공학박사학위논문

낸드 플래시 기반 저장장치의 수명 향상을 위한 계층 교차 최적화 기법

Cross-Layer Optimization Techniques for Extending Lifetime of NAND Flash-Based Storage Devices

2016년 2월

서울대학교 대학원
전기·컴퓨터 공학부
정재용

낸드 플래시 기반 저장장치의 수명 향상을 위한 계층 교차 최적화 기법

Cross-Layer Optimization Techniques for Extending Lifetime of NAND Flash-Based Storage Devices

지도교수 김 지 홍

이 논문을 공학박사 학위논문으로 제출함
2015년 11월

서울대학교 대학원
전기·컴퓨터 공학부
정재용

정재용의 공학박사 학위논문을 인준함
2015년 12월

위 원 장 문 봉 기 (인)

부위원장 김 지 홍 (인)

위 원 유 승 주 (인)

위 원 경 계 현 (인)

위 원 정 명 수 (인)

Abstract

Replacing HDDs with NAND flash-based storage devices (SSDs) has been one of the major challenges in modern computing systems especially in regards to better performance and higher mobility. Although uninterrupted semiconductor process scaling and multi-leveling techniques lower the price of SSDs to the comparable level of HDDs, the decreasing lifetime of NAND flash memory, as a side effect of recent advanced device technologies, is emerging as one of the major barriers to the wide adoption of SSDs in high-performance computing systems.

In this dissertation, we propose new cross-layer optimization techniques to extend the lifetime (in particular, endurance) of NAND flash memory. Our techniques are motivated by our key observation that erasing a NAND block with a lower voltage or at a slower speed can significantly improve NAND endurance. However, using a lower voltage in erase operations causes adverse side effects on other NAND characteristics such as write performance and retention capability. The main goal of the proposed techniques is to improve NAND endurance without affecting the other NAND requirements.

We first present Dynamic Erase Voltage and Time Scaling (DeVTS), a unified framework to enable a system software to exploit the tradeoff relationship between the endurance and erase voltages/times of NAND flash memory. DeVTS includes erase voltage/time scaling and write capability tuning, each of which brings a different impact on the endurance, performance, and retention capabilities of NAND flash memory.

Second, we propose a lifetime improvement technique which takes advantage of idle times between write requests when erasing a NAND block with a slower speed or when writing data to a NAND block erased with a lower voltage. We have implemented a DeVTS-enabled FTL, called dvsFTL, which optimally adjusts the erase voltage/time and write performance of NAND devices in an automatic fashion. Our experimental results show that dvsFTL can improve NAND endurance by 62%, on average, over DeVTS-unaware FTL with a negligible decrease in the overall write performance.

Third, we suggest a comprehensive lifetime improvement technique which exploits variations of the retention requirements as well as the performance requirement of SSDs when writing data to a NAND block erased with a lower voltage. We have implemented dvsFTL+, an extended version of dvsFTL, which fully utilizes DeVTS by accurately predicting the write performance and retention requirements during run times. Our experimental results show that dvsFTL+ can further improve NAND endurance by more than 50% over dvsFTL while preserving all the NAND requirements.

Lastly, we present a reliability management technique which prevents retention failure problems when aggressive retention-capability tuning techniques are employed in real environments. Our measurement results show that the proposed technique can recover corrupted data from retention failures up to 23 times faster over existing data recovery techniques. Furthermore, it can successfully recover severely retention-failed data, such as ones experienced 8 times longer retention times than the retention-time specification, that were not recoverable with the existing technique.

Based on the evaluation studies for the developed lifetime improve-

ment techniques, we verified that the cross-layer optimization approach has a significant impact on extending the lifetime of NAND flash-based storage devices. We expect that our proposed techniques can positively contribute to not only the wide adoption of NAND flash memory in datacenter environments but also the gradual acceleration of using flash as main memory.

Keywords: NAND Flash Memory, Solid State Drive, Storage Management, Storage Reliability, Storage Lifetime, Embedded Software

Student Number: 2012-30229

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

NAND flash-based solid-state drives (SSDs) are widely used in personal computing systems as well as mobile embedded systems. However, in enterprise environments, SSDs are employed in only limited applications because SSDs are not yet cost competitive with HDDs [1]. Fortunately, the prices for SSDs have fallen to the comparable level of HDDs by continuous semiconductor process scaling (e.g., 10 nm-node process [2]) combined with multi-leveling technologies (e.g., MLC [3] and TLC [4]). However, the limited endurance of NAND flash memory, which have declined further as a side effect of the recent advanced device technologies, is emerging as another major barrier to the wide adoption of SSDs. (*NAND endurance* is the ability of a memory cell to endure program/erase (P/E) cycling, and is quantified as *the maximum number* $N_{P/E}^{max}$ *of P/E cycles* that the cell can tolerate while maintaining its reliability requirements [5].) For example, although the NAND capacity per die doubles every two years, the actual lifetime (which is proportional to the total NAND capacity and $N_{P/E}^{max}$) of SSDs does not increase as much as projected in the past seven years because $N_{P/E}^{max}$ has declined by 70% during that period [6]. In order for SSDs to be commonplace in enterprise environments, the issues concerning NAND endurance

should be properly resolved.

Since the Lifetime L_C of an SSD with the total capacity C is proportional to the maximum number $N_{P/E}^{max}$ of P/E cycles, and is inversely proportional to the total written data W_{day} per day, L_C (in days) can be expressed as follows (assuming a perfect wear leveling):

$$L_C = \frac{N_{P/E}^{max} \times C}{W_{day} \times WAF} \quad , \tag{1.1}$$

where WAF is a write amplification factor which represents the efficiency of an FTL algorithm. Many existing lifetime-enhancing techniques have mainly focused on reducing WAF by increasing the efficiency of an FTL algorithm. For example, by avoiding unnecessary data copies during garbage collection, WAF can be reduced [7]. In order to reduce W_{day}, various system-level techniques were proposed. For example, data de-duplication [8], data compression [9], and write traffic throttling [10] are such techniques. On the other hand, only a few system/software-level techniques have been proposed to increase $N_{P/E}^{max}$. Although several conceptual device-level techniques (e.g., a self-healing SSD [11]) were suggested regarding $N_{P/E}^{max}$, it is difficult for these to be employed in real systems because of their unrealistic hardware settings and critical side-effects.

By exploiting the tradeoff relationships between the NAND characteristics (e.g., capacity, performance, retention, and endurance), several cross-layer optimization techniques have been suggested. In order to improve SSD performance, for example, the retention relaxation technique [12] temporarily relaxes the NAND retention capability while FlexFS [13] flexibly reorga-

nizes the NAND capacity between SLC and MLC regions. Although these techniques exploited the device-level physical characteristics in the similar fashion of our work, their main goals are quite different from ours. Up until now, there have been a few particular suggestions to improve the NAND endurance by exploiting the tradeoff relationships between the NAND capabilities.

1.2 Dissertation Goals

In this dissertation, we propose new cross-layer optimization techniques to extend the lifetime of NAND flash-based storage devices by exploiting the tradeoff relationship among NAND capabilities such as endurance, performance, and retention. The primary goals of this dissertation is as follows:

- Enabling a system software to exploit the tradeoff relationship between the endurance and the other capabilities of NAND flash memory.
- Developing system-level techniques to improve NAND endurance while maintaining the other NAND requirements.
- Providing reliability preservation techniques for NAND flash-based storage systems when flash-optimization techniques are widely employed in real environments.

1.3 Contributions

The proposed cross-layer approach in this dissertation adds a new dimension to the decreasing lifetime problem of NAND flash-based storage devices as follows:

- **A unified NAND endurance model** which captures the tradeoff relationship between NAND endurance and the performance/retention capabilities of NAND flash memory is proposed. We reveal that endurance degradation is primarily caused by excessive erase operations, and suggest effective device-level means (i.e., various write-capability tuning techniques) of alleviating the negative impact of erase operations on NAND endurance. Based on the proposed NAND endurance model, a system software can adjust the internal operation voltages and times of NAND flash memory in a reliable fashion.

- **System-level lifetime improvement techniques** for NAND flash-based storage devices are presented. Based on the NAND endurance model, the proposed techniques dynamically change the NAND performance and retention capabilities for each program operation so that endurance-enhancing erase operations can be frequently used. Since the proposed lifetime improvement techniques can efficiently adapt to varying characteristics of I/O workload by accurately predicting the write performance and retention requirements, the overall performance and reliability requirements of storage systems are maintained while significantly improving NAND endurance.

- **Reliability management techniques** for NAND flash-based storage systems are suggested. Since the proposed lifetime improvement techniques aggressively tune down the NAND retention capability to improve NAND endurance, the retention-failure problem can be a serious technical issue for power/temperature-unstable computing environments. In order to preserve the data durability of the stored data in NAND flash memory, we introduce a novel data recovery technique which can efficiently and quickly recover corrupted data from retention failures.

Although this dissertation has mainly focused on improving NAND endurance, our proposed techniques can be extended to improve other requirements (e.g., performance, retention, and read-disturbs resistance) of storage systems. Moreover, since our techniques are entirely independent on data content, the existing flash-optimization techniques can be easily integrated into our proposed framework.

1.4 Dissertation Structure

This dissertation consists of seven chapters. The first chapter presents a introduction to this dissertation while the last chapter serves as a conclusion with a summary and future work. The five intermediate chapters are organized as follows:

Chapter 2 reviews the operational principles of NAND flash memory and explains existing SSD lifetime improvement techniques closely related to this dissertation.

Chapter 3 describes the dynamic NAND voltage and time scaling framework which includes erase voltage/time scaling and write capability tuning. Combining erase scaling and write tuning, a unified NAND endurance model for estimating their effects on NAND endurance is also suggested.

Chapter 4 proposes an SSD lifetime improvement technique using write-performance tuning. We explain how to use a lower voltage and a slower speed for an erase operation and how to write data to a NAND block erased with a lower voltage. In addition, the effect of the proposed technique on NAND endurance is presented in detail.

Chapter 5 presents a comprehensive SSD lifetime improvement technique using both write-performance tuning and retention-capability tuning. We describe reliable prediction schemes to accurately predict the write performance and retention requirement and present efficient adaptation schemes to manage the NAND capabilities. We then show how much NAND endurance is improved and whether the overall NAND requirements are preserved.

Chapter 6 suggests a reliability management technique in order to recover data loss due to retention failures. Finally, we show how efficient the proposed technique is in terms of data recovery power and speed.

Chapter 2

Background

In order to improve NAND endurance, reliability and performance parameters are dynamically changed during run time in this dissertation. In this chapter, we review the basics of key *Vth* design parameters and the principals of a NAND program operation.

2.1 Threshold Voltage Window of NAND Flash Memory

NAND flash memory stores data into cells by changing their *Vth* states depending on bit information, and restores data from cells by sensing their *Vth* states. Figure 1 illustrates an example of *Vth* distributions for an MLC NAND device which stores two bits in a cell by using four distinct *Vth* states distinguished by three read reference voltages.

Aside from serving as a non-volatile storage medium, MLC NAND devices are also required to meet the specified NAND requirements [5]. For example, read and program operations of an MLC device should be completed within 100 μs and 1,600 μs, respectively [6]. Moreover, even after 3,000 P/E cycles, it is required to support up to 400,000 read operations [6] as well as to retain its stored data for up to 1 year at 30 °C [14]. Since the *Vth* design parameters shown in Figure 1 are closely related to the NAND

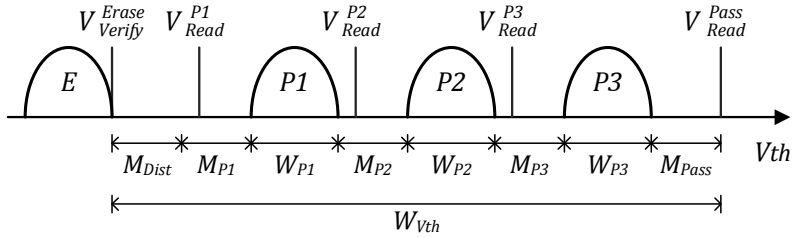


Figure 1: An example of *Vth* distributions for MLC NAND flash memory and primary *Vth* design parameters for the NAND requirements.

requirements, the overall *Vth* distributions should be carefully designed to meet all the NAND requirements under the worst-case operating conditions for a storage product.

The upper *Vth* target V_{Verify}^{Erase} of the E state is one of the key factors in determining the total width W_{Vth} of *Vth* distributions. As V_{Verify}^{Erase} is lowered, W_{Vth} gets widened so that it is easier to optimize the *Vth* parameters for higher performance or longer retention capability. However, as a side-effect of the lowered V_{Verify}^{Erase}, NAND endurance may deteriorate because NAND blocks are more deeply erased [15]. Conversely, when a higher V_{Verify}^{Erase} is used, designing *Vth* distributions becomes more complex because less W_{Vth} is available.

The width W_{Pi} of a *Vth* distribution is mostly determined by the NAND write performance requirement. Since NAND flash memory generally uses the incremental step pulse programming (ISPP) scheme to form *Vth* distributions, W_{Pi} and the program time are directly affected by the ISPP step control. For example, when a fine-grained ISPP step control is used for a program operation, W_{Pi} can be shortened while the program time increases [15]. As a result, W_{Pi} is determined by the minimum achievable

width of a *Vth* distribution under the given program-time requirement.

The *Vth* gap M_{Pi} between two adjacent states is mainly determined by the NAND retention requirement. When NAND memory cells are programmed and left for a long time, charge loss may occur because stress-induced damage in the tunnel oxide layer is likely to loosen stored charges. Since this charge-loss phenomenon may cause *Vth* changes, it is necessary for a sufficient M_{Pi} to tolerate the *Vth* changes. In order to guarantee the NAND retention requirement under the worst-case operating condition, M_{Pi} is determined by the maximum *Vth* change after the maximum number of P/E cycles and the specified retention time.

The *Vth* gap M_{Dist} between the E state and V_{Read}^{P1} primarily affects the program-disturbance resistance and read-disturbance resistance of NAND flash memory. When NAND memory cells are programmed or read, neighbor cells that belong to the E state may be softly programmed so that their *Vth*s move to the right [5][16]. In order to compensate for the *Vth* changes due to these disturbances, a sufficient M_{Dist} should be reserved in the *Vth* window as shown in Figure 1. Typically, M_{Dist} is decided by the *Vth* changes after the maximum number of P/E cycles followed by the maximum number of read cycles.

The read pass voltage V_{Read}^{Pass} which affects the NAND read disturbance is another key factor in deciding the value of W_{Vth}. Since the NAND read disturbance has an exponential dependence on the V_{Read}^{Pass} [17], V_{Read}^{Pass} is usually fixed as low as possible in device design times. The *Vth* gap M_{Pass} between the $P3$ state and V_{Read}^{Pass} is also essential to fully turn on all the NAND memory cells in a block [5].

When the *Vth* design parameters are designated accordingly, all the *Vth* states are placed between V_{Verify}^{Erase} and V_{Read}^{Pass}. Therefore, the total width W_{Vth} of the *Vth* window is expressed as follows (for an MLC NAND device):

$$\begin{aligned} W_{Vth} &= V_{Read}^{Pass} - V_{Verify}^{Erase} \\ &= M_{Dist} + \sum_{i=1}^{3} W_{Pi} + \sum_{i=1}^{3} M_{Pi} + M_{Read}. \end{aligned} \tag{2.1}$$

Since the *Vth* design parameters are highly related to one another, if a certain design parameter is to be changed, we should check its effect on the whole *Vth* window.

2.2 NAND Program Operation

In order to form a threshold voltage distribution within a desired region, NAND flash memory generally uses the incremental step pulse programming (ISPP) scheme. As shown in Figure 2, the ISPP scheme gradually increases the program voltage by the V_{ISPP} step until all the memory cells in a page are located in a desired threshold voltage region. While repeating ISPP loops, once NAND cells are verified to have been sufficiently programmed, those cells are excluded from subsequent ISPP loops.

Since the program time is proportional to the number of ISPP loops (which are inversely proportional to V_{ISPP}), the program time T_{PROG} can

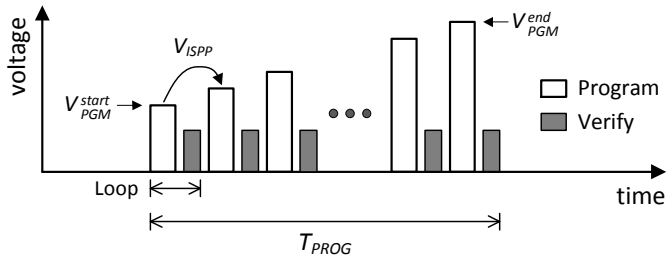


Figure 2: A conceptual timing diagram of the ISPP scheme.

be expressed as follows:

$$T_{PROG} \propto \frac{V_{PGM}^{end} - V_{PGM}^{start}}{V_{ISPP}}. \tag{2.2}$$

Figure 3 shows normalized T_{PROG} variations over different V_{ISPP} scaling ratios. (When a V_{ISPP} scaling ratio is set to $x\%$, V_{ISPP} is reduced by $x\%$ of the nominal V_{ISPP}.) When a narrow threshold voltage distribution is needed, V_{ISPP} should be reduced for a fine-grained control, thus increasing the program time. Since the width of a threshold voltage distribution is proportional to V_{ISPP} [18], for example, if the nominal V_{ISPP} is 0.5 V and the width of a threshold voltage distribution is reduced by 0.25 V, V_{ISPP} also needs to be reduced by 0.25 V (i.e., a V_{ISPP} scaling ratio is 0.5), thus increasing T_{PROG} by 100%.

2.3 Related Work

Since the lifetime of SSDs is inversely proportional to the total written data W_{day} per day and the write amplification factor WAF, existing lifetime-enhancing studies for SSDs have mainly focused on reducing W_{day}

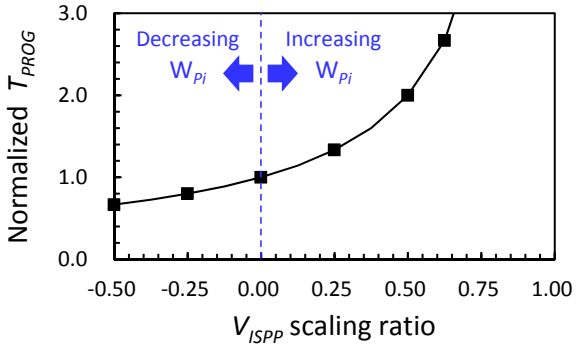


Figure 3: Normalized T_{PROG} variations over different V_{ISPP} scaling ratios.

or WAF. In this section, we briefly review typical examples of existing lifetime-enhancing techniques that reduce W_{day} and WAF, and explain a device-level technique for improving NAND endurance. Finally, we describe one of the cross-layer optimization techniques for better SSD performance, which is an integral motivation behind our work.

2.3.1 System-Level SSD Lifetime Improvement Techniques

Data Compression Technique

In order to reduce W_{day}, many types of flash-aware data compression techniques have been proposed to reduce the logical amount of write traffic to NAND chips. For example, the compression-aware flash translation layer (CaFTL) [9] was suggested to make key FTL modules (e.g., address mapping table and garbage collector) compression-aware so that compression efficiency could be maximized. Figure 4 shows the overall architecture of CaFTL with a page mapping table and a specially-designed data structure

(i.e., a data chunk table) for managing compressed data.

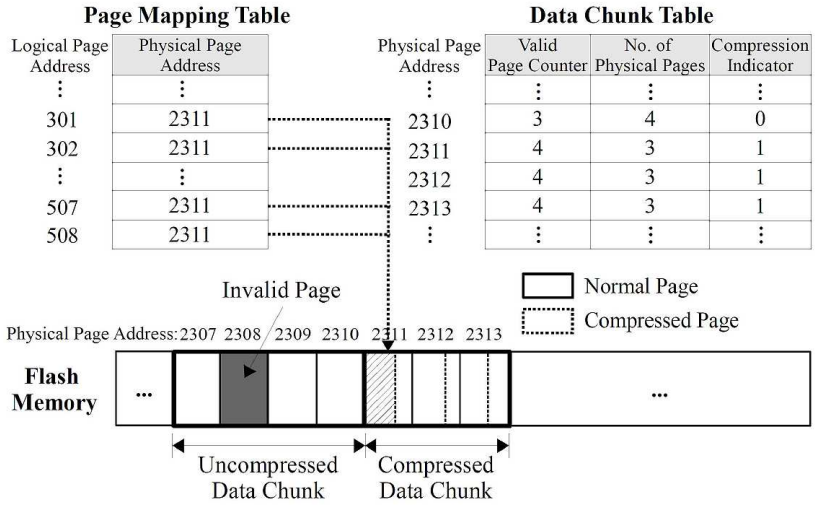


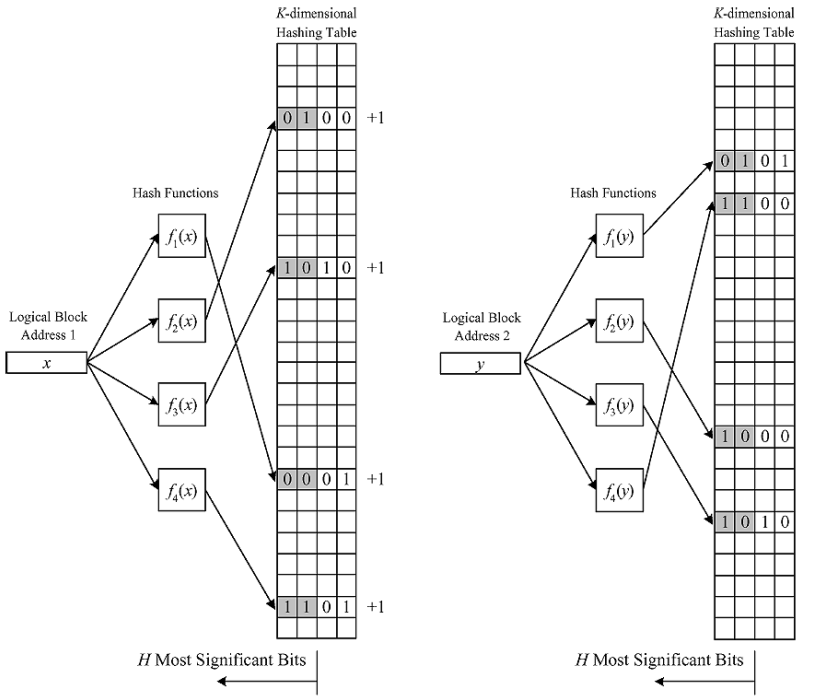
Figure 4: An overall organization of CaFTL.

In order to mitigate page fragmentation issues, CaFTL temporarily stores compressed data in a data buffer, and flushes four stored pages to NAND flash memory simultaneously. After flushing, compression-related information as shown in Figure 4 is updated to the data chunk table. Based on the data chunk table, CaFTL efficiently handles read requests and finds the most appropriate victim block during garbage collection. Moreover, CaFTL monitors the compression-ratio changes of input data so that unnecessary compression is avoided. Although data compression is an effective solution for reducing W_{day} in general, when poorly compressed data (e.g., multimedia data) are continuously incoming, the method's effectiveness in extending the SSD lifetime is significantly degraded. However, since our proposed techniques does not depend on data content, it can improve SSD lifetime

even when all the requested data is not compressed.

Data Separation Technique

Since NAND flash memory does not allow *in-place-updates*, unnecessary data copies occur during garbage collection so that the logical amount of written data is actually amplified by WAF. In order to minimize WAF, several flash optimization techniques (e.g., advanced mapping schemes, TRIM command and data separation techniques) have been introduced. For example, Hsieh *et al.* suggested a multi-hash function based data separation technique for separating hot data (i.e., frequently updated data) and cold data (i.e., rarely updated data) with a reasonable hardware overhead [7]. Since hot data are updated within a short time, if such hot data are aggregated in the same NAND block, there is a high probability that a dead block (i.e., a NAND block where all the pages are invalidated) or a near-dead block can be selected during garbage collection, thus reducing WAF. Figure 5 shows an example of the hot data identification process with K independent hash functions to hash a given LBA into the multiple entries of an M-entry hash table [7]. Whenever write requests are issed, each counter entry corresponding to a hashed value is incremented. In order to capture *recent hot data*, all the counter entries are decayed every predefined number of input requests. If the H most significant bits of every counter corresponding to K hash functions contain a non-zero value, that LBA is classified as hot data. Although the main purpose of the data separation technique is quite different from our proposed technique, its ability to identify hot data can contribute to increasing the efficiency of the proposed lifetime improvement techniques. For ex-



(a) the counter updating of an LBA (b) the hot data identification of an LBA

Figure 5: Examples of the counter updating and the hot data identification of an LBA.

ample, if the data separator can accurately identify hot data, the retention-time requirements of such hot data can be relaxed because hot data will be updated in the near-future.

2.3.2 Device-Level Endurance-Enhancing Technique

Wu *et al.* presented a device-level endurance enhancement technique that boosts self-recovery speed by heating a flash chip under high temperature [11]. Figure 6 shows the self-healing SSD architecture and its self-healing process. When a sick chip (i.e., a NAND chip that is almost worn-

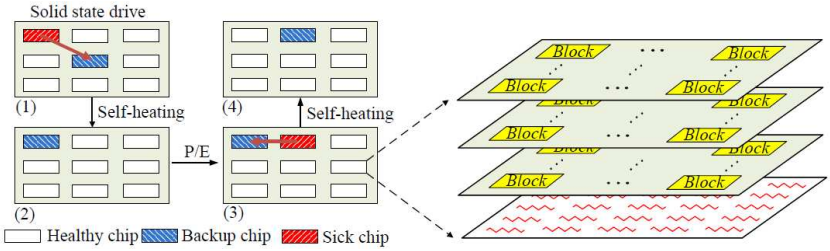


Figure 6: Illustration of self-healing SSD and an example of self-healing process.

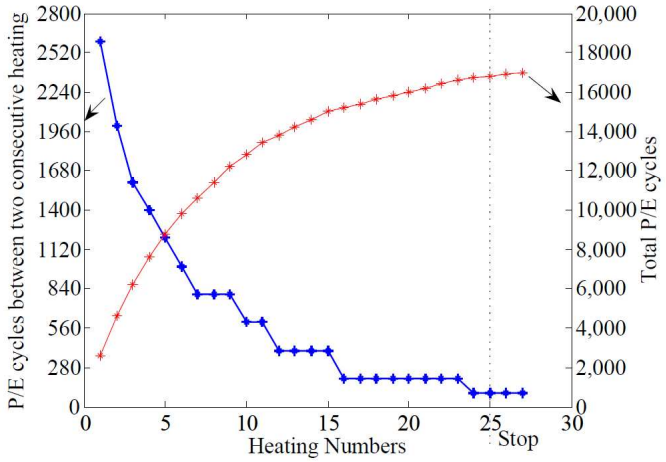
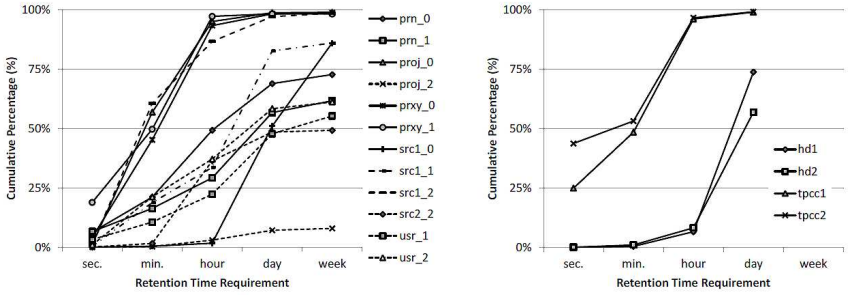


Figure 7: The effect of the self-heating on increasing $N_{P/E}^{max}$.

out) is detected, its entire data is copied to the extra backup chip during device idle times. After data copy operations are completed, a sick chip is heated at 200 °C for 35 minutes. Figure 7 shows the effect of self-heating on increasing $N_{P/E}^{max}$. By leveraging the temperature-accelerated recovery, it improved the endurance of SSDs up to fivefold. A major drawback of this approach is that it requires extra energy consumption to heat flash chips and lowers the reliability of a storage device. Our proposed technique improves



(a) MSRC workloads

(b) MapReduce and TPC-C workloads

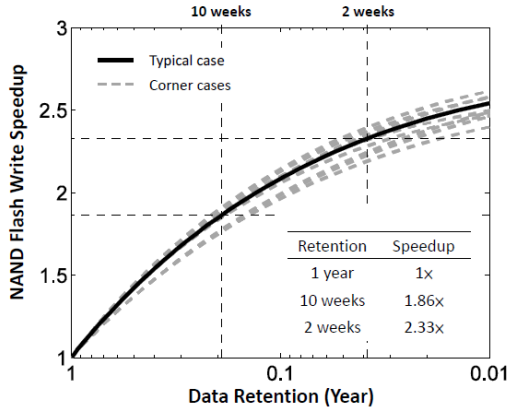
Figure 8: Example distributions of the data retention requirements.

the endurance of NAND devices by lowering the erase voltage and slowing down the erase speed without any serious side effects.

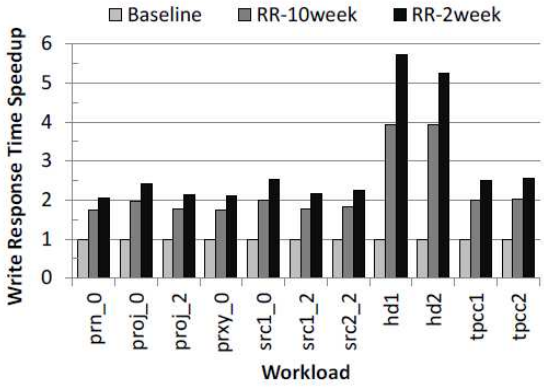
2.3.3 Cross-Layer Optimization Techniques Exploiting NAND Tradeoffs

Liu *et al.* proposed a retention relaxation technique to improve SSDs by relaxing their NAND retention capabilities [12]. This technique is motivated by their observation that in typical enterprise workloads, a considerable portion of written data to SSDs is likely to be updated soon (e.g., less than a day as shown in Figure 8). Since this observed updated time is much shorter than the NAND retention-time specification (i.e., 1 year), the retention relaxation technique increases the ISPP step voltage so that the NAND write performance is increased while shortening the retention capability.

Figure 9(a) shows how much the write speed increases as the retention-time requirements are relaxed. For example, if the retention-time requirement is relaxed to 2 weeks, the NAND write speed can be increased to 2.33x



(a) Relationship between the NAND retention times and the NAND write speedup.



(b) SSD write response time speedup.

Figure 9: Experimental results for the SSD write response time speedup.

of the write speed when 1-year retention-time is required. Figure 9(b) shows the overall write speedup of SSDs for 11 workloads under different retention capabilities. When the retention capability is relaxed to only 2 weeks, the overall SSD write response time was reduced by 160% on average.

The main weakness of this technique is that its effectiveness on improving SSD performance is entirely dependent on the workload conditions.

Since this technique always relaxes the NAND retention capability without consideration of data characteristics (e.g., the update frequency), when most of the written data are not updated within a predefined retention time, this data should be rewritten by a background data refresh process. When there is enough idle times between consecutive write requests, the side-effect of such background data refresh operations can be hidden as shown in the hd1 and hd2 cases of Figure 9(b). However, when the idle time is not sufficient, the write performance speedup may decrease as shown in the prn_0 case of Figure 9(b).

Another technical issue is that this technique did not take into account *retention-failure problems*. When a power failure occurs and continues for a long time, retention-relaxed data may not be retrieved because a background data refresh process does not work during power failures. In order for this kind of aggressive flash optimization techniques to be widely employed, the retention-failure problem should be adequately resolved.

Although the main goal of this technique is quite different from ours, its technical concept is one of the important motivations in the way that it actively exploited the tradeoff relationships between the NAND capabilities. For example, the concept of a retention relaxation substantially contributed to the development of our write-age tuning mode.

Chapter 3

Dynamic Erase Voltage and Time Scaling

In this chapter, we propose a unified framework, called **D**ynamic **E**rase **V**oltage and **T**ime **S**caling (**DeVTS**), which enables a system software to exploit the tradeoff relationship between the NAND endurance and erase voltages/times. The DeVTS framework is motivated by our NAND device physics study that NAND endurance is degraded primarily during erase operations. Since the probability of oxide damage (which is known as the main cause of endurance degradation) has an exponential dependence on the stress voltage [19], reducing the stress voltage (i.e., the erase voltage) is the most effective means of improving NAND endurance. Moreover, given an erase operation, since a nominal erase voltage tends to excessively damage NAND memory cells in the beginning of an erase operation [20], slowing down the erase speed (i.e., monotonically increasing the erase voltage from a low voltage to the nominal voltage over a sufficiently long time period) can minimize the damage [15][21], thus additionally improving NAND endurance. By modifying a NAND device to support multiple erase voltage and time scaling modes (which have different impacts on NAND endurance), and allowing a flash software to select the most appropriate erase scaling modes depending on a workload, DeVTS has a significant potential to in-

crease $N_{P/E}^{max}$.

However, in order to write data to a NAND block erased with a lower erase voltage, it is required to use special write modes that can form threshold voltage (*Vth*) distributions within a narrower *Vth* window. Since the *Vth* window (i.e., the total width of *Vth* margins for a NAND cell) is tightly designed to guarantee all the specified NAND requirements (i.e., endurance, performance and retention), in order to assign more *Vth* margin to the endurance, the *Vth* margin for the other requirements needs to be reduced instead. For example, a slow write mode with a fine-grained program control can shorten the width of a *Vth* distribution so that the required *Vth* margin for performance can be saved while the NAND program time increases [15]. Similarly, a short-retention write mode, which reduces the *Vth* gap between two adjacent *Vth* states, can save the required *Vth* margin for retention while the retention capability is sacrificed [16] [12].

In order to estimate the impact of the special write modes (i.e., slow write modes or short-retention write modes) on NAND endurance, we develop a unified NAND endurance model which accurately captures the trade-off relationships between NAND endurance and NAND performance/retention capabilities. Based on the NAND endurance model, when a slow or short-retention write mode is used at the expense of the performance or retention capability, we can estimate how much the erase voltage can be lowered and its impact on NAND endurance.

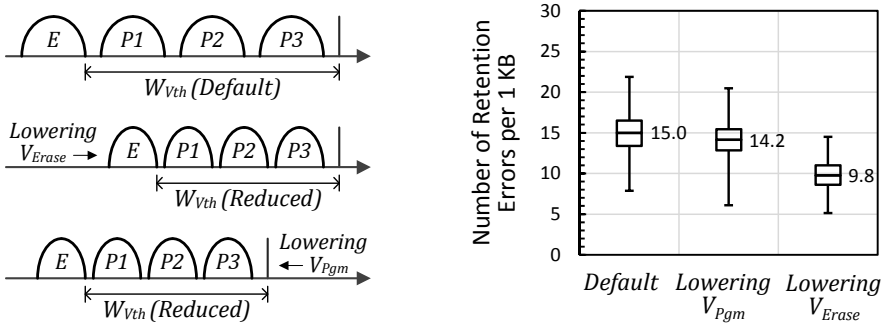
3.1 Erase Voltage and Time Scaling

3.1.1 Motivation

The physical mechanism of endurance degradation is closely related to stress-induced damage in the tunnel oxide layer of a NAND memory cell [16]. Since the probability of oxide damage has an exponential dependence on the stress voltage [19], lowering the stress voltage (i.e., the program voltage V_{Pgm} or the erase voltage V_{Erase}) during P/E cycles can be an effective means of improving NAND endurance.

Although the maximum V_{Pgm} to complete a program operation is usually higher than V_{Erase}, NAND endurance is primarily degraded during erase operations. This is because the stress time interval of an erase operation is about 100 times longer than that of a program operation. Furthermore, since written data on a certain cell is likely to be changed randomly, the probability that the cell consecutively experiences the maximum V_{Pgm} during P/E cycles is very low. On the contrary, all the cells in a NAND block experience V_{Erase} at all times during P/E cycles. Therefore, we can assume that changing V_{Erase} has a more significant impact on NAND endurance.

In order to verify our assumption, we evaluated the effects of two different stress-voltage-reduction policies, shown in Figure 10(a), on NAND endurance. In the 'lowering V_{Erase}' policy, W_{Vth} shrinks to the right direction (compared to the default case) so that V_{Erase} is lowered by 1 V while V_{Pgm} is not changed. On the other hand, in the 'lowering V_{Pgm}' policy, W_{Vth} shrinks to the left direction so that the maximum V_{Pgm} is reduced by 1 V while V_{Erase} is maintained. In our evaluation, ten blocks out of two



(a) Illustrations of two different stress-voltage-reduction policies over the default case.

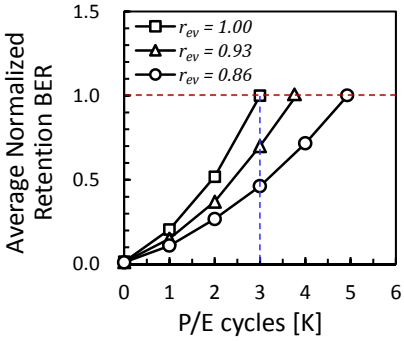
(b) Variations of the numbers of retention errors per 1 KB under three different policies.

Figure 10: Comparison of the impacts of lowering V_{Pgm} and V_{Erase} on NAND retention errors.

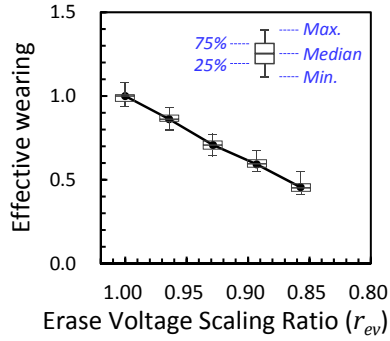
20-nm node NAND chips were selected for each policy. As the main evaluation metric, we measured the number of retention errors (i.e., bit errors after 3K pre-cycling and 1 hour's baking at 100 °C [22]) per 1-KB cells because it reflects the effective degree of NAND wearing [15]. As shown in Figure 10(b), when the 'lowering V_{Pgm}' policy was used, the number of retention errors was reduced by only 5.3%, on average, over the default case. However, when the 'lowering V_{Erase}' policy was used, the number of retention errors was reduced by 34.7%, on average, over the default case. These results clearly show that lowering V_{Erase} is much more effective than lowering V_{Pgm} in improving NAND endurance.

3.1.2 Erase Voltage Scaling

In order to evaluate the effect of erase voltage scaling on NAND endurance, we performed NAND cycling tests by using different V_{Erase}'s. In a



(a) Average variations of normalized retention BER over varying P/E cycles under different r_{ev}'s.



(b) Variations of effective wearing over varying r_{ev}'s.

Figure 11: The effect of erase voltage scaling on NAND endurance.

cycling test, program and erase operations are repeated 3,000 times. Our cycling tests for each case were performed with 100 blocks out of five 20-nm node NAND chips. After cycling tests, we measured the NAND retention BER (i.e., the number of retention errors divided by the total number of cells) for each block as a measure of wearing degree of NAND memory cells. The measured BERs were normalized over the retention BER when the nominal erase voltage $V_{Erase}^{nominal}$ was used. Figure 11(a) shows how the retention BER changes, on average, as the number of P/E cycles increases while different V_{Erase}'s are used. We represent different V_{Erase}'s using an erase voltage scaling ratio r_{ev} $(0 \leq r_{ev} \leq 1)$. When r_{ev} is set to x, V_{Erase} is reduced by $(1-x) \times V_{Erase}^{nominal}$. As shown in Figure 11(a), the more V_{Erase} is reduced (i.e., the lower r_{ev}'s), the lower the retention BERs. For example, when r_{ev} is set to 0.93, the normalized retention BER is reduced by 30% after 3K P/E cycles over the $V_{Erase}^{nominal}$ case.

Since different V_{Erase}'s affect NAND endurance by different amounts,

we introduce a new endurance metric, called *effective wearing*, which represents the effective degree of NAND wearing per a P/E cycle. Based on a linear approximation model[1] which simplifies the NAND wear-out behavior over P/E cycles as shown in Figure 11(a), we represent effective wearing with a normalized retention BER after 3K P/E cycles. For example, when $V_{Erase}^{nominal}$ is used (i.e., $r_{ev} = 1.00$), effective wearing is 1.00. On the other hand, when V_{Erase} is reduced by 7% (i.e., $r_{ev} = 0.93$), effective wearing becomes 0.70. As shown in Figure 11(b), since effective wearing has a near-linear dependence on r_{ev}, effective wearing for a different r_{ev} can be estimated by a linear regression model. In this dissertation, we will use a NAND endurance model with five erase voltage modes $\texttt{EVmode}_i$'s which have five different r_{ev}'s.

The effect of lowering V_{Erase} on NAND endurance can be estimated by accumulating effective wearing for each P/E cycle. After 3K P/E cycles, for example, the total sum ΣEW of effective wearing with $V_{Erase}^{nominal}$ is 3,000 ($= 1.00 \times 3000$), but when r_{ev} is set to 0.93, ΣEW is only 2,100 ($= 0.70 \times 3000$). Since NAND reliability is maintained until ΣEW reaches 3,000, $N_{P/E}^{max}$ can be increased by 1,286 ($= (3000 - 2100)/0.70$) when V_{Erase} is reduced by 7% over $V_{Erase}^{nominal}$.

Since we did not have access to NAND chips from different manufacturers, we could not prove that our test results can be generalized. However,

[1] In this dissertation, we use a linear approximation model which simplifies the wear-out behavior over P/E cycles. Our current linear model can overestimate the effective wearing under low erase voltage scaling ratios while it can underestimate the effective wearing under high erase voltage scaling ratios. We verified that, by the combinations of over-/underestimations of the effective wearing in our model, the current linear model achieves a reasonable accuracy with an up to 10% overestimation [20] while supporting a simple software implementation.

since our tests are based on widely-known device physics which have been investigated by many device engineers and researchers, we are convinced that the consistency of our results would be maintained as long as NAND flash memories use the same physical mechanism (i.e., FN-tunneling) for program and erase operations. We believe that our results will also be effective for future NAND devices as long as their operations are based on the FN-tunneling mechanism. It is expected that current 2D NAND devices will gradually be replaced by 3D NAND devices, but the basis of 3D NAND is still the FN-tunneling mechanism.

3.1.3 Erase Time Scaling

Endurance degradation is directly proportional to V_{Erase} in an erase operation as described in Section 3.1.2. When V_{Erase} is applied to a NAND block, however, NAND memory cells are likely to be over-damaged by V_{Erase}. Since the actual voltage across the tunnel oxide layer is the sum of V_{Erase} and the *Vth* of a cell [20], an unintended higher (than V_{Erase}) voltage may cause additional damage (which is dependent on the cell's *Vth*) to the cell until all the programmed cells are sufficiently erased. For example, NAND memory cells which have higher *Vth*'s (e.g., the $P3$ state) are more damaged than those that have lower *Vth*'s (e.g., the E state).

In order to minimize oxide damage in the beginning of an erase operation, it is necessary to properly control the applied V_{Erase} so that the actual voltage across the tunnel oxide layer does not exceed V_{Erase} throughout the erase operation. We implemented this idea by modifying the existing incremental step pulse erasing (ISPE) scheme [23] so that the applied V_{Erase}

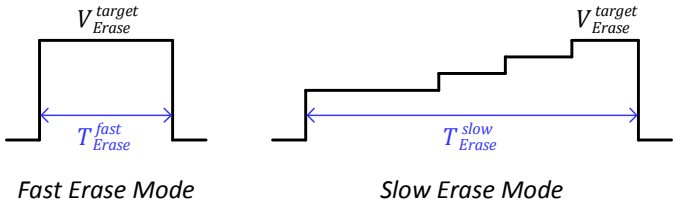
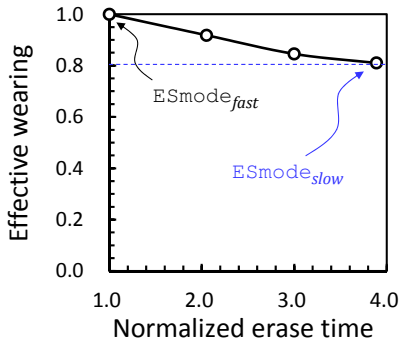
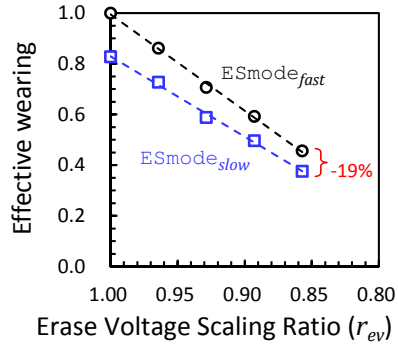


Figure 12: An illustration of an erase voltage control for the proposed erase time scaling.



(a) Variations of effective wearing over varying erase times.



(b) Variations of effective wearing over varying r_{ev}'s under two different erase time settings.

Figure 13: The effect of erase time scaling on NAND endurance.

gradually increases from a low voltage (e.g., V_{Erase} − the average *Vth* of the $P3$ state) to V_{Erase} over a sufficiently long time period as shown in Figure 12. However, when the modified ISPE scheme is used for an erase operation, the erase time (e.g., T_{Erase}^{slow} shown in Figure 12) inevitably increases because more ISPE loops are needed to complete the erase operation.

As shown in Figure 13(a), effective wearing decreases near-linearly as the erase time increases. For example, when the erase time increases three-fold, effective wearing is reduced, on average, by 19%. We represent the erase speed mode with a default erase time by $\texttt{ESmode}_{fast}$ while that with

a long erase time is represented by $\texttt{ESmode}_{slow}$. As shown in Figure 13(b), the effect of $\texttt{ESmode}_{slow}$ on improving NAND endurance can be exploited whenever longer erase times are acceptable regardless of r_{ev}.

3.2 Write Capability Tuning

If a NAND block is *shallowly erased* (i.e., erased with a lower voltage), the available *Vth* window for a program operation is also reduced. This is because W_{Vth} is mainly affected by V_{Verify}^{Erase} (which determines the requirement of V_{Erase}) as explained in Section 2.1. For example, as shown in Figure 14, if a NAND block is shallowly erased with a low erase voltage V_{Erase}^{low} (which is lower than $V_{Erase}^{nominal}$), W_{Vth} is reduced by a saved *Vth* margin ΔW_{Vth} (which is proportional to the voltage difference between $V_{Erase}^{nominal}$ and V_{Erase}^{low}). Since *Vth* distributions should be formed within the given *Vth* window, when V_{Erase}^{low} is used in an erase operation, it is necessary to use special write modes which adjust *Vth* design parameters (e.g., W_{Pi}, M_{Pi} and M_{Dist}) so that W_{Vth} is reduced by at least ΔW_{Vth}. In this sec-

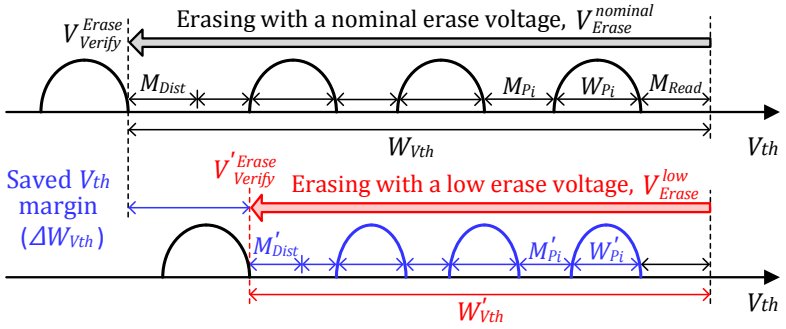
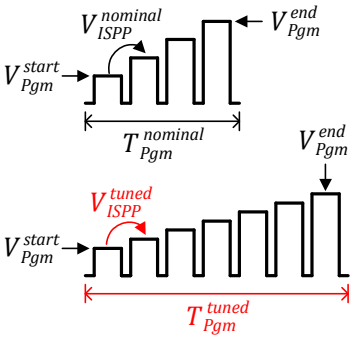


Figure 14: An example of NAND capability tuning for writing data to a shallowly erased NAND block.

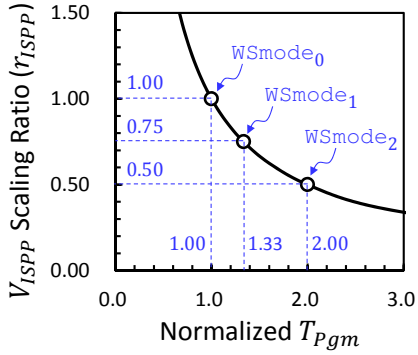
tion, we describe several write capability tuning techniques to save W_{Vth}, and present the NAND endurance model to estimate the impact of the proposed tuning techniques on NAND endurance.

3.2.1 Write Performance Tuning

In order to reduce W_{Pi}'s, a fine-grained ISPP step control is needed because W_{Pi} is directly proportional to the ISPP step voltage V_{ISPP} [18]. However, since the number of ISPP loops to complete a program operation is inversely proportional to V_{ISPP} [15], the program time T_{Pgm} inevitably increases as shown in Figure 15(a) if narrow *Vth* distributions are required. Figure 15(b) shows how much V_{ISPP} can be reduced as T_{Pgm} increases. T_{Pgm} was normalized over the nominal program time $T_{Pgm}^{nominal}$ (e.g., 1,300 μs [3]). We denote V_{ISPP} scaling ratio over the nominal ISPP step voltage $V_{ISPP}^{nominal}$ by r_{ISPP} $(0 \leq r_{ISPP} \leq 1)$. When r_{ISPP} is set to x, V_{ISPP} is reduced by $(1-x) \times V_{ISPP}^{nominal}$.



(a) An illustration of the write performance tuning technique.



(b) The relationship between the normalized T_{Pgm} and r_{ISPP}.

Figure 15: The proposed write performance tuning.

In our proposed write-performance tuning technique, we define three different write-speed modes, $\texttt{WSmode}_0$, $\texttt{WSmode}_1$, and $\texttt{WSmode}_2$, as shown in Figure 15(b). $\texttt{WSmode}_0$ is the fastest write mode which has the same T_{Pgm} as that of the nominal write mode, but cannot reduce V_{ISPP}. Alternatively, $\texttt{WSmode}_2$, the slowest write mode, has a T_{Pgm} two times longer (i.e., the normalized T_{Pgm} is 2.0) than the nominal write mode, but can reduce V_{ISPP} by 50% (i.e., r_{ISPP} is 0.50) over $V_{ISPP}^{nominal}$.

Since W_{Pi} has a linear dependence on V_{ISPP} (which is determined by the write-performance requirement as shown in Figure 15(b)), ΔW_{Vth} by tuning T_{Pgm} is expressed as follows (for an MLC NAND device):

$$\Delta W_{Vth} = \sum_{i=1}^{3} \Delta W_{Pi} = \sum_{i=1}^{3} (1 - r_{ISPP}) \times V_{ISPP}^{nominal}. \qquad (3.1)$$

For example, if $V_{ISPP}^{nominal}$ is 400 mV, and a longer T_{Pgm} two times as long as $T_{Pgm}^{nominal}$ is acceptable, W_{Vth} can be reduced by 600 mV ($= 3 \times ((1 - 0.50) \times 400$ mV)).

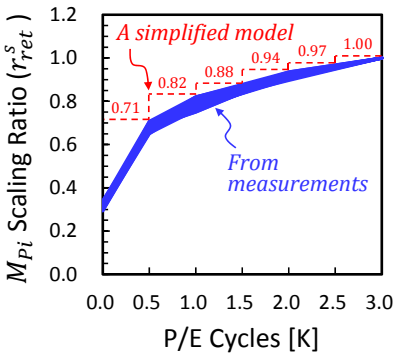
3.2.2 Retention Capability Tuning

NAND flash memory is required to retain its stored data for the specified retention time (e.g., 1 year at 30 °C [14]). In order to guarantee the NAND retention requirement throughout the storage lifespan, M_{Pi}'s are usually *fixed* during device design times to cover the maximum *Vth* change under the worst-case operating condition (i.e., the maximum number of P/E cycles and the specified retention time). However, since such worst-case operating conditions rarely occur, M_{Pi}'s are not fully needed in most common

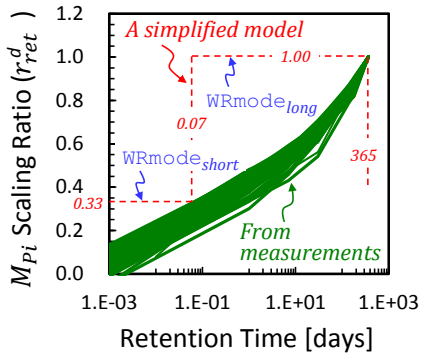
cases. For example, since the *Vth* change due to the charge-loss phenomenon is proportional to the number of P/E cycles [16], only part of M_{Pi} is enough for young NAND memory cells (that have experienced fewer P/E cycles) to meet the retention-time requirement. Moreover, since the *Vth* change is also proportional to a retention time [16], when written data are updated frequently within a short time period, M_{Pi} for such data can be further reduced.

Static Retention Tuning

In order to determine how much M_{Pi} is required as the number of P/E cycles increases, we performed NAND cycling tests over varying P/E cycles. A cycling test for each case was performed with more than 2,000 NAND pages (from 20 blocks out of 2 NAND chips). After the cycling tests, we measured the average change in *Vth* for each block after 1 hour's baking at $100\,^{\circ}\mathrm{C}$. Measured average *Vth* change was normalized over the maximum required *Vth* margin M_{Pi}^{max} under the worst-case operating condition (i.e., 3K P/E cycles and 1-year retention time). We represent the normalized average *Vth* change over varying P/E cycles as the static M_{Pi} scaling ratio r_{ret}^{s}. Figure 16(a) shows r_{ret}^{s} variations over varying P/E cycles. After 0.5K P/E cycles, for example, only 71% of M_{Pi}^{max} is required (i.e., r_{ret}^{s} is 0.71). Based on the measurement results, we constructed a simplified static M_{Pi} scaling model where r_{ret}^{s} changes every 0.5K P/E cycles as shown by the dotted line in Figure 16(a). For a given number of P/E cycles, ΔW_{Vth} by tuning the NAND retention capability is expressed as follows (for an MLC



(a) r^s_{ret} variations over varying P/E cycles.



(b) r^d_{ret} variations over varying retention times.

Figure 16: The simplified M_{Pi} scaling models for retention capability tuning.

NAND device):

$$\Delta W_{Vth} = \sum_{i=1}^{3} \Delta M_{Pi} = \sum_{i=1}^{3} (1 - r^s_{ret}) \times M^{max}_{Pi}. \tag{3.2}$$

For example, if the sum of three M^{max}_{Pi}'s is 900 mV, and the number of P/E cycles is less than 0.5K, W_{Vth} is reduced by 261 mV ($= (1 - 0.71) \times 900$ mV).

Dynamic Retention Tuning

In order to determine how much M_{Pi} is required as the retention time increases, we performed NAND cycling tests over varying retention times and measured the average change in *Vth* for each retention time interval. Measured average *Vth* change was normalized over M^{max}_{Pi}. We represent the normalized average *Vth* change over varying retention times as the dynamic M_{Pi} scaling ratio r^d_{ret}. The solid lines in Figure 16(b) show r^d_{ret} vari-

ations over varying retention times with more than 2,000 NAND pages. In order to minimize the management overhead, we simplify the r^d_{ret} changes over varying retention times into two different write-retention modes (i.e., $\texttt{WRmode}_{short}$ and $\texttt{WRmode}_{long}$) as shown by the dotted line in Figure 16(b). $\texttt{WRmode}_{long}$ is the long-retention write mode which fully supports the specified retention time (i.e., 1 year), but cannot reduce M_{Pi} (i.e., r^d_{ret} is 1.00). Alternatively, $\texttt{WRmode}_{short}$ is the short-retention write mode which supports only a 0.07-day retention time while requiring only 33% of M^{max}_{Pi} (i.e., r^d_{ret} is 0.33). By combining r^s_{ret} with r^d_{ret}, Equation 3.2 is re-expressed as follows:

$$\Delta W_{Vth} = \sum_{i=1}^{3} \Delta M_{Pi} = \sum_{i=1}^{3} (1 - r^s_{ret} \times r^d_{ret}) \times M^{max}_{Pi}. \tag{3.3}$$

For example, when P/E cycle count is less than 0.5K, and the retention requirement is less than 0.07 days, W_{Vth} is reduced by 689 mV ($= (1 - 0.71 \times 0.33) \times 900$ mV).

3.2.3 Disturbance Resistance Tuning

Since the program disturbance and the read disturbance of NAND flash memory are proportional to the number of P/E cycles [16], we measured how much M_{Dist} is required as the number of P/E cycles increases. After performing NAND cycling tests with varying P/E cycles and a specified number (i.e., 400K [6]) of read cycles, we measured the average change in *Vth* in the E state. Our tests were performed with more than 2,000 NAND pages. Measured average *Vth* change was normalized over the maximum

Table 1: A simplified r_{dist} model over varying P/E cycles.

P/E Cycles [K]	0.5	1.0	1.5	2.0	2.5	3.0
r_{dist}	0.43	0.57	0.74	0.90	0.95	1.00

required *Vth* margin M_{Dist}^{max} under the worst-case operating condition (i.e., 3K P/E cycles and 400K read cycles). We represent the normalized average *Vth* change caused by NAND disturbance as r_{dist} $(0 \leq r_{dist} \leq 1)$. Table 1 summarizes our simplified r_{dist} model over varying P/E cycles. For example, after 0.5K P/E cycles, only 43% of M_{Dist}^{max} is required (i.e., r_{dist} is 0.43). For a given number of P/E cycles, ΔW_{Vth} by tuning the NAND disturbance resistance can be expressed as follows:

$$\Delta W_{Vth} = \Delta M_{Dist} = (1 - r_{dist}) \times M_{Dist}^{max}. \tag{3.4}$$

Given that M_{Dist}^{max} is 400 mV, and the number of P/E cycles is less than 0.5K, W_{Vth} is reduced by 228 mV $(= (1 - 0.43) \times 400$ mV$)$.

3.3 NAND Endurance Model

Combining the proposed NAND capability tuning (i.e., write-performance tuning, retention-capability tuning, and disturbance-resistance tuning) with erase voltage/time scaling, we developed a novel NAND endurance model which can be used with DeVTS-enabled NAND chips. In order to construct the NAND endurance model, we calculate ΔW_{Vth} for each combination of NAND capability tuning modes by using Equations 3.1, 3.3, and 3.4. Since a reduced erase voltage $(= (1 - r_{ev}) \times V_{Erase}^{nominal})$ is proportional to ΔW_{Vth},

Table 2: An example of a parameter set used to estimate effective wearing.

Parameter	$V_{Erase}^{nominal}$	M_{Dist}^{max}	$V_{ISPP}^{nominal}$	$\sum M_{Pi}^{max}$	α_c
Value	14 V	400 mV	400 mV	900 mV	0.6

r_{ev} can be re-expressed as follows:

$$r_{ev} = 1 - \frac{\Delta W_{Vth}}{V_{Erase}^{nominal} \times \alpha_c}, \tag{3.5}$$

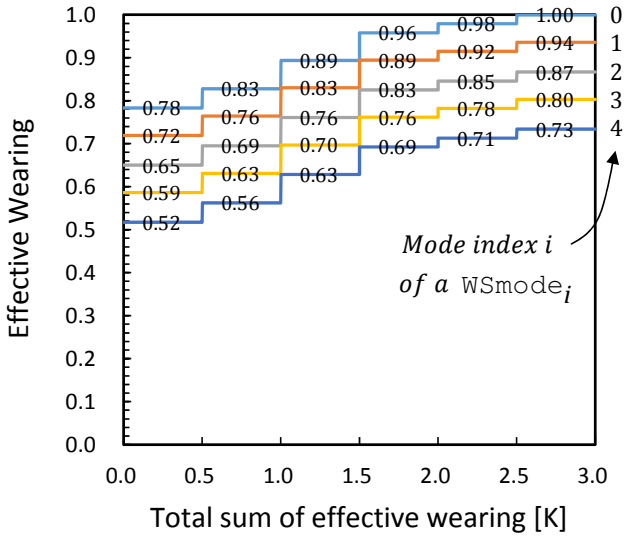
where α_c is the empirical scaling parameter which represents the impact of the V_{Erase} change on the *Vth* window. For example, if α_c is 0.60 and V_{Erase} is reduced by 1.00 V, W_{Vth} can be effectively reduced by 0.60 V. When r_{ev} is calculated from Equation 3.5 for a given ΔW_{Vth}, the corresponding effective wearing can be estimated by the linear equation described in Section 3.1.2. Table 2 summarizes the parameter set used to construct the NAND endurance model in this dissertation. All the data in our model is based on measurement results with 20-nm node NAND chips.

As summarized in Table 3, $\texttt{EVmode}_j$'s are decided by the combinations of two write-retention modes (i.e., $\texttt{WRmode}_{long}$ and $\texttt{WRmode}_{short}$) and five write-speed modes (i.e., $\texttt{WSmode}_0 \sim \texttt{WSmode}_4$) because r_{ev}'s are different for each combination. Figures 17 and 18 show our DeVTS-enabled NAND endurance (i.e., the effective wearing) model with two erase speed modes (i.e., $\texttt{ESmode}_{fast}$ and $\texttt{ESmode}_{slow}$) and two write-retention modes (i.e., $\texttt{WRmode}_{long}$ and $\texttt{WRmode}_{short}$). Since ΔW_{Vth}'s are also affected by static retention-capability tuning and disturbance-resistance tuning, the values of effective wearing vary whenever ΣEW exceeds 0.5K. If the total

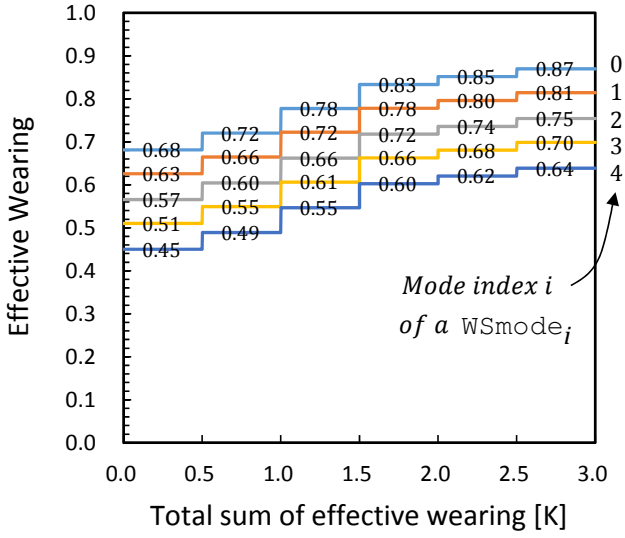
EVmode	0	1	2	3	4	5	6	7	8	9
WSmode	0	1	2	3	4	0	1	2	3	4
WRmode	*long*					*short*				

Table 3: The $\texttt{EVmode}_j$ decision rule.

sum of the effective wearing is less than 0.5K, for example, when a NAND block is slowly erased before writing with the short-retention write mode (i.e., $\texttt{WRmode}_{short}$) and the slowest write-speed mode (i.e., $\texttt{WSmode}_4$), the lowest erase voltage (i.e., $\texttt{EVmode}_9$) can be used for an erase operation. In this case, the effective wearing is only 0.29. The NAND endurance model not only presents effective wearing for each combination of $\texttt{EVmode}_j$ and $\texttt{ESmode}_k$ used in an erase operation, but also specifies corresponding write capability tuning modes (i.e., $\texttt{WSmode}_i$ and $\texttt{WRmode}_m$) when writing data to a NAND block erased with $\texttt{EVmode}_j$.

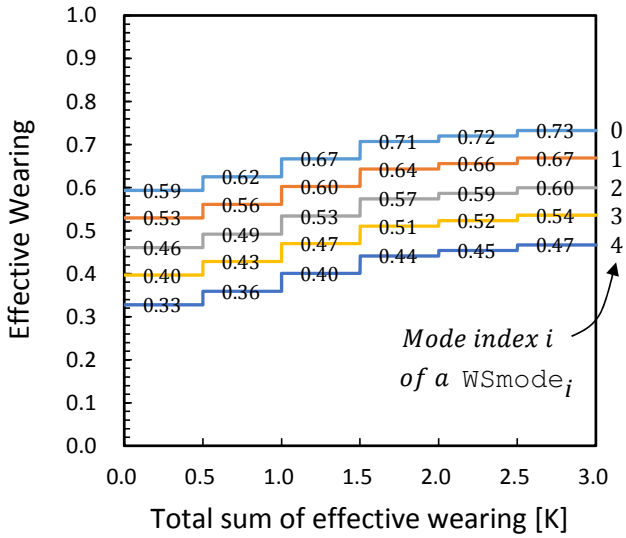


(a) The endurance model for $\texttt{ESmode}_{fast}$ under five write speed modes.

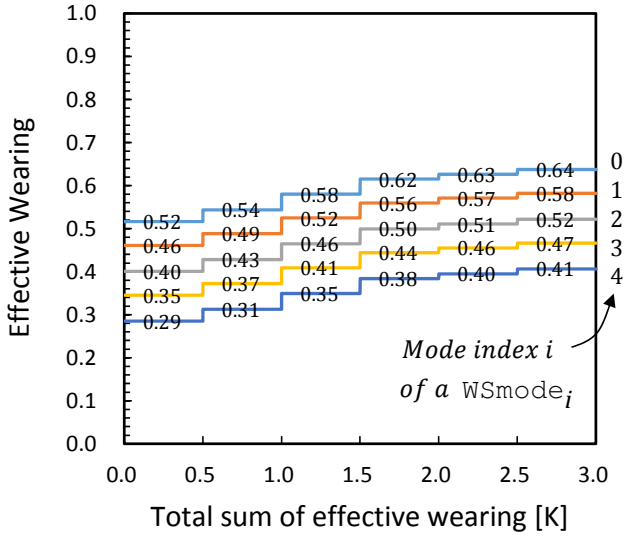


(b) The endurance model for $\texttt{ESmode}_{slow}$ under five write speed modes.

Figure 17: The proposed NAND endurance models for DeVTS-enabled NAND chips when long-retention write mode $\texttt{WRmode}_{long}$ is used.



(a) The endurance model for $\texttt{ESmode}_{fast}$ under five write speed modes.



(b) The endurance model for $\texttt{ESmode}_{slow}$ under five write speed modes.

Figure 18: The proposed NAND endurance models for DeVTS-enabled NAND chips when short-retention write mode $\texttt{WRmode}_{short}$ is used.

Chapter 4

Lifetime Improvement Technique Using Write-Performance Tuning

In this chapter, we propose an SSD lifetime improvement technique, called Dynamic Erase Voltage and Time scaling with Write Performance Tuning (DeVTS-wPT), using the write-performance tuning technique based on the DeVTS framework. Our DeVTS-wPT technique exploits the trade-off relationships between the NAND endurance and erase voltages/speeds at the firmware-level (or the software level in general) so that NAND endurance is improved while the overall write throughput is not affected. For example, since the maximum performance of NAND flash memory is not always needed in real workloads, a DeVTS-wPT based technique can exploit idle times between consecutive write requests for shortening the width of threshold voltage distributions so that shallowly erased NAND blocks, which were erased by lower erase voltages, can be used for most write requests. Idle times can be also used for slowing down the erase speed. If such idle times can be automatically estimated by a firmware/system software, the DeVTS-wPT based technique can choose the most appropriate write speed for each write request or select the most suitable erase voltage/speed for each erase operation. By aggressively selecting endurance-enhancing erase modes (i.e., a slow erase with a lower erase voltage) when a large idle

time is available, NAND endurance can be significantly improved because less damaging erase operations are more frequently used.

We have implemented the first DeVTS-wPT aware FTL, called *dvsFTL*, which dynamically adjusts write and erase modes in an automatic fashion, thus improving NAND endurance with a negligible degradation in the overall write throughput. In dvsFTL, we also revised key FTL software modules (such as garbage collector and wear-leveler) to make them DeVTS-wPT aware for maximizing the effect of DeVTS-wPT on NAND endurance. Since no NAND chip currently allows an FTL firmware to change its program and erase voltages/times dynamically, we evaluated the effectiveness of dvsFTL with the *extFlashBench* emulation environment [24] using a DeVTS-wPT-enabled NAND simulation model (which supports multiple write and erase modes). Our experimental results using various I/O traces show that dvsFTL can improve $N_{P/E}^{max}$ by 61.2% over an existing DeVTS-wPT-unaware FTL with less than 2.2% decrease in the overall write throughput.

4.1 Design and Implementation of dvsFTL

4.1.1 Overview

Based on our NAND endurance model presented in Section 3.3, we have implemented dvsFTL, the first DeVTS-wPT-aware FTL, which automatically changes write and erase modes depending on write throughput requirements. dvsFTL is based on a page-level mapping FTL with additional modules for DeVTS-wPT support. Figure 19 shows an organizational

overview of dvsFTL. The DVS manager, which is the core module of dvsFTL, selects a write-speed mode $\texttt{WSmode}_i$ for a write request and decides both an appropriate erase voltage mode $\texttt{EVmode}_j$ and erase speed mode $\texttt{ESmode}_k$ for each erase operation. In determining appropriate modes, the mode selector bases its decisions on the estimated write throughput requirement using a circular buffer. dvsFTL maintains per-block mode information and NAND setting information as well as logical-to-physical mapping information in the extended mapping table. The per-block mode table keeps track of the current write mode and the total sum of the effective wearing for each block. The NAND setting table is used to choose appropriate device settings for the selected write and erase modes, which are sent to NAND chips via a new interface *DeviceSettings* between dvsFTL and NAND chips. dvsFTL also extends both the garbage collector and wear leveler to be DeVTS-wPT-aware.

4.1.2 Write-Speed Mode Selection

In order to select the most appropriate write-speed mode (i.e., the slowest write mode among available write-speed modes which does not affect the overall write performance), *the Wmode selector* in the DVS manager estimates the write-performance requirement for a given write request based on the utilization u^{wb} of a write buffer. Since the write buffer queues incoming requests before they are written, u^{wb} changes depending on the difference between the incoming rate r^{in} of write requests from a host system and the outgoing rate r^{out} to NAND devices. When writes are requested in a sporadic fashion (i.e., $r^{in} < r^{out}$), u^{wb} may decrease. In this case, the Wmode

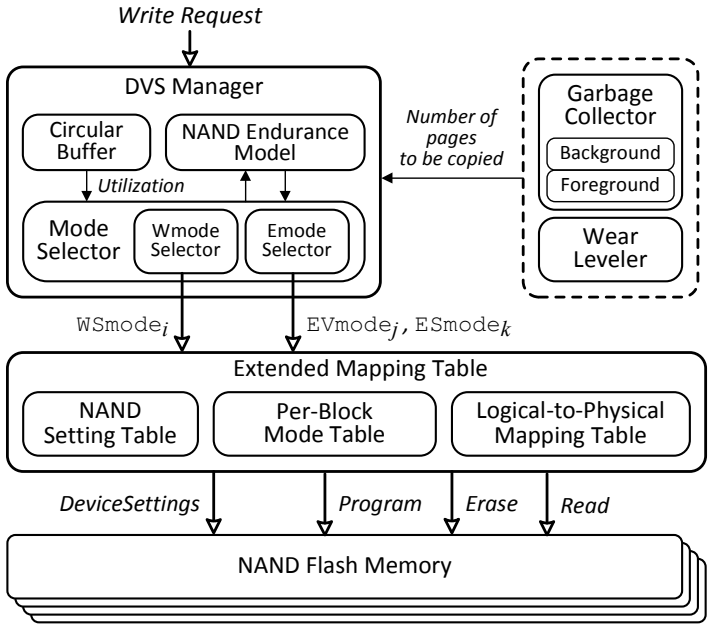


Figure 19: An organizational overview of dvsFTL.

selector estimates that the maximum write performance of NAND devices is not fully needed. On the contrary, when write requests are so intensive (i.e., $r^{in} > r^{out}$) that u^{wb} increases, it is estimated that queued requests should be written as fast as possible.

Figure 20 shows an overview of the write-speed mode selection in dvsFTL. In our implementation, the write-performance requirement is classified into five levels by four buffer utilization boundaries as shown in Figure 20. For example, when u^{wb} is lower than 0.20, the requests queued in the write buffer is written to a NAND page with $\texttt{WSmode}_4$, the slowest write mode. However, when u^{wb} is higher than 0.80, in order to satisfy the urgent requirement of write performance, the write-speed mode is changed to $\texttt{WSmode}_0$, the fastest write mode.

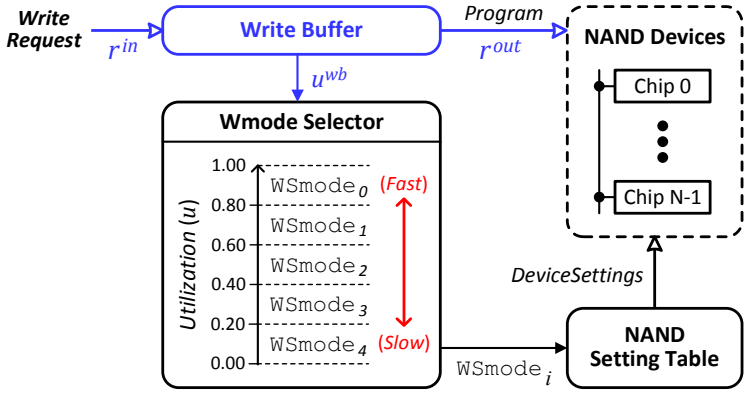


Figure 20: An overview of the write-speed mode selection in dvsFTL.

Our proposed write-speed mode selection technique can efficiently adapt to varying r^{in} as well as r^{out} (which is proportional to the number of available NAND chips that are ready to be written). When NAND chips are not available due to garbage collection, r^{out} is significantly reduced [25]. For example, when garbage collection operations are performed in half of NAND chips, r^{out} is reduced by 50%. If r^{out} reaches below r^{in} so that u^{wb} increases, a faster write mode is more suited to mitigate the side effect of garbage collection. Since our estimation metric is based on u^{wb} which depends on both r^{out} and r^{in}, the Wmode selector can determine the most proper write-speed mode by taking into account the variations in both r^{in} and r^{out} during run times.

4.1.3 Erase Voltage Mode Selection

On-Demand Selection

Selecting the most appropriate erase-voltage mode is the most essential step in dvsFTL because the erase voltage has a significant impact on NAND endurance as well as the overall write performance as described in Sections 3.1.2 and 3.2.1, respectively. When $\texttt{EVmode}_5$ (which uses the lowest erase voltage) is always used in erase operations, NAND endurance can be improved to the fullest extent. However, since a NAND block erased with $\texttt{EVmode}_4$ allows only $\texttt{WSmode}_4$ (which is the slowest write-speed mode) in a program operation, when intensive write operations are requested, the write performance can be degraded significantly. On the contrary, when $\texttt{EVmode}_0$ (which uses the highest erase voltage) is used at all times, DeVTS-wPT cannot reach its full potential while still maintaining the overall write-performance requirement. Therefore, similar to the write mode selections, estimating the requirements of future write requests is also a critical step in selecting the right erase-voltage mode.

When a foreground garbage collection process is invoked, since the write-speed mode and write-retention mode of a received write request have already been chosen by the Wmode selector, the victim block can be erased with the corresponding erase-voltage mode as defined in the NAND endurance model. For example, if ΣEW is less than 0.5K for a victim block, and $\texttt{WSmode}_0$ has been chosen, the Emode selector decides $\texttt{EVmode}_0$ as the appropriate erase-voltage mode.

However, when a background garbage collection process is invoked, it

is difficult to estimate the requirements of subsequent write requests. This is because background garbage collection is activated when write requests are not issued for more than the threshold time interval so that the recent history of write requests is nearly initialized. In our implementation, the Emode selector postpones deciding the right erase-voltage mode and selects $\texttt{EVmode}_4$ as the default so that a victim block is shallowly erased (with the lowest erase voltage) during the background garbage collection process. The right erase-voltage mode is *lazily* decided when the next phase of write requests (after the background garbage collection process) is written to that block. If the selected write modes are not compatible with $\texttt{EVmode}_4$, the selected block is additionally erased using the *lazy erase* operation (of which latency is about 1,000 μs), described in the next section. Although the write latency for the first page in the block is increased by 77% because the lazy erase operation is performed in advance of the first-page write, its negative impact on the overall write performance is less than 0.6% while the potential of DeVTS-wPT can be fully utilized in terms of the lifetime improvement.

Lazy Selection

As explained in Section 3.2.1, when a NAND block was erased with $\texttt{EVmode}_i$, a page in the shallowly erased block can be programmed using specific $\texttt{WSmode}_j$'s (where $j \geq i$) only because the requirement of the saved threshold voltage margin cannot be satisfied with a faster write-speed mode $\texttt{WSmode}_k$ $(k < i)$. In order to write data with a faster write-speed mode to the shallowly erased NAND block, the shallowly erased block should be erased further before it is written. We propose a lazy erase scheme

which additionally erases the shallowly erased NAND block, when necessary, with a small extra erase time (i.e., 20% of the nominal erase time). Since the effective wearing mainly depends on the maximum erase voltage used, erasing a NAND block by a high erase voltage in a lazy fashion does not incur any extra damage than erasing it with the initially high erase voltage. Although it takes a longer erase time, the total sum of the effective wearing by lazily erasing a shallowly erased block is less than that by erasing with the initially high erase voltage. This can be explained in a similar fashion as why the erase time scaling is effective in improving the NAND endurance as discussed in Section 3.1.3. The endurance gain from using two different starting erase voltages is higher than the endurance loss from a longer erase time.

4.1.4 Erase Speed Mode Selection

The Emode selector chooses a proper erase-speed mode which can offer an additional lifetime benefit without affecting the overall write performance. Since write requests waiting in the write buffer cannot be programmed to NAND chips during an erase operation, when writes are continuously requested, the buffer utilization will increase. The increase Δu^{erase} in the buffer utilization due to the erase operation can be estimated by how many write requests are fulfilled during that time interval. As a result, the effective buffer utilization u^* after the erase operation is expressed as the sum of the current buffer utilization u^{wb} and Δu^{erase}. In selecting an erase-speed mode, the Emode selector first checks whether or not erasing with $\texttt{ESmode}_{slow}$ raises u^* above 1.0. If it is estimated that u^* will be higher

than 1.0, in order to avoid buffer overflow, $\texttt{ESmode}_{fast}$ is selected. Otherwise, the Emode selector additionally checks whether or not erasing with $\texttt{ESmode}_{slow}$ causes a change in the current write-speed mode. If u^* is increased above the current buffer utilization boundary (e.g., 0.20, 0.40, 0.60, 0.80, or 1.00 as shown in Figure 20), subsequent write requests will be written with a faster write mode. In this case, since the endurance gain by using a slower erase mode is smaller than the endurance gain lost by using a faster write mode as shown in Figures 17 and 18, $\texttt{ESmode}_{slow}$ is not a suitable choice in terms of the lifetime improvement. If it is confirmed that $\texttt{ESmode}_{slow}$ will not affect the overall write performance and actually has a lifetime benefit, it is then selected for the erase operation.

4.1.5 DeVTS-wPT Aware FTL Modules

Extended Mapping Table

Since erase operations are performed at the NAND block level, the per-block mode table maintains five linked lists of blocks which were erased using the same erase voltage mode. When the DVS manager decides a write-speed mode for a write request, the corresponding linked list is consulted to locate a destination block for the write request. Also, the DVS manager informs a NAND chip how to configure appropriate device settings (e.g., ISPP/ISPE voltages, the erase voltage, and reference voltages for read/verify operations) for the current write-speed mode using the per-block mode table. Once NAND chips are set to a certain mode, an additional setting is not necessary as long as the write and the erase modes are maintained. For a

read request, since different write-speed modes require different reference voltages for read operations, the per-block mode table keeps track of the current write mode for each block so that a NAND chip changes its read references before serving a read request.

In order to retrieve the per-block mode table, maintained in volatile RAM, after an SSD is rebooted, dvsFTL writes the device-setting information for each block into the spare area of the first page of that block. Since the read reference voltages for a block is also unknown just after rebooting, dvsFTL first searches the right reference voltages among the predefined set of the read reference voltages corresponding to each erase mode. After appropriate voltages are found, the device-setting information for that block can be recovered by reading the spare area of its first page.

DeVTS-wPT Enabled Garbage Collection

When a garbage collection process is invoked, selecting the most suitable write-speed mode for data copy operations is also a challenging issue to maximize the efficiency of DeVTS-wPT. If valid data is copied with the fastest write mode at all times, the performance overhead of a garbage collection process can be minimized. However, since free pages in deeply erased blocks (which are compatible with the fastest write mode) are frequently used, the probability of erasing blocks with the highest erase voltage is increased inevitably. Conversely, if the slowest write mode is always used in data copy operations, the overall write performance may be significantly degraded. Since write requests waiting in the write buffer cannot be programmed to NAND chips during data copy operations, the buffer utilization

may be effectively increased by Δu^{copy} which is proportional to the number of valid pages to be copied. Consequently, the effective buffer utilization u^* after the data copy operation is expressed as the sum of the current buffer utilization u^{wb} and Δu^{copy}. Similar to the erase-speed mode selection, if it is estimated that u^* will be raised above 1.0, the Wmode selector selects the fastest write mode (i.e., $\texttt{WSmode}_0$). Otherwise, the Wmode selector selects the fastest write mode among available write-speed modes that does not change the current write-speed mode.

DeVTS-wPT Enabled Wear leveling

Since different erase voltage/time affects the NAND endurance differently as described in Section 3.1, the reliability metric (based on the number of P/E cycles) of the existing wear leveling algorithm [26] is no longer valid in a DeVTS-wPT-enabled NAND flash chip. In dvsFTL, the DeVTS-wPT-aware wear leveler uses the total sum of the effective wearing instead of the number of P/E cycles as a reliability metric, and tries to evenly distribute the total sum of the effective wearing among NAND blocks.

Device Setting Interfaces

As semiconductor technologies reach their physical limitations, it is necessary to use cross-layer optimization between system software and NAND devices. As a result, some of internal device interfaces are gradually opened to public in the form of additional 'user interface'. For example, in order to track bit errors caused by data retention, a new 'device setting interface'

which adjusts the internal reference voltages for read operations is recently opened to public [27][28]. There are already many set and get functions for modifying or monitoring NAND internal configurations in the up-to-date NAND specifications such as the toggle mode interface and ONFI. For the measurements presented here, we were fortunately able to work in conjunction with a flash manufacturer to adjust erase voltage as we wanted.

4.2 Experimental Results

4.2.1 Experimental Settings

We evaluated the effectiveness of the proposed dvsFTL with *extextFlash-Bench*, an extended version of an existing unified development environment for NAND flash-based storage systems [24]. In order to keep track of temporal interactions among various NAND operations, extFlashBench emulates the key operations of DeVTS-wPT-enabled NAND devices in a timing-accurate fashion using high-resolution timers (or hrtimers) (which are available in a recent Linux kernel [29]).. Our validation results on an 8-core Linux server system show that the extFlashBench is very accurate. For example, variations on the program time and erase time of our DRAM-based NAND emulation models are less than 0.8% of T_{PROG} and 0.3% of T_{ERS}, respectively.

For our evaluation, we modified a NAND flash model in extFlashBench to support DeVTS-wPT-enabled NAND flash chips with five write modes, five erase voltage modes, and two erase speed modes. Each NAND flash chip employed 128 blocks which were composed of 128 8-KB pages. The

Table 4: The latency variations of NAND functions used in the experiments.

NAND function	Speed mode	Latency [3]
Program	$\texttt{WSmode}_0$	1,300 μs
	$\texttt{WSmode}_1$	1,482 μs
	$\texttt{WSmode}_2$	1,729 μs
	$\texttt{WSmode}_3$	2,080 μs
	$\texttt{WSmode}_4$	2,600 μs
Erase	$\texttt{ESmode}_{fast}$	5,000 μs
	$\texttt{ESmode}_{slow}$	20,000 μs
Read	-	100 μs

maximum number of P/E cycles was set to 3,000. The nominal page program time (i.e., T_{PROG}) and the nominal block erase time (i.e., T_{ERS}) were set to 1.3 ms and 5.0 ms, respectively. Table 4 summarizes the latency variations of write-speed modes and erase-speed modes used in our evaluations.

We evaluated the proposed dvsFTL in two different environments, mobile and enterprise environments. Since the organizations of mobile storage systems and enterprise storage systems are quite different, we used two extFlashBench configurations for different environments as summarized in Table 5. For a mobile environment, extFlashBench was configured to have two channels, and each channel has a single NAND chip. Since mobile systems are generally resource-limited, the size of a circular buffer for a mobile environment was set to 80 KB only (i.e., equivalently 10 8-KB pages). For an enterprise environment, extFlashBench was configured to have eight channels, each of which was composed of four NAND chips. Since enter-

Environments	Channels	Chips	Buffer
Mobile	2	2	80 KB
Enterprise	8	32	32 MB

Table 5: Summary of two extFlashBench configurations.

prise systems can utilize more resources, the size of a circular buffer was set to 32 MB (which is a typical size of data buffer in HDD) for enterprise environments.

We carried out our evaluations with two different techniques: baseline and dvsFTL. Baseline is an existing DeVTS-wPT-unaware FTL that always uses the highest erase voltage mode and the fast erase mode for erasing NAND blocks, and the fastest write mode for writing data to NAND blocks. dvsFTL is the proposed DeVTS-wPT-aware FTL which decides the erase voltage and the erase time depending on the characteristic of a workload and the write-performance tuning techniques, described in Sections 3.1, 3.2.1 and 4.1, so it can maximally exploit the benefits of dynamic program and erase scaling.

Our evaluations were conducted with various I/O traces from mobile and enterprise environments. In order to replay I/O traces on top of the extFlashBench, we developed a trace replayer. The trace replayer fetches I/O commands from I/O traces and then issues them to the extFlashBench according to their inter-arrival times to a storage device. After running traces, we measured the maximum number of P/E cycles, $N_{P/E}^{max}$, which was actually conducted until flash memory became unreliable. We then compared it with that of Baseline. The overall write throughput is an important metric

that shows the side-effect of dvsFTL on storage performance. For this reason, we also measured the overall write throughput while running each I/O trace.

4.2.2 Workload Characteristics

We used 8 different I/O traces collected from Android-based smartphones and real-world enterprise servers. The `m_down` trace was recorded while downloading a system installation file (whose size is about 700 MB) using a mobile web-browser through 3G network. The `m_p2p1` trace included I/O activities when downloading multimedia files using a mobile P2P application from a lot of rich seeders. Six enterprise traces, `hm_0`, `proj_0`, `prxy_0`, `src1_2`, `stg_0`, and `web_0`, were from the MS-Cambridge benchmarks [30]. However, since enterprise traces were collected from old HDD-based server systems, their write throughputs were too low to evaluate the performance of modern NAND flash-based storage systems. In order to partially compensate for low write throughput of old HDD-based storage traces, we accelerated all the enterprise traces by 100 times so that the peak throughput of the most intensive trace (i.e., `src1_2`) can fully consume the maximum write throughput of our NAND configuration. (In our evaluations, therefore, all the enterprise traces are 100x-accelerated versions of the original traces.)

Since recent enterprise SSDs utilize lots of inter-chip parallelism (multiple channels) and intra-chip parallelism (multiple planes), peak throughput is significantly higher than that of conventional HDDs. We tried to find appropriate enterprise traces which satisfied our requirements to (1) have

public confidence; (2) can fully consume the maximum throughput of our NAND configuration; (3) reflect real user behaviors in enterprise environments; (4) are extracted from under SSD-based storage systems. To the best of our knowledge, we could not find any workload which met all of the requirements at the same time. In particular, there are few enterprise SSD workloads which are opened to public.

Table 6 summarizes the distributions of inter-arrival times of our I/O traces. Inter-arrival times were normalized over $T_{PROG}^{effective}$ which reflects parallel NAND operations supported by multiple channels and multiple chips per channel in the extFlashBench. For example, for an enterprise environment, since up to 32 chips can serve write requests simultaneously, $T_{PROG}^{effective}$ is about 40 μs (i.e., 1300 μs of T_{PROG} is divided by 32 chips.). On the other hand, for a mobile environment, since there are only 2 chips can serve write requests at the same time, $T_{PROG}^{effective}$ is 650 μs. Although the mobile traces collected from Android smartphones (i.e., `m_down` [31] and `m_p2p1`) exhibit very long inter-arrival times, normalized inter-arrival times over $T_{PROG}^{effective}$ are not much different from the enterprise traces, except that the mobile traces show distinct bimodal distributions which no write requests in $1 <$t≤ 2.

4.2.3 Endurance Gain Analysis

In order to understand how much $N_{P/E}^{max}$ is improved by DeVTS-wPT, each trace was repeated until the total sum of the effective wearing reached 3K. Measured $N_{P/E}^{max}$ values were normalized over that of Baseline. Figure 21 shows normalized $N_{P/E}^{max}$ ratios for eight traces with two different

Trace	Distributions of normalized inter-arrival times t over $T_{PROG}^{effective}$ [%]		
	$t \leq 1$	$1 < t \leq 2$	$t > 2$
proj_0	40.6%	47.0%	12.4%
src1_2	41.0%	55.6%	3.4%
hm_0	14.2%	72.1%	13.7%
prxy_0	8.9%	34.6%	56.5%
stg_0	7.1%	81.5%	11.4%
web_0	5.4%	36.7%	56.9%
m_down	45.9%	0.0%	54.1%
m_p2p1	49.5%	0.0%	50.5%

Table 6: Normalized inter-arrival times of write requests for eight traces used for evaluations.

techniques. Overall, the improvement on $N_{P/E}^{max}$ is proportional to inter-arrival times as summarized in Table 6; the longer inter-arrival times are, the more likely slow write modes are selected.

dvsFTL improves $N_{P/E}^{max}$ by 69%, on average, over baseline for the enterprise traces. For proj_0 and src1_2 traces, improvements on $N_{P/E}^{max}$ are less than 50% because inter-arrival times of more than 40% of write requests are shorter than $T_{PROG}^{effective}$ so that it is difficult to use the lowest erase voltage mode. For the other enterprise traces, $N_{P/E}^{max}$ is improved by 79%, on average, over baseline.

On the other hand, for the mobile traces, dvsFTL improves $N_{P/E}^{max}$ by only 38%, on average, over baseline. Although more than 50% of write requests have inter-arrival times twice longer than $T_{PROG}^{effective}$, dvsFTL could not improve $N_{P/E}^{max}$ as much as expected. This is because the size of the

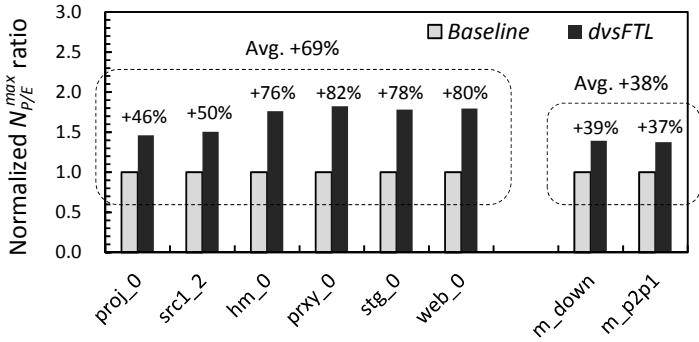


Figure 21: Comparisons of normalized $N_{P/E}^{max}$ ratios for eight traces.

circular buffer is too small for buffering the increase in the buffer utilization caused by the garbage collection. For example, when a NAND block is erased by the fast speed erase mode, the buffer utilization is increased by 40% for the mobile environment while the effect of the fast erase mode on the buffer utilization is less than 0.1% for the enterprise environment. Moreover, by the same reason, the slow erase speed mode cannot be used in the mobile environment.

4.2.4 Overall Write Throughput Analysis

Although dvsFTL uses slow write modes frequently, the decrease in the overall write throughput over Baseline is less than 2.2% as shown in Figure 22. For `proj_0` trace, the overall write throughput is decreased by 2.2%. This is because, in `proj_0` trace, the circular buffer may become full by highly clustered write requests. When the circular buffer becomes full, if the foreground garbage collection should be invoked, the write response time of NAND chips can be directly affected. Although inter-arrival times

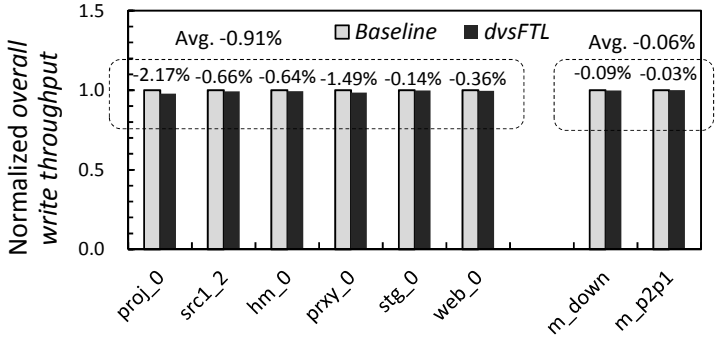


Figure 22: Comparisons of normalized overall write throughputs for eight traces.

in `prxy_0` trace are relatively long over other enterprise traces, the overall write throughput is degraded more than the other enterprise traces. This is because almost all the write requests exhibit inter-arrival times shorter than 10 ms so that the background garbage collection is not invoked at all. (In our dvsFTL setting, the background garbage collection is invoked when a idle time between two consecutive requests is longer than 300 ms.) As a result, the foreground garbage collection is more frequently invoked, thus increasing the write response time.

We also evaluated if there is an extra delay from a host in sending a write request to the circular buffer because of DeVTS-wPT. Although dvsFTL introduced a few extra queueing delay for the host, the increase in the average queueing delay per request was negligible compared to $T_{PROG}^{effective}$. For example, for `src1_2` trace, 0.4% of the total programmed pages were delayed, and the average queueing delay per request was 2.6 μs. For `stg_0` trace, less than 0.1% of the total programmed pages were delayed, and the average queueing delay per request was 0.1 μs.

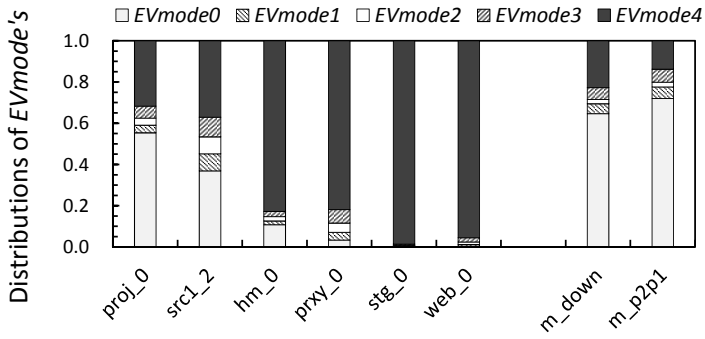
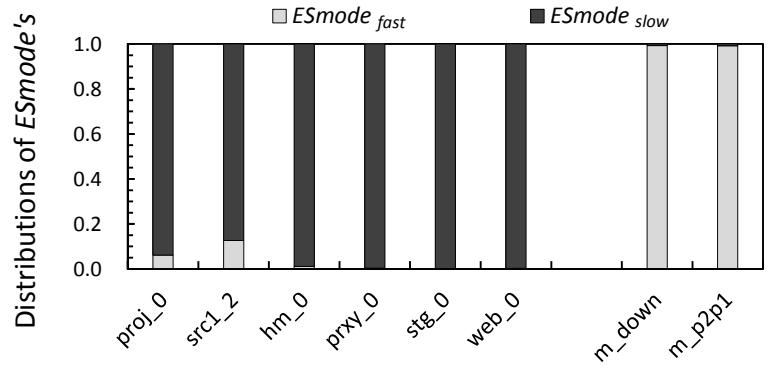


Figure 23: Distributions of `EVmode`'s used.

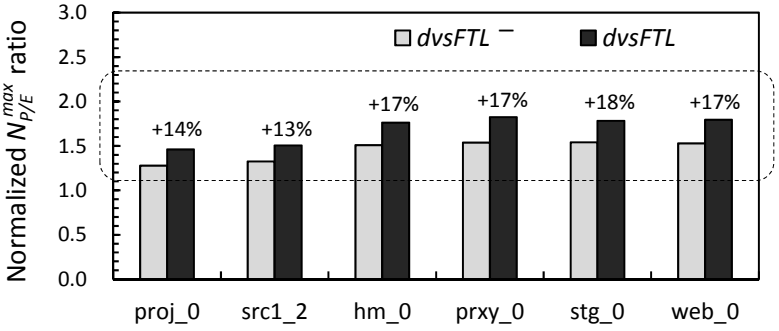
4.2.5 Detailed Analysis

We performed a detailed analysis on the relationship between the erase voltage/speed modes and the improvement of $N_{P/E}^{max}$. Figure 23 presents distributions of `EVmode`'s used for eight I/O traces. Distributions of `EVmode`'s exactly correspond to the improvements of $N_{P/E}^{max}$ as shown in Figures 17 and 18; the more frequently a low erase voltage mode is used, the higher the endurance gain is. In our evaluations for eight I/O traces, lazy erases are rarely used for all the traces.

Figure 24(a) shows distributions of `ESmode`'s for eight I/O traces. Since the slow erase mode is selected by using the effective buffer utilization, there are little chances for selecting the slow erase mode for the mobile traces because the size of the circular buffer is only 80 KB. On the other hand, for the enterprise environment, there are more opportunities for selecting the slow erase mode. Even for the traces with short inter-arrival times such as `proj_0` and `src1_2`, only 5%$\sim$10% of block erases used the fast erase mode.



(a) Distributions of ESmode's used.



(b) The effect of $\texttt{ESmode}_{slow}$ on improving $N_{P/E}^{max}$.

Figure 24: Distributions of ESmode's used and the effect of ESmode's on $N_{P/E}^{max}$.

We also evaluated the effect of the slow erase mode on the improvement of $N_{P/E}^{max}$. For this for evaluation, we modified our dvsFTL so that $\texttt{ESmode}_{fast}$ is always used when NAND blocks are erased. (We represent this technique by dvsFTL$^-$.) As shown in Figure 24(b), the slow erase mode can improve the NAND endurance gain up to 18%. Although the slow erase mode can increase the buffer utilization, its effect on the write throughput was almost negligible.

Chapter 5

Lifetime Improvement Technique Using Retention-Capability Tuning

In this chapter, we propose a comprehensive SSD lifetime improvement technique, called Dynamic Erase Voltage and Time Scaling with Write Performance and Retention Capability Tuning (DeVTS-wPRT), which utilizes both the write performance tuning and retention capability tuning so that the potential of the DeVTS framework reaches the fullest extent. Our DeVTS-wPRT technique actively exploits the tradeoff relationships between the NAND requirements at a software level so that NAND endurance can be improved while the overall write performance and retention requirements of SSDs are not affected. For example, when incoming write requests are not so intensive that the maximum performance of NAND devices is not fully required, a DeVTS-wPRT-enabled technique takes advantage of idle times between consecutive write requests to tune down the program or the erase speed as slowly as possible. In addition, when some of data is updated frequently such that a long retention time is not needed, a DeVTS-wPRT-enabled technique decides to tune down the retention capability of such data as low as possible. If such a low-performance requirement or short-retention requirement is detected, the DeVTS-wPRT-enabled technique selects the most proper speed and age modes for each program operation, or

chooses the most suitable voltage and speed modes for each erase operation. By actively employing endurance-enhancing erase modes (i.e., a slow erase mode with a lower erase voltage) depending on workload conditions, $N_{P/E}^{max}$ is significantly increased because less damaging erase operations are more frequently used.

We have implemented a DeVTS-wPRT-aware FTL, called *dvsFTL+*, which dynamically adjusts the erase voltage and speed modes by properly tuning the performance and retention capabilities of write requests. dvsFTL+ selects the most proper write speed mode and erase voltage/speed modes based on the utilization of a write buffer. In order to decide the most appropriate write-retention mode, an existing data separator in SSDs is redesigned to securely predict the future update time of the current write request. When it predicts that the written data will not be updated until its retention deadline expires, a data reclaim process is proactively invoked to avoid retention failures. The existing key FTL modules (e.g., mapping table, garbage collector and wear leveler) were also revised to make them DeVTS-wPRT-aware to maximize the efficiency of dvsFTL+. We evaluated the effectiveness of dvsFTL+ with an *extFlashBench* emulation environment [24] where the DeVTS-wPRT-enabled NAND emulation model was integrated. Our experimental results using various I/O traces, collected from enterprise servers, show that dvsFTL+ can increase $N_{P/E}^{max}$ by 52%, on average, over dvsFTL (which exploits only write-performance tuning). dvsFTL+ can increase $N_{P/E}^{max}$ by 94%, on average, over an existing DeVTS-wPRT-unaware FTL without sacrificing the performance and retention requirements of SSDs.

5.1 Design and Implementation of dvsFTL+

5.1.1 Overview

In order to improve NAND endurance without affecting the other NAND requirements, we have implemented a DeVTS-wPRT-aware FTL, *dvsFTL+*, which dynamically changes erase scaling modes and write capability tuning modes based on the NAND endurance model. Figure 25 illustrates an organizational overview of dvsFTL+ based on an existing page-level mapping FTL with additional modules for supporting DeVTS-wPRT. *The DVS manager* is the key module which selects the most appropriate erase scaling mode and write capability tuning mode for a given write request depending on the performance and retention requirements. Firstly, the write-speed mode ($\texttt{WSmode}_i$) and erase-speed mode ($\texttt{ESmode}_k$) are selected based on the write-performance requirement estimated using the write buffer. Secondly, the write-retention mode ($\texttt{WRmode}_m$) is chosen based on the retention requirement predicted by the retention-time predictor. Finally, the DVS manager decides the erase-voltage mode ($\texttt{EVmode}_j$) by considering selected write capability tuning modes. In order to preserve the retention requirement, *the retention keeper* periodically checks the remaining retention time of written data and rewrites them to another NAND page when their retention deadline approaches.

The Wmode selector decides the most proper write-retention mode for a given write request based on the predicted future update time (i.e., retention-time requirement) of that request. If it is predicted that a request will be updated within the predefined time period, which is much shorter

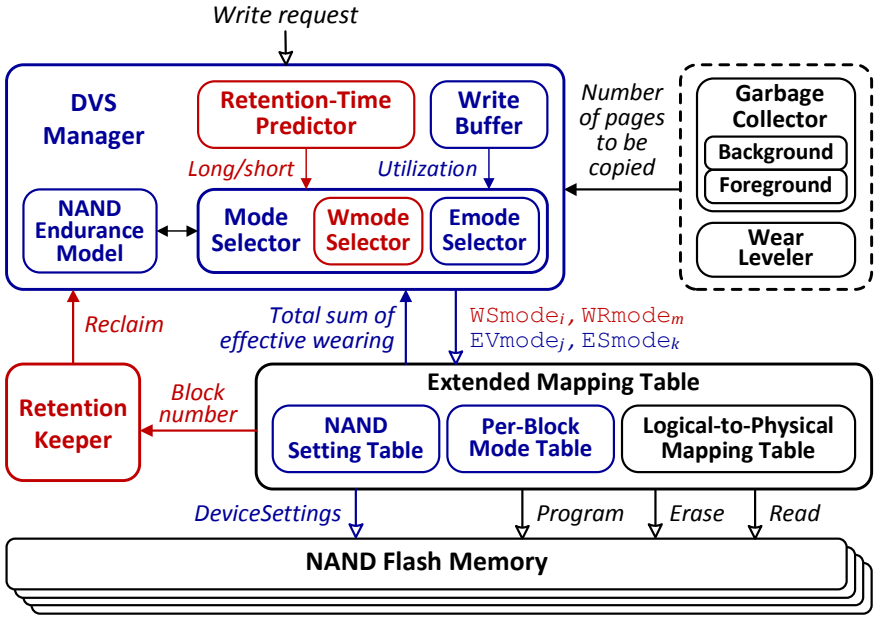


Figure 25: An organizational overview of dvsFTL+.

than the nominal retention-time specification of NAND devices, the Wmode selector selects the short-retention write mode (i.e., $\texttt{WRmode}_{short}$) for that request. When a prediction regarding the future retention-time requirement is incorrect, a reclaim process [28] should be performed to preserve the durability of retention tuned data. However, since too frequent reclaim operations can substantially cancel the lifetime benefit of retention-capability tuning as well as interfere with foreground activities to serve user requests, it is required to minimize the number of reclaimed pages and the overhead of a reclaim operation.

5.1.2 Retention Requirement Prediction

Our proposed retention-time predictor estimates the future retention-time requirement of a write request based on the average update interval for recent requests. Since there are only two write-retention modes in our NAND endurance model, it is necessary to classify whether or not the average update interval is shorter than the predefined short retention-time interval T_{ret}^{short} (e.g., 0.07 days as defined in Section 3.2.2). In our implementation, the retention-time predictor is based on an existing data separator [26] with a different control policy for capturing the average update interval and for making a reliable decision on the write-retention mode (as will be described in Section 5.1.3).

Each LBA is mapped to multi-dimensional counters incremented whenever corresponding write requests are issued. In order to compare the update interval to T_{ret}^{short}, all the counters are decayed regularly after a designated time interval T_{decay} (in this dissertation, $T_{decay} = T_{ret}^{short}$). If the update interval of an LBA is shorter than T_{decay}, the corresponding counter value will increase. Otherwise, the counter values will decrease. After multiple decaying intervals, when the counter value is greater than the predefined threshold value, the retention-time predictor decides that the recent update interval of that LBA is shorter than T_{ret}^{short} on average. In this case, the retention-time predictor predicts that the current write request will be also updated within T_{ret}^{short} by exploiting the temporal locality of I/O requests.

Figure 26 shows a functional overview of our proposed retention-time predictor. Since maintaining all the counters for each LBA is too expensive

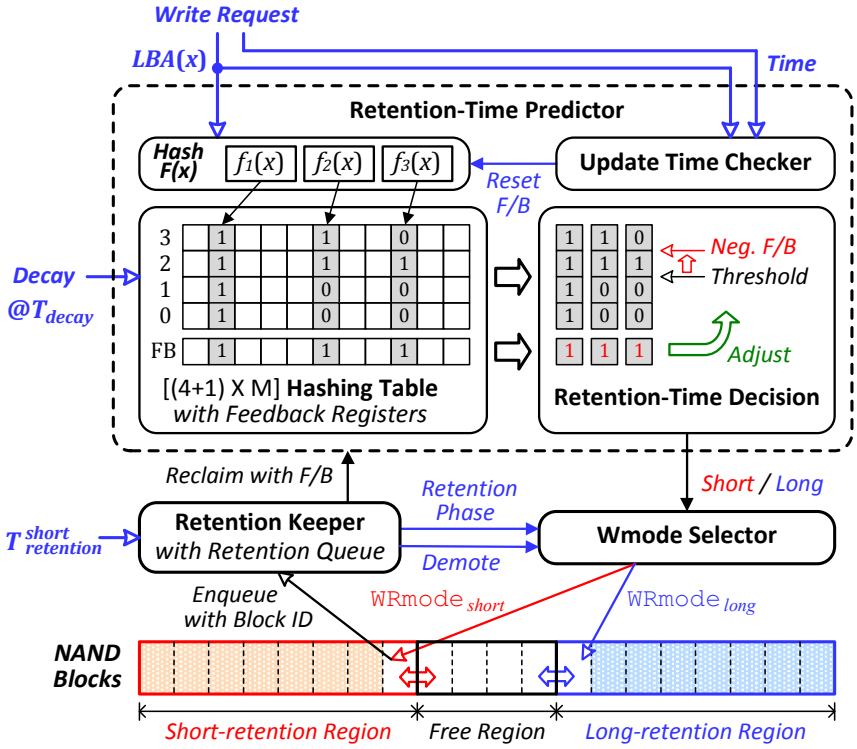


Figure 26: A functional overview of the write-retention mode selection and retention requirement management procedures.

to be implemented in practice, the proposed retention-time predictor keeps only a limited number of counters which are referenced by three hash functions. In deciding the retention-time requirement of an LBA, all the counters corresponding to that LBA are considered simultaneously. The retention-time predictor can be implemented with a small space overhead (i.e., 64 KB per 1-GB storage capacity). Furthermore, if the data separator is already employed in a storage system, the retention-time predicting scheme can be easily implemented by revising the existing data separator with a negligible space overhead.

5.1.3 Maximization of Endurance Benefit

In order to maximize the endurance benefit of retention-capability tuning, minimizing reclaimed pages is one of the key design challenges of the write-retention mode selection.

Misprediction Control

One of the main sources behind misprediction is *hash collisions* in the hashing table as shown in Figure 26. When counters corresponding to cold data (which are rarely updated) are unintentionally incremented due to hash collisions, such cold data can be mispredicted as short-retention data. (We denote this misprediction as *false-short*.) Once mispredicted data is written with $\texttt{WRmode}_{short}$, such data will be eventually reclaimed before T_{ret}^{short}.

In order to minimize the false-short ratio due to hash collisions, we introduce a misprediction control technique based on the past false-short history. As shown in Figure 26, each counter has an additional feedback register which is set to one when the written data is reclaimed. The purpose of these feedback registers is to impose a penalty for the mispredicted write so that consecutive mispredictions for that request is prevented. If all the corresponding feedback registers were already set, the retention-time predictor determines the retention-time requirement in a conservative fashion by raising the decision threshold level. For example, when the counter values of a request are 15, 12, and 4, and all the dedicated feedback registers are already set, this request is classified as long-retention data instead of short-retention data because the decision threshold level is raised from the normal

level (e.g., 4) to the higher level (e.g., 8). When prediction is correct (i.e., data written with $\texttt{WRmode}_{short}$ is updated within T_{ret}^{short}), the feedback registers are reset so that the decision threshold is reverted back to the normal level.

Selective Retention Tuning

When the update characteristics of I/O requests are changed so that too many retention-tuned pages are reclaimed, it is more beneficial to suspend retention-capability tuning. For example, if the number of pages per a block is 100, in order to write 5,000 pages with a combination of $\texttt{WRmode}_{short}$ and $\texttt{WSmode}_0$, 50 blocks erased with $\texttt{EVmode}_2$ (of which effective wearing is 0.59 as summarized in Figures 17 and 18) are consumed. In this case, the total endurance gain of retention-tuned writes is 20.50 ($= (1.00 - 0.59) \times 50$). However, when 60 pages per block are reclaimed with $\texttt{WRmode}_{long}$, 30 ($= 60 \times 50/100$) blocks erased with $\texttt{EVmode}_0$ (of which effective wearing is 0.78) are consumed during reclaim operations. In this case, the total endurance loss of reclaimed writes is 23.40 ($= 0.78 \times 30$). Since the endurance loss is larger than the endurance gain in this example, it is better not to use the short-retention write mode. In order to make such a decision, we estimate the *break-even point* at which the endurance gain of retention-tuned writes is equal to the endurance loss of reclaimed writes. In the previous example, the break-even number (i.e., N^{be}) of reclaimed pages per a block is 52.6 ($= (1.00 - 0.59)/0.78 \times 100$). The retention keeper continuously monitors the average number of reclaimed pages per a block. When the average number of reclaimed pages becomes greater than N^{be}, the retention

keeper switches the *retention-tuning phase* from the *enable phase* to the *suspend phase*. In the *suspend phase*, the Wmode selector always selects $\texttt{WRmode}_{long}$ regardless of retention-time prediction results. When beneficial I/O characteristics are detected, the retention keeper resumes retention-capability tuning again.

5.1.4 Minimization of Reclaim Overhead

Since it is difficult to completely eliminate mispredicted writes, minimizing the overhead of a reclaim operation is also required in order not to affect foreground activities. The main goal of the reclaim operation is to preserve the durability of stored data written with $\texttt{WRmode}_{short}$. In order to reliably rewrite mispredicted data before its retention deadlines expire, the retention keeper periodically (e.g., one tenth of the short retention-time interval) checks its remaining retention time. However, since maintaining the retention deadline for each written page requires excessive system resources as well as high checking overheads, we have developed a simple but effective reclaim technique. As shown in Figure 26, data for each write-retention mode is written to different regions, i.e, the short-retention region and the long-retention region. This separation technique can provide another advantage in reducing the write amplification factor over the existing data separation technique [26]. Since data written with $\texttt{WRmode}_{short}$ are likely to be updated within T_{ret}^{short}, when such a block is chosen in the garbage collection process, if it is not yet reclaimed, the number of unnecessary data copies can be significantly reduced. After short-retention data is written to a free block chosen from the free region, the block id is inserted into the

retention queue in the retention keeper with the written time. Although following data is written to the block at different times, the worst-case retention deadline is still determined by the earliest written time. Since the retention queue maintains its entries in a FIFO fashion, the remaining retention times for each block are automatically sorted in ascending order, thus simplifying the checking process. When the retention keeper identifies a block whose retention deadline has almost expired, mispredicted pages in the identified block are reclaimed to a free block, selected from the free region, with a *demoted* write-retention mode (i.e., $\texttt{WRmode}_{long}$). After reclaim operation is completed, the newly written block is inserted into the long-retention region.

5.2 Experimental Results

5.2.1 Experimental Settings

We evaluated the effectiveness of the proposed dvsFTL+ with *extFlashBench*, an extended version of an existing unified development environment for NAND flash-based storage systems [24]. extFlashBench emulates the key operations of DeVTS-wPRT-enabled NAND devices in a timing-accurate fashion so that it is possible to keep track of temporal interactions among various NAND operations [15]. In order to reflect the chip-level parallelism (which is one of the key factors affecting the maximum write performance of an SSD), extFlashBench was configured to have eight channels, each of which was composed of four NAND chips. Each NAND chip employed 512 blocks which were composed of 128 8-KB pages. The size of a write buffer was set to 16 MB which was about 0.1% of the total NAND

capacity. In recent SSDs, the total DRAM capacity is usually set to about 1% of the total NAND capacity. However, since most of the DRAM area is already being used to maintain the meta data such as a mapping table, we set the buffer size to only 10% of the available DRAM capacity. In Section 5.2.6, we discuss the effect of the different buffer size in detail.

Our evaluations were performed with two different techniques: baseline and dvsFTL+. Baseline is an existing DeVTS-wPRT-unaware FTL that does not use the erase scaling modes and the write tuning modes. dvsFTL+ is the proposed DeVTS-wPRT-aware FTL which fully exploits DeVTS-wPRT-enabling techniques, described in Sections 3.1 and 3.2, depending on workload characteristics so that the lifetime benefit of DeVTS-wPRT can be maximally achieved while still satisfying all the NAND requirements. Each technique was evaluated by replaying various I/O traces on top of extFlashBench. When I/O requests were issued according to their timing information in the trace files, corresponding NAND operations were performed in extFlashBench. We continuously replayed the traces on NAND blocks until they became unreliable and measured the maximum number of P/E cycles, $N_{P/E}^{max}$. We also measured the overall write throughput and retention times which are related to the side effect of DeVTS-wPRT.

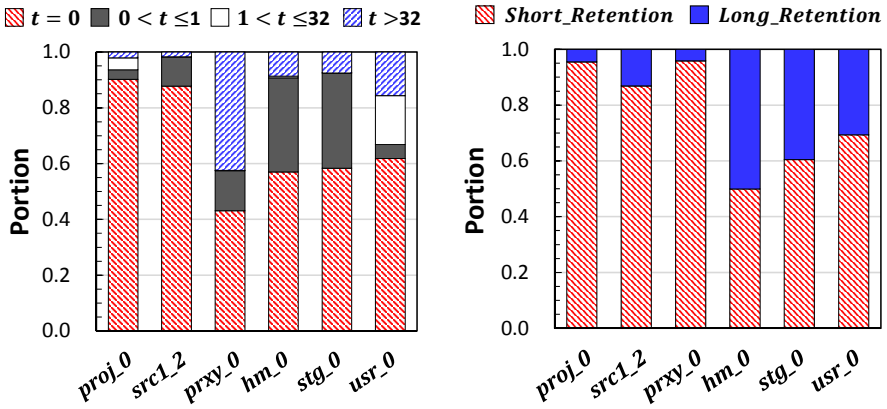
5.2.2 Workload Characteristics

In our evaluation, we used six I/O traces, *proj_0*, *src1_2*, *prxy_0*, *hm_0*, *stg_0*, and *usr_0*, selected from the MSR Cambridge traces [30]. Although these traces include I/O characteristics in real-world enterprise servers, their I/O rates were too low to meaningfully stimulate the temporal behavior of

high-performance NAND flash-based storage systems. In order to utilize these traces in our evaluations, we accelerated I/O rates of all the traces by 100 times so that the peak I/O rate of the most write intensive trace is comparable to the maximum write performance of our extFlashBench configuration [15][32].

Figure 27(a) shows the distributions of the inter-arrival times for write requests of six traces. Inter-arrival times were normalized over the effective program time $T_{Pgm}^{effective}$ of extFlashBench. Since up to 32 NAND chips can serve write requests simultaneously, $T_{Pgm}^{effective}$ is 32 times shorter than the nominal program time T_{Pgm} (i.e., the write latency of $\texttt{WSmode}_0$) of a single chip. When there were multiple pages in a write request, their inter-arrival times t were classified as the '$t = 0$' case in Figure 27(a). Alternatively, when write requests, containing only one page, were issued in a sporadic fashion, they were classified as the '$t > 32$' case. It is expected that the overall endurance gain for a sporadic trace (e.g., *prxy_0*) will be higher than that for an intensive trace (e.g., *proj_0*) because slower write and erase modes can be more frequently used in a sporadic trace.

Figure 27(b) shows the distributions of the retention times for write requests of six traces. The short-retention group and long-retention group were classified by T_{ret}^{short} (i.e., 0.07 days). In our evaluation, we set T_{ret}^{short} to 0.07 days as shown in Figure 16(b). This is because the total time of traces was only about 1 day. If our DeVTS-wPRT technique is to be employed in real systems, it is better to increase T_{ret}^{short} to 1 day for more reliable retention management. In this case, the lifetime benefit of dynamic retention tuning is slightly reduced because corresponding r_{ret}^{d} increases from 0.33



(a) Distributions of normalized inter-arrival times t over $T_{Pgm}^{effective}$.

(b) Distributions of retention times classified by T_{ret}^{short}.

Figure 27: Characteristics of write requests for six traces.

to 0.50. An interesting aspect is that there is a strong correlation between the distribution of inter-arrival times and those of retention times (except *prxy_0*). The more intensively write requests are issued, the more frequently they are updated. Therefore, for intensive traces, it is expected that the weakness of the write-speed tuning mode can be partly compensated for by the write-retention tuning mode.

5.2.3 Endurance Gain Analysis

In order to measure $N_{P/E}^{max}$ (i.e., the effective lifetime of a NAND device as defined in Section 3.1.2), each trace was repeated until ΣEW reached 3K [6]. Measured $N_{P/E}^{max}$ values were normalized over 3K. Figure 28(a) shows $N_{P/E}^{max}$ ratios for six traces with two different techniques. dvsFTL+ extends $N_{P/E}^{max}$ by 94%, on average, over Baseline.

As we expected, the improvements on $N_{P/E}^{max}$ for each trace clearly

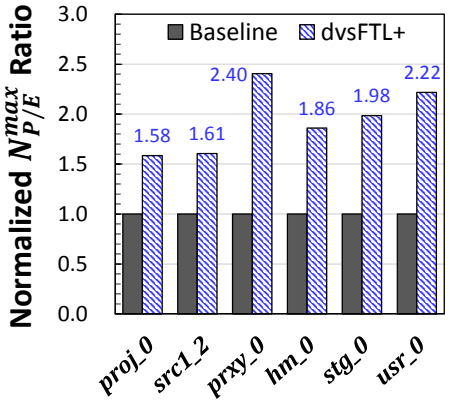
exhibit similar trends as the distributions of inter-arrival times and retention times as shown in Figures 27(a) and 27(b), respectively. In the case of *proj_0* trace, $N_{P/E}^{max}$ is improved by only 58% because most of the write requests are issued instantaneously so that 40% of erase operations cannot take advantage of endurance-enhancing modes at all as shown in Figure 28(b). However, since a considerable part of the rest of erase operations exploits the short-retention write mode, the limited $N_{P/E}^{max}$ ratio due to highly clustered consecutive writes is partly compensated for. (For more detail, see Section 5.2.6.) Alternatively, for the *usr_0* trace, $N_{P/E}^{max}$ is improved by up to 122% because more than half of erase operations are performed with the lowest erase voltage. In particular, for the *prxy_0* trace, the improvement ratio of $N_{P/E}^{max}$ goes up to 140%. This is because most NAND operations frequently utilize both the slow-speed write mode and short-retention write mode as expected in the characteristics of *prxy_0* trace.

5.2.4 NAND Requirements Analysis

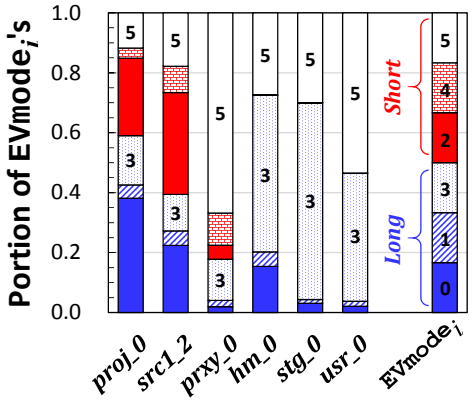
Since the main goal of dvsFTL+ is to extend $N_{P/E}^{max}$ while the other NAND requirements are left untouched, we checked whether or not the overall write-performance and retention-time requirements were preserved.

Overall Write-Performance Requirement

When a write request is issued and the write buffer is full (i.e., u is 1.0), serving that request is delayed until one of requests queued in the buffer is written to a NAND chip so that u is decreased below 1.0. This delay time



(a) Comparisons of normalized $N_{P/E}^{max}$ ratios.



(b) Distributions of the erase voltage modes.

Figure 28: Comparisons of the endurance gain and distributions of the $\texttt{EVmode}_i$'s for six traces.

can be further amplified when the foreground garbage collection process is performed in NAND chips. Although dvsFTL+ frequently uses slow-speed write and erase modes, since such slow-speed modes are selected only when the write-performance requirement is not urgent, dvsFTL+ does not incur an additional delay over Baseline as summarized in Table 7.

For the *proj_0* trace, the overall write throughput is improved by 0.5%

Table 7: Comparisons of the overall write performance for six traces.

		proj_0	*src1_2*	*prxy_0*	*hm_0*	*stg_0*	*usr_0*
Overall Write Throughput [MB/s]	Baseline	23.65	7.59	14.15	3.95	2.74	2.27
	dvsFTL+	23.78	7.60	14.16	3.95	2.74	2.27
Average Read Response Time [us]	Baseline	315	242	333	258	208	194
	dvsFTL+	310	245	300	288	206	194
Portion of Queuing Delay	Baseline	5.6%	1.1%	0.1%	0.2%	0.0%	0.0%
	dvsFTL+	4.8%	1.0%	0.0%	0.1%	0.0%	0.0%
WAF	Baseline	1.15	1.07	1.11	1.14	1.25	1.09
	dvsFTL+	1.10	1.07	1.03	1.06	1.13	1.05

because the worst-case delay time due to the garbage collection process is reduced. For example, the write amplification factor (WAF), which represents the average garbage collection overhead, is reduced by about 0.4% so that the portion of delayed requests among total requests is reduced from 5.6% to 4.8%. For other traces, the overall write throughput and portion of delayed requests of dvsFTL+ are maintained at the same level as those of Baseline.

Overall Retention Requirement

Since $\texttt{WRmode}_{short}$ aggressively reduces the retention capability of NAND pages, in order to guarantee the durability of the stored data, mispredicted pages whose retention deadlines are imminent should be properly reclaimed. However, when there are too many mispredicted pages, retention

failures may occur because the number of reclaimable pages for the given checking period is limited. For example, if the retention checking period is 60 s and the shortest program latency is 1,300 μs, the retention keeper can reclaim up to 46,153 pages (which is 2.2% of the total NAND pages in extFlashBench). Although dvsFTL+ frequently uses $\texttt{WRmode}_{short}$ for writing data onto NAND pages, retention failures did not occur in our evaluations. This is mainly because the misprediction ratio is sufficiently suppressed by the misprediction control techniques, described in Section 5.1.3, so that the numbers of mispredicted pages are maintained below the maximum number of reclaimable pages at all times.

In this dissertation, we assume that the power is always supplied. However, if the power is cut off, since the retention keeper cannot work without the power supply, retention failures may occur. This *predictable* data loss can be prevented by rewriting valid pages written with $\texttt{WRmode}_{short}$ to other NAND pages with $\texttt{WRmode}_{long}$ during the power hold-up time supported by a storage system [33].

5.2.5 Detailed Analysis of Retention-Time Predictor

The Overall Accuracy of Retention Time Predictor

In order to predict the retention-time requirement of future write requests, we proposed the retention-time predictor as described in Section 5.1.2. Since the main goal of the retention-time predictor is to minimize the misprediction (in particular, false-short) ratio with a reasonable resource overhead, we performed a detailed analysis on how accurate our proposed resource-

optimized predictor is. Table 8 summarizes the analysis results for four traces with four different techniques: History, DA H_noFB, and H_FB. History is a history-based prediction technique which directly utilizes the previous update time interval to predict the next update time [34]. DA, H_noFB, and H_FB are retention-time prediction techniques based on the recent update frequency maintained in multiple update counters, but with different configurations. DA uses a direct-address mapping which keeps the number of counters as many as that of LBAs. Alternatively, since H_noFB has a limited number of counters mapped to corresponding LBAs by hash functions, mispredictions may occur due to hash collisions. H_FB is the proposed prediction technique which employs additional feedback registers and makes an adaptive decision so that the misprediction ratio is substantially reduced.

For the *src1_2* trace, the false-short ratio under History is too high (i.e., 4.8%) to avoid retention failures while the ratio under H_FB is sufficiently suppressed below the tolerable level. Comparing H_FB with H_noFB, the false-short ratio is reduced from 2.3% to 0.9%, a value similar to that of DA. For the other traces, the false-short ratios are also maintained at a low level. These results clearly indicate that our proposed misprediction control technique can efficiently reduce the misprediction ratio, caused by hash collisions, to the comparable level of DA.

However, as summarized in Table 8, another misprediction ratio, i.e., the *false-long* ratio, is increased for write-intensive traces (e.g., *src1_2*). This is because the retention-time predictor in this dissertation mostly focuses on minimizing the false-short ratio. If the false-long ratio is too high, the potential of retention-capability tuning cannot be fully exploited. In order

Table 8: Accuracy of the retention-time predictor under different data separation techniques.

Prediction	Result	*proj_0*				*src1_2*			
		History	DA	no FB	FB	History	DA	no FB	FB
Short	*True*	92.5%	55.2%	75.1%	47.9%	81.9%	49.4%	60.9%	42.6%
	False	2.7%	0.4%	1.6%	0.7%	4.8%	0.8%	2.3%	0.9%
Long	*True*	1.8%	4.1%	2.9%	3.8%	8.3%	12.3%	10.8%	12.2%
	False	3.0%	40.2%	20.4%	47.6%	5.0%	37.5%	26.0%	44.3%

Prediction	Result	*prxy_0*				*hm_0*			
		History	DA	no FB	FB	History	DA	no FB	FB
Short	*True*	94.3%	89.3%	89.8%	85.4%	43.3%	28.1%	28.8%	28.1%
	False	1.3%	0.5%	0.6%	0.6%	5.9%	0.5%	0.7%	0.8%
Long	*True*	2.9%	3.6%	3.5%	3.6%	44.2%	49.6%	49.4%	49.3%
	False	1.5%	6.5%	6.1%	10.5%	6.6%	21.8%	21.1%	21.8%

Prediction	Result	*stg_0*				*usr_0*			
		History	DA	no FB	FB	History	DA	no FB	FB
Short	*True*	56.8%	30.7%	30.9%	29.6%	63.6%	52.8%	53.1%	52.1%
	False	3.2%	0.3%	0.3%	0.3%	4.5%	0.9%	1.0%	1.1%
Long	*True*	36.4%	39.3%	39.3%	39.2%	26.2%	29.7%	29.7%	29.6%
	False	3.6%	29.7%	29.6%	30.8%	5.7%	16.6%	16.3%	17.3%

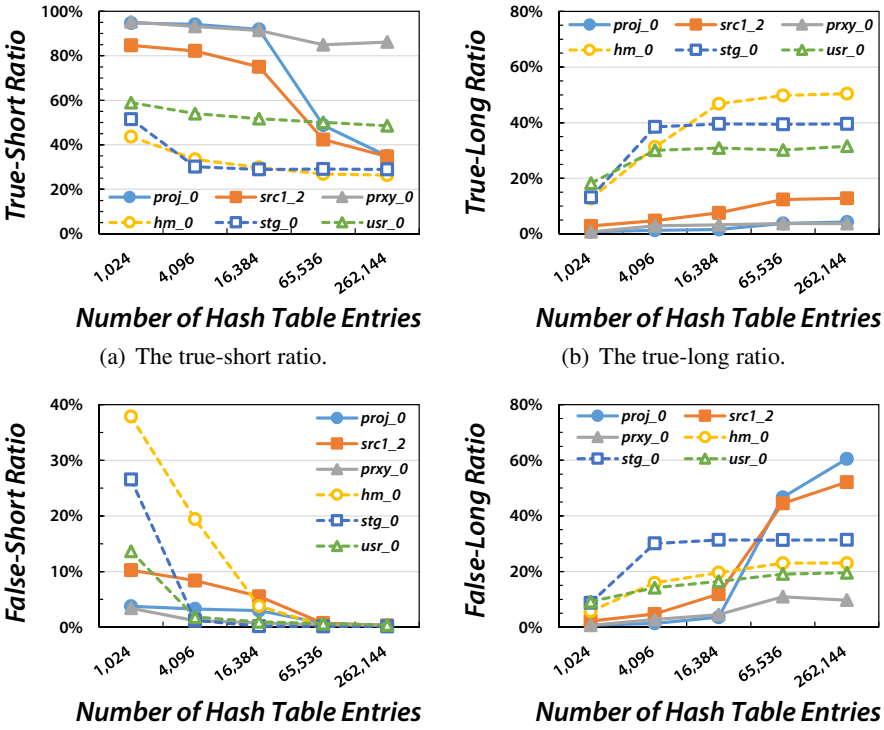
to further extend $N_{P/E}^{max}$ for write-intensive traces, reducing the false-long ratio is also required. Our future work involves developing a more accurate retention-time predictor capable of consistent performance regardless

of varying characteristics of I/O workload.

Sensitivity of Number of Hash Table Entries

The accuracy of the proposed retention-time predictor mainly depends on the number M of hash table entries as shown in Figure 26. Since the probability of hash collisions, which cause mispredictions, is inversely proportional to M, too small M may lower the prediction accuracy. On the contrary, although large M can improve the prediction accuracy, the space overhead may increase. In this dissertation, M was set to 65,536 which is 0.006% of the total storage capacity (i.e., 16 GB) as described in Section 5.1.2. In order to understand how sensitive M is to the prediction accuracy and to check whether or not our selected M is reasonable in terms of both cost and accuracy, we evaluated the prediction accuracy over different M's. Figure 29(c) shows variations of the false-short ratios (i.e., the probability that the prediction of 'short retention' is false) over different M's for six traces. As we expected, the false-short ratios decrease as M increases. In order to reduce the false-short ratio below the target level (i.e., 1% in this dissertation), M is required to be larger than 65,536.

On the other hand, for *proj_0* and *src1_2* traces, the false-long ratios (i.e., the probability that the prediction of 'long retention' is false) increase as M increases. This is because our proposed retention-time predictor estimates the retention-time requirement of a write request by comparing its counter value (which reflects the previous update history for that request) with the predefined threshold level. When M is small (e.g., 1,024), since the counter can be incremented not only by the request corresponding to



(a) The true-short ratio.

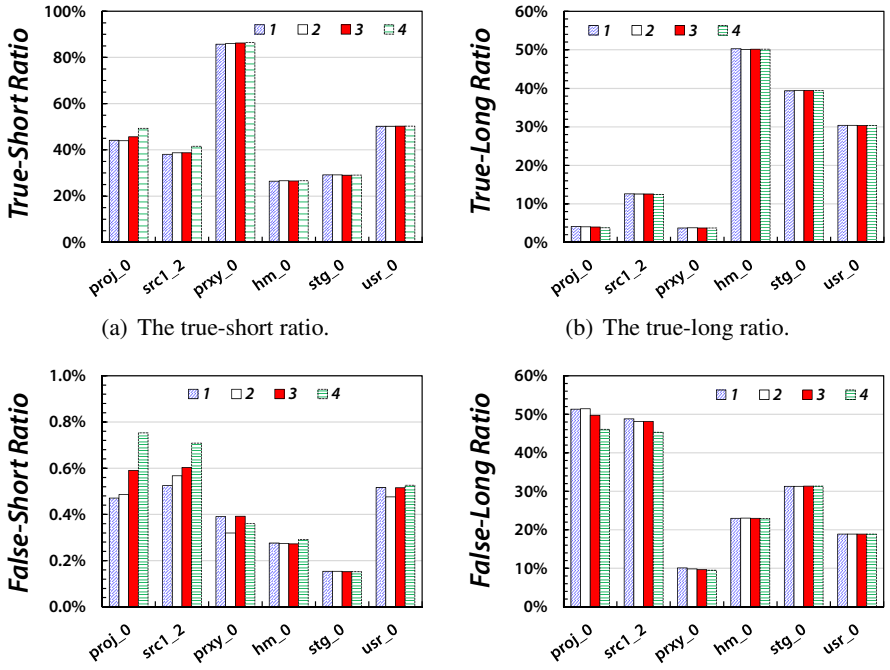
(b) The true-long ratio.

(c) The false-short ratio.

(d) The false-long ratio.

Figure 29: Variations of the prediction accuracy over different numbers of hash table entries.

that counter, but also by other requests due to hash collisions, the counter value is likely to be raised unintentionally depending on the probability of hash collisions. As a result, when the probability of hash collisions is too high (e.g., $M \leq$ 16,384), the false-short ratio increases while lowering the false-long ratio as shown in Figures 29(c) and (d). On the contrary, since the probability of hash collisions is very low when M is sufficiently large (e.g., $M \geq$ 65,536), the counter value can be incremented only by the corresponding request. As a result, a frequently-updated request may be mispredicted as 'long-retention' data until the counter value exceeds the threshold level.



(a) The true-short ratio.

(b) The true-long ratio.

(c) The false-short ratio.

(d) The false-long ratio.

Figure 30: The effects of different numbers of hash functions on the accuracy of the retention-time predictor.

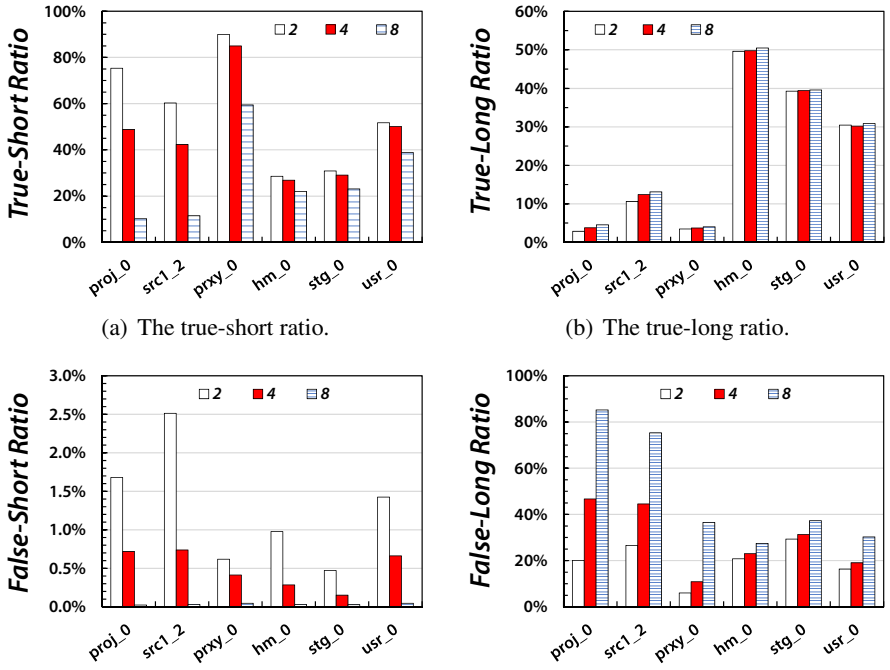
Sensitivity of Number of Hash Functions

The number N_{hash} of hash functions is also one of key design parameters in our proposed retention-time predictor. Although using large N_{hash} has a positive impact on the identification accuracy, too large N_{hash} can unintentionally increase the probability of hash collisions as a side effect. In this dissertation, we used three hash functions in the retention-time predictor. In order to understand how sensitive N_{hash} is to the prediction accuracy, we performed evaluations with different N_{hash}'s (i.e., 1, 2, 3, and 4) as shown in Figures 30 (a), (b), (c), and (d).

As we expected, the overall accuracy is improved as more hash functions are used. For example, when N_{hash} increases from one to four, the true-short ratio and false-long ratio are improved by 14% and 10%, respectively, as shown in Figures 30 (a) and (d). However, as a side effect of large N_{hash}, the false-short ratio also increases as shown in Figure 30 (c). For example, when N_{hash} is four, the average false-short ratio is 0.76%. Although the false-short ratio does not exceed 1% (i.e., the target accuracy) in this case, we set N_{hash} to three instead of four to minimize the negative impact of reclaim operations on the foreground activities.

Sensitivity of Retention Decision Level

Our proposed retention-time predictor determines the retention-time requirement of a write request as 'short' when the counter value is higher than the predefined threshold level N_{th} (e.g., 4) as shown in Figure 26. In this comparison process, an appropriate N_{th} is the key parameter to achieve a reasonable prediction accuracy. In this dissertation, we set N_{th} to four. In order to understand how sensitive N_{th} is to the prediction accuracy, we compared the accuracy over different N_{th}'s as shown in Figure 31. When N_{th} is set to two, the false-short ratio as well as the true-short ratio are higher than those when N_{th} is four because the retention-time predictor is more likely to select 'short retention' in this case. On the contrary, when N_{th} is set to eight, those two ratios are reduced significantly because it is difficult for the predictor to select 'short retention'. Although the false-short ratio is extremely minimized in this case, the endurance benefit of DeVTS cannot be fully achieved because there are little requests classified as 'short



(a) The true-short ratio.

(b) The true-long ratio.

(c) The false-short ratio.

(d) The false-long ratio.

Figure 31: The effects of different decision levels on the accuracy of the retention-time predictor.

retention'. Therefore, we conclude that four is the most optimized setting in our retention-time prediction scheme.

5.2.6 Detailed Analysis of Endurance Gain

Breakdown of Endurance Gain

In order to understand the effect of each endurance-enhancing technique on the overall $N_{P/E}^{max}$ improvement ratio in detail, we modified our dvsFTL+ so that each technique can be enabled separately. Figure 32 shows the increase in $N_{P/E}^{max}$ ratios for six traces when each endurance-enhancing

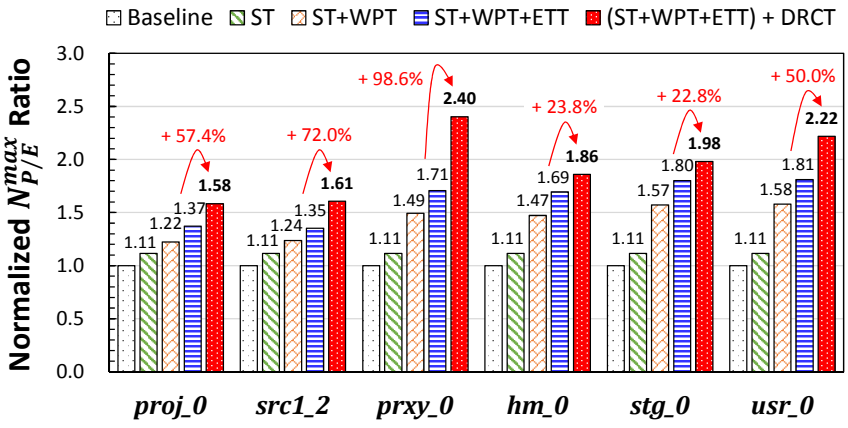


Figure 32: Variations of the normalized $N_{P/E}^{max}$ ratios under different endurance-enhancing techniques for six traces.

technique (i.e., ST, WPT, ETT and DRCT) is enabled one by one on top of baseline. ST is the Static Tuning technique described in Sections 3.2.2 and 3.2.3. WPT is the Write-Performance Tuning technique, and ETT is the Erase-Time Tuning technique, as described in Sections 3.2.1 and 3.1.3, respectively. DRCT is the Dynamic Retention-Capability Tuning technique described in Section 3.2.2. Our proposed dvsFTL+ fully utilizes all the aforementioned techniques.

Among the endurance-enhancing techniques implemented in dvsFTL+, DRCT has the most significant impact on extending $N_{P/E}^{max}$. DRCT is responsible for 34%, on average, of the total endurance gain. The effect of DRCT strongly depends on the true-short ratio summarized in Table 8. For example, for the *prxy_0* trace, predicting short-retention requests is very accurate (i.e., 85.4%). As a result, $N_{P/E}^{max}$ is significantly extended (i.e., 98.6%) by DRCT. However, for the *hm_0* trace, its effect is marginal. The effect of WPT is comparable to that of DRCT. The effects of ETT and ST account for

about 20% and 12%, respectively, of the total endurance gain. In the lifetime improvement technique using write-performance tuning (i.e., dvsFTL) presented in Chapter 4, only ST, WPT, and ETT were employed. In this case, the average $N_{P/E}^{max}$ ratio is only 1.62. Our proposed dvsFTL+ (where DRCT has been added) further extends $N_{P/E}^{max}$ by about 52% over dvsFTL. As shown in Figure 32, since DRCT is more effective for write-intensive traces where the effect of WPT is limited, DRCT can substantially make up for the weaknesses of WPT.

Sensitivity of Buffer Size

The large size of the write buffer offers an advantage to extend $N_{P/E}^{max}$ because the probability of using the slow write and erase modes is increased. However, since an excessively large buffer size is not cost effective in most practical storage systems, we set the buffer size to only 16 MB, only 0.1% of the total storage capacity. In order to understand how sensitive $N_{P/E}^{max}$ ratio is to the buffer size, we performed evaluations with different write buffer sizes as summarized in Table 9. When the buffer size is reduced to 4 MB, the average $N_{P/E}^{max}$ is decreased by 3.6%. Alternatively, with a 64 MB-size write buffer, the average $N_{P/E}^{max}$ ratio is increased by 4.1% as we expected. The effect of a reduced-size buffer on the overall write throughput is negligible because the Wmode selector efficiently chooses the proper write mode.

Table 9: Variations of $N_{P/E}^{max}$ ratios and the overall write throughput over different buffer sizes for six traces.

	Buffer Size	*proj_0*	*src1_2*	*prxy_0*	*hm_0*	*stg_0*	*usr_0*	Avg.
Normalized	4 MB	1.54	1.53	2.17	1.81	1.97	2.18	1.87
$N_{P/E}^{max}$	16 MB	1.58	1.61	2.40	1.86	1.98	2.22	1.94
Ratio	64 MB	1.66	1.72	2.57	1.93	2.00	2.26	2.02
Overall Write	4 MB	23.70	7.59	14.11	3.95	2.74	2.28	9.06
Throughput	16 MB	23.78	7.60	14.16	3.95	2.74	2.27	9.08
[MB/s]	64 MB	24.14	7.61	14.15	3.94	2.74	2.27	9.14

Chapter 6

Reliability Management Technique for NAND Flash Memory

NAND flash memory is a non-volatile memory device which can retain stored data even when power is turned off. Since NAND flash memory stores data as quantities of charges held on floating gates (that are electrically isolated by insulating layers), in theory, NAND flash memory can permanently store its data without a power source if the insulating layers work perfectly. However, actual NAND cells are limited in their data retention capability because various defects in the insulating layers occur during program/erase (P/E) operations. These defects in the NAND cells make charges in the floating gate loosened, thus guaranteeing the integrity of stored data only up to a finite retention time [5]. Since the probability of charge loss due to defects has an exponential dependence on temperature [16], the NAND retention time is specified under a specific operating temperature. For example, NAND flash memory for client-class applications is often required to retain its stored data for at least 1 year at 25 °C [14].

If NAND flash memory is used beyond the specified retention time, the data stored in the NAND flash memory may not be correctly retrieved because of excessive retention errors. For example, when NAND flash memory is left for more than two times longer than the specified retention time, reten-

tion failures may occur, losing the stored data. Moreover, since the NAND retention time decreases exponentially as temperature rises [35], an increase in temperature can significantly degrade the NAND retention capability. For example, when temperature rises to 70 °C, the specified NAND retention time of 1 year (at 25 °C) may be reduced to only 32 hours[1]. Furthermore, the retention-failure problem can be a more serious technical issue when more aggressive flash-optimization techniques (e.g., [12][34]) are widely employed. Since these flash optimization techniques aggressively reduce the NAND retention capability during run time for higher NAND performance [12] or longer NAND endurance [34], retention errors are likely to increase. Thus, there is a strong demand for *efficient on-line* data recovery techniques for retention failures in NAND flash memory.

In order to deal with the NAND retention-failure problem, several data recovery techniques such as the retention failure recovery (`RFR`) technique [36] and the data retention-error recovery pulse (`RFR`) technique [37] have been proposed. However, since `RFR` requires to heat NAND chips, it can be used only as an *off-line* recovery solution. Although `RFR` can be implemented as an on-line recovery solution, it is quite limited because its recovering process is very slow and its recovery capability is rather restricted for recent 20-nm node (or below) NAND flash memory.

[1]This estimation is based on the Arrhenius equation used to calculate thermal acceleration factors for NAND devices [35].

6.1 Overview

In this chapter, we propose an efficient *on-line* data recovery technique, called *Flash Defibrillator* (FD), which can be effective for recent NAND flash memory [38]. The proposed FD technique is motivated by our observations on the characteristics of retention-failed NAND cells in recent 20-nm node NAND flash memory. The key finding is that when read operations are repeated, highly-damaged cells (that probably contributed to retention failures) are more likely to experience abnormal charge-transient behavior (e.g., random charge fluctuation [39] or charge detrapping [40]). Since the abnormal charge-transient behavior of NAND cells (under repeated reads) were rarely observed in 3x-nm NAND flash memory, the existing technique such as RFR (which was developed for 3x-nm NAND flash memory) cannot adequately handle retention errors from this new charge movement phenomenon. The proposed FD takes this behavior (as well as retention loss) into account of recovering retention-failed cells, thus resulting in a more efficient on-line data recovery solution.

The proposed FD technique consists of two main steps, a diagnostic step and a post-processing step. In the diagnostic step of FD, as done in RFR [37], a sequence of diagnostic pulses (i.e., effectively read operations) is applied to NAND cells. The main goal of the diagnostic step is to recharge retention-loss cells so that these cells can be read at the correct state. Since diagnostic pulses add extra charges to NAND cells, a threshold voltage (*Vth*) distribution tends to shift to the right after the diagnostic step, thus making some of retention-failed cells be recovered.

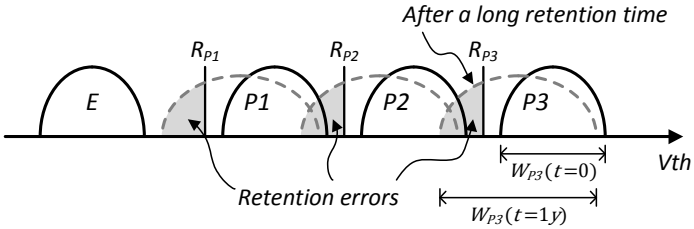
In the following post-processing step, FD identifies retention-failed cells as ones whose *Vth* states were shifted to the right. This heuristic, as used in RFR [37], reversely exploits the retention-loss mechanism in that highly retention-loss cells are more likely to be recharged with a low voltage. Furthermore, in order to avoid the negative effect of the abnormal charge movements on the FD's recovery capability, FD identifies retention-failed data in a progressive fashion using a selective error-correction procedure. The selective error-correction procedure, which identifies retention-recovered cells as early as possible, is based on a simple but effective heuristic: *If a NAND cell* ***c*** *is shifted to a higher Vth state after the diagnostic step, the cell* ***c*** *is identified as a retention-failed cell and its Vth state is corrected to the higher Vth state.* As soon as the cell ***c*** is corrected by our heuristic, it is no longer considered in the remaining steps of FD. Although our heuristic seems to be very simple, it is quite effective in handling abnormal charge movements (after the diagnostic step) observed in recent NAND flash memory, thus significantly improving FD's data recovery capability over RFR. The result of the post-processing step is stored to an internal buffer. If bit errors of the buffered data can be fully corrected by ECC, FD completes its recovery procedure and the fully recovered data are rewritten to a free page. Otherwise, two FD steps are repeated to the buffered data. After a pre-set maximum iteration count is reached, FD stops the recovery procedure.

6.2 Motivation

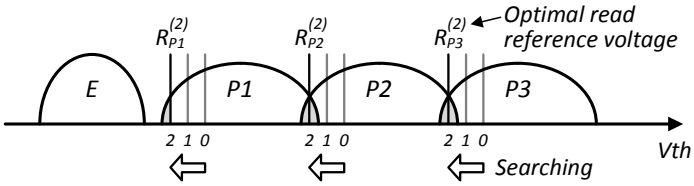
6.2.1 Limitations of the Existing Retention-Error Management Policy

When NAND flash memory is programmed and left for a long time, retention errors may occur due to retention loss. Figure 33(a) illustrates an example of *Vth*-distribution changes after 3K P/E cycling and a 1-year retention time. Since the overall *Vth* distributions shift down after a long retention time, a lot of bit errors may occur when the initial read reference voltages (R_{Pi}'s) are used in a read operation. If the number of bit errors exceeds the error-correction capability (e.g., 40 bits per 1 KB for an MLC device [6]) of ECC, a read-retry procedure is invoked to manage retention errors [27]. Read retry is a searching algorithm for the optimal read reference voltage, which iteratively performs read-and-check routines with different read reference voltages until all the bit errors are corrected. For example, as shown in Figure 33(b), read retry was performed two times to find the optimal read reference voltages ($R_{Pi}^{(2)}$'s).

However, if NAND flash memory is left beyond the specified retention time, the stored data cannot be retrieved even with read retry. This is because read retry cannot *actively* reduce bit errors, but *just find* the optimal read reference voltages where the number of bit errors can be minimized for given *Vth* distributions. Since retention loss tends to cause shifting the overall *Vth* distributions as well as widening W_{Pi}'s, after a long retention time, two adjacent *Vth* distributions may overlap each other. For example, as shown in Figure 33(a), since $W_{P3}(t{=}1y)$ after 1-year retention time gets



(a) An example of *Vth*-distribution changes due to retention loss.



(b) An example of a read retry procedure to find the optimal read reference voltage.

Figure 33: Examples of a NAND retention-error management policy.

wider than $W_{P3}(t{=}0)$, the $P3$ state is overlapped with the $P2$ state. As a result, remaining bit errors cannot be further reduced by read retry as shown in Figure 33(b). If there are more bit errors than the error-correction capability at the optimal read reference voltages, there is no way of retrieving the stored data with the existing error management policy.

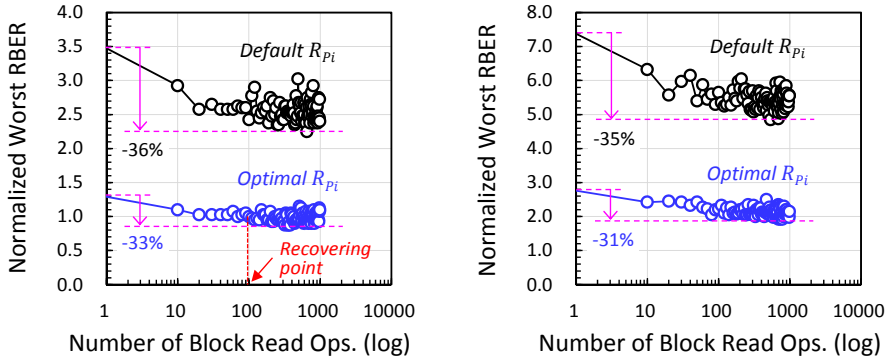
6.2.2 Limitations of the Existing Retention-Failure Recovery Technique

Before we describe the proposed `FD` technique in detail, we present our evaluation results of an existing data recovery technique for recovering retention failed cells in recent 20-nm node NAND flash memory. For our evaluation, we used the data retention-error recovery pulse (`DRRP`) tech-

nique [8], which was considered as one of the most effective data recovery techniques for 3x-nm NAND chips. As will be discussed below, our evaluation results strongly suggest a need for better data recovery techniques for recent 20-nm node (or below) NAND flash memory, which was the main motivation for developing our proposed FD technique.

In order to recover retention-failed cells, DRRP repeatedly applies weak-stress pulses (e.g., 3 V [37]) to retention-failed cells so that the *Vth*'s of retention-failed cells can be recovered to their original *Vth* state. Measurement results with 3x-nm node NAND chips showed that DRRP could reduce the RBER of severely retention-failed cells (who experienced 3K P/E cycling and 3 days' baking at $85\,^{\circ}\mathrm{C}$) by 56%, on average, after applying 500 weak-stress pulses [37].

However, our measurement results show that the effectiveness of DRRP as an *on-line* recovery solution is quite limited because its data recovery process is very slow for recent 20-nm node (more advanced technology over 3x-nm node by about two generations) NAND flash memory. Since applying a weak-stress pulse is not allowed in our test environment, we used *read operations* (which can apply the read voltage of about 6 V) instead of the weak-stress pulse. Figure 34(a) shows worst-case RBER (i.e., the RBER of a 1-KB sector which has the highest number of bit errors) variations over different numbers of read operations after 3K P/E cycling and 2-year retention times. The measured RBERs were normalized over the maximum error-correction capability of ECC. We denote the normalized worst-case RBER by W-RBER. When the default R_{Pi}'s were used, DRRP could reduce W-RBER by 36%. However, it could not lower W-RBER below 1.00.



(a) W-RBER variations in the 2-year retention time condition.

(b) W-RBER variations in the 8-year retention time condition.

Figure 34: Normalized worst-case RBER (W-RBER) variations over varying numbers of read operations under DRRP.

On the other hand, when the optimal R_{Pi}'s were used, DRRP could reduce W-RBER below 1.00. However, this reduction was reached after 100 read operations. If the average page read time is 100 μs, for example, it takes about 10 ms for each NAND page to be recovered, which is too slow to be employed as an on-line run-time technique.

Moreover, the data recovery capability of DRRP is quite restricted in recent NAND flash memory. Figure 34(b) shows W-RBER changes with varying numbers of read operations after 8-year retention times (8x longer than the specified retention time). When the optimal R_{Pi}'s were used, DRRP could reduce W-RBER by up to 31%, however, W-RBER was not reduced below 1.00 until 1000 read operations. In the 4-year retention case, DRRP still could not reduce W-RBER below 1.00. These measurement results show that DRRP can recover retention-failed data which experienced up to 2x longer retention time than the specified retention time.

Our evaluation results show that DRRP is less effective with recent 20-nm NAND flash memory in recovering retention-failed data and its recovering speed is very slow. Our main goal was to improve DRRP so that it can be as effective with 20-nm NAND chips as with 3x-nm NAND chips while its recovering speed is fast enough so that it can be used as an on-line run-time solution.

6.3 Retention Error Recovery Technique

In this section, we describe a charge movement model which can capture abnormal charge-transient behavior observed in recent 20-nm node (or below) NAND flash memory. Based on the charge movement model, we present a selective error-correction procedure and the implementation of our proposed FD technique in detail.

6.3.1 Charge Movement Model

Since applying multiple read pulses can partially recharge retention-loss cells, *Vth*'s of these cells can shift to the right [37]. On the other hand, it is reported that as a side-effect of recent advanced NAND technologies, another type of charge loss may occur due to multiple read pulses [40] so that *Vth*'s of some highly-damaged cells can shift to the left. If these abnormal charge-loss components are not negligible, the effectiveness of the recharging process can be substantially cancelled. Furthermore, it is also reported that in recent advanced NAND cells, *Vth*'s of some weak cells may randomly fluctuate because several charges are periodically trapped

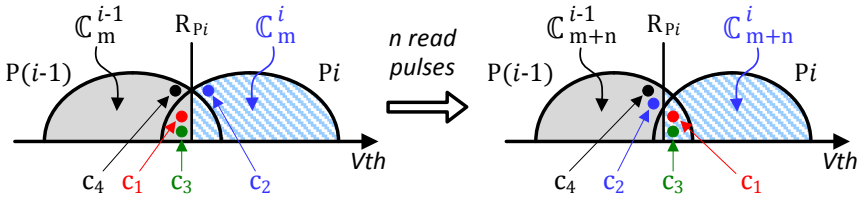


Figure 35: An example of charge movement after n read pulse applications.

and detrapped due to the random telegraph noise (RTN) effect [39]. These randomly-fluctuated components may negatively affect the recharging process.

In order to understand how read pulses affect NAND cells, we built a simple charge movement (CM) model. Figure 35, the CM model can be expressed based on this set definition. We denote $\mathbb{C}^{i}_{m}$ as a set of cells that are read as the i^{th} state after the m^{th} read pulse. For example, in Figure 35, $\mathbb{C}^{i-1}_{m} = \{c_1, c_3, c_4, \cdots\}$ while $\mathbb{C}^{i}_{m} = \{c_2, \cdots\}$. After the n read pulses are applied, if the read value of a cell c_1 changes from $P(i-1)$ to Pi (for example, because of recharging), we say the cell c_1 belongs to the set $\mathbb{C}^{(i-1)\rightarrow i}$. That is,

$$c_1 \in \mathbb{C}^{i-1}_{m} \cap \mathbb{C}^{i}_{m+n} = \mathbb{C}^{(i-1)\rightarrow i}. \tag{6.1}$$

On the other hand, if the read value of a cell c_2 changes from Pi to $P(i-1)$ (for example, because of charge detrapping), we say the cell c_2 belongs to the set $\mathbb{C}^{i\rightarrow(i-1)}$. That is,

$$c_2 \in \mathbb{C}^{i}_{m} \cap \mathbb{C}^{i-1}_{m+n} = \mathbb{C}^{i\rightarrow(i-1)}. \tag{6.2}$$

Finally, if the read value of a cell c_3 periodically changes between $P(i-1)$

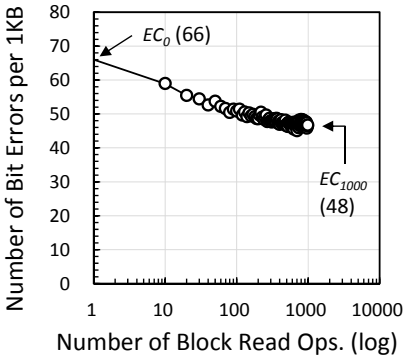
and Pi (for example, because of random charge fluctuation), we say the cell c_3 belongs to the set $\mathbb{C}^{(i-1)\leftrightarrow i}$. That is,

$$\begin{aligned} c_3 &\in \mathbb{C}_m^{i-1} \cap \mathbb{C}_{m+n}^{i} \cap \mathbb{C}_{m+2n}^{i-1} \cap \cdots \\ &= \left[\bigcap_{k=0} \mathbb{C}_{m+2k\cdot n}^{i-1}\right] \cap \left[\bigcap_{k=0} \mathbb{C}_{m+(2k+1)\cdot n}^{i}\right] = \mathbb{C}^{(i-1)\leftrightarrow i}. \end{aligned} \tag{6.3}$$

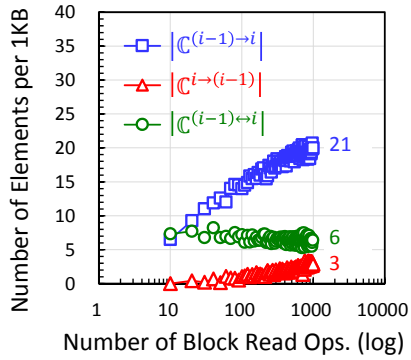
After applying the m^{th} read pulse, since the number EC_m^{DRRP} of bit errors under DRRP decreases by the number of recharged cells while it increases by the number of additionally detrapped cells, EC_m^{DRRP} can be expressed as follows:

$$EC_m^{DRRP} = EC_0 - |\mathbb{C}^{(i-1)\rightarrow i}| + |\mathbb{C}^{i\rightarrow(i-1)}|, \tag{6.4}$$

where EC_0 is the initial number of bit errors before applying read pulses. This estimation is based on the assumption that the *Vth* states of the upper-tail cells (e.g, a cell c_4 in Figure 35) in a widened *Vth* distribution (due to retention loss) rarely change from $P(i-1)$ to Pi after read pulse applications. When retention loss occurs, a *Vth* distribution gets widen as shown in Figure 33(a), which reflects that the lower-tail cells are more likely to lose charges over the upper-tail cells. As a result, the upper-tail cells have a much lower probability to be recharged over the lower-tail cells so that their effect on the error-correction process can be ignored. Moreover, it is not necessary to include the number $|\mathbb{C}^{(i-1)\leftrightarrow i}|$ of randomly-fluctuated cells in EC_m^{DRRP} because DRRP does not distinguish the set $\mathbb{C}^{(i-1)\leftrightarrow i}$ from the set $\mathbb{C}^{(i-1)\rightarrow i}$ or the set $\mathbb{C}^{i\rightarrow(i-1)}$.



(a) Variations of the total bit-error count over varying the numbers of read operations.



(b) Variations of the numbers of each CM element over varying the numbers of read operations.

Figure 36: Measurement results for tracing the CM-component changes in response to multiple read pulses.

In order to trace the overall trend of CM-element changes in response to read pulses, we measured the average number of each CM element (per 1-KB unit) every ten read pulses. Figure 36(a) shows EC^{DRRP} variations over varying numbers (e.g., $m = 0 \sim 1000$) of read operations after 3K P/E cycling and the 8-year (equivalent) retention times. In this example, EC_0 was 66 while EC_{1000} after 1,000 read pulses is reduced to 48. Figure 36(b) shows measurement results for each CM element, which can explain the cause of retention-error reductions as shown in Figure 36(a). As read operations are repeated, $|\mathbb{C}^{(i-1)\to i}|$ grows rapidly in the early stage (i.e., $\sim$100 read pulses) but slowly at the end. (Note that the x-axis of Figure 36(b) is a log scale.) On the other hand, $|\mathbb{C}^{i\to(i-1)}|$ grows slowly from beginning to the end. Since the differences between $|\mathbb{C}^{(i-1)\to i}|$ and $|\mathbb{C}^{i\to(i-1)}|$ are nearly saturated after one thousand read pulses, further read pulses have little effect on reducing EC_m^{DRRP}. Measurement result of the total bit-error count (e.g.,

$EC_{1000}^{DRRP} = 48$) almost matched the estimation (i.e., $66 - 27 + 9 = 48$) from Equation 6.4. An interesting observation is that $|\mathbb{C}^{(i-1)\leftrightarrow i}|$ starts with non-zero counts which is comparable with $|\mathbb{C}^{(i-1)\rightarrow i}|$ in the early stage. However, since DRRP expects only $\mathbb{C}^{(i-1)\rightarrow i}$ elements after multiple read pulses, $\mathbb{C}^{(i-1)\leftrightarrow i}$ elements (as well as $\mathbb{C}^{i\rightarrow(i-1)}$ elements) are not considered in its error-correcting process.

6.3.2 A Selective Error-Correction Procedure

By progressively taking CM elements into account of a data recovery process, the proposed FD can more efficiently recover retention failures over DRRP. Since non-zero $|\mathbb{C}^{i\rightarrow(i-1)}|$ indicates the occurrence of additional charge loss during the recovery process, if those elements can be identified from the read data, the data recovery capability can be enhanced. Moreover, since random charge fluctuation is more active in highly-damaged cells [39] (which probably contributed to retention errors [16]), taking $\mathbb{C}^{(i-1)\leftrightarrow i}$ elements as retention-failed cells can be an effective way of correcting retention errors. Another important advantage of considering $\mathbb{C}^{(i-1)\leftrightarrow i}$ elements is that the data recovery speed can be accelerated. Since $\mathbb{C}^{(i-1)\leftrightarrow i}$ elements are frequently observed even in the early stage of the recovery process as shown in Figure 36(b), if the error-correction process can consider these elements, the error-correction capability nearly doubles in the early stage. Since each CM element can be separately extracted from the read data as shown in Figure 36(b), conceptually, the total number EC_m^{FD} of bit errors

under FD after the m^{th} read pulse can be expressed as follows:

$$EC_m^{FD} = EC_m^{DRRP} - |\mathbb{C}^{i\rightarrow(i-1)}| - |\mathbb{C}^{(i-1)\leftrightarrow i}| \tag{6.5}$$

For example, if $|\mathbb{C}^{(i-1)\rightarrow i}|$, $|\mathbb{C}^{i\rightarrow(i-1)}|$ and $|\mathbb{C}^{(i-1)\leftrightarrow i}|$ are 21, 3 and 6, respectively, after 1000 read pulses as shown in Figure 36(b), EC_{1000}^{DRRP} is 48 ($= 66 - 21 + 3$) while EC_{1000}^{FD} is only 39 ($= 66 - 21 - 6$). In this example, DRRP reduces retention errors by 27% while FD reduces retention errors by 41%.

6.3.3 Implementation

Based on the charge movement model, we have implemented FD with the selective error-correction procedure. Figure 37 shows an overview of the current FD implementation which consists of two main steps, a diagnostic step and a post-processing step.

In the diagnostic step, a sequence of diagnostic pulses is applied to retention-failed cells. The main role of the diagnostic step is two-fold. First, it recharges retention-loss cells (same as DRRP [37]). Second, it senses the *Vth* changes in response to diagnostic pulses for the following post-processing step. In order to achieve these two functions at the same time, we use a read operation as a diagnostic pulse. Since a read operation senses the data of a selected page while it applies the read voltage (e.g., $\sim$6 V) to unselected pages in a NAND block, when read operations are sequentially executed to all of pages in a block, recharging the unselected pages and sensing the selected page can be executed in a pipelined fashion. Since the effect

of just one read pulse on recharging may not be noticeable for causing *Vth* changes, it is more efficient to use a sequence of (consecutive) read pulses as a unit operation of the diagnostic step. For example, ten consecutive read pulses are required to cause *Vth* changes in our measurements. On the other hand, in order to detect randomly-fluctuated cells (i.e., cells in $\mathbb{C}^{(i-1)\leftrightarrow i}$) as early as possible, the post-processing step is invoked after every read operations in the early stage (e.g., less than one hundred read pulses) of `FD`. If the number of consecutive read pulses is conditionally changed (we call this policy the *variable-length sequence policy*), although the data recovery capability may not be improved, the data recovery speed may be substantially enhanced.

In the following post-processing step, `FD` identifies retention-loss cells by a selective error-correction procedure so that retention errors can be pro-

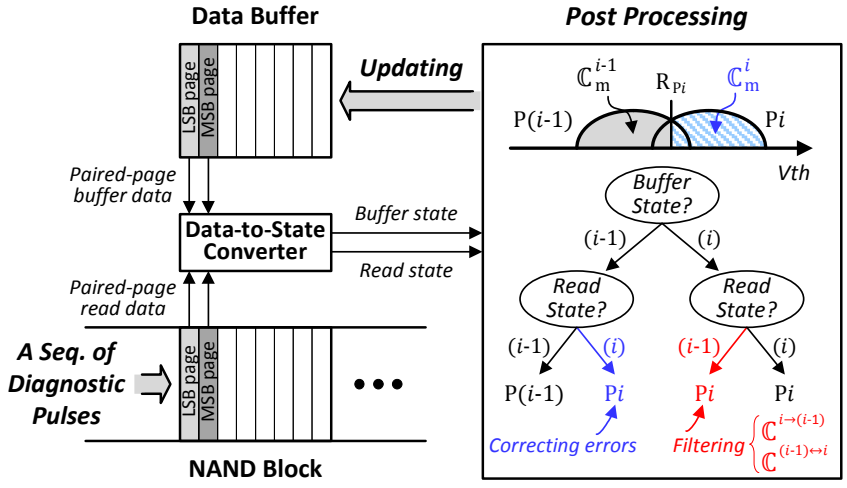


Figure 37: An overview of the current `FD` implementation with a selective error-correction procedure.

gressively corrected. Since CM is based on *Vth* states as presented in Section 6.3.1, it is necessary to convert raw data to *Vth* states before the post-processing step as shown in Figure 37. The selective error-correction procedure is based on simple, but effective heuristics: *(1) When a buffer state is $P(i)$, if the corresponding read state is $P(i-1)$ or Pi, then the buffer state is not changed. On the other hand, (2) when a buffer state is $P(i-1)$, if the corresponding read state is $P(i)$, then the buffer state is changed to $P(i)$* . The first heuristic can avoid the negative impacts of *Vth*-decreased cells (i.e., cells in $\mathbb{C}^{i\rightarrow(i-1)}$ or $\mathbb{C}^{(i-1)\leftrightarrow i}$) on correcting retention errors. On the other hand, the second heuristic takes *Vth*-increased cells (i.e., cells in $\mathbb{C}^{(i-1)\rightarrow i}$ or $\mathbb{C}^{(i-1)\leftrightarrow i}$) as retention-failed cells so that retention errors can be selectively corrected. In `FD`, once a retention-failed cell is corrected by the second heuristic, then the corrected cell is no longer considered in the remaining post-processing steps by the first heuristic. However, since `DRRP` takes only a cell belongs to a set $\{\mathbb{C}_0^{i-1} \cap \mathbb{C}_m^i\}$ (after the m^{th} read pulse) as a retention-failed cell regardless of the error-correction history, `DRRP` cannot properly handle the negative impacts of *Vth*-decreased cells on its data recovery capability.

The result of the post-processing step is updated to the data buffer as shown in Figure 37 so that retention errors in the buffer can be progressively corrected. If the buffered data is correctable by ECC, `FD` completes its recovery procedure and rewrites the recovered data to a free page. Otherwise, two `FD` steps are repeated until a pre-set maximum iteration count (e.g., one thousand) is reached.

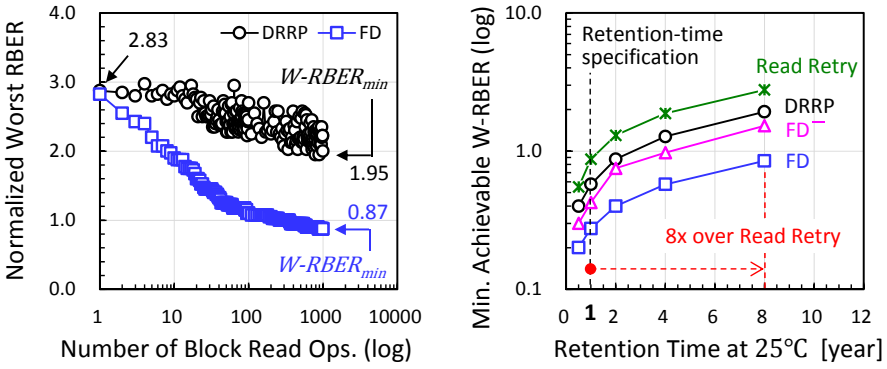
Our proposed `FD` implementation as shown in Figure 37 requires a data

buffer with a single block size (e.g., 1 MB in an MLC device) and several state registers with a single page size (e.g., 8 KB). Moreover, since the post-processing step and the diagnostic step can be performed independently for each other, FD can exploit a pipelined execution between the diagnostic step and the post-processing step so that the total FD execution time can be partially reduced.

6.4 Experimental Results

We evaluated the effectiveness of FD for recovering retention failures with ten blocks (pre-cycled for 3K P/E cycles) out of five 20-nm node NAND chips. As a main evaluation metric, we measured RBERs of about 10,000 sectors and computed W-RBER (i.e., the normalized worst-case RBER as defined in Section 6.2.2) among measured sectors. In order to emulate a long retention state such as a 2-year retention time condition, we baked selected chips at 100 °C for a equivalent retention time (e.g., 4 hours) estimated by the Arrhenius equation [35].

In order to compare the data recovery capability of DRRP and FD, we measured W-RBER while varying the number of read pulses in a very long range (without applying the stopping condition of the error-correction procedures). Figure 38(a) shows the data recovery capability of both techniques in the 8-year retention case. Since DRRP cannot lower its W-RBER even with 1,000 read pulses, it cannot recover retention-failed data under the 8-year retention condition. On the other hand, FD can recover retention-failed data under the same retention condition after about 360 read pluses. In order



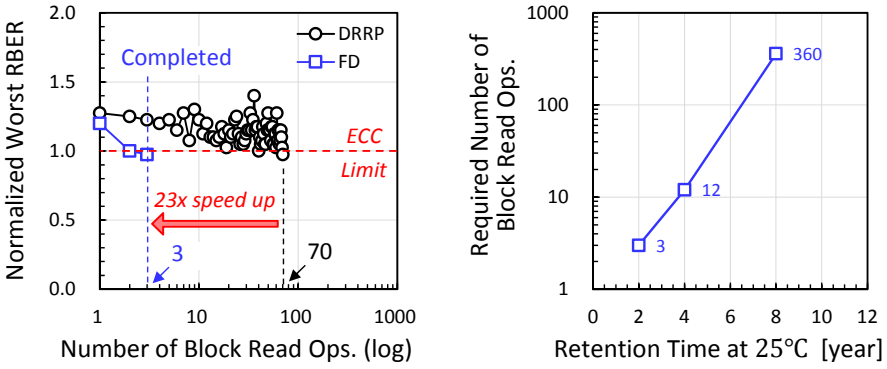
(a) W-RBER variations over varying the numbers of read operations in the 8-year retention time condition.

(b) $W\text{-}RBER_{min}$ variations over varying retention time condition under different techniques.

Figure 38: Comparisons of the data recovery capability under different data recovery techniques.

to compare the data recovery capability of various techniques under varying retention time conditions, we computed the minimum achievable W-RBER, denoted as $W\text{-}RBER_{min}$, of each technique for a given retention condition. For example, in Figure 38(a), $W\text{-}RBER_{min}$ of `FD` is 0.87 while $W\text{-}RBER_{min}$ of `DRRP` is 1.95. Intuitively, $W\text{-}RBER_{min}$ indicates the maximum data recovery power of a given technique. Figure 38(b) shows $W\text{-}RBER_{min}$ variations under different retention time conditions for several different techniques. As shown in Figure 38(b), `FD` can effectively extend the NAND retention capability by up to 8 years (which is eight times longer than the retention-time specification) while `DRRP` can guarantee only 2-year retention times.

The enhanced error-correction capability of `FD` over `DRRP` mainly comes from the selective error-correction procedure which can efficiently identify retention-loss cells by finely distinguishing $\mathbb{C}^{i\to(i-1)}$ and $\mathbb{C}^{(i-1)\leftrightarrow i}$ elements



(a) W-RBER variations over varying the number of read operations in the 2-year retention time condition.

(b) Required numbers of read operations to complete FD over different retention-time conditions.

Figure 39: Comparisons of the data recovery speed between DRRP and FD.

as explained in Section 6.3.2. In order to understand the impact of the fine-grained cell classification on the data recovery capability, we disabled the $\mathbb{C}^{(i-1)\leftrightarrow i}$ identification step from FD. We denote this modified FD technique by $\texttt{FD}^{-}$. The only difference between DRRP and $\texttt{FD}^{-}$ is for $\texttt{FD}^{-}$ to filter cells in $\mathbb{C}^{i\rightarrow(i-1)}$. As shown in Figure 38(b), DRRP can extend the NAND retention capability by up to 2 years. On the other hand, $\texttt{FD}^{-}$ can extend the NAND retention capability by up to 4 years while FD can extend it by up to 8 years. This result indicates that identifying cells in $\mathbb{C}^{(i-1)\leftrightarrow i}$ in the data recovery procedure significantly strengthens the data recovery capability of FD over $\texttt{FD}^{-}$.

In order to compare the data recovery speed of DRRP and FD, we tested both techniques under three different retention conditions. Figure 39(a) shows the data recovery speed of DRRP and FD in the 2-year retention condition. In DRRP, W-RBER slowly decreases as read pulses are repeated, and all the

Table 10: Required numbers of read pulses to complete FD.

Retention time	Variable-length sequence policy	Fixed-length sequence policy
2 years	3	10
4 years	12	30
8 years	360	370

retention errors are corrected (i.e., W-RBER $\leq$ 1.0) after applying 70 read pulses. On the other hand, in FD, retention errors can be fully corrected only after 3 read pulses. Once all the data are correctable, FD is completed. As a result, FD can recover retention failures up to 23x faster over DRRP for the 2-year retention case. When the average page read time is, for example, 100 μs, it takes about 7 ms for DRRP to recover retention failures while only 300 μs is required for FD. In order to further compare the data recovery speed in longer retention cases, we performed additional experiments for 4-year and 8-year retention conditions. As shown in Figure 39(b), in the 4-year and 8-year retention cases, FD can successfully recover retention failures after applying 12 and 360 read pulses, respectively. On the other hand, in both cases, DRRP could not recover retention failures until 1,000 read pulses. (In our evaluations, the maximum number of read pulses was set to 1,000.)

We also evaluate if *the variable-length sequence policy* (described in Section 6.3.3) is effective in speeding up the overall data recovery procedure. Under the variable-length sequence policy, until the total number of read pulses reaches 100, a single diagnostic pulse is applied to NAND cells between consecutive post-processing step. Once the total number of

read pulses reaches 100, ten consecutive read pulses are applied in a row between consecutive post-processing step. In order to evaluate the effectiveness of the variable-length sequence policy, we compared it with the *the fixed-length sequence policy* (which always applies ten consecutive read pulses at a time). As summarized in Table 10, in the 2-year and 4-year retention cases, the variable-length sequence policy can reduce the total data recovery time by 70% and 60%, respectively, over the fixed-length sequence policy. This is because, in an early stage of FD, frequently reading retention-failed cells can increase the probability of detecting cells in $\mathbb{C}^{(i-1)\leftrightarrow i}$ so that they can be excluded quickly from the remaining data recovery procedure. However, in the 8-year retention case, the variable-length sequence policy has a little benefit over the fixed-length sequence policy. This is because the main advantage of the variable-length sequence policy is to detect cells in $\mathbb{C}^{(i-1)\leftrightarrow i}$ early. For severely retention-failed data such as the 8-year retention case, after most of cells in $\mathbb{C}^{(i-1)\leftrightarrow i}$ are detected early, other components such as $\mathbb{C}^{(i-1)\rightarrow i}$ (not yet classified) are the dominant source of the retention errors. As a result, the overall recovery time of FD is decided by how long it takes to find cells in $\mathbb{C}^{(i-1)\rightarrow i}$ (which is similar under two polices).

Our experimental results with 20-nm node NAND chips show that FD can recover retention failures up to 23x faster over the existing DRRP technique. Furthermore, since FD can recover severely retention-failed data, it effectively extends the NAND retention time. Our result indicates that the NAND retention time can be effectively extended by up to 8x over the specified retention time.

Chapter 7

Conclusions

7.1 Summary and Conclusions

The cost-per-bit of NAND flash-based solid-state drives (i.e., SSDs) has steadily improved through uninterrupted semiconductor process scaling and multi-leveling so that they are how widely employed in not only mobile embedded systems but also personal computing systems. However, the limited lifetime of NAND flash memory, as a side effect of recent advanced device technologies, is emerging as one of the major concerns for recent high-performance SSDs, especially for datacenter applications.

In this dissertation, we proposed several cross-layer optimization techniques to improve the lifetime (particularly endurance) of NAND flash memory. Although the performance and reliability requirements of NAND flash memory are designed under the worst-case operating conditions of a storage product, the maximum capabilities of NAND devices are not fully utilized in most cases. This observation has motivated us to propose a versatile device-level framework (i.e., DeVTS), including a NAND endurance model and newly defined device setting interfaces, that allows a flash software to exploit the tradeoffs between the endurance and performance/retention capabilities of NAND flash memory.

We have developed several SSD lifetime improvement techniques based

on the DeVTS framework that supports various erase scaling modes and write capability tuning modes, each of which has a different impact on NAND endurance. By accurately predicting the NAND requirements of write requests, our proposed techniques optimally tune the performance and retention capabilities of NAND devices. We have implemented dvsFTL+, based on the DeVTS framework and proposed lifetime improvement techniques, that dynamically selects erase voltage/time scaling modes and write performance/retention capability tuning modes depending on varying workload conditions. The existing garbage collector and wear leveler are also redesigned to maximize the efficiency of dvsFTL+. Since the performance and retention capabilities of NAND devices are frequently relaxed, dvsFTL+ manages the NAND requirements in a reliable fashion.

In order to evaluate the effectiveness of the proposed lifetime improvement techniques, we have built a timing-accurate NAND simulation environment which accurately emulates temporal interactions between varying I/O requests and various NAND operations. Our experimental results show that when the write-performance tuning technique is employed, NAND endurance is improved by 62% on average. When the retention-capability tuning technique is added to dvsFTL+, NAND endurance is further improved by 94%, on average, over an existing DeVTS-unaware FTL. In our evaluation, the overall write performance and retention requirements of storage systems are reliably maintained.

Since our proposed lifetime improvement techniques aggressively tune down the retention capability of NAND flash memory, data loss may occur due to retention failures when power is suddenly cut off. Consequently, we

have suggested a new data recovery technique to recover corrupted data from retention failures by exploiting the unique retention loss mechanism of NAND flash memory. Our experimental results show that our proposed data recovery technique can recover from retention failures up to 23x faster over the existing recovery technique. Furthermore, it effectively extends the NAND retention time by up to 8x over the specified retention time.

Since the proposed lifetime improvement techniques and reliability management techniques require only a small resource overhead and a negligible time overhead, they can easily be implemented into the existing NAND flash-based storage systems with minimal changes in flash software modules.

7.2 Future Work

7.2.1 Lifetime Improvement Technique Exploiting The Other NAND Tradeoffs

The lifetime improvement techniques in this dissertation take advantage of variations in the write performance and retention requirements. However, if variations in the maximum required number of read counts for each NAND page is additionally exploited, NAND endurance can be further improved. For example, if the maximum read count of an MLC NAND block is reduced from 1,000K [41] to 1,000, the *Vth* window can be additionally saved by about 500 mV. Since the saved *Vth* window by retention-capability tuning is about 500 mV, the effect of read-disturb resistance tuning on improving NAND endurance will be comparable to that of retention-capability

tuning.

However, unlike the performance and retention capability tuning techniques, there is a challenging issue in the implementation of a read-disturb resistance tuning technique. Since the proposed performance and retention capability tuning techniques exploit the spatial and temporal locality of write requests, it is possible to accurately predict the characteristics of the near-future write requests. On the contrary, it is difficult to predict the future read intensity of the current write request in a storage software layer. In order to decide whether or not the read-disturb resistance of the current write request can be relaxed, it is necessary to exploit more higher-level hints from file systems or applications. If such useful information for the future read intensity of write requests can be exploited, the endurance gain of the proposed techniques is maximized.

7.2.2 Development of Extended Techniques for DRAM-Flash Hybrid Main Memory Systems

As big data analytics based on massive data, rapidly generated and processed, become commonplace in real environments, there is a strong demand on high-performance computing systems that can efficiently store and process such massive data in real time. The most critical requirement on the next-generation information systems, such as intelligent self-driving control systems, based on the big data analytics is to keep extremely high performance in a consistent manner. In order to satisfy such a requirement, most of existing optimization techniques have mainly focused on *in-memory pro-*

cessing that can prevent from accessing to slow storage systems. The existing DRAM-based main memory system, however, is not a practical solution for such big data applications because of its pool cost/energy efficiencies. In order to implement a cost-efficient main memory system with a huge capacity as well as low power consumption, several system-level approaches have been suggested by taking advantage of both DRAM and NAND flash memory through a new software architecture [42] or hardware architecture [43]. However, the limited lifetime of NAND flash memory can be also a serious reliability issue when such DRAM-Flash hybrid main memory systems are actively employed in real environments.

If the operating systems can directly manage the proposed lifetime improvement techniques by exploiting various new interactions between DRAM and NAND flash under an NVDIMM-like setting, it is possible to extend the lifetime of NAND flash to the fullest extent. For example, by exploiting many useful hints, disappeared while passing through I/O stacks, in the host system, the performance and retention requirements of the requests can be more reliably and directly classified. Moreover, since our proposed techniques can easily be combined with existing data reduction techniques such as data compression and data de-duplication, NAND lifetime can be further extended. Our proposed lifetime improvement techniques can be a crucial breakthrough in the new type of main memory systems.

7.2.3 Development of Specialized SSDs

Recently, in order to optimally exploit the unique superiorities (e.g., non-volatility, high write throughput, and low access latency) of NAND de-

vices, several types of specialized SSDs are required in datacenter environments [44]. For example, when SSDs are used as cache, lower latency as well as higher endurance is needed. On the other hand, when SSDs are used as a *cold storage*, a higher capacity with a longer retention time is more preferable. However, existing SSD products do not fulfil such various requirements in a single device because most of capabilities of NAND flash memory usually fixed during device design times. In order to meet such requirements from datacenter applications, it is required to develop a multi-purpose SSD whose capabilities can be flexibly adjusted on demand.

The primary goal of this dissertation is to improve NAND endurance by conditionally tuning down the other NAND capabilities. In order to achieve this research goal, we propose the NAND endurance model which accurately captures the tradeoff relationship among the NAND capabilities. Since the relationship between each NAND capability is expressed as the saved *Vth* window by each tuning technique, the proposed NAND endurance model can be utilized for other purpose such as booting the write performance or retention capability of a storage device. For example, when urgent write requests are issued to a storage system, the write performance of NAND devices can be rapidly boosted by temporarily sacrificing the endurance and retention capabilities of NAND devices. Similarly, when cold data are to be written, the retention capability of NAND devices can be further enhanced by sacrificing the write performance of NAND devices.

Bibliography

[1] C. Albrecht, A. Merchant, M. Stokely, M. Waliji, F. Labelle, N. Coehlo, X. Shi, and C. E. Schrock, "Janus: Optimal Flash Provisioning for Cloud Storage Workloads," in *Proceedings of the USENIX Annual Technical Conference* (ATC), 2013.

[2] Mario Sako Y. Watanabe, T. Nakajima, J. Sato, K. Muraoka, M. Fujiu, F. Kouno, M. Nakagawa, M. Masuda, K. Kato, Y. Terada, Y. Shimizu, M. Honma, A. Imamoto, T. Araya, H. Konno, T. Okanaga, T. Fujimura, X. Wang, M. Muramoto, M. Kamoshida, M. Kohno, Y. Suzuki, T. Hashiguchi, T. Kobayashi, M. Yamaoka, and R. Yamashita, "A Low-Power 64Gb MLC NAND-Flash Memory in 15 nm CMOS Technology," in *Proceedings of the IEEE International Solid-State Circuits Conference* (ISSCC), 2015.

[3] C. Kim, J. Ryu, T. Lee, H. Kim, J. Lim, J. Jeong, S. Seo, H. Jeon, B. Kim, I. Lee, D. Lee, P. Kwak, S. Cho, Y. Yim, C. Cho, W. Jeong, K. Park, J.-M. Han, D. Song, K. Kyung, Y.-H. Lim, and Y.-H. Jun, "A 21 nm High Performance 64 Gb MLC NAND Flash Memory with 400 MB/s Asynchronous Toggle DDR Interface," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 4, pp. 981–989, 2012.

[4] S.-H. Shin, D.-K. Shim, J.-Y. Jeong, O.-S. Kwon, S.-Y. Yoon, M.-H. Choi, T.-Y. Kim, H.-W. Park, H.-J. Yoon, Y.-S. Song, Y.-H. Choi, S.-W. Shim, Y.-L. Ahn, K.-T. Park, J.-M. Han, K.-H. Kyung, and Y.-

H. Jun, “A New 3-bit Programming Algorithm using SLC-to-TLC Migration for 8 MB/s High Performance TLC NAND Flash Memory,” in *the IEEE Symposium on VLSI Circuits Digest of Technical Papers* (VLSI), 2012.

[5] J. E. Brewer and M. Gill, “Nonvolatile Memory Technologies with Emphasis on Flash,” *Wiley*, 2008.

[6] A. A. Chien and V. Karamcheti, “Moore’s Law: The First Ending and a New Beginning,” *IEEE Computer Magazine*, vol. 46, no. 12, pp. 48–53, 2013.

[7] J.-W. Hsieh, T.-W. Kuo, and L.-P. Chang, “Efficient Identification of Hot Data for Flash Memory Storage Systems,” *ACM Transactions on Storage*, vol. 2, no. 1, pp. 22–40, 2006.

[8] F. Chen, T. Luo, and X. Zhang, “CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory Based Solid State Drives,” in *Proceedings of the USENIX Conference on File and Storage Technologies* (FAST), 2011.

[9] S. Lee, J. Park, K. Fleming, Arvind, and J. Kim, “Improving Performance and Lifetime of Solid-State Drives Using Hardware-Accelerated Compression,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1732–1739, 2011.

[10] S. Lee, T. Kim, K. Kim, and J. Kim, “Lifetime Management of Flash-Based SSDs Using Recovery-Aware Dynamic Throttling,” in *Proceed-*

ings of the USENIX Conference on File and Storage Technologies (FAST), 2012.

[11] Q. Wu, G. Dong, and T. Zhang, "Exploiting Heat-Accelerated Flash Memory Wear-Out Recovery to Enable Self-Healing SSDs," in *Proceedings of the USENIX Workshop on Hot Topics in Storage and File Systems* (HotStorage), 2011.

[12] R.-S. Liu, C.-L. Yang, and W. Wu, "Optimizing NAND Flash-Based SSDs via Retention Relaxation," in *Proceedings of the USENIX Conference on File and Storage Technologies* (FAST), 2012.

[13] S. Lee, K. Ha, K. Zhang, J. Kim, and J. Kim, "FlexFS: A Flexible Flash File System for MLC NAND Flash Memory," in *Proceedings of the USENIX Annual Technical Conference* (ATC), 2009.

[14] A. Cox, "JEDEC SSD Specifications Explained," Available: http://www.jedec.org.

[15] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Lifetime Improvement of NAND Flash-based Storage Systems Using Dynamic Program and Erase Scaling," in *Proceedings of the USENIX Conference on File and Storage Technologies* (FAST), 2014.

[16] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill, "Bit Error Rate in NAND Flash Memories," in *Proceedings of the International Reliability Physics Symposium* (IRPS), 2008.

[17] M. Kang, K.-T. Park, Y. Song, S. Hwang, B. Y. Choi, Y. Song, Y. Lee, and C. Kim, "Improving Read Disturb Characteristics by Self-Boosting Read Scheme for Multilevel NAND Flash Memories," *Japanese Journal of Applied Physics*, vol. 48, no. 4, pp. 04C062-1–04C062-6, 2009.

[18] K.-D. Suh, B.-H. Suh, Y.-H. Lim, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, J.-H. Choi, J.-R. Kim, and H.-K. Lim, "A 3.3 V 32 Mb NAND Flash Memory with Incremental Step Pulse Programing Scheme," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, 1995.

[19] K. F. Schuegraf and C. Hu, "Effects of Temperature and Defects on Breakdown Lifetime of Thin SiO2 at Very Low Voltages," *IEEE Transactions on Electron Devices*, vol. 41, no. 7, pp. 1227–1232, 1994.

[20] J. Jeong and J. Kim, "Dynamic Program and Erase Scaling in NAND Flash-based Storage Systems," *Technical Report, Seoul National University*, cares.snu.ac.kr/download/TR-CARES-01-14, 2014.

[21] S. Cho, "Improving NAND Flash Memory Reliability with SSD Controllers," in *Proceedings of the Flash Memory Summit*, 2013.

[22] JEDEC standard, "Stress-Test-Driven Qualification of Integrated Circuits," JESD47H.01, 2011.

[23] D. W. Lee, S. Cho, B. W. Kang, S. Park, B. Park, M. K. Cho, K.-O. Ahn, Y. S. Yang, and S. W. Park, "The Operation Algorithm for Im-

proving the Reliability of TLC (Triple Level Cell) NAND Flash Characteristics," in *Proceedings of the IEEE International Memory Workshop* (IMW), 2011.

[24] S. Lee, J. Park, and J. Kim, "FlashBench: A Workbench for a Rapid Development of Flash-based Storage Devices," in *Proceedings of the IEEE International Symposium on Rapid System Prototyping* (RSP), 2012.

[25] M. Jung and M. Kandermir, "Revisiting Widely Held SSD Expectations and Rethinking System-Level Implications," in *Proceedings of the ACM SIGMETRICS*, 2013.

[26] L.-P. Chang, "On Efficient Wear Leveling for Large-Scale Flash-Memory Storage Systems," in *Proceedings of the ACM Symposium on Applied Computing*, 2007.

[27] J. Yang, "High-Efficiency SSD for Reliable Data Storage Systems," in *Proceedings of the Flash Memory Summit*, 2011.

[28] R. Frickey, "Data Integrity on 20 nm NAND SSDs," in *Proceedings of the Flash Memory Summit*, 2012.

[29] IBM, "Kernel APIs, Part 3: Timers and Lists in the 2.6 Kernel," http://www.ibm.com/developerworks/library/l-timers-list/.

[30] D. Narayanan, A. Donnelly, and A. Rowstron, "Write Off-Loading: Practical Power Management for Enterprise Storage," in *Proceedings*

of the USENIX Conference on File and Storage Technologies (FAST), 2008.

[31] http://www.ubuntu.com/download

[32] J. Lee, Y. Kim, G. M. Shipman, S. Oral, and J. Kim, "Preemptible I/O Scheduling of Garbage Collection for Solid State Drives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 2, pp. 247–260, 2013.

[33] K. Bang, K.-I. Im, D.-G. Kim, S.-H. Park, and E.-Y. Chung, "Power Failure Protection Scheme for Reliable High-Performance Solid State Disks," *IEICE Transactions on Information and Systems*, vol. E96-D, no. 5, pp. 1078–1085, 2013.

[34] L. Shi, K. Wu, M. Zhao, C. J. Xue, and E. H.-M. Sha, "Retention Trimming for Wear Reduction of Flash Memory Storage Systems," in *Proceedings of the Design Automation Conference* (DAC), 2014.

[35] JEDEC standard, "Method for Developing Acceleration Models for Electronic Component Failure Mechanisms," *JESD91A*, Aug. 2003.

[36] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data Retention in MLC NAND Flash Memory: Characterization, Optimization, and Recovery," in *Proceedings of the IEEE Symposium on High-Performance Computing Architecture* (HPCA), 2015.

[37] S. Tanakamaru, Y. Yanagihara, and K. Takeuchi, "Error-Prediction LDPC and Error-Recovery Schemes for Highly Reliable Solid-State

Drives (SSDs)," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 11, pp. 2920–2933, 2013.

[38] J. Jeong, Y. Song, and J. Kim, "FlashDefibrillator: A Data Recovery Technique for Retention Failures in NAND Flash Memory," in *Proceedings of the IEEE Non-Volatile Memory System and Applications Symposium* (NVMSA), 2015.

[39] K. Fukuda, Y. Shimizu, K. Amemiya, M. Kamoshida, and C. Hu, "Random Telegraph Noise in Flash Memories - Model and Technology Scaling," in *Proceedings of the IEEE International Electron Devices Meeting* (IEDM), 2007.

[40] B. Tang, C. Robinson, W. D. Zhang, J. F. Zhang, R. Degraeve, P. Blomme, M. Toledano-Luque, G. V. den bosch, B. Govoreanu, and J. V. Houdt, "Read and Pass Disturbance in the Programmed States of Floating Gate Flash Memory Cells With High-k Interpoly Gate Dielectric Stacks," *IEEE Transactions on Electron Devices*, vol. 60, no. 7, pp. 2261–2267, 2013.

[41] K. Ha, J. Jeong, and J. Kim, "An Integrated Approach for Managing Read Disturbs in High-Density NAND Flash Memory," accepted for publication in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

[42] A. Badam and V. S. Pai, "SSDAlloc: Hybrid SSD/RAM Memory Management Made Easy," in *Proceedings of the USENIX Conference on Networked Systems* (NSDI), 2011.

[43] B. Jacob, I. Bhati, M.-T. Chang, P. Rosenfeld, J. Stevens, P. Tschirhart, Z. Chishti, S.-L. Lu, J. Ang, D. Resnick, and A. Rodrigues, "A Journaled, NAND-Flash Main-Memory System," *Technical Report, University of Maryland*, UMD-SCA-2010-12-01, December 2010, updated 2014.

[44] J. Taylor, "Flash at Facebook," in *Proceedings of the Flash Memory Summit*, 2013.

초 록

컴퓨팅 시스템의 성능 향상을 위해, 기존의 느린 HDD를 빠른 낸드 플래시 메모리 기반 저장장치(SSD)로 대체하고자 하는 연구가 최근 활발히 진행되고 있다. 지난 수십 년에 걸친 반도체 미세 공정 기술과 다치화 기술 등의 디바이스 수준의 발전으로 인해 최근 SSD의 가격은 HDD 수준으로 낮아졌지만, 이와 같은 최신 반도체 기술의 부작용으로 낸드 플래시 메모리의 수명이 급격히 감소되는 문제가 지속적으로 제기되고 있다. 이와 같은 낮은 수명 문제는, SSD가 고성능 컴퓨팅 시스템에 널리 사용되기 위해 반드시 해결되어야 하는 가장 중요한 기술적 이슈중의 하나이다.

본 논문에서는, 낸드 플래시 메모리의 수명(특히, 내구성)을 향상시키기 위한, 디바이스와 시스템간의 계층 교차 최적화 기법을 제안한다. 제안하는 기법들은 우리의 중요한 발견, 즉 낸드 블락을 낮은 전압으로 그리고 느린 속도로 소거할수록 낸드의 내구성이 현격히 개선된다는 새로운 사실을 기반으로 개발되었다. 그러나, 소거 동작에서 낮은 전압을 사용할 수록, 쓰기 성능 및 데이터 보존 능력 등의 디바이스의 특성이 저하되는 부작용이 발생한다. 제안된 기법의 주된 목표는, 전체적으로 저장 장치의 특성에 미치는 부작용 없이 낸드 디바이스의 내구성을 향상시킴으로써, SSD의 수명을 개선하고자 하는 것이다.

첫번째로, 시스템 소프트웨어가 낸드 플래시 메모리의 내구성과 소거 전압/시간 간의 보상관계를 직접적으로 활용할 수 있는 기반이 되는, 동적 소거 전압 및 시간 조절 기법(DeVTS)이 제안된다. DeVTS는 소거 전압/시간 및 쓰기 능력을 변경하는 다양한 모드를 제공하는데, 이와 같은 모드를 이용하여 낸드 플래시 메모리의 내구성, 성능 그리고 데이터 보존 능력을 상황에 맞게 서로 다르게 조절할 수 있게 된다.

두번째로, 낸드 블락을 느리게 소거하거나 또는 낮은 전압으로 소거된 블락에 데이터를 쓸 때, 쓰기 요청들 간의 유휴 시간을 활용하여, 전체적인 성능 저하없이 SSD의 수명을 향상시킬 수 있는 기법이 제안된다. 또한, 이와 같은 기법이 적용된 dvsFTL이라는 FTL (Flash Translation Layer)이 제안되는데, 이는

낸드 디바이스의 소거 전압/속도와 쓰기 성능을 상황에 맞게 자동적으로 조절함으로써 낸드 디바이스의 내구성을 극대화하는 기능을 구비한다. 실험 결과에 의하면, dvsFTL은 낸드 디바이스의 내구성을 평균적으로 62% 개선하는 효과를 보이고, 이때 전체적인 SSD의 성능 저하는 극히 미미한 수준이었다.

세번째로, 낮은 전압으로 소거된 블락에 데이터를 쓸 때, SSD에 요구되는 성능 및 데이터 보존 시간에 대한 변동을 종합적으로 활용하여, SSD의 수명 향상 효과를 극대화 시키는 기법이 제안된다. 이어서, dvsFTL 대비 개선된 FTL인, dvsFTL+가 제안되는데, 이는 쓰기 성능 및 데이터 보존 시간에 대한 요구량의 변화를 실시간으로 정확하게 예측함으로써, SSD의 수명 개선 효과를 보다 향상시킬 수 있게 한다. 실험 결과에 의하면, dvsFTL+는 dvsFTL보다 낸드 디바이스의 내구성을 50% 이상 더 향상시킬 수 있고, 이때 저장 장치에 대한 모든 요구 수준을 충분히 보장할 수 있었다.

마지막으로, 데이터 보존 능력을 적극적으로 조절하는 기법이 실제 컴퓨팅 환경에 적용되는 경우 발생할 수 있는, 데이터 유실문제를 해결하기 위한 신뢰성 개선 기법이 추가로 제안된다. 실험 결과에 의하면, 제안된 기법은, 기존에 낸드 플래시 메모리가 보장하는 데이터 보존 시간을 최대 8배 향상시킬 수 있을 뿐 아니라, 기존에 제시된 데이터 복구 기법들보다 최대 23배 빠르게 손상된 데이터를 복구할 수 있었다.

지금까지 제시된 다양한 실험 결과를 바탕으로, 우리는 제안된 계층 교차 최적화 기법들이 낸드 플래시 기반 저장장치의 수명 향상에 큰 효과가 있음을 확인하였다. 향후 제안된 기법들이 보다 발전된다면, 낸드 플래시 메모리가 초고속 컴퓨팅 시스템의 주 저장 장치로 널리 사용될 수 있을 뿐 아니라 메인 메모리로도 충분히 활용될 수 있을 것으로 기대된다.

키워드: 낸드 플래시 메모리, 플래시 기반 저장장치, 저장장치 제어, 저장장치 신뢰성, 저장장치 수명, 임베디드 소프트웨어
학번: 2012-30229