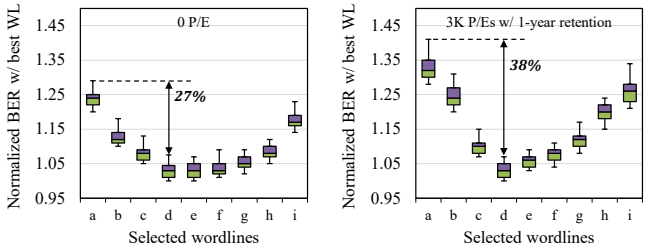
3 Reliability Characterization of 3D Blocks

In this section, we explain two key observations on 3D NAND flash characteristics which motivated the proposed GuardedErase scheme.

[2] NAND flash memory consists of multiple blocks. Each block has multiple WLs (e.g., 128 - 256 WLs) and each WL consists of a group of flash cells (e.g., 8K - 16K cells).



(a) Per-WL BER of 3D flash. (b) Per-WL BER of 3D flash.



(c) Per-WL BER of 2D flash. (d) Per-WL BER of 2D flash.

Figure 3: Per-WL BER variations in 3D NAND flash and 2D NAND flash.

3.1 Large Reliability Variations Among WLs

Ideally, we would expect all flash cells in a block (or chip) to have identical characteristics. However, in practice, significant electrical/physical characteristic variations between flash cells are unavoidable due to many unexpected process fluctuations in a complex NAND flash manufacturing system. For example, in 2D NAND flash memory, it is known that the reliability of WLs varies depending on the physical location of a WL within a flash block. Furthermore, it is widely accepted that such process variability would be much stronger in 3D NAND flash memory because of its unique manufacturing process. To quantify the reliability variations between WLs, we examined the inter-WL BER variations using 160 real 3D TLC NAND flash chips[3]

Figure 3(a) shows significant inter-WL BER variations exist within a tested block even when the blocks experience no program/erase cycles. All BER values were normalized over that of the most reliable WL. (For simplicity, we show the evaluation result of the selected 9 WLs.) The BER of the worst WL (WL h, which is near the top layer) is about 60% higher than that of the best WL (WL e, which is around the middle layer). When flash blocks get aged (from a large number of P/E cycles), as shown in Figure 3(b), the BER of the worst WL may be twice as large as that of the best WL. We

[3]Our flash chips are fabricated by 3D charge-trap technology (which is known as *SMArT* [21] or *TCAT* [22]). All the commercial 3D NAND flash memories have similar structures and cell types, and it is commonly believed that different 3D TLC flash chips share key device-level characteristics. Therefore, our characterization results on reliability variability agree with general trends with other 3D NAND flash chips.

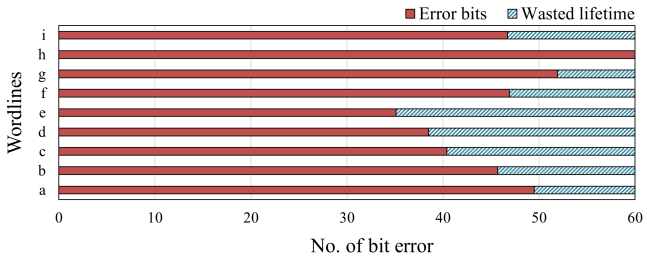


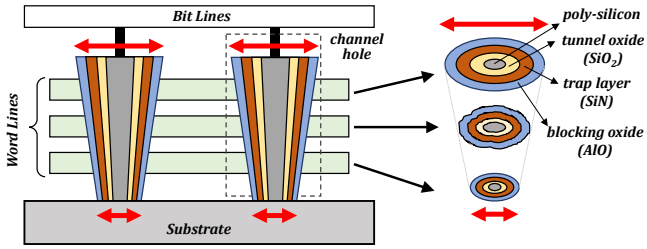
Figure 4: Inter-WL BER variations when the worst WL becomes bad.

also examined the inter-WL reliability variations of 1x-nm 2D TLC NAND flash memory. Unlike 3D NAND flash memory, as shown in Figures 3(c) and 3(d), the BER difference between the worst WL and best WL is much smaller.

In order to understand the impact of large reliability variations among WLs on the NAND block lifetime, we examined per-WL BER variations of different WLs when the number of bit errors from the worst WL exceeds the maximum ECC correction capacity. Figure 4 shows that when the number of bit errors from the worst WL, WL h, exceeds the ECC correction capacity, 60 bits per 1-KB sector, the rest of the WLs in the block are still fairly reliable. If we decide that the block is bad at this time, a significant amount of the block lifetime is wasted. For example, the BER of the best WL, WL d, is just about 60% of that of the worst WL, WL h. Therefore, to fully utilize the lifetime of 3D NAND flash memory, it is important to manage the reliability of a flash block at the individual WL level instead of the conventional block level because there exist large reliability variations among WLs in a flash block.

The root cause of large reliability variations is related to a unique manufacturing process to form the vertical architecture of 3D NAND flash memory. Figure 5(a) shows a detailed organization of a vertical layer in 3D NAND flash memory using a cross-sectional view along the y-z plane and a top-down view (of three cross sections along the x-y plane). The channel holes are formed at the early stage of 3D NAND flash manufacturing by an etching process [23].

Under an ideal manufacturing process, all the channel holes would have the same geometrical structure regardless of their physical locations to achieve the homogeneous reliability



(a) A cross-sectional view. (b) A top-down view.

Figure 5: Inter-WL variations in 3D NAND flash memory.

characteristics among flash cells. However, the cylindrical channel hole cannot avoid suffering from severe structural variations depending on its vertical (z-directional) location. For example, as shown in Figure 5(b), the diameter of the channel holes varies significantly over the height of an h-layer. Furthermore, the shape of channel holes is also affected depending on vertical positions. Different channel hole diameters as well as their shapes can cause large variations in the characteristics of flash cells. Therefore, even when the same program/erase voltage is applied to flash cells, they experience different electrical stress depending on the diameter or shape of a channel hole, resulting in different reliability characteristics along vertical locations.

3.2 Erase Stress on Flash Reliability

Before we design a WL-level stress mitigation technique for 3D NAND flash memory, in order to quantitatively understand the main source of flash stress, we performed a comprehensive characterization study using real 160 3D TLC NAND flash chips with 48 horizontal layers where each layer consists of 4 WLs. Since both a program operation and an erase operation incur a significant amount of wear stress on flash cells, the main goal of our study was to measure the relative impact of these operations on the reliability characteristics of 3D flash cells. We tested a total of 3,686,400 WLs (11,059,200 pages) to obtain statistically significant experimental results. Following a standard evaluation metric commonly used in NAND flash reliability studies, we measured changes in RBER (Raw Bit-Error Rate) after each measurement scenario.

To quantify the impact of the program and erase operations on the NAND endurance stress, we designed three measurement scenarios. In each scenario, one of three operation sequences is repeated until the lifetime limit of the block. Table 2 summarizes three operation sequences with their member operations. Note that all three sequences include dummy operations so that unexpected factors do not affect the accuracy of measurement results.[4]

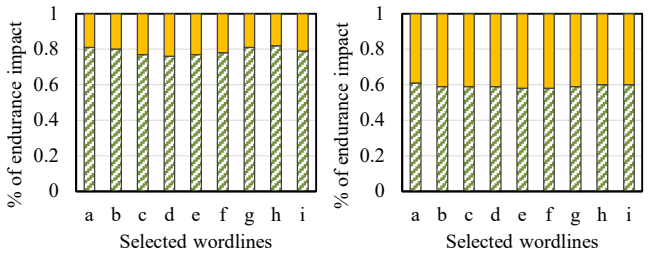
Figures 6(a) and 6(b) show how erase and write operations affect the lifetime of a flash block in 2D and 3D NAND flash memory, respectively. As expected, the endurance impact of an erase operation was significantly larger than that of a program operation in 3D NAND flash memory. The endurance stress of erase operations was responsible for almost 80% of the total stress of flash cells in 3D NAND flash memory. Note

[4] As explained in Section 2.2, the flash cell's wear-out is mainly caused by trapped charges in the tunnel oxide layer during program and erase operations. If the flash cell is sufficiently erased, there is little charge remaining that can be transferred from the SiN layer to the substrate. Therefore, for example, repeating only erase operations in `Erase-only Sequence` cannot accurately measure the endurance effect of the erase operation. As shown in Table 2, dummy program operations are performed before erase operations. Dummy erase operations were added to the sequence because they were included in `Modified Base Sequence` so that the effect of erase operation can be effectively measured by comparing BER values of two sequences.

Table 2: A summary of three operation sequences.

Sequence type	Operations order
`Modified Base Sequence`	P → dummy P → E → dummy E
`Erase-only Sequence`	dummy P → E → dummy E
`Program-only Sequence`	dummy E → P → dummy P

P : program, E : erase



(a) 3D NAND flash. (b) 2D NAND flash.

Figure 6: Endurance impact of erase and program operations.

that the impact of erase operations on the flash endurance is about 33% higher in 3D NAND flash memory over 2D NAND flash memory. The high impact of erase operations over program operations was observed similarly regardless of WL locations. Our measurement study clearly indicates that a low-stress mechanism should be focused on erase operations, not program operations.

3.3 Erase Stress Reduction

As described in Section 2.2, electrical stress during P/E cycles could have a detrimental effect on the tunnel oxide layer of flash cells. As the amount of damage to the tunnel oxide layer increases, flash cells eventually get worn out. As shown in Section 3.2, lowering the erase stress is essential in reducing the amount of damage to the tunnel oxide layer. Since the amount of damage to the tunnel oxide layer increases exponentially as the erase voltage increases [24], the proposed GuardedErase scheme offers a low-stress erase mode by lowering erase voltage.

4 GuardedErase: Mechanism and Modes

4.1 Basic Idea

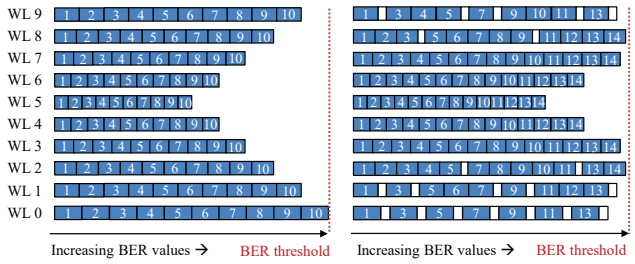
Since an erase operation is a major source of flash wear-out, it is important to reduce the erase stress on flash cells when they are erased if the flash cells can be used for more P/E cycles. In order to reduce the erase stress when necessary, GuardedErase supports the low-stress erase mode as well as the normal erase mode. The key motivation behind GuardedErase is that when a WL is erased by a lower erase voltage, the lifetime of the WL increases [14]. GuardedErase protects weak WLs from

experiencing high erase stress by lowering their erase voltage, thus extending the maximum number of block erasures for a block (i.e., the lifetime of a flash block).

Consider an example 3D flash block with 10 WLs in Figure 7(a). Reflecting strong process variability among WLs, there are significant variations on BER values for the same number of P/E cycles. For example, when the normal block erase operation is used, WL 0, which is the weakest WL, reaches its maximum BER value (i.e., the BER threshold value of the block) after 10 P/E cycles. On the other hand, after 10 P/E cycles, WL 5, which is the strongest WL, barely reached 50% of its maximum BER value. However, since WL 0 reached its maximum BER value, the block is no longer usable, thus becoming a bad block. If we employ a fine-grained BPM policy, the block can continue to be used with nine WLs after 10 P/E cycles. However, since the effective block capacity is reduced by 10% (which, in turn, may introduce a severe WAF increase), the BPM policy is rather limited in increasing the total amount of written data to the block while incurring no I/O performance degradation. In the example block in Figure 7(a), even if an SSD can tolerate up to 20% of the average block capacity reduction (thanks to its OP space), only 9 more WLs can be written to the block before WL 1 becomes unusable after one more block erasure.

Figure 7(b) illustrates how the lifetime of the example block can be extended using the proposed GuardedErase scheme. When BER values of weak WLs (such as WL 0, WL 1 and WL 9) are higher than those of other WLs, weak WLs are erased using the low-stress erase mode that reduces wear by approximately 1/3 as will be shown in Section 4.2. For example, after the 1st erase cycle, the BER value of WL 0 is more than double that of WL 5. In order to protect WL 0, the low-stress erase mode, which is indicated by a white box in Figure 7(b), is used for WL 0 in the 2nd erase cycle. WL 0 is erased six more times using the low-stress erase mode under similar conditions. Figure 7(b) shows that the lifetime of WL 1 and WL 9 is extended to the 14-th erase cycle with five applications of the low-stress erase mode while that of WL 2 and WL 8 is extended to the 14-th erase cycle with two applications of the low-stress erase mode.

When weak WLs are erased using the low-stress erase mode, they cannot store data for the following program cycle (i.e., it cannot reliably store data). In Figure 7(b), WL 0 is not used to store data 7 times out of 14 block erasures. (White boxes indicate the unstable WL state.) Although the total amount of written data to WL 0 is significantly reduced, the lower erase voltage prolongs the lifetime of WL 0, which is the weakest WL of the block. By delaying WL 0 to reach its BER threshold, we can increase the lifetime of the block to 14 block erasures, thus more data are written to the block. In Figure 7(b), the total amount of written data (in the number of WLs) increases from 100 WLs (in Figure 7(a)) to 119 WLs. That is, by using the lower erase voltage for weak WLs, we were able to increase the total amount of data written to a



(a) Using the normal erase mode. (b) Using the gErase mode.

Figure 7: Per-WL BER changes under the normal block erase and the gErase block erase.

block by 19%. This, in turn, extends the SSD lifetime in terms of the total amount of written bytes.

Note that the GuardedErase scheme outperforms the BPM scheme in the total number of WLs written to the block by writing 10 more WLs. Furthermore, unlike the BPM scheme where the effective block capacity is monotonically non-increasing over P/E cycles, the GuardedErase scheme can dynamically control the effective block capacity up to the maximum block capacity depending on which erase mode is used. If the low-stress erase mode can be adaptively applied by exploiting the future write demand and intensity characteristics, the GuardedErase scheme can efficiently extend the SSD lifetime without an I/O performance degradation. In order to achieve the full potential of the GuardedErase scheme, therefore, we need to design an efficient mechanism for supporting the low-stress erase mode and devise an intelligent management policy of applying the low-stress erase mode to proper WLs at the right time.

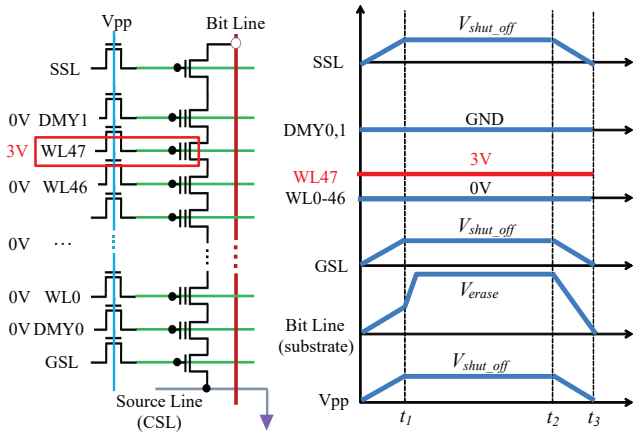
4.2 Per-WL Low-Stress Erase Mode

4.2.1 Implementation

There are two implementation options for the low-stress erase mode. One is to reduce the erase time (i.e., erase time scaling [14]) and the other is to reduce the erase voltage (i.e., erase voltage scaling [14]). However, since it is not easy to control the erase time at the WL granularity, we employed a scheme that lowers the erase voltage. In order to reduce the erase voltage for a specific WL, we exploited a test-mode command [15] for 3D NAND flash memory that allows to vary the driving voltage setting at the WL granularity. Although we cannot directly apply a lower erase voltage to a specific WL, this test-mode command can be used to effectively lower the erase voltage for the target WL[5].

Figure 8(a) illustrates how to reduce the erase stress on WL 47 using the proposed method. Since the voltage difference between the control gate voltage and the voltage applied to

[5]It has been known that all NAND manufactures have their own hidden test-mode commands to modify the internal operating voltage.



(a) Differential voltage driving to control gates.

(b) Voltage changes over time for a block erase.

Figure 8: An implementation of the low-stress erase mode.

the substrate acts as the effective erase voltage, when a higher voltage is applied to the control gate of a WL, its erase voltage is effectively reduced, thus reducing the erase stress on the WL. In Figure 8(a), 3V, not the usual 0V, is applied to the control gate of WL 47, thus reducing the effective erase voltage of WL 47 by 3V. Figure 8(b) shows how various voltages are driven when the low-stress erase mode is used for WL 47 while the rest of WLs are erased using the normal erase mode. When the nominal erase voltage V_{erase} is applied to the bit line (or substrate) from time t_1 to t_2, the NAND flash cells of WL 47 experience a smaller electrical potential difference by 3V over the flash cells of the other WLs. Considering that V_{erase} is approximately $17V$ and the erase stress is exponentially proportional to an electrical potential difference, a significant amount of erase stress is reduced to the NAND flash cells of WL 47 during an erase operation.[6]

4.2.2 Stress Mitigation Effect

In order to understand the endurance impact of the low-stress erase mode on WLs, we performed a comprehensive process characterization study using state-of-the-art 3D TLC NAND flash chips. For endurance comparisons, we formed two block groups whose member blocks were evenly selected from different physical locations using 30 NAND flash chips. Two block groups were erased using different erase modes, one group using the normal erase mode only and the other group using the normal erase mode and the low-stress erase mode half and half,[7] respectively. For each WL of a tested block,

[6] When the low-stress erase mode is applied to a WL, we increase the verify voltage setting for the WL so that the erase operation can take the same number of erase loops as in the normal erase mode.

[7] We experimented with different combinations of two erase modes (e.g., 25% : 75%). However, we could not find any difference in the experimental conclusions.

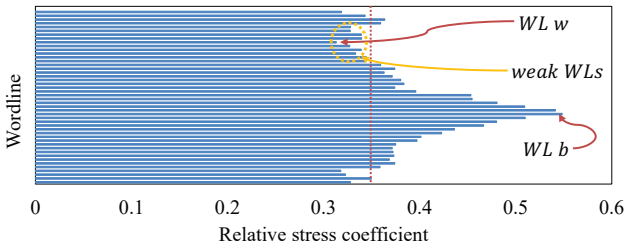


Figure 9: Per-WL relative stress coefficients.

we collected the maximum number of P/E cycles that keeps the BER value of the WL below the BER threshold value.

Based on the measured maximum P/E cycles per WL, we computed the relative stress coefficient S_k of WL_K that indicates how much the erase stress is mitigated when the low-stress erase mode is used. S_k is a per-WL quantity that indicates how much WL_K wears out when it is erased using the low-stress erase mode over the normal erase mode. For example, if the maximum number of P/E cycles of WL_P was 10K when the normal erase mode was used, but it was increased to 20K when the low-stress erase mode is used, the relative stress coefficient S_p is 0.5. Figure 9 summarizes our measured relative stress coefficients for different WLs in a block.

Although there are significant differences in S_k values depending on WLs, we observed that the relative stress coefficients of weak WLs (for which most of the low-stress erase mode are applied) are quite similar: all the coefficients belong to an interval [0.31, 0.34] with the mean of 0.33. Although the low-stress erase mode may be applied to a few strong WLs with larger relative stress coefficients in GuardedErase, when we evaluate the stress mitigation effect of the low-stress erase mode, we assume that all the WLs have the same relative stress coefficient, 0.35, for a simple analysis.[8]

4.3 Per-Block Erase Modes

In order for an FTL to effectively exploit the endurance-capacity trade-off of the low-stress erase mode, we support nine block erase modes, gE(1), ..., gE(9). Table 3 summarizes the proposed nine block erase modes with varying numbers of protected WLs and different erase relief ratios[9]. The higher n in gE(n), the more WLs are erased using the low-stress erase mode with a higher erase relief ratio. Therefore, when a block is erased with a higher gErase mode, the maximum number of P/E cycles of the block is increased. For example, when a block is erased with gE(9) only, the maximum number of P/E

[8] In GuardedErase, a fixed number N of WLs are selected for applying the low-stress erase mode. Although N is fixed, selected N WLs change over time because N WLs with the worst BER values are protected when a block is erased. Since the relative stress coefficient of selected WLs is mostly less than 0.35 with few stronger WLs, our assumption of 0.35 is rather conservative in understanding real stress mitigation levels.

[9] The erase relief ratio of gE(n) is defined as a ratio of applying the gE(n) block erase mode over the normal block erase operation.

Table 3: A summary of nine gErase modes.

	gErase mode(n) == gE(n)								
	gE(1)	gE(2)	gE(3)	gE(4)	gE(5)	gE(6)	gE(7)	gE(8)	gE(9)
No. of protected WLs	8	12	16	20	24	24	24	28	32
Erase relief ratio	25%	33%	38%	40%	42%	50%	50%	50%	50%
Block capacity reduction	1.04%	2.08%	3.13%	4.17%	5.21%	6.25%	7.29%	8.33%	9.38%
Norm. Max P/E cycles	1.19	1.26	1.30	1.33	1.37%	1.39	1.41	1.43	1.45

cycles of the block is increased by 45% whereas when the block is erased with gE(1), it is increased by 19% only. On the other hand, as shown in Table 3, the higher the gErase mode, the more WLs become unusable for the next program cycle, thus increasing WAF values. As described in Section 5, it is an important task for an FTL to choose a proper gErase mode when it erases a block.

In the remainder of Section 4.3, we provide a high-level description on how we designed nine block erase modes. (For additional details, see [25].) In Table 3, the percentage of block capacity reduction in gE(n) is given by (1.04 × n)%. Since our flash block has 192 WLs, when a block is erased by gE(n), $2n$ WLs of the block become unusable for the next program cycle[10]. When a target percentage of block capacity reduction, say 1.04% in gE(1), is given, there can be multiple options to achieve the target block capacity reduction ratio using the low-stress erase mode. Let n_{WL} and f denote the number of protected WLs and the erase relief ratio. A combination $(k \times n_{WL}, f/k)$ over any k achieves the same block capacity reduction ratio as when $(n_{WL} \times f/100)$ WLs are always protected. For example, (4 WLs, 50%), (8 WLs, 25%) and (12 WLs, 16.7%) can achieve the same 1.04% reduction percentage in a 192-WL blocks. From multiple candidate combinations, we prefer ones with a lower f because such a combination can achieve higher I/O efficiency by allowing more flexible applications of gErase modes by an FTL [25]. For example, we prefer (8 WLs, 25%) and (12 WLs, 16.7%) over (4 WLs, 50%). On the other hand, if f is too low (i.e., n_{WL} is too large), weak WLs may not be fully protected, thus limiting their endurance improvements [25]. For example, (12 WLs, 16.7%) cannot maximally extend the lifetime of the weakest WL, WL 0, because f should be larger than 18.3% if WL 0 could fully achieve its maximum expected endurance [25]. Therefore, as shown in Table 3, (8 WLs, 25%) was selected for gE(1).

5 Design and Implementation of LongFTL

5.1 Overview

Based on the proposed gErase modes of Table 3, we implemented a gErase-enabled FTL, called longFTL, which exploits the key trade-off relationship of gErase between the block endurance extension and the block capacity reduction. The main goal of longFTL is to significantly extend the block lifetime with a negligible performance degradation by intelligently choosing a gErase mode under varying I/O workloads. Figure 10 shows an overall organization of longFTL. LongFTL, which is based on a typical page-level mapping FTL, employs three gErase-specific modules, the weak WL detector, the WAF monitor, and the gErase mode selector. The weak WL detector dynamically identifies weak WLs in a block, the WAF monitor tracks WAF changes, and the gErase mode selector determines the optimal gErase mode for the current workload characteristics estimated by the WAF monitor. The extended flash block manager selectively applies the optimal gErase mode in producing free blocks during its free-block allocation step.

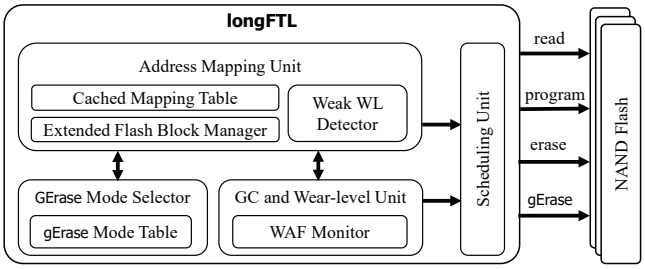


Figure 10: An organizational overview of longFTL.

5.2 Weak WL Detector

In order to protect weak WLs to extend the endurance of a block, it is required to identify which WLs are weak. [11] The weak WL detector (WLD) module is responsible for maintaining WLs according to their BER values so that when N weakest WLs are requested by the gErase mode selector, the WLs with the N largest BER values can be quickly identified. The WLD module employs $(N_{ecc}^{max} - 9)$ linked lists where N_{ecc}^{max} represents the maximum number of bit errors that can be corrected by an ECC module. A linked list L_k contains all the WLs with k bit errors. Figure 11 shows an example of BER-sorted linked lists used for detecting weak WLs.

The BER-sorted linked list is used temporarily when weak WLs are identified. (In the current implementation, we identify weak WLs once in 100 P/E cycles.) We maintain the identified weak WLs using a separate bitmap data structure. For each WL, we allocate a 1-bit flag that indicates the WL is weak or not. Using this bitmap information, longFTL recognizes which WLs should be erased by the low-stress erase mode. The memory footprint for supporting the bitmap of weak WLs is small. For example, if a NAND die consists of 2000 blocks and there are 192 WLs per block, the required memory is less than 48 KB, which is less than 0.3% of the memory requirement of a page-level mapping table.

[10] (2/192) × n ≃ 0.0104 × n.

[11] Since each WL may experience different erase modes (i.e., normal and 9 gErase modes), we maintain a 16-bit relief count for each WL that indicates the current wear status of the WL. However, for a simpler presentation, we assume that each WL is managed based on their BER value. The memory overhead of supporting per-WL relief count can be limited to about 2% of the logical-to-physical mapping table size.

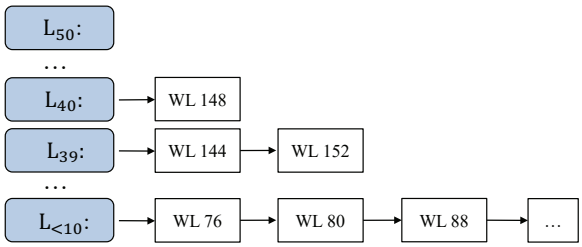


Figure 11: BER-sorted linked lists for detecting weak WLs.

To identify weak WLs of a block, we construct the BER-sorted linked list for the block after every 100 P/E cycles. Since we use the BER-sorted linked list only when weak WLs are identified, the BER-sorted linked list for the block is allocated temporarily from a heap memory and returned after use. That is, the memory requirement for one BER-sorted linked list is sufficient for a whole NAND die with 2000 blocks. Assuming that N_{ecc}^{max} is 50, the maximum memory requirement is less than 2 KB per NAND die when each block has 192 WLs. For example, assuming 4 bytes for a list pointer and 2 bytes for WL information, (50 - 8) headers $\times$ 4 bytes/header + 192 WLs $\times$ (4+2) bytes/WL = 1320 bytes.

Since weak WLs of a block are identified every 100 P/E cycles of the block, its time overhead is minimal. For the current implementation, we check the BER value of a WL by reading its worst page (e.g., MSB page). The total overhead of checking BER values of WLs is typically less than 0.1% of total I/O time in our evaluation. Furthermore, we can distribute the overhead of the BER checking step of each block to multiple P/E cycles (e.g., 91-st to 100-th P/E cycles) by overlapping the weak WL identification step among multiple blocks (e.g., 10 blocks). Since this requires ten BER lists at the same time, the memory requirement increases to about 13 KB (which is less than 0.1% of the page-level mapping table size). However, we only need to check BER values from one-tenth of the total WLs during one P/E cycle interval, thus further reducing the time overhead. Since the BER value of a WL changes slowly, the distributed scheme does not affect the accuracy of identifying weak WLs.

In addition to selecting N weakest WLs from a block, the WLD module ensures that gErase modes are evenly applied to all the blocks. If a large number of gErase modes are used for a few blocks only, the SSD lifetime may not be extended at all because the rest of blocks are erased using the normal erase mode only. We maintain a separate linked list of free blocks by the number of gErase operations. When we select a free block, we prefer blocks with fewer gErase counts[12].

5.3 WAF Monitor

The WAF monitor is responsible for tracking a WAF value of an SSD. Since a block capacity is reduced when the block is erased using a gErase mode, it is important to understand if the current effective SSD capacity is adequate to properly process the current I/O workload. If the effective SSD capacity were reduced too much from aggressive gErase mode applications by the gErase mode selector, the SSD may suffer a significant I/O performance degradation. On the other hand, if gErase modes were applied too conservatively, the effectiveness of GuardedErase is quite limited. In order to prevent the I/O performance fluctuations from frequent gErase-mode changes, we modify gErase modes only when the current I/O workload has been relatively stable so that transient I/O workload variations do not cause an SSD capacity reduction from mispredicted mode changes.

The main function of the WAF monitor is to observe WAF fluctuations, which we interpret as I/O workload changes, and decide if the current WAF value is *stable* enough for the gErase mode selector to make a proper mode decision. Although it is difficult to precisely define what a stable WAF value means, in the current implementation, we assume that a WAF value is stable if the last 9 measured WAF, as well as the current WAF, have been *steady* in their respective observation intervals. A WAF value is called *steady* for an observation interval $(t_s, t_e]$ if all the WAF values observed in $(t_s, t_e]$ are within $[0.98w, 1.02w]$ where w is the WAF value at t_s. Since we are interested in knowing long-term workload characteristics instead of fast changing short-term workload characteristics, a single observation interval is defined as the time it takes to perform a sufficient number (e.g., 5% of total blocks) of garbage collections (GC). The WAF monitor also tracks the change in WAF values (i.e., WAF history) so that the recent WAF trend can be considered by the gErase mode selector.

5.4 GErase Mode Selector

The gErase mode selector decides which erase mode would be used for the next block erase. Figure 12 describes the key steps in selecting the next gErase mode.

In deciding the next erase mode, the gErase mode selector considers two pieces of information from the WAF monitor: WAF stability and WAF history. The gErase mode selector with the current gErase mode gE(cur) invokes the procedure in Figure 12 whenever a new WAF value w is measured (i.e., every observation interval). If the WAF value is stable, we compute the expected TBW values (from Equation 1) for three neighboring gErase modes, gE(cur-1), gE(cur) and gE(cur+1), and select the gErase mode with the largest TBW value as the next gErase mode to be used. If not all three TBW values are computable (i.e., because of WAF history for gE(cur+1) is not available), we set the next gErase mode as gE(cur+1), so that we can explore the potential benefit of using a more aggressive gErase mode. Note that this explorational step is tried in a limited fashion only after an I/O workload signif-

[12] Since a free block may be selected under a different criterion (such as low P/E cycles), leveling gErase counts among different blocks is supported in a combined fashion with other selection criteria.

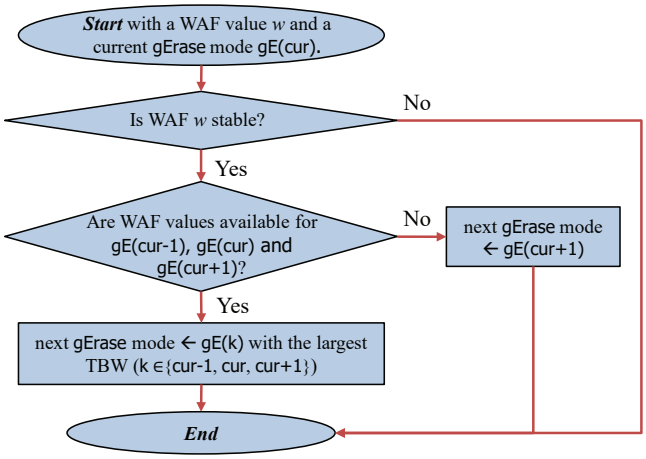


Figure 12: A procedure for selecting the next gErase mode.

icantly changes when most previous WAF values become unstable. Once a WAF value becomes unstable for gE(n), the TBW value for gE(n) cannot be computed until the WAF value get re-stabilized again.

Figure 13 illustrates how the gErase mode selector works over varying I/O workloads. The gErase mode is started with gE(0). When the WAF is saturated (❶), there is no WAF history for the adjacent gErase mode, so it is changed to gE(1) without a TBW comparison. When the WAF is saturated after the mode change (❷, ❸) a higher mode is used because TBW gets higher over the previous mode. However, the TBW of gE(3) is lower than that of gE(2), the mode changes back to gE(2). Since the newly calculated TBW of gE(2) is lower than the TBW of gE(1) calculated by WAF history (❺), it is changed back to gE(1). In the actual gE(1) execution, the TBW is rather low (❻), so the mode is changed back to gE(2). After that, the WAF remains stable, and the TBW is maximized at the mode, gE(2).

5.5 Extended Flash Block Manager

Once the next gErase mode is determined, the extended flash block manager decides whether to apply the gErase mode before generating free blocks that can be used for incoming allocation requests. In order to minimize the negative impact on the I/O performance from applying gErase modes, we take a conservative policy in employing gErase modes.

In the current implementation, the extended flash block manager uses gErased free blocks only for storing hot data from host requests. The main rational behind this policy is that the negative performance impact of the SSD capacity reduction from gErase modes would last longer if cold data (that are not likely to be updated) are allowed to be stored to gErased free blocks. Since a gErased block with cold data is less likely to be selected as a victim block of garbage collection, the unused WLs from the gErased block will take longer to be reclaimed for future usage. Therefore, the impact

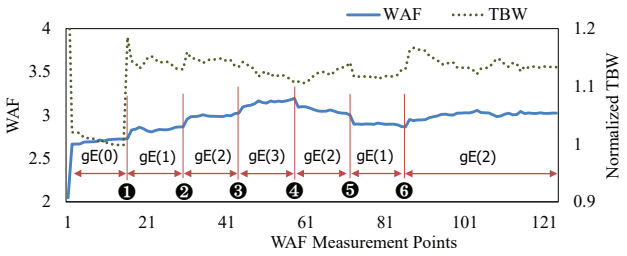


Figure 13: An example gErase mode selection.

of the SSD capacity reduction on WAF will last longer. Note that our current policy prevents gErased free blocks from storing valid data moved during a garbage collection process because they are implicitly cold data [26]. Since valid data moved during the garbage collection process are considered cold, we do not store them to gErased blocks, thus reducing I/O performance degradation.

6 Experimental Results

6.1 Experimental Settings

To evaluate the effectiveness of the proposed technique, we implemented longFTL as an extension on a well-known FTL simulator, MQSim [16]. For our evaluation, we configured the FTL simulator to support the 32-GB storage capacity for faster experiments. Our simulated storage system has four channels with one NAND flash chip per channel. In setting the key NAND flash configuration parameters, recent large-block flash chips [5, 6] were used as references. Each NAND flash chip has a 2-plane configuration and each plane has 1822 blocks. Each block consists of 192 WLs and each WL can store three 8-KB pages. The average page program time and the block erase time were set to 700 μs and 4000 μs, respectively. The overprovisioning ratio was set to 10% as commonly done in commercial SSDs. To effectively exploit the overprovisioned area, GC is invoked when the number of free blocks is less than 0.2% of the total number of blocks.

We compared longFTL with two existing schemes: Baseline, and BPM [11]. Baseline represents a standard page-level mapping FTL without any special scheme for optimizing large-block NAND flash chips. BPM is the same FTL as Baseline with the bad-page management scheme [11]. All evaluation results were normalized over those from Baseline.

We have carried out our experiments with various I/O traces with different I/O characteristics from MSR trace workloads [27] and synthetic I/O traces generated from Sysbench [28] and Filebench [29]. Table 4 summarizes key I/O characteristics of these workloads. Since longFTL improves its lifetime at the expense of temporarily reduced SSD capacity, we selected I/O traces (e.g., prxy0 and src10) with a large amount of written data so that we can better understand how longFTL works under reduced SSD capacity scenarios. When the total amount

Table 4: I/O characteristics of traces used for evaluations.

	proj0	prxy1	prxy0	src10	proj2	OLTP	fileserver	varmail
Read:Write	6:94	64:36	5:95	46:54	86:14	70:30	40:60	40:60
Total writes (GB)	144	725	54	302	169	639	249	312
WAF (without gErase)	1.08	1.16	1.24	2.66	3.55	1.89	2.49	2.98

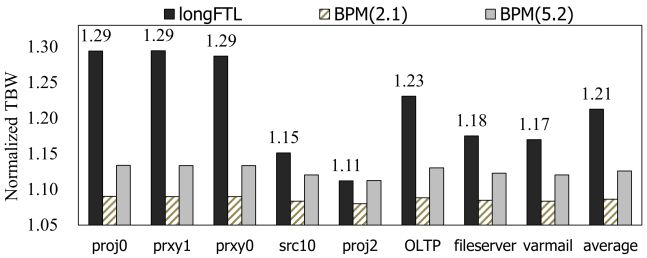


Figure 14: Comparisons of lifetime extensions.

of written data from an I/O trace was not sufficient, the same traces were repeated multiple times so that sufficient write requests can be generated. In Table 4, prxy0, proj0 and proj2 were such cases. Since the MSR traces were extracted from slow HDD-based storage systems, we accelerated the I/O intensity from a host machine by scaling down inter-request intervals by an appropriate factor (e.g., $\frac{1}{3000}$) so that fast SSD processing speed can be properly considered.

6.2 Lifetime Improvement

In order to understand how longFTL improves the SSD lifetime, we measured TBW values for eight I/O traces. Figure 14 shows normalized TBW values for the benchmark I/O traces over Baseline. For the BPM techniques, we used two versions, BPM(2.1) and BPM(5.2), where BPM(r) allows the block capacity to be reduced by r% of the block capacity. LongFTL achieved the highest average improvement ratio, 21%, on the lifetime extension in all eight traces, followed by BPM(5.2) and BPM(2.1). As expected, the lifetime improvement of both BPM(2.1) and BPM(5.2) was insensitive on I/O characteristics of each trace, achieving 9% and 13% improvement ratios mainly depending on the upper limit of the block capacity reduction.

Although longFTL outperformed BPM(2.1) and BPM(5.2) on all the benchmark traces, the efficiency of longFTL significantly varies over different traces. For example, longFTL improved the SSD lifetime by 29% in proj0, prxy1 and prxy0. On the other hand, longFTL can improve the SSD lifetime only by 11% for proj2. The main reason behind a rather large efficiency difference is because the impact on the WAF value of each trace is quite different when gErase modes are applied.

Figure 15 shows how the WAF values change in longFTL. In Figure 15, we included the most commonly used gErase mode for each I/O trace (which we call the dominant gErase mode). In general, the more the number of protected WLs, the higher the WAF value. However, as shown in Figure 15, the exact relationship is workload-dependent. For example, in proj0 and prxy1, where longFTL was the most effective in

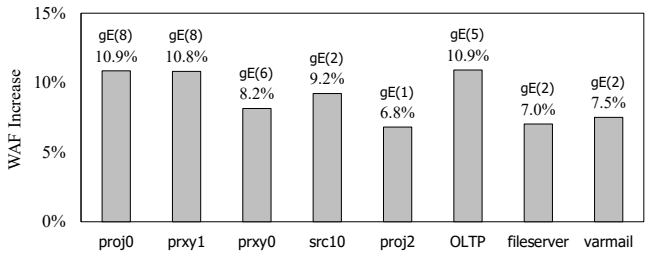


Figure 15: Comparisons of WAF changes under dominant gErase modes used.

increasing the SSD lifetime, the WAF value is only increased by 11% even when the dominant gErase mode protects 28 WLs with gE(8). On the contrary, in proj2, its WAF was very sensitive to the number of protected WLs. Thus we mostly used gE(1) with 8 protected WLs only. When 1.04% of the physical capacity was reduced by gE(1), the WAF value was increased by 7%. Therefore, no higher gE(n) mode was used for proj2, limiting the effect of longFTL on the SSD lifetime. This workload dependency is the main motivation for the WAF Monitor in longFTL. Furthermore, it emphasizes the importance of an FTL for the low-erase modes to be effectively utilized.

6.3 Performance Overhead

Although longFTL selectively decreases the effective SSD capacity from gErase modes, its negative impact on the I/O performance is very small because longFTL applies gErase operation mostly for hot data (as described in Section 5.5). Figure 16 compares normalized IOPS values of three techniques over the Baseline scheme. LongFTL experiences the average 3% IOPS degradation over Baseline. On the contrary, BPM(5.2) degrades its IOPS on average by 10%. Furthermore, BPM(2.1) achieves a similar IOPS level as longFTL but it is ineffective in improving the SSD lifetime. BPM(5.2) may not be employed in practice because it violates the common SSD performance requirement specification (such as the 10% upper limit on the SSD performance degradation at the end of the rated product lifetime [30]). BPM(5.2) can experience more than 20% performance degradation as in proj2.

6.4 Effectiveness of Block Relief Policy

As explained in Section 4.3, longFTL prefers limited applications of gErase modes. For example, when we build the gErase modes of Table 3, we selected ones with the lowest erase relief ratio. Furthermore, longFTL applied gErase modes only for storing host-requested data as described in Section 5.5. That is, no gErase mode is used when storing data during GC. In order to understand the effectiveness of the current conservative policy of longFTL, we compared this policy with a more aggressive policy where gErase modes were always used when erasing blocks.

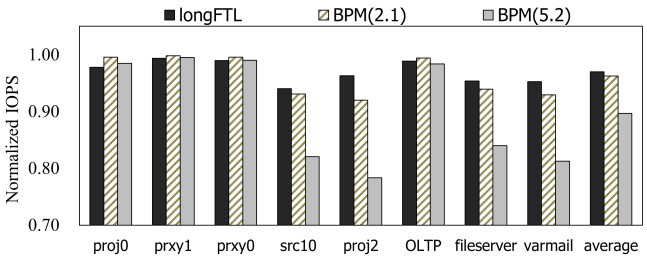


Figure 16: Comparisons of performance overhead.

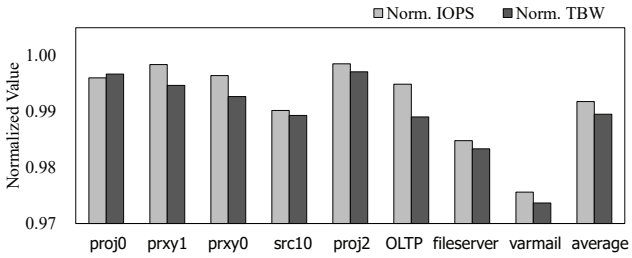


Figure 17: Performance and lifetime comparisons of an aggressive gErase application policy.

Figure 17 shows how the more aggressive policy works over the current one. All the values were normalized over those of longFTL. The evaluation result shows that TBW and IOPS can degrade up to 3% (as in varmail). In the aggressive policy, the TBW improvement ratio can be reduced by up to 15.2%[13] over the current conservative policy.

7 Related Work

Improving the SSD lifetime has been an active research topic because the endurance of modern flash memory is continuously reduced. To overcome the limited lifetime problem, several system-level techniques have been proposed by exploiting the physical characteristics of NAND flash memory [11–14].

Jeong *et al.* conducted a study to improve the NAND endurance by adjusting the erase voltage and erase time [13, 14]. It is similar to our gErase scheme in that it improves the block lifetime by reducing the stress voltage during erase operations. However, unlike the gErase scheme, their technique does not exploit the intra-block reliability variations (i.e., between pages within a single block). It only focuses on reducing the erase-caused stress at the block granularity.

Jimenez *et al.* focused on the intra-block reliability variations and conducted a study to improve NAND endurance through a program stress relief technique [12]. However, this approach is not effective in modern 3D NAND flash memory because the main cause of NAND flash wear-out is erase operations, not program operations. Our gErase scheme focuses on erase operations as a stress relief target. Furthermore, longFTL based on gErase handles a stress relief approach in a more complete fashion, considering the impact of gErase modes on WAF changes.

Debao *et al.* proposed a BPM technique that improves the lifetime by continuously using the remaining pages instead of classifying the entire block as a bad block when a read error occurs on one of pages in the block [11]. The BPM technique may be considered as a valid solution to tackle large reliability variations among pages in the block. However, it is difficult to apply the BPM technique without large I/O performance degradation. Unlike our gErase scheme, the BPM technique cannot recover bad pages for future write requests. Therefore, once the SSD capacity is reduced, the SSD capacity is permanently reduced so that future write-intensive requests cannot be properly serviced.

[13] (1 - (17.8/21.0)) × 100 ≃ 15.2%.

8 Conclusions

We have presented the GuardedErase scheme, a new block erase scheme for modern 3D NAND flash memory, that protects weak WLs from reaching their maximum endurance too early so that the lifetime of a block can be extended. The GuardedErase scheme exploits the trade-off relationship between the NAND endurance and the block capacity so that the SSD lifetime can be effectively extended with minimal impact on I/O performance.

Based on the per-WL low-stress erase mode that was devised from an extensive 3D NAND flash characterization study, we proposed nine different gErase modes and implemented a gErase-aware FTL, longFTL, which uses appropriate gErase modes under varying I/O workload characteristics. Our experimental results show that longFTL can improve the SSD lifetime by 21% on average with negligible degradation on the SSD performance.

The current version of longFTL can be further improved in several directions. For example, in the current version, the impact of a newly selected gE(n) on the future WAF change is measured *after* gE(n) is applied. If it could be predicted in advance with high accuracy, a better gE(n) could be selected earlier, thus further improving the SSD lifetime. It may be an interesting future direction to devise such a predictor using data-driven machine learning techniques.

9 Acknowledgments

We would like to thank Brad Settlemyer, our shepherd, and the anonymous reviewers for their valuable comments that greatly improved our paper. This work was supported by Samsung Research Funding & Incubation Center of Samsung Electronics, Republic of Korea, under Project Number SRFC-IT2002-06, and by Samsung Electronics Co., Ltd. (IO201207-07809-01). The ICT at Seoul National University provided research facilities for this study. (Corresponding Author: Jihong Kim)

References

[1] Hyunsuk Kim, Su-Jin Ahn, Yu Gyun Shin, Kyupil Lee, and Eunseung Jung. Evolution of nand flash memory: From 2d to 3d as a storage market leader. In *Proceedings of IEEE International Memory Workshop (IMW)*, 2017.

[2] Mark LaPedus. "3d nand's vertical scaling race". https://semiengineering.com/3d-nands-vertical-scaling-race, 2020.

[3] Ki-Tae Park, Sangwan Nam, Daehan Kim, Pansuk Kwak, Doosub Lee, Yoon-He Choi, Myung-Hoon Choi, et al. Three-dimensional 128gb mlc vertical nand flash-memory with 24-wl stacked layers and 50mb/s high-speed programming. In *Proceedings of International Solid-State Circuits Conference*, 2014.

[4] Woopyo Jeong, Jae-woo Im, Doo-Hyun Kim, Sang-Wan Nam, Dong-Kyo Shim, Myung-Hoon Choi, Hyun-Jun Yoon, et al. A 128gb 3b/cell v-nand flash memory with 1gb/s i/o rate. In *Proceedings of International Solid-State Circuits Conference*, 2015.

[5] Dongku Kang, Woopyo Jeong, Chulbum Kim, Doo-Hyun Kim, Yong Sung Cho, Kyung-Tae Kang, Jinho Ryu, et al. 256gb 3b/cell v-nand flash memory with 48 stacked wl layers. In *Proceedings of International Solid-State Circuits Conference*, 2016.

[6] Ryuji Yamashita, Sagar Magia, Tsutomu Higuchi, Kazuhide Yoneya, Toshio Yamamura, Hiroyuki Mizukoshi, Shingo Zaitsu, et al. A 512gb 3b/cell flash memory on 64-word-linelayer bics technology. In *Proceedings of International Solid-State Circuits Conference*, 2017.

[7] Seungjae Lee, Chulbum Kim, Minsu Kim, Sung-min Joe, Joonsuc Jang, Seungbum Kim, Kangbin Lee, et al. A 1tb 4b/cell 64-stacked-wl 3d nand flash memory with 12mb/s program throughput. In *Proceedings of International Solid-State Circuits Conference*, 2018.

[8] Chun-Hsiung Hung, Meng-Fan Chang, Yih-Shan Yang, Yao-Jen Kuo, Tzu-Neng Lai, Shin-Jang Shen, Jo-Yu Hsu, , et al. Layer-aware program-and-read schemes for 3d stackable vertical-gate be-sonos nand flash against cross-layer process variations. *IEEE Journal of Solid-State Circuits*, 50(6):1491–1501, 2015.

[9] Youngseop Shim, Myungsuk Kim, Myoungjun Chun, Jisung Park, Yoona Kim, and Jihong Kim. Exploiting process similarity of 3d flash memory for high performance ssds. In *Proceedings of IEEE/ACM International Symposium on Microarchitecture*, 2019.

[10] Micron. TN-29-59: Bad block management. https://www.micron.com/-/media/client/global/documents/products/technical-note/nand-flash/tn2959_bbm_in_nand_flash.pdf, 2011.

[11] Wei Debao, Qiao Liyan, Zhang Peng, and Peng Xiyuan. Bpm: A bad page management strategy for the lifetime extension of flash memory. In *Proceedings of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2015.

[12] Xavier Jimenez, David Novo, and Paolo Ienne. Wear unleveling: Improving nand flash lifetime by balancing page endurance. In *Proceedings of USENIX Conference on File and Storage Technologies*, 2014.

[13] Jaeyong Jeong, Sangwook Shane Hahn, Sungjin Lee, and Jihong Kim. Lifetime improvement of nand flash-based storage systems using dynamic program and erase scaling. In *Proceedings of USENIX Conference on File and Storage Technologies*, 2014.

[14] Jaeyong Jeong, Youngsun Song, Sangwook Shane Hahn, Sungjin Lee, and Jihong Kim. Dynamic erase voltage and time scaling for extending lifetime of nand flash-based ssds. *IEEE Transactions on Computers*, 66(4):616–630, 2016.

[15] Rino Micheloni, Luca Crippa, and Alessia Marelli. *Inside NAND flash memories*. Springer Science & Business Media, 2010.

[16] Arash Tavakkol, Juan Gómez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu. Mqsim: A framework for enabling realistic studies of modern multi-queue {SSD} devices. In *Proceedings of USENIX Conference on File and Storage Technologies*, 2018.

[17] Jungdal Choi and Kwang Soo Seol. 3d approaches for non-volatile memory. In *Proceedings of Symposium on VLSI Technology-Digest of Technical Papers*, 2011.

[18] Youngwoo Park, Jaeduk Lee, Seong Soon Cho, Gyoyoung Jin, and Eunseung Jung. Scaling and reliability of nand flash devices. In *Proceedings of the IEEE International Reliability Physics Symposium*, 2014.

[19] Yixin Luo, Saugata Ghose, Yu Cai, Erich F Haratsch, and Onur Mutlu. Heatwatch: Improving 3d nand flash memory device reliability by exploiting self-recovery and temperature awareness. In *Prpc. International Symposium on High Performance Computer Architecture (HPCA)*, 2018.

[20] Samsung v-nand technology, white paper. https://studylib.net/doc/8282074/samsung-v-nand-technology, 2014.

[21] Eun-Seok Choi and Sung-Kye Park. Device considerations for high density and highly reliable 3d nand flash cell in near future. In *Proceedings of IEEE International Electron Devices Meeting (IEDM)*, 2012.

[22] Jaehoon Jang, Han-Soo Kim, Wonseok Cho, Hoosung Cho, Jinho Kim, Sun Il Shim, Jae-Hun Jeong, et al. Vertical cell array using tcat (terabit cell array transistor) technology for ultra high density nand flash memory. In *Proceedings of IEEE Symposium on VLSI Technology (VLSI)*, 2009.

[23] Jaehoon Jang, Han-Soo Kim, Wonseok Cho, Hoosung Cho, Jinho Kim, Sun Il Shim, Jae-Hun Jeong, et al. Vertical cell array using TCAT (Terabit Cell Array Transistor) technology for ultra high density NAND flash memory. In *Proceedings of the IEEE Symposium on VLSI Technology (VLSI)*, 2009.

[24] Klaus F Schuegraf and Chenming Hu. Effects of temperature and defects on breakdown lifetime of thin sio/sub 2/at very low voltages. In *Proceedings of IEEE International Reliability Physics Symposium (IRPS)*, 1994.

[25] Duwon Hong. *Optimizing the Reliability and Performance of Ultra-large SSDs*. PhD thesis, Dept. of Computer Science and Engineering, College of Engineering, Seoul National University, 2021.

[26] Benny Van Houdt. On the necessity of hot and cold data identification to reduce the write amplification in flash-based ssds. *Performance Evaluation*, 82:1–14, 2014.

[27] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write off-loading: practical power management for enterprise storage. In *Proceedings of USENIX Conference on File and Storage Technologies*, 2008.

[28] Sysbench. http://github.com/akopytov/sysbench.

[29] Filebench. http://filebench.sourceforge.net.

[30] Ulrich Hansen. The ssd endurance race: Who's got the write stuff? https://www.flashmemorysummit.com/English/Collaterals/Proceedings/2012/20120821_TC11_Hansen.pdf, 2012.