# 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**—Replacing HDDs with NAND flash-based SSDs has been one of the main concerns in enterprise storage systems. Although uninterrupted semiconductor process scaling lowers the price for 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 another major barrier to a wide adoption of SSDs. In this paper, we propose an integrated approach, called dynamic erase voltage and time scaling (DeVTS), for extending the lifetime (particularly, endurance) of NAND flash memory. The DeVTS approach is based on our key observations from the NAND physics that slowly erasing a NAND block with a lower erase voltage can significantly improve the NAND endurance. In order to utilize our observations, we exploit that when I/O requests are not so intensive, the NAND program speed can be slowed down so that endurance-enhancing erase modes are frequently used in those cases. We also exploit that for frequently updated data, the NAND retention requirement can be relaxed so that the erase voltage is further reduced. By modifying NAND chips to support multiple write and erase modes with different voltages and times, we have implemented the DeVTS-aware FTL, called autoFTL+, which exploits the tradeoff relationships between the NAND endurance and erase voltage/time under optimal program time and retention tuning. Our experimental results show that autoFTL+ can improve the NAND endurance by 94.2% over an existing DeVTS-unaware FTL with less than only 0.1% decrease in the overall write throughput while preserving the retention requirement.

**Index Terms**—NAND flash memory, performance, reliability, lifetime, FTL.

## 1 Introduction

NAND flash-based solid-state drives (SSDs) are widely used in personal computing systems as well as mobile embedded systems. On the other hand, in enterprise environments, SSDs are adopted in only limited applications because SSDs are not yet cost competitive with HDDs [2]. Fortunately, the recent prices for SSDs are fallen to the comparable level of HDDs by continuous semiconductor process scaling (e.g., 10 nm-node process) combined with innovative advances in circuit technologies (e.g., MLC and TLC). However, the limited endurance of NAND flash memories, which declines further as a side effect of recent advanced device technologies, is emerging as another major barrier against a wide adoption of SSDs. (*The NAND endurance* is the ability of a memory cell to endure program/erase (P/E) cycling-induced stress, and is quantified as *the maximum number $N_{P/E}^{max}$ of P/E cycles* that a cell can tolerate while maintaining the reliability requirements [3].) For example, although the NAND capacity per die doubles every two years, the actual lifetime (which is proportional to the NAND capacity and $N_{P/E}^{max}$) of SSDs does not increase as much as projected in the past seven years because $N_{P/E}^{max}$ declined by 70% during that period [4]. In order for SSDs to be the mainstream of enterprise storage systems, the decreasing NAND endurance problem 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**), which can significantly improve the NAND endurance by slowly erasing a NAND block with a lower erase voltage. The key motivation of DeVTS, from our NAND device physics study on the endurance degradation, is that the NAND endurance is mainly degraded during erase operations. Since the probability of the oxide damage (which is known as the main cause of the NAND endurance degradation) has an exponential dependence on the stress voltage [6], reducing the stress voltage (i.e., the erase voltage) is the most effective way of improving the NAND endurance. Moreover, for a given erase operation, since a nominal erase voltage tends to excessively damage NAND memory cells in the beginning period of an erase operation [7], 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 excessive damage [8] [1], thus additionally improving the NAND endurance. By modifying a NAND device to support multiple erase voltage and time scaling modes (which have different impacts on the 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 necessary to use some special write modes that partially relax the NAND performance or retention capabilities. Since the *Vth* window, the total width of a *Vth* margin for a NAND cell, is tightly assigned to the specified NAND requirements (i.e., endurance, performance and retention time), in order to assign more *Vth* margin to endurance (i.e., to lower the

- *An earlier version of this paper was presented at the USENIX Conference on File and Storage Technologies, February 17–20, 2014 [1].*
- *J. Jeong, 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: {jyjeong, ysunsong, shanehahn, jihong}@davinci.snu.ac.kr.*
- *S. Lee is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139.*
*E-mail: chamdoo@csail.mit.edu.*

erase voltage), it is needed to reduce the *Vth* margin for other requirements. For example, a slow write mode with a fine-grained program control can shorten the width of a *Vth* distribution so that the *Vth* margin for performance can be saved while the NAND program time increases [1]. For another example, a short-lived write mode which reduces the *Vth* gap between two adjacent *Vth* states can save the *Vth* margin for retention while the retention capability is sacrificed [5] [9].

In order to estimate the impact of the special write modes (i.e., slow write modes or short-lived write modes) on the NAND endurance, we develop a comprehensive NAND endurance model which accurately captures the tradeoff relationships between the NAND endurance and the NAND performance/retention characteristics. Based on the NAND endurance model, when a slow or short-lived 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 the NAND endurance.

Our DeVTS approach exploits the tradeoff relationships between the NAND endurance and the NAND performance/retention capability at the software level so that the NAND endurance is improved while the overall write performance and data retention requirements of SSDs are not affected. For example, when incoming write requests are not so intensive that the maximum performance of NAND devices in an SSD is not fully needed, a DeVTS-enabled technique can exploit idle times between consecutive write requests for tuning the the program or erase speed as slowly as possible (i.e., select the most appropriate mode among the write-speed modes or the erase-speed modes). For another example, when some of written data are frequently updated and do not need long data retention times, a DeVTS-enabled technique can decide to tune down the retention capability of such data as low as possible (i.e., select the most suitable mode among the write-age modes). If such long idle times and short retention requirements can be automatically identified by a system software, the DeVTS-enabled technique can select the most appropriate speed/age mode for each program operation, or choose the most suitable voltage/speed mode for each erase operation. By actively employing endurance-enhancing erase modes (i.e., a slow erase mode with a lower erase voltage) when long idle times are detected or written data are frequently updated, $N_{P/E}^{max}$ can be significantly increased because less damaging erase operations are more frequently used.

We have implemented the DeVTS-aware FTL, called *autoFTL+*, which dynamically changes erase voltage and speed modes in an *automatic* fashion under properly tuning the performance and retention capabilities of write requests. AutoFTL+ determines 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-age mode, the existing data separator in an SSD is redesigned to conservatively predict the future update time of a write request. When it is expected that the written data will not be updated until the predicted retention time, a data reclaim process is proactively invoked to avoid the retention failure. Key FTL modules (e.g., mapping table, garbage collector, and wear leveler) were also revised to make them DeVTS-aware for maximizing the efficiency of autoFTL+. We evaluated the effectiveness of autoFTL+ with an extended *FlashBench* emulation environment [10] where a DeVTS-enabled NAND simulation model is integrated. Our experimental results using various I/O traces extracted from enterprise servers show that autoFTL+ can increase $N_{P/E}^{max}$ by 94.2% on average over an existing DeVTS-unaware FTL with less than only 0.1% decrease in the overall write throughput while completely preserving the retention requirement of SSDs.

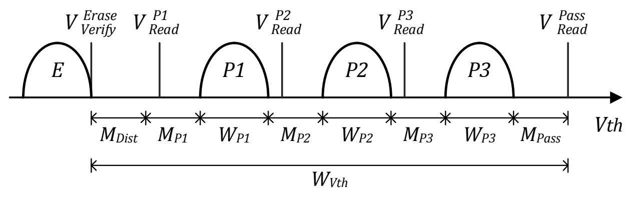


Fig. 1. An example of *Vth* distributions for MLC NAND flash memory and primary *Vth* design parameters for the NAND requirements.

The rest of the paper is organized as follows. Before our proposed techniques are presented, we briefly review the design principle of NAND flash memories in Section 2. Section 3 explains the erase voltage and time scaling. In Section 4, we present the write tuning methods and the NAND simulation model. In Section 5, we describe our DeVTS-aware FTL in detail, and report experimental results including the endurance gain in Section 6. After Section 7 summarizes related works, we finally conclude with summary in Section 8.

## 2 Background

NAND flash memory stores data into cells by electrically 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 MLC NAND flash memory which stores two bits in a cell by using four distinct *Vth* states, i.e., $E$, $P1$, $P2$ and $P3$. Four *Vth* states are distinguished by three read reference voltages, i.e., $V_{Read}^{P1}$, $V_{Read}^{P2}$ and $V_{Read}^{P3}$.

Not only to just serve as a non-volatile storage medium, NAND flash memory is required to meet the performance and reliability requirements (e.g., endurance, retention and read disturbance [3]). For example, read and program operations of an MLC device should be completed within 100 $\mu$s and 1,600 $\mu$s, on average, respectively [4]. At the same time, even after 3,000 program/erase (P/E) cycles, it is required to endure up to 400,000 read operations [4] as well as to retain its stored data for up to 1 year at 30 °C [12]. Since the *Vth* design parameters as shown in Fig. 1 are mainly affected by the NAND requirements, the total width $W_{Vth}$ of Vth distributions should be carefully designed to meet all the NAND requirements under the worst-case operating conditions of a storage product.

The upper *Vth* target $V_{Verify}^{Erase}$ of the $E$ state is one of the key factors to decide $W_{Vth}$. As $V_{Verify}^{Erase}$ is lowered, $W_{Vth}$ can be widened so that it is easy to design *Vth* distributions for a higher performance or a longer retention capability. However, as the side-effect of the lower $V_{Verify}^{Erase}$, the NAND endurance may get worse because a NAND block is more deeply erased [1]. On the other hand, when a higher $V_{Verify}^{Erase}$

is used, the complexity of designing $Vth$ distributions increases because an available $W_{Vth}$ is reduced.

The width $W_{Pi}$ of a $Vth$ distribution is mainly 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 decided 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 makes stored charges loosened. 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 decided by the maximum $Vth$ change after the maximum number of P/E cycles and the specified retention-time requirement of a storage device.

The $Vth$ gap $M_{Dist}$ between the $E$ state and $V_{Read}^{P1}$ mainly affects the program-disturbance resistance and the read-disturbance resistance of NAND flash memory. When NAND memory cells are programmed or read, neighbor cells which belong to the $E$ state may be softly programmed so that their $Vth$'s move to the right [3] [5]. In order to cope with the $Vth$ changes due to the program/read disturbance, a sufficient $M_{Dist}$ should be reserved in the $Vth$ window as shown in Fig. 1. In general, the $M_{Dist}$ requirement is conservatively decided by the maximum $Vth$ changes after the maximum number of P/E cycles and the maximum number of read cycles.

The read pass voltage $V_{Read}^{Pass}$ which affects the NAND read disturbance is another key factor to decide $W_{Vth}$. Since the NAND read disturbance has an exponential dependence on the $V_{Read}^{Pass}$ [13], $V_{Read}^{Pass}$ is usually fixed in device design times as low as possible. As shown in Fig. 1, the $Vth$ gap $M_{Pass}$ between the $P3$ state and $V_{Read}^{Pass}$ is essential to fully turn on all the NAND memory cells in a block [3].

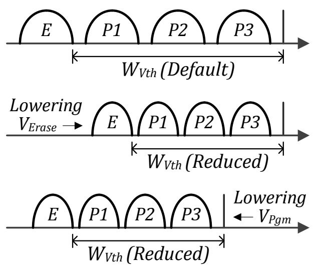
Based on the proper $Vth$ design parameters, program $Vth$ states (i.e., $P1$, $P2$ and $P3$) are placed between $V_{Verify}^{Erase}$ and $V_{Read}^{Pass}$. Therefore, the total width $W_{Vth}$ of the $Vth$ window can be expressed as follows (for MLC NAND flash memory):

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

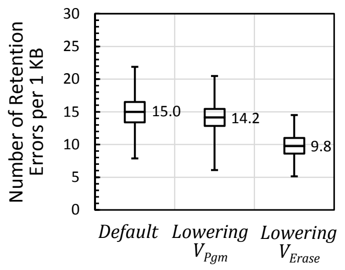
Since the $Vth$ design parameters are highly inter-related each other, in order to change a certain capability of NAND flash memory, we should check the effect of it on the whole $Vth$ window.

## 3 Erase Voltage and Time Scaling

The key intuition of the DeVTS approach is that changing the erase voltage as well as the erase time can significantly



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



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

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

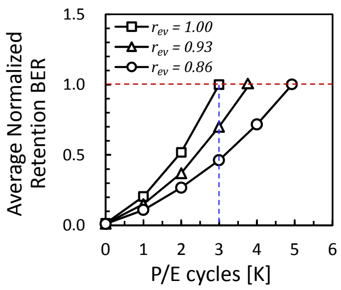
affects the NAND endurance. In this section, we describe the motivation of DeVTS and present the effect of the erase voltage/time scaling on improving the NAND endurance.

### 3.1 Motivation

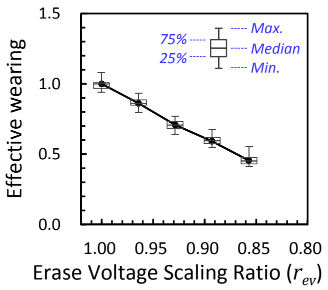
The physical mechanism of the NAND endurance degradation is closely related to stress-induced damage in the tunnel oxide layer of a NAND memory cell [5]. Since the probability of stress-induced damage has an exponential dependence on the stress voltage [6], lowering the stress voltage (i.e., the program voltage $V_{Pgm}$ and the erase voltage $V_{Erase}$) during P/E cycles is the most effective way of improving the NAND endurance.

Although the maximum $V_{Pgm}$ to complete a program operation is usually higher than $V_{Erase}$, the NAND endurance is mainly 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 to a cell is randomly changed (e.g., by a data randomizer [14]), the probability that the cell consecutively experiences the maximum $V_{Pgm}$ during P/E cycles is very low. On the other hand, all the cells in a NAND block always experience $V_{Erase}$ during P/E cycles. Therefore, we can assume that changing $V_{Erase}$ has a more significant impact on the NAND endurance.

In order to verify our assumption, we evaluated the effects of two different stress-voltage-reduction policies on the NAND endurance as shown in Fig. 2(a). In the 'lowering $V_{Erase}$' policy, $W_{Vth}$ shrinks to the right direction (compared to the default case) so that $V_{Erase}$ can be lowered 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 $V_{Pgm}$ can be lowered while $V_{Erase}$ is maintained. In our evaluations, ten blocks out of two 20-nm node NAND chips were selected for each policy. As a 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 [15]) per 1-KB unit because it reflects the effective degree of the NAND wearing [1]. As shown in Fig. 2(b), when the 'lowering $V_{Pgm}$' policy was used, the number of retention errors was reduced by only 5.3% ($= 1 - 14.2/15.0$), on average, over the default case. On the other hand, when the 'lowering $V_{Erase}$' policy was used, the number of retention errors was reduced by 34.7% ($= 1 - 9.8/15.0$), on average, over the



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



(b) Effective wearing variations over different $r_{ev}$'s.

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

default case. These results clearly show that lowering $V_{Erase}$ is much more effective on improving the NAND endurance over lowering $V_{Pgm}$.

## 3.2 Erase Voltage Scaling and Its Effect on Endurance

In order to evaluate the effect of erase voltage scaling on the NAND endurance, we performed NAND cycling tests by using different $V_{Erase}$'s. In a 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 for quantifying the 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. 3(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 Fig. 3(a), the more $V_{Erase}$ is reduced (i.e., the lower $r_{ev}$'s), the less 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 degrade the NAND endurance by different amounts, we introduce a new endurance metric, called *effective wearing*, which represents the effective degree of NAND wearing after a P/E cycle. Based on a linear approximation model [7] which simplifies the NAND wear-out behavior over P/E cycles as shown in Fig. 3(a), we represent the effective wearing by a normalized retention BER after 3K P/E cycles. For example, when $V_{Erase}^{nominal}$ is used (i.e., $r_{ev} = 1.00$), the effective wearing is 1.00. On the other hand, when $V_{Erase}$ is reduced by 7% (i.e., $r_{ev} = 0.93$), the effective wearing becomes 0.70. As shown in Fig. 3(b), since the effective wearing has a near-linear dependence on $r_{ev}$, the 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 the NAND endurance can be estimated by accumulating the effective wearing for each P/E cycle. After 3K P/E cycles, for example, the total sum $\Sigma EW$ of the effective wearing with $V_{Erase}^{nominal}$ is 3,000 $(= 1.00 \times 3000)$. On the other hand, when $r_{ev}$ is set to 0.93, $\Sigma EW$ is only 2,100 $(= 0.70 \times 3000)$. Since the NAND reliability can be maintained until $\Sigma EW$ 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 $V_{Erase}^{nominal}$.

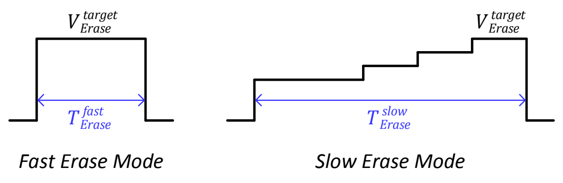
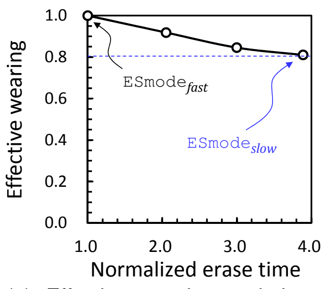
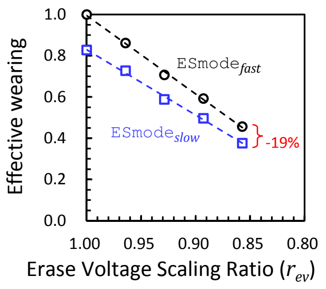


Fig. 4. An illustration of an erase voltage control for the proposed erase time scaling.



(a) Effective wearing variations over different erase times.



(b) Effective wearing variations over varying $r_{ev}$'s under two different erase time settings.

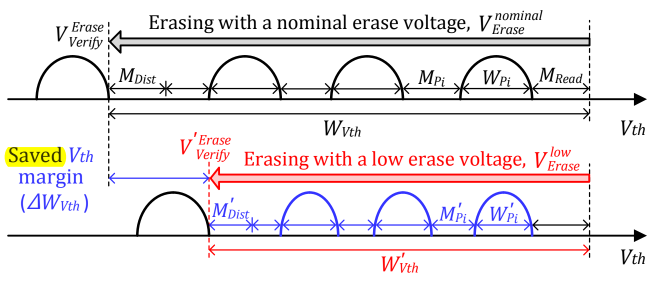
Fig. 5. The effect of erase time scaling on the NAND endurance.

## 3.3 Erase Time Scaling and Its Effect on Endurance

The endurance degradation is mainly proportional to the target erase voltage $V_{Erase}^{target}$ in an erase operation as described in Section 3.2. When $V_{Erase}^{target}$ is applied to a NAND block, however, NAND memory cells can be temporarily over-damaged by $V_{Erase}^{target}$. Since the actual erase voltage across the tunnel oxide layer is the sum of $V_{Erase}^{target}$ and the *Vth* of a cell [7], an unintended higher (than $V_{Erase}^{target}$) 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 over those have lower *Vth*'s (e.g., the $E$ state).

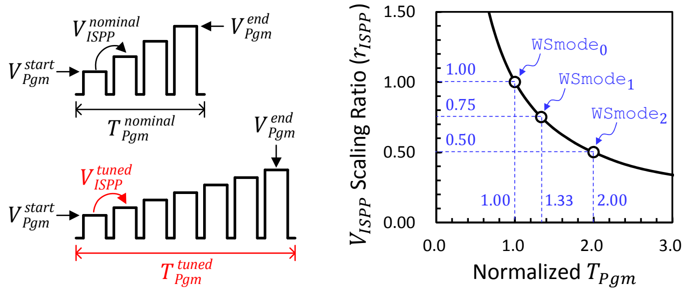
In order to minimize additional damage in the beginning period of an erase operation, it is necessary to properly control an applied $V_{Erase}$ so that the actual voltage across the tunnel oxide layer does not exceed $V_{Erase}^{target}$ throughout the erase operation. We implemented this idea by modifying the existing incremental step pulse erasing (ISPE) scheme [16] in that the applied $V_{Erase}$ gradually increased from a low voltage (e.g., $V_{Erase}^{target}$ − the average *Vth* of the $P3$ state) to $V_{Erase}^{target}$ 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., $T_{Erase}^{slow}$ as shown in Fig. 4) inevitably increases because more ISPE loops are needed for completing the erase operation.

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



$V^{Erase}_{Verify}$
Erasing with a nominal erase voltage, $V^{nominal}_{Erase}$
$M_{Dist}$
$M_{Pi}$
$W_{Pi}$
$M_{Read}$
$W_{Vth}$
Vth
Saved Vth margin ($\Delta W_{Vth}$)
$V'^{Erase}_{Verify}$
Erasing with a low erase voltage, $V^{low}_{Erase}$
$M'_{Dist}$
$M'_{Pi}$
$W'_{Pi}$
$W'_{Vth}$
Vth

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



(a) An illustration of the program time tuning technique.

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

Fig. 7. The proposed program time tuning.

the erase time increases 3 times, the effective wearing is reduced, on average, by 19%. We represent the erase speed mode with a default erase time by `ESmode`$_{fast}$ while the erase speed mode with a long erase time by `ESmode`$_{slow}$. As shown in Fig. 5(b), the effect of `ESmode`$_{slow}$ on improving the NAND endurance can be exploited regardless of $r_{ev}$ whenever longer erase times are acceptable.

## 4 Program Time and Retention Tuning

If a NAND block is *shallowly erased* (i.e., erased with a lower erase voltage), the available *Vth* window for a program operation is also reduced. This is because $W_{Vth}$ is mainly decided by $V^{Erase}_{Verify}$ (which affects the requirement of $V_{Erase}$) as explained in Section 2. For example, as shown in Fig. 6, if a NAND block is shallowly erased with a low erase voltage $V^{low}_{Erase}$ (which is lower than $V^{nominal}_{Erase}$), $W_{Vth}$ is reduced by a saved *Vth* margin $\Delta W_{Vth}$ (which is proportional to the voltage difference between $V^{nominal}_{Erase}$ and $V^{low}_{Erase}$). Since *Vth* distributions should be formed only within the given *Vth* window, when $V^{low}_{Erase}$ is used in an erase operation, it is necessary to use some special write modes which adjust *Vth* design parameters (e.g., $W_{Pi}$, $M_{Pi}$ and $M_{Dist}$) so that $W_{Vth}$ can be saved by at least $\Delta W_{Vth}$. In this section, we describe program-time tuning and retention-capability tuning methods for saving $W_{Vth}$ and present the NAND endurance model for estimating the impact of the proposed tuning methods on the NAND endurance.

### 4.1 Dynamic Program Time Tuning

In order to reduce $W_{Pi}$'s, a fine-grained ISPP step control is necessary because $W_{Pi}$ is directly proportional to the ISPP step voltage $V_{ISPP}$ [11]. However, since the number of ISPP loops for completing a program operation is inversely proportional to $V_{ISPP}$ [1], the program time $T_{Pgm}$ inevitably increases as shown in Fig. 7(a) if narrow *Vth* distributions are required. Fig. 7(b) shows how much $V_{ISPP}$ can be reduced as $T_{Pgm}$ increases. $T_{Pgm}$ was normalized over the nominal program time $T^{nominal}_{Pgm}$ (e.g., 1,300 $\mu$s [14]). We denote $V_{ISPP}$ scaling ratio over the nominal ISPP step voltage $V^{nominal}_{ISPP}$ by $r_{ISPP}$ ($0 \leq r_{ISPP} \leq 1$). When $r_{ISPP}$ is set to $x$, $V_{ISPP}$ is reduced by $(1-x) \times V^{nominal}_{ISPP}$.

In our proposed program-time tuning method, we define three different write-speed modes, `WSmode`$_0$, `WSmode`$_1$ and `WSmode`$_2$, as shown in Fig. 7(b). `WSmode`$_0$ is the fastest write mode which has the same $T_{Pgm}$ as that of the nominal write mode, but cannot reduce $V_{ISPP}$. On the other hand, `WSmode`$_2$ is the slowest write mode which has a two times longer $T_{Pgm}$ (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 $V^{nominal}_{ISPP}$.

Since $W_{Pi}$ has a linear dependence on $V_{ISPP}$ (which can be determined by the $T_{Pgm}$ requirement as shown in Fig. 7(b)), $\Delta W_{Vth}$ by tuning $T_{Pgm}$ can be expressed as follows (for MLC NAND flash memory):

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

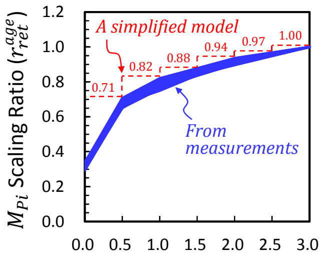
If $V^{nominal}_{ISPP}$ was 400 mV, for example, when two times longer $T_{Pgm}$ is acceptable, $W_{Vth}$ can be saved by 600 mV ($= 3 \times ((1-0.50) \times 400$ mV)).

### 4.2 Retention Capability Tuning

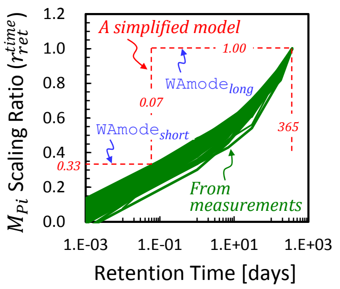
NAND flash memory is required to retain its stored data for the specified retention time (e.g., 1 year in client-class applications at room temperature [12]). In order to guarantee the NAND retention requirement throughout the storage lifespan, $M_{Pi}$'s are usually *fixed* during device design times to cover the maximum *Vth* shift 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 amount of *Vth* shift due to the charge-loss phenomenon is proportional to the number of P/E cycles [5], 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 amount of *Vth* shift is also proportional to the retention time [5], when written data are frequently updated within a short time period, $M_{Pi}$ for such frequently updated data can be further reduced.

#### 4.2.1 *Age-dependent 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 with different numbers of P/E cycles. A cycling test for each case were performed with more than 2,000 NAND pages (from 20 blocks out of 2 NAND chips). After cycling tests, we measured the average amount of *Vth* changes for each block after 1 hour's baking at 100 °C. Measured average *Vth* changes were normalized over the maximum required *Vth* margin $M^{max}_{Pi}$ under the worst-case



(a) Age-dependent $M_{Pi}$ scaling ratio ($r_{ret}^{age}$) variations over different numbers of P/E cycles.



(b) Time-dependent $M_{Pi}$ scaling ratio ($r_{ret}^{time}$) variations over different retention times.

Fig. 8. Simplified $M_{Pi}$ scaling models for retention capability tuning.

operating condition (i.e., 3K P/E cycles and 1-year retention time). We denote the normalized average *Vth* changes over different numbers of P/E cycles by the age-dependent $M_{Pi}$ scaling ratio $r_{ret}^{age}$. Fig. 8(a) shows $r_{ret}^{age}$ variations over different numbers of P/E cycles. After 0.5K P/E cycles, for example, only 71% of $M_{Pi}^{max}$ is required (i.e., $r_{ret}^{age}$ is 0.71). Based on the measurement results, we construct a simplified age-dependent $M_{Pi}$ scaling model where $r_{ret}^{age}$ is changed every 0.5K P/E cycle interval as shown by dotted line in Fig. 8(a). For a given number of P/E cycles, $\Delta W_{Vth}$ by tuning the NAND retention capability can be expressed as follows (for MLC NAND flash memory):

$$\Delta W_{Vth} = \sum_{i=1}^{3} \Delta M_{Pi} = \sum_{i=1}^{3} (1 - r_{ret}^{age}) \times M_{Pi}^{max}. \qquad (3)$$

If the sum of three $M_{Pi}^{max}$'s was 900 mV, for example, when the number of P/E cycles is less than 0.5K, $W_{Vth}$ can be saved by 261 mV (= (1 − 0.71) × 900 mV).

### 4.2.2 Time-dependent Dynamic Retention Tuning

In order to decide how much $M_{Pi}$ is required as the retention time increases, we performed NAND cycling tests with different retention times and measured the average amount of *Vth* changes after 1 hour's baking at 100 °C. Measured average *Vth* changes were normalized over $M_{Pi}^{max}$. We denote the normalized average *Vth* change over different retention times by the time-dependent $M_{Pi}$ scaling ratio $r_{ret}^{time}$. Solid lines in Fig. 8(b) show $r_{ret}^{time}$ variations over different retention times with more than 2,000 NAND pages. In order to minimize the management overhead, we simplify the $r_{ret}^{time}$ changes over different retention times into two different write-age modes (i.e., WAmode$_{short}$ and WAmode$_{long}$) as shown by dotted line in Fig. 8(b). WAmode$_{long}$ is the long-lived write mode which fully supports the specified retention time (i.e., 1 year), but cannot save $M_{Pi}$ (i.e., $r_{ret}^{time}$ is 1.00). On the other hand, WAmode$_{short}$ is the short-lived write mode which supports only the 0.07-day retention time while requires only 33% of $M_{Pi}^{max}$ (i.e., $r_{ret}^{time}$ is 0.33). By combining $r_{ret}^{age}$ with $r_{ret}^{time}$, Equation 3 (i.e., $\Delta W_{Vth}$ by tuning the NAND retention capability) can be re-expressed as follows (for MLC NAND flash memory):

$$\Delta W_{Vth} = \sum_{i=1}^{3} \Delta M_{Pi} = \sum_{i=1}^{3} (1 - r_{ret}^{age} \times r_{ret}^{time}) \times M_{Pi}^{max}. \qquad (4)$$

TABLE 1
A simplified $r_{dist}$ model over different numbers of 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 |

For example, when the number of P/E cycles is less than 0.5K, and the retention-time requirement is less than 0.07 day, $W_{Vth}$ can be saved by 689 mV (= (1 − 0.71 × 0.33) × 900 mV).

## 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 [5], we measured how much $M_{Dist}$ is required as the number of P/E cycles increases. After performing NAND cycling tests with different numbers of P/E cycles and the specified number (i.e., 400K [4]) of read cycles, we measured the average amount of *Vth* changes in the *E* state. Our tests were performed with more than 2,000 NAND pages. Measured average *Vth* changes were normalized over the maximum required *Vth* margin $M_{Dist}^{max}$ under the worst-case operating condition (i.e., 3K P/E cycles and 400K read cycles). We denote the normalized average *Vth* changes due to the NAND disturbance by $r_{dist}$ ($0 \le r_{dist} \le 1$). Table 1 summarizes our simplified $r_{dist}$ model over different numbers of 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, $\Delta W_{Vth}$ by tuning the NAND disturbance resistance can be expressed as follows:

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

If $M_{Dist}^{max}$ was 400 mV, when the number of P/E cycles is less than 0.5K, $W_{Vth}$ can be saved by 228 mV (= (1 − 0.43) × 400 mV).

## 4.4 NAND Endurance Model

Combining the proposed write tuning techniques (i.e., program time 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 $\Delta W_{Vth}$ for each combination of write-tuning modes by using Equations 2, 4 and 5. Since a reduced erase voltage (= $(1 - r_{ev}) \times V_{Erase}^{nominal}$) is proportional to $\Delta W_{Vth}$, $r_{ev}$ can be re-expressed as follows:

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

where $\alpha_c$ is the empirical scaling ratio which represents the impact of the $V_{Erase}$ change on the *Vth* window. If $\alpha_c$ is 0.60, for example, when $V_{Erase}$ is reduced by 1.00 V, $W_{Vth}$ can effectively decrease by 0.60 V. When $r_{ev}$ is calculated from Equation 6 for a given $\Delta W_{Vth}$, the corresponding effective wearing for $r_{ev}$ can be estimated by a linear equation as described in Section 3. Table 2 summarizes the parameter set used to construct the NAND endurance model in this

TABLE 2
An example of a parameter set used to estimate the effective wearing.

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

TABLE 3
The NAND endurance (i.e., the effective wearing) model over the total sum of the effective wearing (i.e., $\Sigma EW$) under write tuning modes.

| EVmode | 0 | 1 | 3 | 2 | 4 | 5 |
|---|---|---|---|---|---|---|
| ESmode | *fast* | | | | | |
| WAmode | *long* | | | *short* | | |
| WSmode | 0 | 1 | 2 | 0 | 1 | 2 |
| $0 \leq \Sigma EW \leq 0.5K$ | 0.78 | 0.65 | 0.52 | 0.59 | 0.46 | 0.33 |
| $0.5K < \Sigma EW \leq 1.0K$ | 0.83 | 0.69 | 0.56 | 0.62 | 0.49 | 0.36 |
| $1.0K < \Sigma EW \leq 1.5K$ | 0.89 | 0.76 | 0.63 | 0.67 | 0.53 | 0.40 |
| $1.5K < \Sigma EW \leq 2.0K$ | 0.96 | 0.83 | 0.69 | 0.71 | 0.57 | 0.44 |
| $2.0K < \Sigma EW \leq 2.5K$ | 0.98 | 0.85 | 0.71 | 0.72 | 0.59 | 0.45 |
| $2.5K < \Sigma EW \leq 3.0K$ | 1.00 | 0.87 | 0.73 | 0.73 | 0.60 | 0.47 |

| EVmode | 0 | 1 | 3 | 2 | 4 | 5 |
|---|---|---|---|---|---|---|
| ESmode | *slow* | | | | | |
| WAmode | *long* | | | *short* | | |
| WSmode | 0 | 1 | 2 | 0 | 1 | 2 |
| $0 \leq \Sigma EW \leq 0.5K$ | 0.68 | 0.57 | 0.45 | 0.52 | 0.40 | 0.29 |
| $0.5K < \Sigma EW \leq 1.0K$ | 0.72 | 0.60 | 0.49 | 0.54 | 0.43 | 0.31 |
| $1.0K < \Sigma EW \leq 1.5K$ | 0.78 | 0.66 | 0.55 | 0.58 | 0.46 | 0.35 |
| $1.5K < \Sigma EW \leq 2.0K$ | 0.83 | 0.72 | 0.60 | 0.62 | 0.50 | 0.38 |
| $2.0K < \Sigma EW \leq 2.5K$ | 0.85 | 0.74 | 0.62 | 0.63 | 0.51 | 0.40 |
| $2.5K < \Sigma EW \leq 3.0K$ | 0.87 | 0.75 | 0.64 | 0.64 | 0.52 | 0.41 |

paper. All the data in our model are based on measurement results with 20-nm node NAND chips.

Table 3 shows our DeVTS-enabled NAND endurance (i.e., the 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 maximally improve the NAND endurance while $\texttt{EVmode}_0$ is the highest erase-voltage mode which has the lowest endurance gain. $\texttt{EVmode}_i$'s are decided by the combinations of two write-age modes (i.e., $\texttt{WAmode}_{long}$ and $\texttt{WAmode}_{short}$) and three write-speed modes (i.e., $\texttt{WSmode}_0 \sim \texttt{WSmode}_2$) because $r_{ev}$'s are different for each combination. Since $\Delta W_{Vth}$'s are also affected by age-dependent static retention tuning and disturbance resistance tuning, the values of the effective wearing vary whenever $\Sigma EW$ exceeds 0.5K. For example, if $\Sigma EW$ is less than 0.5K, when a NAND block is slowly erased before writing with the short-lived write mode (i.e., $\texttt{WAmode}_{short}$) and the slowest write-speed mode (i.e., $\texttt{WSmode}_2$), the lowest erase voltage (i.e., $\texttt{EVmode}_5$) can be used for an erase operation. In this case, the effective wearing is only 0.29.

## 5 Design and Implementation of autoFTL+

### 5.1 Overview of autoFTL+

In order to improve the NAND endurance without affecting the other NAND requirements, we have implemented DeVTS-aware FTL, called *autoFTL+*, which automatically changes erase-scaling modes and write-tuning modes based on the NAND endurance model described in Section 4.4. Figure 9 illustrates an organizational overview of autoFTL+ which is 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-tuning mode for a given write request depending on the performance and retention requirements. Firstly, the write-speed mode ($\texttt{WSmode}_i$) or erase-speed mode ($\texttt{ESmode}_k$) is selected based on the estimated write-performance requirement using a write buffer. Secondly, the write-age mode ($\texttt{WAmode}_m$) is decided based on the predicted retention-time requirement using a retention-time predictor. Finally, the DeVTS manager determines the erase-voltage mode $\texttt{EVmode}_j$ considering the selected write-tuning modes. In order to preserve the retention requirement, *the retention keeper* periodically checks the age of written data and rewrites old-aged data to another NAND page before their age exceeds the specified retention time. The extended mapping table maintains *per-block mode information* (e.g., the erase/write mode, the age and $\Sigma EW$ for each block) as well as logical-to-physical mapping information. When the write mode or erase mode is changed, autoFTL+ reconfigures NAND chips through a new device setting interface, *DeviceSettings*, by consulting *the NAND setting table*. The existing garbage collector and wear leveler are also revised for maximizing the efficiency of DeVTS.

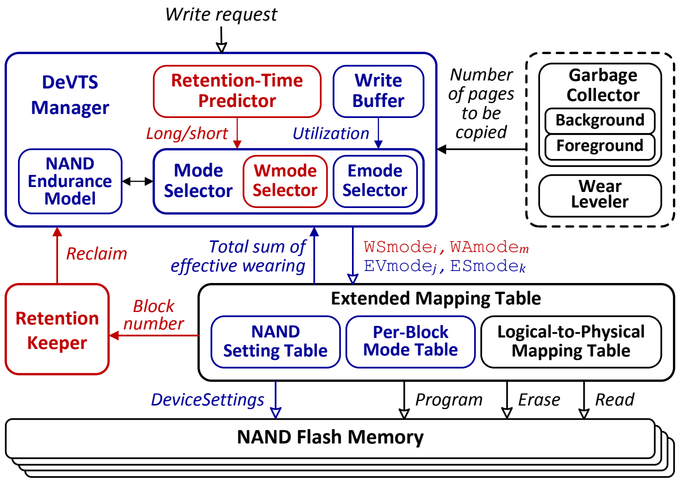


Fig. 9. An organizational overview of autoFTL+.

### 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), *the Wmode selector* in the DeVTS manager estimates the write-performance requirement for a given write request based on the utilization of a write buffer. Since the write buffer queues incoming write requests before they are written, the utilization $u^{wb}$ of the write buffer 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. Figure 10 shows an overview of the write-speed mode selection in autoFTL+. When write requests are

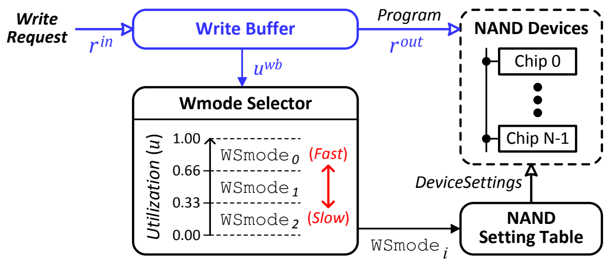


Fig. 10. An overview of the write-speed mode selection in autoFTL+.

incoming in a sporadic fashion (i.e., $r^{in} < r^{out}$), $u^{wb}$ may decrease. In this case, the Wmode selector estimates that the maximum write performance of NAND devices is not fully needed. On the other hand, when write requests are so intensive (i.e., $r^{in} > r^{out}$) that $u^{wb}$ increases, the Wmode selector decides that the write-performance requirement becomes more urgent.

In our implementation, the write-performance requirement is classified into three levels by two buffer utilization boundaries as shown in Figure 10. For example, when $u^{wb}$ is lower than 0.33, the write request queued in the write buffer can be programmed to a NAND page with WSmode$_2$ which is the slowest write mode. On the other hand, 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 WSmode$_0$ which is the fastest write mode.

Our proposed write-speed mode selection technique can efficiently adapt to not only varying $r^{in}$ but also changing $r^{out}$ (which is proportional to the number of available NAND chips which are ready to write data). When some NAND chips are not available due to internal FTL operations (e.g., garbage collection), $r^{out}$ can be significantly reduced [17]. For example, when garbage collection operations are performed in half of NAND chips, effective $r^{out}$ is reduced by 50%. If effective $r^{out}$ gets lowered below $r^{in}$ so that $u^{wb}$ increases, a faster write mode is better to mitigate the side effect of the internal FTL operations. Since our estimation metric is based on $u^{wb}$ which depends on both $r^{out}$ and $r^{in}$, the Wmode selector can select the most proper write-speed mode by taking into account of the variations in both $r^{in}$ and $r^{out}$.

## 5.3 Write-Age Mode Selection

The Wmode selector decides the most proper write-age 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 the write request will be updated within the predefined time period which is much shorter than the nominal retention-time specification of NAND devices, the Wmode selector selects the short-aged write mode (i.e., WAmode$_{short}$) for that request. When the prediction on the future retention-time requirement is incorrect, a background reclaim process [18] should be performed to preserve the durability of write-age tuned data. However, too frequent reclaim operations can partially cancel the lifetime benefit of DeVTS and incur substantial delays in serving user requests. In this section, we present a write-age mode selection procedure with the retention-time predictor and describe conservative write-age management techniques for maximizing the efficiency of DeVTS.

### 5.3.1 *Retention-Time Requirement Prediction*

Our proposed retention-time predictor estimates the future retention-time requirement of a write request based on the average update interval for each request in recent past. Since there are only two write-age modes in our NAND endurance model, it is necessary to classify whether the average update interval is shorter than the predefined short retention-time interval $T^{short}_{retention}$ (e.g., 0.07 day as defined in Section 4.2.2) or not. In our implementation, the retention-time predictor is based on the existing data separator [19] with different control policy for capturing the average update interval and for making a conservative decision on the write-age mode (as will be described in Section 5.3.2).

Each LBA is mapped to multi-dimensional counters which are incremented whenever corresponding write requests are incoming. In order to compare the update interval with $T^{short}_{retention}$, all the counters are decayed every pre-set time interval $T_{decay}$ (in this paper, $T_{decay} \leq T^{short}_{retention}$). If the update interval of an LBA is shorter than $T_{decay}$, the corresponding counter value will eventually increase. Otherwise, the counter values are maintained or decreased. After multiple decaying intervals, when the counter value is greater than the predefined threshold value, the retention-time predictor can decide that the recent update interval of that LBA is shorter than $T^{short}_{retention}$ on average. In this case, the retention-time predictor predicts that the current write request will be also updated within $T^{short}_{retention}$ by exploiting the temporal locality of I/O requests.

Figure 11 shows a functional overview of our proposed retention-time predictor. Since maintaining all the counters for each LBA may be too expensive to be implemented in practice, the proposed retention-time predictor keeps only 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.

### 5.3.2 *Conservative Write-Age Management Techniques*

Since the Wmode selector mainly relies on the retention-time predictor in selecting the write-age mode, improving the prediction accuracy for the retention-time requirement is one of the key design goals in autoFTL+. Moreover, since it is practically difficult to completely avoid the mispredicted writing, minimizing the overhead of a reclaim operation is also necessary to maximize the lifetime benefit as well as to minimize the performance penalty of autoFTL+. In order to achieve such design goals, we have developed several write-age management techniques in a conservative fashion.

*Adaptive Decision Threshold Control*

One of the main reasons of misprediction is *hash collision* in the hashing table as shown in Figure 11. When counters cor-

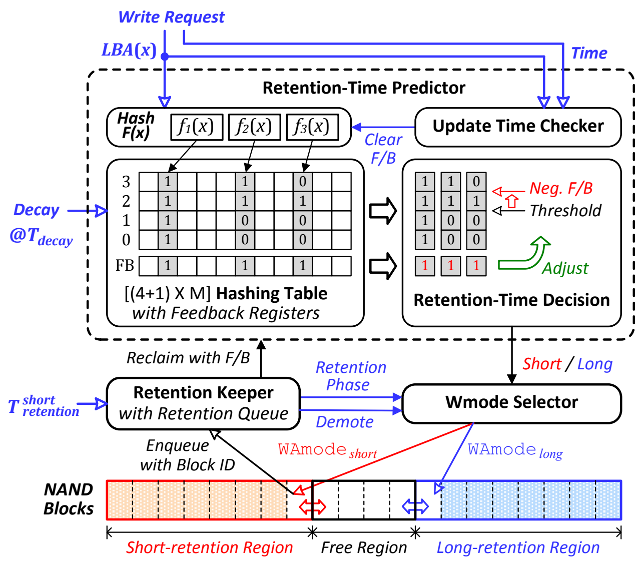


Fig. 11. A functional overview of the write-age mode selection procedure and the write-age management techniques.

responding cold data (which are rarely updated) are unintentionally incremented due to hash collision, such cold data may be mispredicted as the short-retention data. (We denote this misprediction by *false-short*.) Once mispredicted data are written with $\texttt{WAmode}_{short}$, such data should be rewritten to other NAND pages before $T^{short}_{retention}$. Too frequent rewriting caused by inaccurate prediction can unnecessarily degrade the NAND endurance. In order to minimize the misprediction ratio due to hash collision, we introduce an adaptive retention-time decision scheme based on the previous misprediction history. As shown in Figure 11, each counter has an additional (negative) feedback register which is set to one when the written data are reclaimed by the retention keeper. If all the corresponding feedback registers were set, the retention-time predictor conservatively decides the retention-time requirement by increasing the decision threshold level. For example, when the counter values are 15, 4 and 5, this request can be classified as a short-retention data with a normal threshold level (e.g., 4). On the other hand, if all the feedback registers were set, this request is classified as a long-retention data because all the counter values are not greater than the conservative threshold level (e.g., 8). When data written with $\texttt{WAmode}_{long}$ start to be frequently updated, the feedback registers are reset so that the decision threshold is changed to the normal level. Since the feedback registers keeps track of mispredicted requests due to hash collision, the false-short ratio can be minimized.

### *Optimized Reclaim Operation*

The main goal of the reclaim operation is to preserve the durability of the stored data written with $\texttt{WAmode}_{short}$. In order to reliably rewrite mispredicted data before their retention deadlines are expired, the retention keeper periodically[1] checks their remaining retention times. However, since maintaining the retention deadline for each written page requires too large system resources as well as too high checking overhead, we have developed a simple but effective reclaim technique. As shown in Figure 11, data for each write-age mode are written to different regions, i.e, the short-retention region[2] and the long-retention region. After short-retention data are written to a NAND block chosen from the free region, the block id is inserted into the retention queue in the retention keeper with the firstly written time. Although the following data are written to the block at different times, the worst-case retention deadline is still decided by the firstly 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. As a result, the time overhead of the checking process can be minimized. When a block whose retention deadline is almost expired is identified, mispredicted pages in the block are copied to a free block with *demoted* write-age mode (i.e., $\texttt{WAmode}_{long}$). After page copies are completed, that block becomes a long-retention block.

### *Selective Write-Age Tuning technique*

When too many pages in short-retention blocks are reclaimed so that the lifetime benefit of the write-age tuning mode is completely cancelled by the side effect of additional data copies, it is better to suspend a write-age tuning. The *break-even point* which represents the maximum number $N^{be}$ of reclaimed pages can be calculated based on the effective wearing per a program operation as summarized in Table 3. For example, if the effective wearing without the write-age tuning mode is 1.0, $\Sigma EW$ for writing $N$ pages is $N$. On the other hand, if the effective wearing with $\texttt{WAmode}_{short}$ is 0.6, $\Sigma EW$ for writing $N$ pages and reclaiming $N^{be}$ pages is $0.6 \times N + N^{be}$. Considering these two cases, in order to gain the lifetime benefit, $\Sigma EW$ with the short-retention mode should be less than that with the long-retention mode. As a result, in this case, $N^{be}$ is $0.4 \times N$. When the update characteristics of I/O requests are temporarily changed so that the average number of rewritten pages in recent reclaim operations become greater than $N^{be}$, the retention keeper changes the retention phase to the suspend phase. In this case, the Wmode selector selects only $\texttt{WAmode}_{long}$ regardless of the retention-time prediction results until the number of reclaimed pages is sufficiently reduced below $N^{be}$. In order to decide when the write-age tuning can be resumed, the retention keeper monitors the number of mispredicted pages in the background. Therefore, the lifetime benefit of DeVTS can be maximized by selectively tuning the write-age mode of requests.

## 5.4 Erase-Voltage Mode Selection

Selecting the most appropriate erase-voltage mode is the most essential step in autoFTL+ because the erase voltage has significant impacts on the NAND endurance as well

1. In order not to interfere with the foreground activities, a long checking period is preferable as long as all the mispredicted data can be reclaimed within the time interval. In our implementation, the checking period was set to one tenth of the retention time interval.

2. This separation technique can provide another advantage in reducing the write amplification factor over the existing data separation technique [19]. Since data written with $\texttt{WAmode}_{short}$ are likely to be updated within $T^{short}_{retention}$, when such block is chosen in the garbage collection, if it is not yet reclaimed, the number of unnecessary data copies can be significantly reduced [20].

as the overall write performance as described in Sections 3 and 4. When $\texttt{EVmode}_5$ (which uses the lowest erase voltage) is always used in erase operations, the NAND endurance can be maximally improved. However, 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 requests are incoming, the write performance may be significantly degraded. Furthermore, since $\texttt{EVmode}_5$ assumes hot data which will be updated within the near future, when cold data are inevitably written to such a shallowly erased block, the endurance gain may be almost cancelled by the reclaim operation. On the other hand, when $\texttt{EVmode}_0$ (which uses the highest erase voltage) is used at all times, the potential of DeVTS cannot be fully exploited[3] while the overall write-performance requirement can be maintained. Therefore, similar to the write mode selections, estimating the requirements of future write requests is also a critical technical issue in selecting the right erase-voltage mode.

When a foreground garbage collection is invoked, since the write-speed mode and write-age mode of a near-future write request are already 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 $\Sigma EW$ is less than 0.5K for the victim block, and $\texttt{WAmode}_{short}$ and $\texttt{WSmode}_0$ are chosen, the Emode selector decides $\texttt{EVmode}_2$ as the right erase-voltage mode.

On the other hand, when a background garbage collection is invoked, it is difficult to estimate the requirements of near-future write requests. This is because the background garbage collection is activated when outstanding write requests are not incoming for more than the threshold time interval so that the recent history of write requests is almost initialized. In our implementation, the Emode selector postpones deciding the right erase-voltage mode and always selects $\texttt{EVmode}_5$ so that a victim block is partially erased (with the lowest erase voltage) during the background garbage collection. The right erase-voltage mode is decided on-demand when the next phase of write requests (after the background garbage collection) is written to that block. If the selected write modes are not compatible with $\texttt{EVmode}_5$, the selected block is additionally erased under the *lazy erase* operation [1]. Although the write latency for the first page in the block is slightly increased (e.g., by 40%) due to this lazy erase, it is possible to fully utilize the potential of DeVTS in terms of the lifetime improvement. In order to minimize the negative impact of the lazy erase on the overall write performance, when the buffer utilization is low, the free blocks reclaimed by the background garbage collection are consumed with the highest priority.

### 5.5 Erase-Speed Mode Selection

The Emode selector selects a proper erase-speed mode which can offer an *actual* 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 write requests are continuously incoming, the buffer utilization may be increased. The increase $\Delta u^{erase}$ in the buffer utilization due to the erase operation can be estimated by how many write requests can be written 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 $\Delta u^{erase}$. In selecting an erase-speed mode, the Emode selector firstly checks whether erasing with $\texttt{ESmode}_{slow}$ raises $u^*$ above 1.0 or not. 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 erasing with $\texttt{ESmode}_{slow}$ causes a change in the current write-speed mode or not. If $u^*$ is increased above the original buffer utilization boundary (e.g., 0.33 or 0.66 as shown in Figure 10), the next write requests will be written with a faster write mode. In this case, since an endurance gain by using a slower erase mode is smaller than an endurance lose 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 selected for an erase operation.

### 5.6 Write Mode Selection in Garbage Collection

When a garbage collection is invoked, selecting the most suitable write-speed mode for data copy operations is also a challenging issue for maximizing the efficiency of DeVTS. If valid data are copied with the fastest write mode at all times, the performance overhead of a garbage collection can be minimized. However, since free pages in deeply erased blocks (which are compatible with the fastest write mode) are frequently consumed, the probability of erasing blocks with the highest erase voltage is substantially increased. On the other hand, 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 $\Delta u^{copy}$ which is proportional to the number of valid pages to be copied. As a result, the effective buffer utilization $u^*$ after the data copy operation is expressed as the sum of the current buffer utilization $u^{wb}$ and $\Delta u^{copy}$. 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 which does not change the current write-speed mode.

## 6 Experimental Results

### 6.1 Experimental Settings

We evaluated the effectiveness of the proposed autoFTL+ with *FlashBench+* which is an extended version of an existing unified development environment for NAND flash-based storage systems [10]. FlashBench+ emulates the key operations of DeVTS-enabled NAND devices in a timing-accurate fashion so that it is possible to keep track of temporal

3. Although the slowest write-speed mode and the long-retention mode are used at all times, the NAND endurance can be still improved by the age-dependent static retention tuning and the disturbance resistance tuning as described in Sections 4.2.1 and 4.3, respectively. This is because these two write tuning techniques can be exploited regardless of workload conditions.

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

| NAND functions | DeVTS speed Mode | Latency [14] |
|---|---|---|
| Program | $\mathtt{WSmode}_0$ | 1,300 $\mu s$ |
| | $\mathtt{WSmode}_1$ | 1,730 $\mu s$ |
| | $\mathtt{WSmode}_2$ | 2,600 $\mu s$ |
| Erase | $\mathtt{ESmode}_{fast}$ | 5,000 $\mu s$ |
| | $\mathtt{ESmode}_{slow}$ | 20,000 $\mu s$ |

interactions among various NAND operations [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 to affect the maximum write performance of an SSD), FlashBench+ 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%[4] of the total NAND capacity.

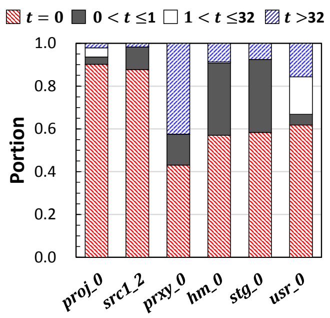
Our evaluations were performed with two different techniques: baseline and autoftl+. Baseline is an existing DeVTS-unaware FTL which does not use the erase scaling modes and the write tuning modes, described in Sections 3 and 4. Autoftl+ is the proposed DeVTS-aware FTL which fully exploits DeVTS-enabling techniques depending on workload characteristics so that the lifetime benefit of DeVTS can be maximally achieved while all the NAND requirements are preserved. Each technique was evaluated by replaying various I/O traces on top of FlashBench+. (For more details on I/O traces , please see Section 6.2.) When I/O requests were issued according to their timing information in trace files, corresponding NAND operations were performed in FlashBench+. After traces were replayed, we measured the maximum number of P/E cycles, $N_{P/E}^{max}$, which was actually experienced by NAND blocks until they became unreliable. We also measured the overall write throughput and retention times which were related to the side effect of DeVTS.

## 6.2 Workload Characteristics

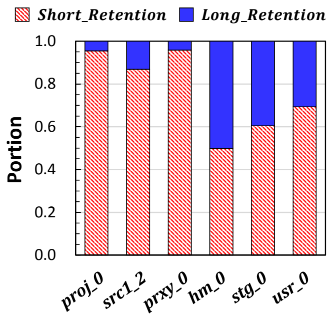
In our evaluations, we used six I/O traces, *proj_0*, *src1_2*, *prxy_0*, *hm_0*, *stg_0*, and *usr_0*, selected from the MSR Cambridge traces [21]. Although these traces include I/O characteristics in real-world enterprise servers, their I/O rates were too low to meaningfully stimulate the temporal 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 with the maximum write performance of our FlashBench+ configuration [22] [1].

Figure 12(a) shows the distributions of inter-arrival times of write requests for six traces. Inter-arrival times were normalized over the effective program time $T_{Pgm}^{effective}$ of FlashBench+. Since up to 32 NAND chips can serve write

4. 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 used for maintaining the meta data such mapping table, we set the buffer size to only 10% of the available DRAM capacity. In Section 6.5.3, we discuss the effect of the different buffer size in detail.



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



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

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

requests simultaneously, $T_{Pgm}^{effective}$ is 32 times shorter than the nominal program time $T_{Pgm}$ (i.e., the write latency of $\mathtt{WSmode}_0$) of one chip. When there were multiple pages in a write request, their inter-arrival times $t$ were classified as the '$t = 0$' case in Figure 12(a). On the other hand, when write requests containing only one page were incoming 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 a intensive trace (e.g., *proj_0*) because endurance-enhancing write and erase modes can be more frequently used in a sporadic trace.

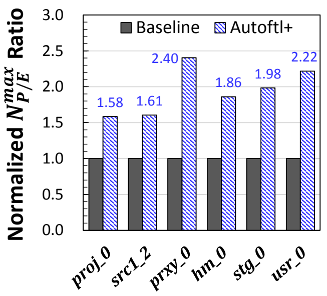
Figure 12(b) shows the distributions of retention times of write requests for six traces. The short-retention group and long-retention group were classified by $T_{retention}^{short}$ (i.e., 0.07 days[5]). A remarkable aspect is that there is a strong correlation between the distributions of inter-arrival 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 by the write-age tuning mode.

## 6.3 NAND Lifetime Analysis

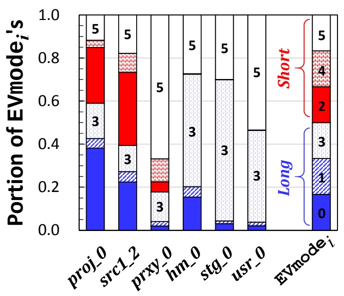
In order to measure $N_{P/E}^{max}$ (i.e., the NAND lifetime as defined in Section 3.2), each trace was repeated until $\Sigma EW$ reaches 3K (for MLC NAND devices [4]). Measured $N_{P/E}^{max}$ values were normalized over 3K. Figure 13(a) shows $N_{P/E}^{max}$ ratios for six traces with two different techniques. Autoftl+ 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 12(a) and 12(b), respectively.

For *proj_0* trace, $N_{P/E}^{max}$ is improved by only 58% because most of write requests are issued instantaneously so that 40% of erase operations cannot exploit endurance-enhancing modes at all as shown in Figure 13(b). However, since a considerable part of the rest of erase operations can

5. In our evaluation, we set $T_{retention}^{short}$ to 0.07 days as shown in Figure 8(b). This is because the total length of traces was only about 1 day. If our DeVTS technique is employed in real systems, it is better to increase $T_{retention}^{short}$ to 1 day for more reliable retention management. In this case, the lifetime benefit of the dynamic retention tuning is slightly reduced because corresponding $r_{ret}^{time}$ is increased 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.

take advantage of the write-age tuning mode, the limited $N_{P/E}^{max}$ ratio due to highly clustered consecutive writes can be partly compensated. (For more detail, see Section 6.5.2.) On the other hand, for *usr_0* trace, $N_{P/E}^{max}$ is improved by 122% because more than half of erase operations can be performed with the lowest erase voltage. In particular, for *prxy_0* trace, the improvement ratio of $N_{P/E}^{max}$ is up to 140%. This is because most of 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 autoftl+ is to extend $N_{P/E}^{max}$ while the other NAND requirements are not affected, we checked whether the overall write-performance and retention-time requirements were preserved or not.

#### 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 $u$ is decreased below 1.0. This delay time can be further increased when the foreground garbage collection is performed in NAND chips. Although autoftl+ frequently uses slow-speed write and erase modes, since such slow-speed modes are selected only when the write-performance requirement is not urgent (i.e., $u$ is less than 0.66), autoftl+ does not incur an additional delay over baseline as summarized in Table 5.

For *proj_0* trace, the overall write throughput is rather improved by 0.5% because the worst-case delay time due to the garbage collection 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 the total requests is reduced from 5.6% to 4.8%. For other traces, the overall write throughput and portion of delayed requests with autoftl+ are maintained at the same level as those with baseline.

#### 6.4.2 Overall Retention-Time Requirement

Since `WAmode`$_{short}$ aggressively reduces the retention capability of NAND pages, in order to guarantee the durability of the stored data, mispredicted pages whose retention

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 |
| | Autoftl+ | 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% |
| | Autoftl+ | 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 |
| | Autoftl+ | 1.10 | 1.07 | 1.03 | 1.06 | 1.13 | 1.05 |

deadlines are imminent should be properly reclaimed. However, when there are too many mispredicted pages, retention failures may occur because the maximum 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 $\mu$s, the retention keeper can reclaim up to 46,153 pages (which is 2.2% of the total NAND pages in FlashBench+). Although autoftl+ frequently uses `WAmode`$_{short}$ for writing data to NAND pages, retention failures did not occur in our evaluations[6]. This is mainly because the misprediction ratio is sufficiently suppressed by the conservative write-age management technique (in particular, the adaptive decision-threshold control) as 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.

### 6.5 Detailed Analysis

#### 6.5.1 Accuracy of the 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 much accurate our proposed resource-optimized predictor is. Table 6 summarizes the analysis results for 4 traces with four different techniques: History, FB_H, noFB_H and DA. History is a history-based prediction technique which utilizes the previous update time interval for predicting the next update time [23]. FB_H, noFB_H and DA are retention-time prediction techniques based on 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. On the other hand, noFB_H has the limited number of counters which are mapped to corresponding LBA by hash functions so that misprediction may occur due to hash collision. FB_H is the proposed prediction technique which employs additional feedback registers and makes an adaptive decision so that misprediction ratio can be substantially reduced.

For *src1_2* trace, the false-short ratio under History is too high (i.e., 4.8%) to avoid retention failures while that

6. In this paper, 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 avoided by rewriting valid pages written with `WAmode`$_{short}$ to other NAND pages with `WAmode`$_{long}$ during the power hold-up time supported by UPS or super capacitors.

TABLE 6
Accuracy of the retention-time predictions under different data separation schemes.

| Pre-diction | Result | *src1_2* | | | | *prxy_0* | | | | *hm_0* | | | | *usr_0* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | History | DA | noFB_H | FB_H | History | DA | noFB_H | FB_H | History | DA | noFB_H | FB_H | History | DA | noFB_H | FB_H |
| Short | True | 81.9% | 49.4% | 60.9% | 42.6% | 94.3% | 89.3% | 89.8% | 85.4% | 43.3% | 28.1% | 28.8% | 28.1% | 63.6% | 52.8% | 53.1% | 52.1% |
| | False | 4.8% | 0.8% | 2.3% | 0.9% | 1.3% | 0.5% | 0.6% | 0.6% | 5.9% | 0.5% | 0.7% | 0.8% | 4.5% | 0.9% | 1.0% | 1.1% |
| Long | True | 8.3% | 12.3% | 10.8% | 12.2% | 2.9% | 3.6% | 3.5% | 3.6% | 44.2% | 49.6% | 49.4% | 49.3% | 26.2% | 29.7% | 29.7% | 29.6% |
| | False | 5.0% | 37.5% | 26.0% | 44.3% | 1.5% | 6.5% | 6.1% | 10.5% | 6.6% | 21.8% | 21.1% | 21.8% | 5.7% | 16.6% | 16.3% | 17.3% |

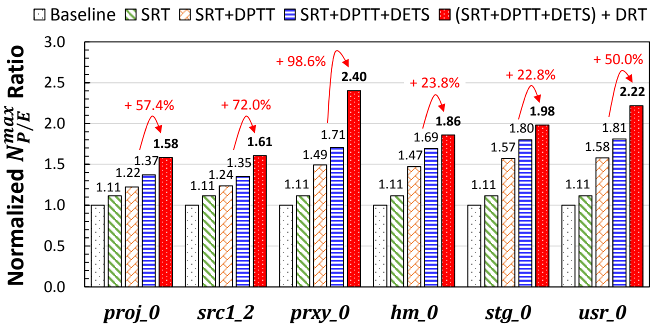


Fig. 14. Variations of the normalized $N_{P/E}^{max}$ ratios under different endurance-enhancing techniques for six traces.

TABLE 7
Variations of the normalized $N_{P/E}^{max}$ ratio and the overall write throughput under 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 |

under FB_H is sufficiently suppressed below the tolerable level. Comparing FB_H with noFB_H, the false-short ratio is reduced from 2.3% to 0.9% which is almost same as that under DA. For other traces, the false-short ratios are also maintained at a low level. These results clearly indicate that our proposed adaptive decision-threshold control technique can significantly reduce the misprediction ratio due to hash collision.

However, as summarized in Table 6, another misprediction ratio, i.e., the false-long ratio, is rather increased for write-intensive traces. This is because the retention-time predictor in this paper mainly focuses on minimizing the false-short ratio. If the false-long ratio is high, the potential of the retention capability tuning technique cannot be fully utilized. In order to more improve $N_{P/E}^{max}$ for write-intensive traces, it is needed to further reduce the false-long ratio as well as false-short ratio. Our future work is to develop more accurate retention-time predictor regardless of varying characteristics of I/O workload.

### 6.5.2 Endurance Gain Breakdown

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 autoFTL+ so that each technique can be selectively enabled. Figure 14 shows the increases in $N_{P/E}^{max}$ ratios for six traces when each endurance-enhancing technique (i.e., ST, DPTT, DETS and DRT) is enabled one by one on top of baseline. ST is the static tuning technique described in Sections 4.2.1 and 4.3. DPTT is the dynamic program-time tuning technique, and DETS is the dynamic erase-time tuning technique, as described in Sections 4.1 and 3.3, respectively. DRT is the dynamic retention-capability tuning technique described in Section 4.2.2. Our proposed autoftl+ fully exploits all the techniques.

Among endurance-enhancing techniques implemented in autoftl+, DRT has the most significant impact on improving $N_{P/E}^{max}$. DRT contributes to 34%, on average, of the total endurance gain (i.e., 94%). The effect of DRT strongly depends on the true-short ratio summarized in Table 6. For example, for *prxy_0* trace, predicting short-retention requests is very accurate (i.e., 85.4%) so that $N_{P/E}^{max}$ is significantly improved (i.e., 98.6%) by DRT. On the other hand, for *hm_0* trace, its effect is marginal. The effect of DPTT is comparable to that of DRT. The effects of DETS and ST account for about 20% and 12%, respectively, of the total endurance gain. In an earlier version (i.e., autoftl [1]) of this paper, only ST, DPTT and DETS were employed. In this case, the average $N_{P/E}^{max}$ ratio is 1.62. Our proposed autoftl+ (where DRT is additionally integrated) more extends $N_{P/E}^{max}$ by about 52% over autoftl. As shown in Figure 14, since DRT is more effective for write-intensive traces where the effect of DPTT is limited, DRT can substantially make up for the weakness of DPTT.

### 6.5.3 Sensitivity of Buffer Size

The large size of the write buffer has an advantage on extending $N_{P/E}^{max}$ because the probability of using the slow write and erase modes is increased. However, since too large buffer size is not cost effective in most of practical storage systems, we set the buffer size to only 16 MB which is 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 size of the write buffer as summarized in Table 7. When the buffer size is reduced to 4 MB, the average $N_{P/E}^{max}$ is also decreased by 3.6%. On the other hand, with 64 MB-size write buffer, the average $N_{P/E}^{max}$ 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 adapts a appropriate speed of write modes.

## 7 Related Works

## 8 Conclusion

## Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT and Future Planning (NRF-2013R1A2A2A01068260). The ICT at Seoul National University and IDEC provided research facilities for this study.

## 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," *Proc. USENIX Conf. File and Storage Tech.* (FAST), 2014.

[2] 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," *Proc. USENIX Annu. Tech. Conf.* (ATC), 2013.

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

[4] 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.

[5] 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," *Proc. Int. Reliability Physics Symp.* (IRPS), 2008.

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

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

[8] S. Cho, "Improving NAND Flash Memory Reliability with SSD Controllers," *Proc. Flash Memory Summit*, 2013.

[9] R.-S. Liu, C.-L. Yang, and W. Wu, "Optimizing NAND Flash-based SSDs via Retention Relaxation," *Proc. USENIX Conf. on File and Storage Tech.* (FAST), 2012.

[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 System Prototyping* (RSP), 2012.

[11] 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 J. Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, 1995.

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

[13] 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," *Jpn. J. Appl. Phys.*, vol. 48, no. 4, pp. 04C062-1–04C062-6, 2009.

[14] 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 J. Solid-State Circuits*, vol. 47, no. 4, pp. 981–989, 2012.

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

[16] 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 Improving the Reliability of TLC (Triple Level Cell) NAND Flash Characteristics," *Proc. IEEE Int. Memory Workshop* (IMW), 2011.

[17] M. Jung and M. Kandermir, "Revisiting Widely Held SSD Expectations and Rethinking System-Level Implications," *Proc. ACM SIGMETRICS*, 2013.

[18] R. Frickey, "Data Integrity on 20 nm SSDs," *Proc. Flash Memory Summit*, 2012.

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

[20] K. Ha and J. Kim, "A Program Context-Aware Data Separation Techniques for Reducing Garbage Collection Overhead in NAND Flash Memory," *Proc. Int. Workshop on Storage Network Architecture and Parallel I/Os* (SNAPI), 2011.

[21] D. Narayanan, A. Donnelly, and A. Rowstron, "Write Off-Loading: Practical Power Management for Enterprise Storage," *Proc. USENIX Conf. File and Storage Tech.* (FAST), 2008.

[22] 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 Design Integr. Circuits Syst.*, vol. 32, no. 2, pp. 247–260, 2013.

[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," *Proc. Design Automation Conf.* (DAC), 2014.

PLACE
PHOTO
HERE

**Jaeyong Jeong** received the BS and MS degrees in radio science and engineering from Korea University, Korea, in 1996 and 1998, respectively. From 1998 to 2012, he was with the Memory Division of Samsung Electronics, Korea, as a principal engineer. He is currently working toward the PhD degree in computer science and engineering at Seoul National University, Korea. His research interests include nonvolatile memory, embedded software and storage systems.

PLACE
PHOTO
HERE

**Youngsun Song**

PLACE
PHOTO
HERE

**Sangwook Shane Hahn**

PLACE
PHOTO
HERE

**Sungjin Lee** received the BE degree in electrical engineering from Korea University in 2005 and the MS and PhD degrees in computer science and engineering from the Seoul National University in 2007 and 2013, respectively. He is currently working as a postdoctoral associate in the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. His research interests include storage systems, operating systems, and embedded software.

PLACE
PHOTO
HERE

**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 at the DSPS R&D Center of Texas Instruments in Dallas, Texas. He is currently a professor at 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.