Dynamic Erase Voltage and Time Scaling for Extending Lifetime of NAND Flash-Based SSDs

Jaeyong Jeong, Youngsun Song, Sangwook Shane Hahn, Sungjin Lee, and Jihong Kim, *Member, IEEE*

Abstract—The decreasing lifetime of NAND flash memory, as a side effect of recent advanced semiconductor process scaling, is emerging as one of major barriers to the wide adoption of SSDs in high-performance computing systems. In this paper, we propose Dynamic Erase Voltage and Time Scaling (DeVTS), an integrated approach to extend the lifetime (particularly, endurance) of NAND flash memory. DeVTS is 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 erase voltage causes adverse side effects on the write performance and retention capability of NAND flash memory. In order to improve NAND endurance without affecting the other NAND requirements, we take advantage of idle times between write requests and variations of the retention requirement when writing data to a NAND block erased with a lower voltage. We have implemented a DeVTS-aware FTL, called dvsFTL, which exploits the tradeoff relationship between the endurance and erase voltages/times by accurately predicting the write performance and retention requirements. Our experimental results show that dvsFTL can improve NAND endurance by 94 percent, on average, over an existing DeVTS-unaware FTL while all the NAND requirements are preserved.

Index Terms—NAND flash memory, solid-state drive, storage system, storage performance, storage reliability

1 Introduction

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 [2]. Fortunately, the prices for SSDs have fallen to the comparable level of HDDs by continuous semiconductor process scaling combined with multi-leveling technologies. 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 [3].) 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 $N_{P/E}^{max}$ have declined by 70 percent during that period [4]. In order for SSDs to be the mainstream of enterprise environments, the issue concerning NAND endurance should be properly resolved.

In this paper, we propose an integrated approach, called **D**ynamic **E**rase **V**oltage and **T**ime **S**caling (**DeVTS**[1]), which significantly improves NAND endurance by dynamically adjusting the erase voltage and time of NAND flash memory. The DeVTS approach 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 [5], reducing the stress voltage (in particular, 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 [6], 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 oxide damage [1], [7], thus additionally improving NAND endurance. By modifying NAND devices 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 can significantly increase $N_{P/E}^{max}$.

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.,

- *J. Jeong is with the Memory Division, Samsung Electronics, 1-1 Samsungjeonja-ro, Hwaseong-si, Gyeonggi-do 18448, Korea. E-mail: jyjeong@davinci.snu.ac.kr.*
- *Y. Song, S.S. Hahn, and J. Kim are with the Department of Computer Science and Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Korea. E-mail: {ysunsong, shanehahn, jihong}@davinci.snu.ac.kr.*
- *S. Lee is with the Department of Computer Science and Information Engineering, Inha University, 100 Inha-ro, Nam-gu, Incheon 22212, Korea. E-mail: sungjin.lee@inha.ac.kr.*

Manuscript received 21 Jan. 2016; revised 24 Sept. 2016; accepted 26 Sept. 2016. Date of publication 3 Oct. 2016; date of current version 17 Mar. 2017. Recommended for acceptance by K. Chakrabarty.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2016.2615038

1. The naming of our proposed technique is highly influenced by the Dynamic Voltage and Frequency Scaling (DVFS) technique that is a very widely-used key technique for low-power CPUs.

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 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 [1]. 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 [8], [9].

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 comprehensive NAND endurance model which accurately captures the tradeoff relationships between the endurance and performance/retention capabilities of NAND flash memory. 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.

Our DeVTS approach 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-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-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-enabled technique selects the most proper speed/retention modes for each program operation, or chooses the most suitable voltage/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-aware FTL, called *dvsFTL*, which dynamically adjusts the erase voltage and speed modes in an *automatic* fashion 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-aware to maximize the efficiency of dvsFTL. We evaluated the effectiveness of dvsFTL with an extended *FlashBench* emulation environment [10] where the DeVTS-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 94 percent, on average, over an existing DeVTS-unaware FTL without sacrificing the performance and retention requirements of SSDs.

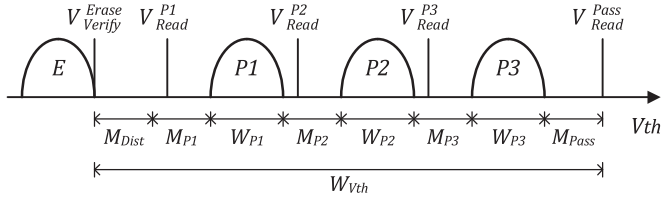


Fig. 1. An illustration of *Vth* distributions for an MLC NAND device and primary *Vth* design parameters.

The rest of the paper is organized as follows. Section 2 briefly reviews main design principles of NAND flash memory and introduces a reliability metric used in this paper. In Section 3, we describe erase voltage and time scaling. Section 4 presents write capability tuning techniques and the NAND endurance model. In Section 5, we explain the proposed dvsFTL in detail, and report experimental results in Section 6. After Section 7 summarizes related work, we finally conclude with a summary in Section 8.

2 Background

2.1 Design Principles 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. Fig. 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 [3]. For example, read and program operations of an MLC device should be completed within 100 μs and 1,600 μs, respectively [4]. Moreover, even after 3,000 P/E cycles, it is required to support up to 400,000 read operations [4] as well as to retain its stored data for up to 1 year at 30 °C [11]. Since the *Vth* design parameters shown in Fig. 1 are closely related to the NAND 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 [1]. 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 [1]. 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 $N_{P/E}^{max}$ followed by 1-year retention times.

The *Vth* margin M_{Dist} between the E state and V_{Read}^{P1} primarily affects the disturbance resistance of NAND devices. When a NAND memory cell is programmed or read, its neighbor cells that belong to the E state may be softly programmed so that their *Vth*s move to the right [3], [8]. 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 Fig. 1. Typically, M_{Dist} is decided by the maximum *Vth* change after $N_{P/E}^{max}$ followed by 400,000 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} [3], 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 memory cells in a NAND block [3].

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

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

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 Reliability Metrics for NAND Memory Cells

Since the physical mechanism of NAND endurance degradation is closely related to stress-induced damage (i.e., defect) in the tunnel oxide layer of NAND cells [8], in order to represent the wearing degree of the NAND cells, it is necessary to measure the defect density in the cells. However, to the best of our knowledge, it is practically impossible to directly measure the defect density in NAND cells without using a destructive inspection.

As an alternative reliability metric for NAND cells, in this paper, we use the number $N_{ret}(x,t)$ of retention errors after t-year retention time for x pre-cycled NAND cells. After several experimental trials with different measures, we observed that $N_{ret}(x,t)$ is one of efficient reliability metrics which can better represent the total defect density in the NAND cells because they have a near-linear dependence relation. In this paper, we extensively use $N_{ret}(x,1)$ for x pre-cycled NAND cells. In order to emulate a 1-year retention time condition, we baked the NAND chips at $100\,^{\circ}\mathrm{C}$ for one hour (which is equivalent to one year at $30\,^{\circ}\mathrm{C}$ [12]) after x pre-cyclings.

A decision to use $N_{ret}(x,t)$ as a reliability metric is largely based on the bit error patterns of recent NAND devices. As the defect density in the cell increases, bit errors occur from two main sources, one from the NAND endurance problem and the other from the NAND retention problem. For example, a common reliability metric $N_{P/E}^{max}$ quantifies the endurance of NAND cells. The data retention of NAND cells can be quantified by the maximum time T_{ret}^{max} that the cells can keep their stored data. Although these two types of bit errors are quantified using different metrics (i.e., $N_{P/E}^{max}$ and T_{ret}^{max}), they are closely related with each other because they are originated from the same physical mechanism, i.e., the total defect density in the cell. For recent NAND devices, however, bit errors from the NAND retention problem are dominating, especially when the defect density in the cell is high, thus focusing on retention errors is an effective approach to model the defect density in NAND cells.

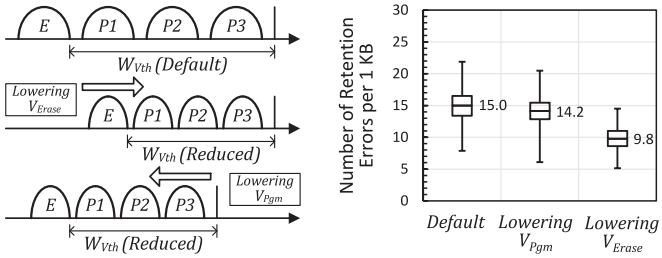
3 Erase Voltage and Time Scaling

In this section, we describe the motivation behind DeVTS and present the effect of erase voltage and time scaling on improving NAND endurance.

3.1 Motivation

Since the probability of oxide damage has an exponential dependence on the stress voltage [5], 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.

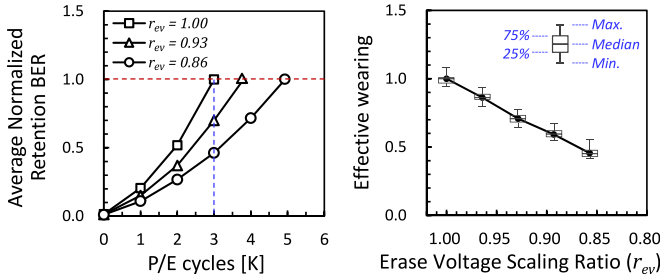
In order to verify our assumption, we evaluated the effects of two different stress-voltage-reduction policies, as shown in Fig. 2a, on NAND endurance. In the 'lowering V_{Erase}' policy, W_{Vth} shrinks to the right direction (compared to the default case) so that V_{Erase} is lowered (e.g., by 1 V) 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 (e.g., by 1 V) while V_{Erase} is maintained. In our evaluation, ten blocks out of two 20-nm node NAND chips [13] were selected for each policy. As the main evaluation metric, we measured $N_{ret}(x,1)$ per 1 KB cells because it reflects the effective degree of NAND wearing [1]. In Fig. 2b, we compare $N_{ret}(3\,K,1)$ values under three different policies. As shown in Fig. 2b, when the 'lowering V_{Pgm}' policy



(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.

Fig. 2. Comparison of the impacts of lowering V_{Pgm} and V_{Erase} on NAND retention errors.



(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.

Fig. 3. The effect of erase voltage scaling on NAND endurance.

was used, the number of retention errors was reduced by only 5.3 percent, 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 percent, on average, over the default case. These results clearly show that if W_{Vth} can be reduced, lowering V_{Erase} is much more effective than lowering V_{Pgm} in improving NAND endurance.

3.2 Erase Voltage Scaling and Its Effect on Endurance

In order to evaluate the effect of erase voltage scaling on NAND endurance, we performed NAND cycling tests with different V_{Erase}'s. In a cycling test, program and erase operations were repeated 3,000 times. Our cycling tests for each case were performed with 100 blocks out of ten 20-nm node NAND chips [13]. After cycling tests, we measured the NAND retention BER (i.e., the number of retention errors divided by the total number of tested 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. Fig. 3a 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 Fig. 3a, 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 after 3 K P/E cycles is reduced by 30 percent 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 [6] which simplifies the NAND wear-out behavior over P/E cycles as shown in Fig. 3a, we represent effective wearing with a normalized retention BER after 3 K 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 percent (i.e., $r_{ev} = 0.93$), effective wearing becomes 0.70. As shown in Fig. 3b, 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 paper, we will use a NAND endurance model with six erase voltage modes $\texttt{EVmode}_i$'s which have six different r_{ev}'s (as described in Section 4.4).

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

3.3 Erase Time Scaling and Its Effect on Endurance

Endurance degradation is directly proportional to V_{Erase} in an erase operation as described in Section 3.2. 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 [6], an unintended higher (than V_{Erase}) voltage may cause additional damage 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 in erase operations 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 [14] so that the applied V_{Erase} 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 Fig. 4. However, when the modified ISPE scheme is used for an erase operation, the erase time (e.g., T_{Erase}^{slow} shown in Fig. 4) inevitably increases because more ISPE loops are needed to complete the erase operation.

As shown in Fig. 5a, effective wearing decreases near-linearly as the erase time increases. For example, when the

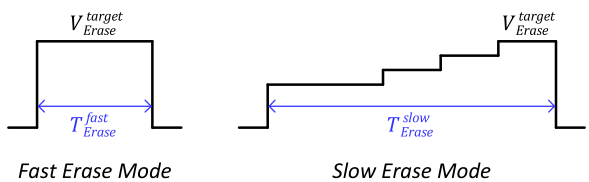
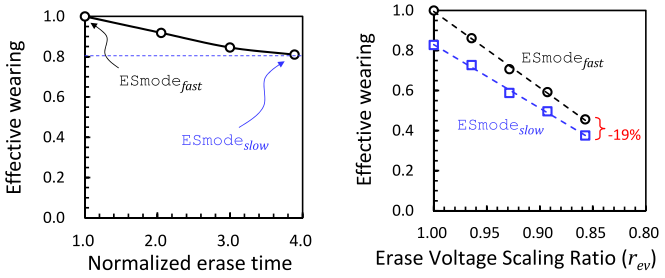


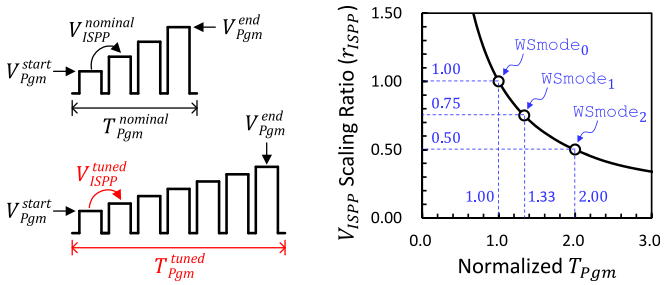
Fig. 4. An illustration of the proposed erase voltage and time controls for the fast and slow erase speed modes.



(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.

Fig. 5. The effect of erase time scaling on NAND endurance.



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

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

Fig. 7. The proposed write-performance tuning technique.

erase time increases threefold, effective wearing is reduced, on average, by 19 percent. We represent the erase speed mode with a default erase time by ESmode$_{fast}$ while that with a long erase time is represented by ESmode$_{slow}$. As shown in Fig. 5b, the effect of ESmode$_{slow}$ on improving NAND endurance can be exploited whenever longer erase times are acceptable regardless of r_{ev}.

4 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. For example, as shown in Fig. 6, if a NAND block is erased with a low erase voltage V_{Erase}^{low} (which is lower than $V_{Erase}^{nominal}$), the available *Vth* window 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, in order to write data to the shallowly erased NAND block, 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 section, 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 improving NAND endurance.

4.1 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} [15]. However, since the number of ISPP loops to complete a program operation is inversely proportional

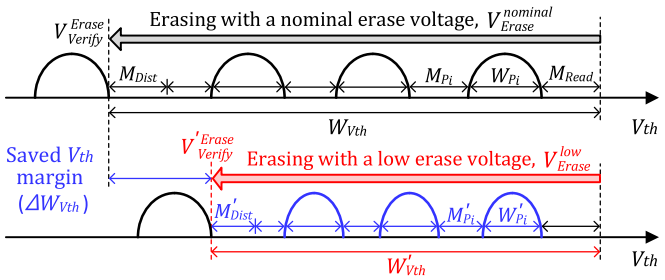


Fig. 6. An example of NAND capability tuning for writing data to a shallowly erased NAND block.

to V_{ISPP} [1], the program time T_{Pgm} inevitably increases as shown in Fig. 7a if narrower *Vth* distributions are required. Fig. 7b 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 [13]). 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 $(1-x) \times V_{ISPP}^{nominal}$.

In our proposed write-performance tuning technique, we define three different write-speed modes, WSmode$_0$, WSmode$_1$, and WSmode$_2$, as shown in Fig. 7b. WSmode$_0$ is the fastest write mode which has the same T_{Pgm} as that of the nominal write mode, and cannot reduce V_{ISPP}. On the contrary, WSmode$_2$, the slowest write mode, has a T_{Pgm} two times longer than $T_{Pgm}^{nominal}$ (i.e., the normalized T_{Pgm} is 2.0), but can reduce V_{ISPP} by 50 percent over $V_{ISPP}^{nominal}$.

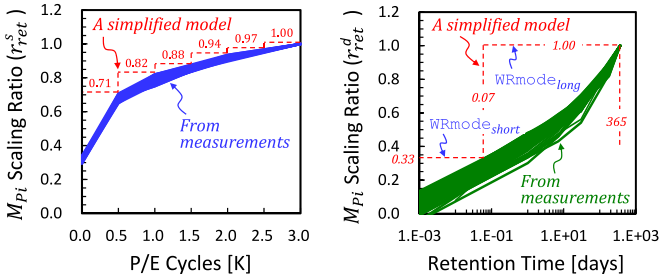
Since W_{Pi} has a linear dependence on V_{ISPP}, ΔW_{Vth} by tuning T_{Pgm} is expressed as follows (for an MLC NAND device):

$$\Delta W_{Vth} = \sum_{i=1}^{3} \Delta W_{Pi} = \sum_{i=1}^{3} (1 - r_{ISPP}) \times V_{ISPP}^{nominal}. \quad (2)$$

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 ($= 3 \times ((1 - 0.50) \times 400\,\text{mV})$).

4.2 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 [11]). 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 conditions (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 utilized 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 [8], only part of M_{Pi} is enough for young NAND memory cells (that have experienced a few P/E cycles) to meet the retention-time requirement. Moreover, since the *Vth* change is also proportional to a retention time [8], when written data are updated frequently within a short time period, M_{Pi} for such data is not fully needed.



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

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

Fig. 8. The simplified M_{Pi} scaling models for retention capability tuning.

4.2.1 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 four NAND chips). After the cycling tests, we measured the average change in *Vth* for each block after 1 hour's baking at 100 °C. Measured average *Vth* change was normalized over the maximum required *Vth* margin M^{max}_{Pi} under the worst-case operating condition (i.e., 3 K 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^s_{ret}. Fig. 8a shows r^s_{ret} variations over varying P/E cycles. After 0.5 K P/E cycles, for example, only 71 percent of M^{max}_{Pi} is required (i.e., r^s_{ret} is 0.71). Based on the measurement results, we constructed a simplified static M_{Pi} scaling model where r^s_{ret} changes every 0.5 K P/E cycles as shown by the dotted line in Fig. 8a. For a given number of P/E cycles, ΔW_{Vth} by tuning the NAND retention capability is expressed as follows (for an MLC NAND device):

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

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

4.2.2 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 Fig. 8b show r^d_{ret} variations 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., WRmode$_{short}$ and WRmode$_{long}$) as shown by the dotted line in Fig. 8b. 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). On the contrary, WRmode$_{short}$ is the short-retention write mode which supports only a 0.07-day retention time while requiring only 33 percent of M^{max}_{Pi} (i.e., r^d_{ret} is 0.33). By combining r^s_{ret} with r^d_{ret}, Eq. (3) is re-expressed as follows:

$$\Delta W_{Vth} = \sum_{i=1}^{3} \Delta M_{Pi} = \sum_{i=1}^{3} (1 - r^s_{ret} \times r^d_{ret}) \times M^{max}_{Pi}. \quad (4)$$

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

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

4.3 Disturbance Resistance Tuning

Since the program disturbance and the read disturbance of NAND flash memory are proportional to the number of P/E cycles [8], 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 followed by a specified number (i.e., 400 K [4]) 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 required *Vth* margin M^{max}_{Dist} under the worst-case operating condition (i.e., 3 K P/E cycles and 400 K read cycles). We represent the normalized average *Vth* change caused by NAND disturbance as r_{dist} $(0 \le r_{dist} \le 1)$. Table 1 summarizes our simplified r_{dist} model over varying P/E cycles. For example, after 0.5 K P/E cycles, only 43 percent of M^{max}_{Dist} 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 is expressed as follows:

$$\Delta W_{Vth} = \Delta M_{Dist} = (1 - r_{dist}) \times M^{max}_{Dist}. \quad (5)$$

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

4.4 NAND Endurance Model

Combining the proposed NAND capability tuning techniques (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 Eqs. (2), (4), and (5). Since a reduced erase voltage $(= (1 - r_{ev}) \times V^{nominal}_{Erase})$ is proportional to ΔW_{Vth}, r_{ev} is expressed as follows (for an MLC NAND devices):

$$r_{ev} = 1 - \frac{\Delta W_{Vth}}{V^{nominal}_{Erase} \times \alpha_c}, \quad (6)$$

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,

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

W_{Vth} can be effectively reduced by 0.60 V. When r_{ev} is calculated from Eq. (6) for a given ΔW_{Vth}, the corresponding effective wearing can be estimated by the linear equation described in Section 3. Table 2 summarizes the parameter set used to construct the NAND endurance model in this paper. All the data in our model is based on measurement results with 20-nm node NAND chips.

Table 3 shows our DeVTS-enabled NAND endurance (i.e., effective wearing) model with six erase-voltage modes (i.e., $\texttt{EVmode}_0 \sim \texttt{EVmode}_5$) and two erase-speed modes (i.e., $\texttt{ESmode}_{fast}$ and $\texttt{ESmode}_{slow}$). $\texttt{EVmode}_5$ is the lowest erase-voltage mode which can maximize improvement on NAND endurance while $\texttt{EVmode}_0$ is the highest erase-voltage with the lowest endurance gain. $\texttt{EVmode}_j$'s are determined by a combination of two write-retention modes (i.e., $\texttt{WRmode}_{long}$ and $\texttt{WRmode}_{short}$) and three write-speed modes (i.e., $\texttt{WSmode}_0 \sim \texttt{WSmode}_2$) because r_{ev}'s are different for each combination. Since ΔW_{Vth}'s are also affected by the static retention-capability tuning and disturbance-resistance tuning, the values of effective wearing vary whenever ΣEW exceeds 0.5 K. For example, if ΣEW is less than 0.5 K, when a NAND block is erased with $\texttt{EVmode}_5$ and $\texttt{ESmode}_{slow}$, the effective wearing for that erase operation 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 $\texttt{EVmode}_j$.

TABLE 3
The NAND Endurance (i.e., Effective Wearing) Model over the Total Sum (i.e., ΣEW) of Effective Wearing under NAND Capability Tuning Modes

EVmode	0	1	3	2	4	5
ESmode	*fast*					
WRmode	*long*			*short*		
WSmode	0	1	2	0	1	2
$0 \le \Sigma EW \le 0.5$ K	0.78	0.65	0.52	0.59	0.46	0.33
0.5 K $< \Sigma EW \le 1.0$ K	0.83	0.69	0.56	0.62	0.49	0.36
1.0 K $< \Sigma EW \le 1.5$ K	0.89	0.76	0.63	0.67	0.53	0.40
1.5 K $< \Sigma EW \le 2.0$ K	0.96	0.83	0.69	0.71	0.57	0.44
2.0 K $< \Sigma EW \le 2.5$ K	0.98	0.85	0.71	0.72	0.59	0.45
2.5 K $< \Sigma EW \le 3.0$ K	1.00	0.87	0.73	0.73	0.60	0.47
EVmode	0	1	3	2	4	5
ESmode	*slow*					
WRmode	*long*			*short*		
WSmode	0	1	2	0	1	2
$0 \le \Sigma EW \le 0.5$ K	0.68	0.57	0.45	0.52	0.40	0.29
0.5 K $< \Sigma EW \le 1.0$ K	0.72	0.60	0.49	0.54	0.43	0.31
1.0 K $< \Sigma EW \le 1.5$ K	0.78	0.66	0.55	0.58	0.46	0.35
1.5 K $< \Sigma EW \le 2.0$ K	0.83	0.72	0.60	0.62	0.50	0.38
2.0 K $< \Sigma EW \le 2.5$ K	0.85	0.74	0.62	0.63	0.51	0.40
2.5 K $< \Sigma EW \le 3.0$ K	0.87	0.75	0.64	0.64	0.52	0.41

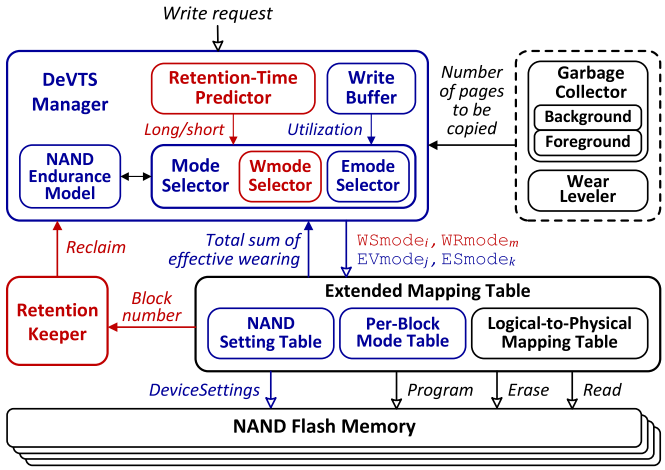


Fig. 9. An organizational overview of dvsFTL.

5 Design and Implementation of dvsFTL

5.1 Overview of dvsFTL

In order to improve NAND endurance without affecting the other NAND requirements, we have implemented a DeVTS-aware FTL, called *dvsFTL*, which dynamically changes erase scaling modes and write capability tuning modes based on the NAND endurance model. Fig. 9 illustrates an organizational overview of dvsFTL based on an existing page-level mapping FTL with additional modules for supporting DeVTS. *The DeVTS 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. First, 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. Second, the write-retention mode ($\texttt{WRmode}_m$) is chosen based on the retention requirement predicted by the retention-time predictor. Finally, the DeVTS 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 deadlines approach. The extended mapping table maintains per-block mode information (e.g., the erase/write modes and ΣEW for each block) as well as logical-to-physical mapping information. When the write mode or erase mode is changed, dvsFTL reconfigures NAND chips through a new device setting interface, *DeviceSettings*, by consulting the NAND setting table. The existing garbage collector and wear leveler[2] are also revised to maximize the efficiency of DeVTS.

5.2 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),

2. Since different erase voltage/time affects NAND endurance differently, the reliability metric (based on $N_{P/E}$) of the existing wear leveler is no longer valid in a DeVTS-enabled NAND chip. In dvsFTL, the DeVTS-aware wear leveler uses ΣEW instead of $N_{P/E}$ as a reliability metric, and tries to evenly distribute ΣEW among NAND blocks.

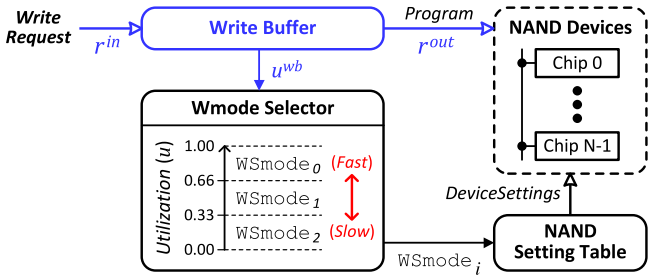


Fig. 10. An overview of the write-speed mode selection in dvsFTL.

the Wmode selector in the DeVTS 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 selector decides 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 decided that queued requests should be written as fast as possible.

Fig. 10 shows an overview of the write-speed mode selection in dvsFTL. In our implementation, the write-performance requirement is classified into three levels by two buffer utilization boundaries as shown in Fig. 10. For example, when u^{wb} is lower than 0.33, the requests queued in the write buffer are written to a NAND page with $\texttt{WSmode}_2$, the slowest write mode. However, when u^{wb} is higher than 0.66, in order to satisfy the urgent requirement of write performance, the write-speed mode is changed to $\texttt{WSmode}_0$, the fastest write mode.

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 [16]. For example, when garbage collection operations are performed in half of NAND chips, r^{out} is reduced by 50 percent. 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.

5.3 Write-Retention Mode Selection

The Wmode selector decides the most proper write-retention mode for a given write request based on the predicted future update time (i.e., the retention-time requirement) of that request. If it is predicted that a request will be updated within the predefined time period, which is shorter than the specified retention-time 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 [17] 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. In this section, we present a write-retention mode selection procedure with the resource-optimized retention-time predictor and describe retention requirement management techniques.

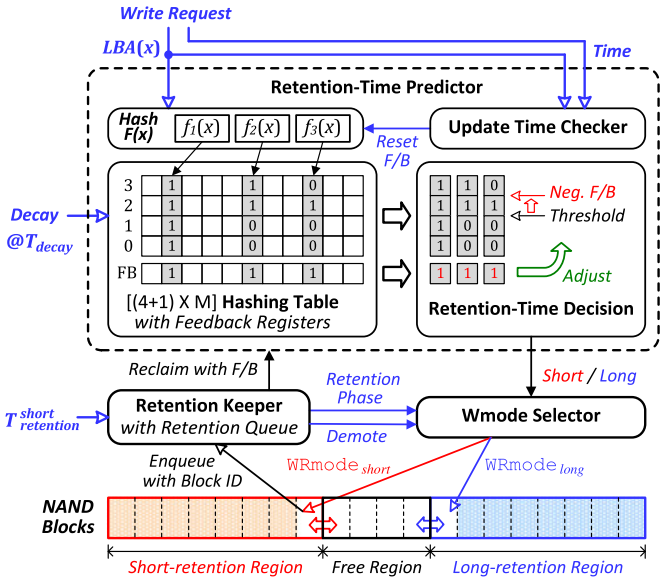


Fig. 11. A functional overview of the write-retention mode selection and retention requirement management procedures.

5.3.1 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^{short}_{ret} (e.g., 0.07 days as defined in Section 4.2.2). In our implementation, the retention-time predictor is based on an existing data separator [18] with a different control policy for making a reliable decision on the write-retention mode (as will be described in Section 5.3.2).

Each LBA is mapped to multi-dimensional counters incremented whenever corresponding write requests are issued. In order to compare the update interval to T^{short}_{ret}, all the counters are decayed regularly after a designated time interval T_{decay} (in this paper, $T_{decay} = T^{short}_{ret}$). 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^{short}_{ret} on average. In this case, the retention-time predictor predicts that the current write request will be also updated within T^{short}_{ret} by exploiting the temporal locality of I/O requests.

Fig. 11 shows a functional overview of our proposed retention-time predictor. Since maintaining all the counters for each LBA is too expensive 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).

5.3.2 Maximization of Endurance Benefit

In order to maximize the endurance benefit of retention-capability tuning, minimizing reclaimed pages caused by misprediction is one of the major 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 Fig. 11. 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.

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 Fig. 11, each counter has an additional feedback register which is set to one when misprediction is detected (i.e., 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 were 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 Table 3) 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 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.

5.3.3 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 deadline expires, the retention keeper periodically (e.g., one tenth of T_{ret}^{short}) 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 Fig. 11, data for each write-retention mode is written to different regions (i.e, the short-retention region and long-retention region). 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 its written time. Although following data is written to that block at different times, the worst-case retention deadline is still determined by the earliest written time for that block. 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.

5.4 Erase-Voltage Mode 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 3.2 and 4.1, 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 $\texttt{EVmode}_5$ allows only $\texttt{WSmode}_2$ (which is the slowest write-speed mode) in a program operation, when intensive write operations are requested, the write performance can be degraded significantly. Furthermore, since $\texttt{EVmode}_5$ assumes the presence of hot data, expected to be updated in the near future, when cold data is inevitably written to such a shallowly erased block, the endurance gain may be rendered invalid by the

reclaim operation. On the contrary, when $\texttt{EVmode}_0$ (which uses the highest erase voltage) is used at all times, DeVTS 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 0.5 K for a victim block, and $\texttt{WRmode}_{short}$ and $\texttt{WSmode}_0$ have been chosen, the Emode selector decides $\texttt{EVmode}_2$ 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 a long time 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 $\texttt{EVmode}_5$ 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 $\texttt{EVmode}_5$, the selected block is additionally erased using the *lazy erase* operation (of which latency is less than 1,000 μs). Although the write latency for the first page in the block is increased by 77 percent 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 percent while the potential of DeVTS can be fully utilized in terms of the lifetime improvement.

5.5 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., 0.33 or 0.66 as shown in Fig. 10), 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 that lost by using a faster write mode as summarized in Table 3, $\texttt{ESmode}_{slow}$ 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.

5.6 DeVTS-Aware Garbage Collection

When a garbage collection process[3] is invoked, selecting the most suitable write-speed mode for data copy operations is also a challenging issue to maximize the efficiency of DeVTS. 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 degraded significantly. 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.

6 Experimental Results

6.1 Experimental Settings

We evaluated the effectiveness of the proposed dvsFTL with *extFlashBench*, an extended version of the existing unified development environment [10] for NAND flash-based storage systems. Since extFlashBench includes various NAND timing information (i.e., the latency of read/program/erase operations of our target NAND chips), it emulates the key operations of DeVTS-enabled NAND devices in a timing-accurate fashion so that it is possible to keep track of temporal interactions among various FTL operations such as garbage collection, wear leveling, as well as reclaiming [1]. Table 4 summarizes the latency variations of write-speed modes and erase-speed modes used in our evaluations. 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 percent[4] of the total NAND capacity.

3. In this paper, we assume that a garbage collector cannot be preempted once it is started.

4. In Section 6.5.3, we discuss the effect of the different buffer size on the overall endurance gain in detail.

TABLE 4
The Latency Variations of NAND Functions Used in the Experiment

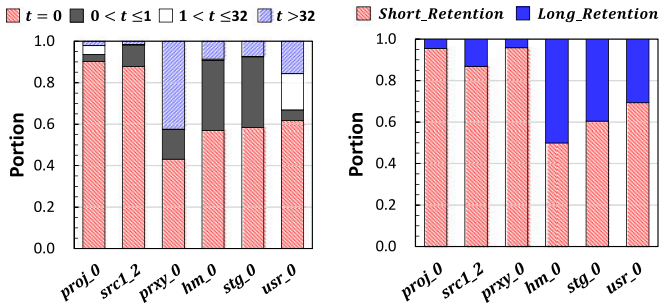
NAND function	Speed mode	Latency [13]
Program	$\texttt{WSmode}_0$	1,300 μs
	$\texttt{WSmode}_1$	1,730 μs
	$\texttt{WSmode}_2$	2,600 μs
Erase	$\texttt{ESmode}_{fast}$	5,000 μs
	$\texttt{ESmode}_{slow}$	20,000 μs

Our evaluations were performed with two different techniques: Baseline and dvsFTL. Baseline is an existing DeVTS-unaware FTL that does not use the erase scaling modes and write tuning modes. dvsFTL is the proposed DeVTS-aware FTL which fully exploits DeVTS-enabling techniques, described in Sections 3 and 4, depending on workload characteristics so that the lifetime benefit of DeVTS 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. (For more details on I/O traces, please see Section 6.2.) 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 effects of DeVTS.

6.2 Workload Characteristics

In our evaluations, 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 [19]. Although these traces included 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[5] so that the peak I/O rate of the most write intensive trace is comparable to the maximum write performance of our extFlashBench configuration [1], [20].

Fig. 12a 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 Fig. 12a. 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., *usr_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.



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

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

Fig. 12. Characteristics of write requests for six traces.

Fig. 12b 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[6]). 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 can be partly compensated for by the write-retention tuning.

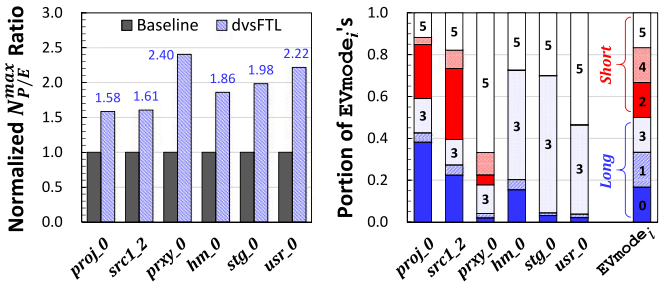
6.3 NAND Lifetime Analysis

In order to measure $N_{P/E}^{max}$ (i.e., the effective lifetime of a NAND device as defined in Section 3.2), each trace was repeated until ΣEW reached 3 K [4]. Measured $N_{P/E}^{max}$ values were normalized over 3 K. Fig. 13a shows $N_{P/E}^{max}$ ratios for six traces with two different techniques. dvsFTL extends $N_{P/E}^{max}$ by 94 percent, 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 Figs. 12a and 12b, respectively. In the case of *proj_0* trace, $N_{P/E}^{max}$ is improved by only 58 percent because most of the write requests are issued instantaneously so that 40 percent of erase operations cannot take advantage of endurance-enhancing modes at all as shown in Fig. 13b. 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 6.5.2.) On the contrary, for the *usr_0* trace, $N_{P/E}^{max}$ is improved by up to 122

5. Since these traces are based on slow HDDs of about 15-20 years ago, we compared the IOPS between these old HDDs and modern SSDs to extract the acceleration ratio for replaying HDD-based traces. We selected 100 as a conservative acceleration ratio because the IOPS differences between the old HDDs and modern SSDs range between several hundreds and tens of thousands. Since we accelerate all the I/O traces, we could have converted long idle I/O intervals of the original I/O traces to short ones. However, our benchmark traces did not have long I/O idle intervals because of steady streams of sporadic I/O requests. Considering this characteristics of our benchmark traces, our evaluation results based on accelerated I/O trace replays are believed to be sufficient to show the effectiveness of the proposed technique.

6. In our evaluation, we set T_{ret}^{short} to 0.07 days as shown in Fig. 8b. This is because the total time of traces was only about 1 day. If our DeVTS 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 r_{ret}^{d} increases from 0.33 to 0.50.



(a) Comparisons of normalized $N_{P/E}^{max}$ ratios. (b) Distributions of the erase voltage modes.

Fig. 13. Comparisons of the endurance gain and distributions of the EVmode$_i$'s for six traces.

percent 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 percent. 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.

6.4 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.

6.4.1 Overall Write-Performance Requirement

When a write request is issued, if 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 decreases below 1.0. This delay time can be further amplified when the foreground garbage collection process is performed in NAND chips. Although dvsFTL frequently uses slow-speed write/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 5.

For the *proj_0* trace, the overall write throughput is rather improved by 0.5 percent because the worst-case delay time due to the garbage collection process is reduced. As summarized in Table 5, the write amplification factor (WAF), which reflects the average garbage collection overhead, is reduced by about 0.4 percent so that the portion of delayed requests among total requests is reduced from 5.6 to 4.8 percent. For other traces, the overall write throughput and portion of delayed requests of dvsFTL are maintained at the same level as those of Baseline.

6.4.2 Overall Retention Requirement

Since 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 percent of the total NAND pages in extFlashBench). Although dvsFTL frequently uses WRmode$_{short}$ for writing data onto NAND pages, retention failures did not occur in our evaluations.[7] This is mainly because the misprediction ratio is sufficiently suppressed by the misprediction control techniques, described in Section 5.3.2, so that the numbers of mispredicted pages are maintained below the maximum number of reclaimable pages at all times.

TABLE 5
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
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

6.5 Detailed Analysis

6.5.1 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 5.3.1. 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 6 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 simply utilizes the previous update time interval to predict the next update time [23]. 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 minimized.

For the *src1_2* trace, the false-short ratio under History is too high (i.e., 4.8 percent) to avoid retention failures while the ratio under H_FB is sufficiently suppressed below the tolerable level (e.g., 1 percent). Comparing H_FB with H_noFB, the false-short ratio is reduced from 2.3 to 0.9 percent, a value similar to that of DA. For the other traces, the false-short ratios are also maintained at a low level. These

7. In this paper, we assume that the power is always supplied. However, if the power is cut off, the durability of the stored data written with WRmode$_{short}$ may not be guaranteed because the retention keeper does not work in this case. This problem can be mitigated by rewriting valid pages written with WRmode$_{short}$ to other NAND pages with WRmode$_{long}$ during the power hold-up time supported by a storage system [21]. Even when the rewriting process is unexpectedly failed, data loss can be recovered by a data recovery procedure [22].

TABLE 6
Accuracy of the Proposed Retention-Time Predictor under Different Data Separation Techniques

Prediction	Result	*src1_2*				*prxy_0*				*hm_0*				*usr_0*			
		History	DA	H_noFB	H_FB	History	DA	H_noFB	H_FB	History	DA	H_noFB	H_FB	History	DA	H_noFB	H_FB
Short	True	81.9%	49.4%	60.9%	42.4%	94.3%	89.3%	89.8%	84.9%	43.3%	28.1%	28.8%	26.9%	63.6%	52.8%	53.1%	50.1%
	False	4.8%	0.8%	2.3%	0.7%	1.3%	0.5%	0.6%	0.4%	5.9%	0.5%	0.7%	0.3%	4.5%	0.9%	1.0%	0.7%
Long	True	8.3%	12.3%	10.8%	12.4%	2.9%	3.6%	3.5%	3.7%	44.2%	49.6%	49.4%	49.8%	26.2%	29.7%	29.7%	30.2%
	False	5.0%	37.5%	26.0%	44.5%	1.5%	6.5%	6.1%	10.9%	6.6%	21.8%	21.1%	23.0%	5.7%	16.6%	16.3%	19.1%

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 6, 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 paper 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 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.

6.5.2 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. Fig. 14 shows the increase in $N_{P/E}^{max}$ ratios for six traces when each endurance-enhancing technique (i.e., ST, WPT, ETT, and DRCT) is enabled one by one on top of Baseline. ST is the combination of the Static Tuning techniques described in Sections 4.2.1 and 4.3. WPT is the Write-Performance Tuning technique, and ETT is the Erase-Time Tuning technique, as described in Sections 4.1 and 3.3, respectively. DRCT is the Dynamic Retention-Capability Tuning technique described in Section 4.2.2. 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 percent, on average, of the total endurance gain. The effect of DRCT strongly depends on the true-short ratio summarized in Table 6. For example, for the *prxy_0* trace, predicting short-retention requests is very accurate (i.e., 84.9 percent). As a result, $N_{P/E}^{max}$ is significantly extended (i.e., 98.6 percent) 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 percent, respectively, of the total endurance gain. In an earlier version (i.e., autoFTL [1]) of this paper, 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 percent over autoFTL. As shown in Fig. 14, 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.

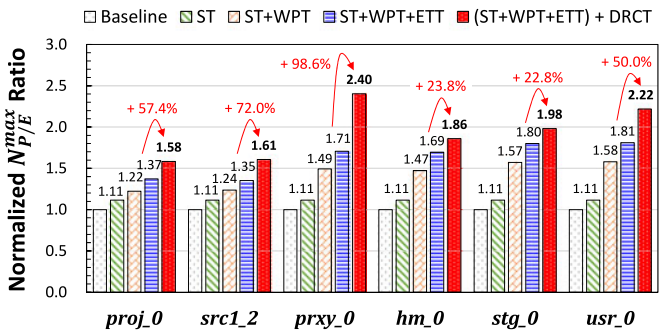


Fig. 14. Variations of the normalized $N_{P/E}^{max}$ ratios under different endurance-enhancing techniques for six traces.

6.5.3 Sensitivity to 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 percent 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 7. When the buffer size is reduced to 4 MB, the average $N_{P/E}^{max}$ is decreased by 3.6 percent. Alternatively, with a 64 MB size write buffer, the average $N_{P/E}^{max}$ ratio is increased by 4.1 percent 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.

7 Related Work

As the endurance of NAND flash memory continuously decreases, several cross-layer optimization techniques, which exploit the physical characteristics of NAND flash memory at a system software level, have been proposed for a longer SSD lifetime.

TABLE 7
Variations of $N_{P/E}^{max}$ Ratios over Different Buffer Sizes for Six Traces

	Buffer Size	*proj_0*	*src1_2*	*prxy_0*	*hm_0*	*stg_0*	*usr_0*	Avg.
Normalized $N_{P/E}^{max}$ Ratio	4 MB	1.54	1.53	2.17	1.81	1.97	2.18	1.87
	16 MB	1.58	1.61	2.40	1.86	1.98	2.22	1.94
	64 MB	1.66	1.72	2.57	1.93	2.00	2.26	2.02

Cai et al. presented a retention error management technique which periodically refreshes or reclaims the written data before their retention deadlines expire [24]. Since retention errors are one of main sources of bit errors which limit the NAND lifetime, if they are accurately controlled, the actual lifetime of NAND flash memory can be extended. Although the main goal of our proposed DeVTS technique is basically same as that of the Cai's work, detailed approaches are quite different. Our DeVTS technique can reduce the NAND cell damage during erase operations so that NAND endurance can be directly improved.

Jeong et al. proposed a novel endurance-enhancing technique which exploits the tradeoff relationship between the endurance and erase voltage of NAND flash memory [1]. In order to lower the erase voltage, the write or erase speed is conditionally slowed down when the maximum NAND performance is not necessary. On the other hand, Shi et al. suggested a retention-trimming technique which is based on the same NAND tradeoff as Jeong's work, but uses a different tuning technique (i.e., the retention-capability tuning) for writing data to NAND pages [23]. Shi's work is effective in extending the SSD lifetime when the maximum retention capability of NAND devices is not fully needed. The lifetime benefit of these two techniques, however, may be quite limited when the overall I/O characteristics are not compatible with the preferable conditions for each technique. On the contrary, since our DeVTS approach provides various write-tuning techniques, each of which brings different impact with different workload conditions, a flash software can easily adapt to varying I/O characteristics for a much longer SSD lifetime.

8 Conclusions

We have presented a highly-integrated cross-layer approach, called DeVTS, for improving the lifetime of flash-based storage systems. The proposed DeVTS approach, which is based on a new NAND endurance model that accurately reflects the NAND device behavior, effectively exploits varying workload characteristics so that appropriate erase scaling modes and write tuning modes are selected for each NAND operation. In order to take advantage of newly supported erase scaling modes in an efficient way, the DeVTS-enabled technique decides more tightly the required performance/retention requirements of written data so that NAND endurance is not wasted by selecting inappropriate modes for NAND write operations. Experimental results show that our DeVTS-aware FTL, dvsFTL, can improve NAND endurance by 94 percent, on average, over an existing FTL without affecting the performance and retention requirements of storage systems. Since the performance tuning and retention tuning are complementary to each other, dvsFTL can more efficiently adapt to various types of I/O workloads.

The lifetime benefit of DeVTS can be further improved in several ways. For example, the current version of DeVTS cannot fully achieve its potential to the fullest because only limited information within a storage system can be exploited in estimating I/O characteristics. As future work, we plan to develop an advanced version of DeVTS that takes advantage of more system-level information (e.g., the amount of near-future write requests to a storage system) for making a better decision in selecting erase scaling modes and write tuning modes.

Acknowledgments

This work was supported by Basic Science Research Program and Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2013R1A2A2A01068260 and NRF-2015M3C4A7065645). The work of Sungjin Lee was supported by the Inha University Research Grant (INHA-53345-1). The ICT at Seoul National University and IDEC provided research facilities for this study. An earlier version of this paper was presented at the USENIX Conference on File and Storage Technologies, 2014 [1]. This work was done while J. Jeong was with the Department of Computer Science and Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Korea. Jihong Kim is the corresponding author.

References

[1] 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 *Proc. 12th USENIX Conf. File Storage Technol.*, 2014, pp. 61–74.

[2] C. Albrecht, et al., "Janus: Optimal flash provisioning for cloud storage workloads," in *Proc. USENIX Conf. Annu. Tech. Conf.*, 2013, pp. 91–102.

[3] J. E. Brewer and M. Gill, *Nonvolatile Memory Technologies with Emphasis on Flash*. Hoboken, NJ, USA: Wiley, 2008.

[4] A. A. Chien and V. Karamcheti, "Moore's law: The first ending and a new beginning," *IEEE Comput.*, vol. 46, no. 12, pp. 48–53, Dec. 2013.

[5] K. F. Schuegraf and C. Hu, "Effects of temperature and defects on breakdown lifetime of thin SiO_2 at very low voltages," *IEEE Trans. Electron Devices*, vol. 41, no. 7, pp. 1227–1232, Jul. 1994.

[6] J. Jeong and J. Kim, "Dynamic program and erase scaling in NAND flash-based storage systems," Seoul Nat. Univ., Seoul, South Korea, Tech. Rep., 2014. [Online]. Available: cares.snu.ac.kr/download/TR-CARES-01–14

[7] S. Cho, "Improving NAND flash memory reliability with SSD controllers," in *Proc. Flash Memory Summit*, 2013, pp. 8–12.

[8] N. Mielke, et al., "Bit error rate in NAND flash memories," in *Proc. IEEE Int. Rel. Phys. Symp.*, 2008, pp. 9–19.

[9] R.-S. Liu, C.-L. Yang, and W. Wu, "Optimizing NAND flash-based SSDs via retention relaxation," in *Proc. 10th USENIX Conf. File Storage Technol.*, 2012, pp. 11–11.

[10] S. Lee, J. Park, and J. Kim, "FlashBench: A workbench for a rapid development of flash-based storage devices," *Proc. IEEE Int. Symp. Rapid Syst. Prototyping*, 2012, pp. 163–169.

[11] A. Cox, "JEDEC SSD Endurance Workloads," in *Proc. Flash Memory Summit*, 2011, pp. 6–7.

[12] *Stress-Test-Driven Qualification of Integrated Circuits*, JEDEC Std. JESD47H.01, 2011.

[13] C. Kim, et al., "A 21 nm high performance 64 Gb MLC NAND flash memory with 400 MB/s asynchronous toggle DDR interface," *IEEE J. Solid-State Circuits*, vol. 47, no. 4, pp. 981–989, Apr. 2012.

[14] D. W. Lee, et al., "The operation algorithm for improving the reliability of TLC (Triple Level Cell) NAND flash characteristics," in *Proc. IEEE Int. Memory Workshop*, 2011, pp. 1–2.

[15] K.-D. Suh, et al., "A 3.3 V 32 Mb NAND flash memory with incremental step pulse programing scheme," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, Nov. 1995.

[16] M. Jung and M. Kandermir, "Revisiting widely held SSD expectations and rethinking system-level implications," in *Proc. ACM SIGMETRICS*, 2013, pp. 203–2016.

[17] R. Frickey, "Data integrity on 20 nm SSDs," presented at the Flash Memory Summit, Santa Clara, CA, USA, 2012.

[18] J.-W. Hsieh, T.-W. Kuo, andL.-P. Chang,"Efficient identification of hot data for flash memory storage systems," *ACM Trans. Storage*, vol. 2, no. 1, pp. 22–40, 2006.

[19] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: Practical power management for enterprise storage," in *Proc. 6th USENIX Conf. File Storage Technol.*, 2008, pp. 253–267.
[20] J. Lee, Y. Kim, G. M. Shipman, S. Oral, and J. Kim, "Preemptible I/O scheduling of garbage collection for solid state drives," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 32, no. 2, pp. 247–260, Feb. 2013.
[21] 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 Trans. Inf. Syst.*, vol. E96-D, no. 5, pp. 1078–1085, May 2013.
[22] J. Jeong, Y. Song, and J. Kim, "FlashDefibrillator: A data recovery technique for retention failures in NAND flash memory," in *Proc. IEEE Non-Volatile Memory Syst. Appl. Symp.*, 2015, pp. 1–6.
[23] 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 *Proc. 51st Annu. Des. Autom. Conf.*, 2014, pp. 1–6.
[24] Y. Cai, et al., "Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime," in *Proc. IEEE Int. Conf. Comput. Des.*, 2012, pp. 94–101.

Jaeyong Jeong received the BS and MS degrees in radio science and engineering from Korea University, Korea, in 1996 and 1998, respectively, and the PhD degree in computer science and engineering from Seoul National University, Korea, in 2016. From 1998 to 2012, he was a senior engineer in the Memory Division, Samsung Electronics, Korea. He is currently a principal engineer at the same company. His research interests include embedded software and storage systems as well as high-density 2D/3D NAND flash memory.

Youngsun Song received the BS and MS degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2005 and 2007, respectively. He is currently working toward the PhD degree in computer science and engineering at Seoul National University, Korea. In 2007, he joined Samsung Electronics, Korea. From 2007 to 2014, he worked on the development of new material NVM devices. His research interests include nonvolatile memory, embedded software, and storage systems

Sangwook Shane Hahn received the BS degree in computer science from Korea Advanced Institute of Science and Technology, in 2011, and the MS degree in computer science and engineering from Seoul National University, Korea, in 2013. He is currently working toward the PhD degree in computer science and engineering at Seoul National University. His research interests include storage systems, operating systems, and embedded software.

Sungjin Lee received the BE degree in electrical engineering from the Korea University, in 2005, and the MS and PhD degrees in computer science and engineering from Seoul National University, in 2007 and 2013, respectively. He is an assistant professor with the Inha University. Before coming to the Inha University, he was a postdoctoral associate in the Computation Structures Group (CSG), MIT CSAIL, where he worked on the development of system software for high-performance storage systems, in particular, for non-volatile memory systems. His current research interests include storage systems, operating systems, and system software.

Jihong Kim received the BS degree in computer science and statistics from Seoul National University (SNU), Korea, in 1986, and the MS and PhD degrees in computer science and engineering from the University of Washington, Seattle, in 1988 and 1995, respectively. Before joining SNU in 1997, he was a technical staff member in the DSPS R&D Center, Texas Instruments, Dallas, Texas. He is currently a professor in the Department of Computer Science and Engineering, Seoul National University. His research interests include embedded software, low-power systems, computer architecture, and storage systems. He is a member of the IEEE.