Optimizing NAND Flash-Based SSDs via Retention Relaxation

Ren-Shuo Liu*, Chia-Lin Yang*, and Wei Wu†
*National Taiwan University and †Intel Corporation
{renshuo@ntu.edu.tw, yangc@csie.ntu.edu.tw, wei.a.wu@intel.com}

Abstract

As NAND Flash technology continues to scale down and more bits are stored in a cell, the raw reliability of NAND Flash memories degrades inevitably. To meet the retention capability required for a reliable storage system, we see a trend of longer write latency and more complex ECCs employed in an SSD storage system. These greatly impact the performance of future SSDs. In this paper, we present the first work to improve SSD performance via retention relaxation. NAND Flash is typically required to retain data for 1 to 10 years according to industrial standards. However, we observe that many data are overwritten in hours or days in several popular workloads in datacenters. The gap between the specification guarantee and actual programs' needs can be exploited to improve write speed or ECCs' cost and performance. To exploit this opportunity, we propose a system design that allows data to be written in various latencies or protected by different ECC codes without hampering reliability. Simulation results show that via write speed optimization, we can achieve 1.8–5.7× write response time speedup. We also show that for future SSDs, retention relaxation can bring both performance and cost benefits to the ECC architecture.

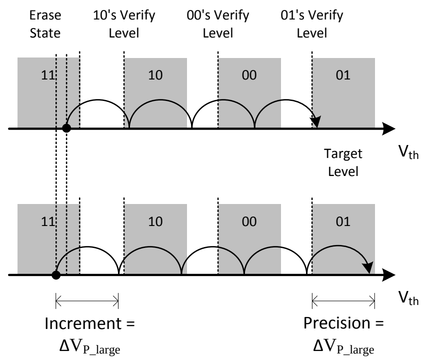
1 Introduction

For the past few years, NAND Flash memories have been widely used in portable devices such as media players and mobile phones. Due to their high density, low power and high I/O performance, in recent years, NAND Flash memories begun to make the transition from portable devices to laptops, PCs and datacenters [6, 35]. As the semiconductor industry continues scaling memory technology and lowering per-bit cost, NAND Flash is expected to replace the role of hard disk drives and fundamentally change the storage hierarchy in future computer systems [14, 16].

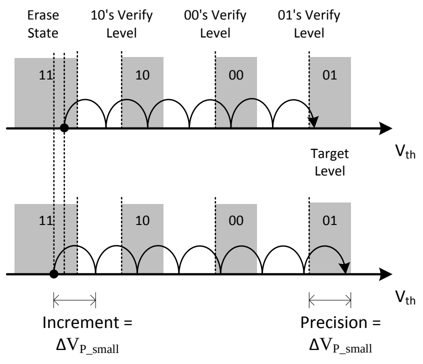
A reliable storage system needs to provide a retention guarantee. Therefore, Flash memories have to meet the retention specification in industrial standards. For example, according to the JEDEC standard JESD47G.01 [19], NAND Flash blocks cycled to 10% of the maximum specified endurance must retain data for 10 years, and blocks cycled to 100% of the maximum specified endurance have to retain data for 1 year. As NAND Flash technology continues to scale down and more bits are stored in a cell, the raw reliability of NAND Flash decreases substantially. To meet the retention specification for a reliable storage system, we see a trend of longer write latency and more complex ECCs required in SSDs. For example, comparing recent 2-bit MLC NAND Flash memories with previous SLC ones, page write latency increased from 200 μs [34] to 1800 μs [39], and the required strength of ECCs went from single-error-correcting Hamming codes [34] to 24-error-correcting Bose-Chaudhuri-Hocquenghem (BCH) codes [8, 18, 28]. In the near future, more complex ECC codes such as low-density parity-check (LDPC) [15] codes will be required to reliably operate NAND Flash memories [13, 28, 41].

To overcome the design challenge for future SSDs, in this paper, we present *retention relaxation*, the first work on optimizing SSDs via relaxing NAND Flash's retention capability. We observe that in typical datacenter workloads, e.g., proxy and MapReduce, many data written into storage are updated quite soon, thereby, requiring only days or even hours of data retention, which is much shorter than the retention time typically specified for NAND Flash. In this paper, we exploit the gap between the specification guarantee and actual programs' needs for SSD optimization. We make the following contributions:

- We propose a NAND Flash model that captures the relationship between raw bit error rates and retention time based on empirical measurement data. This model allows us to explore the interplay between retention capability and other NAND Flash parameters such as the program step voltage for write operations.
- A set of datacenter workloads are characterized for their retention time requirements. Since I/O traces are usually gathered in days or weeks, to analyze retention time requirements in a time span beyond the trace period, we present a retention time projection method based on two characteristics obtained from the traces, the write amount and the write working set size. Characterization results show that for 15 of the 16 traces analyzed, 49–99% of writes require less than 1-week retention time.
- We explore the benefits of retention relaxation for



(a) ISPP with large ΔV_P



(b) ISPP with small ΔV_P

Figure 1: Incremental step pulse programming (ISPP) for programming NAND Flash

speeding up write operations. We increase the program step voltage so that NAND Flash memories are programmed faster but with shorter retention guarantees. Experimental results show that 1.8–5.7× SSD write response time speedup is achievable.

- We show how retention relaxation can benefit ECC designs for future SSDs which require concatenated BCH-LDPC codes. We propose an ECC architecture where data are encoded by variable ECC codes based on their retention requirements. In our ECC architecture, time-consuming LDPC is removed from the critical performance path. Therefore, retention relaxation can bring both performance and cost benefits to the ECC architecture.

The rest of the paper is organized as follows. Section 2 provides background about NAND Flash. Section 3 presents our NAND Flash model and the benefits of retention relaxation. Section 4 analyzes data retention requirements in real-world workloads. Section 5 describes the proposed system designs. Section 6 presents evaluation results regarding the designs in the previous section. Section 7 describes related work, and Section 8 concludes the paper.

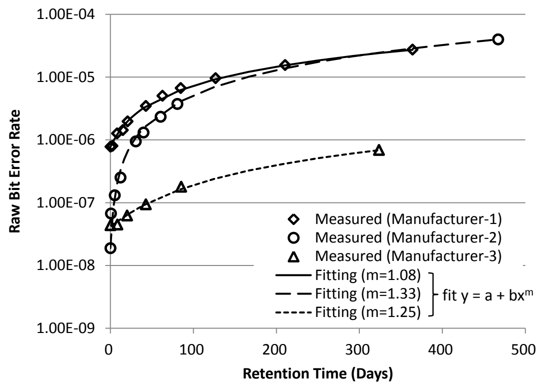
2 Background

NAND Flash memories comprise an array of floating gate transistors. The threshold voltage (V_{th}) of the transistors can be programmed to different levels by injecting different amounts of charge on the floating gates. Different V_{th} levels represent different data. For example, to store N bits data in a cell, its V_{th} is programmed to one of its 2^N different V_{th} levels.

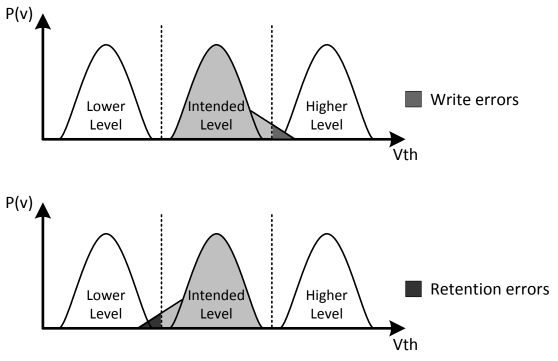
To program V_{th} to the desired level, the incremental step pulse programming (ISPP) scheme is commonly used [26, 37]. As shown in Figure 1, ISPP increases the V_{th} of NAND Flash cells step-by-step by a certain voltage increment (i.e., ΔV_P) and stops once V_{th} is greater than the desired threshold voltage. Because NAND Flash cells have different starting V_{th}, the resulting V_{th} spreads across a range, which determines the precision of cells' V_{th} distributions. The smaller ΔV_P is, the more precise the resulting V_{th} is. On the other hand, smaller ΔV_P means more steps are required to reach the target V_{th}, thereby, resulting in longer write latency [26].

NAND Flash memories are prone to errors. That is, the V_{th} level of a cell may be different from the intended one. The fraction of bits which contain incorrect data is referred to as the raw bit error rate (RBER). Figure 2(a) shows measured RBER of 63–72nm 2-bit MLC NAND Flash memories under room temperature following 10K program/erase (P/E) cycles [27]. The RBER at retention time = 0 is attributed to write errors. Write errors have been shown mostly caused by cells with higher V_{th} than intended because the causes of write errors, such as program disturb and random telegraph noise, tend to over-program V_{th}. The increment of RBER after writing data (retention time > 0) is attributed to retention errors. Retention errors are caused by charge losses which decrease V_{th}. Therefore, retention errors are dominated by cells with lower V_{th} than intended. Figure 2(b) illustrates these two error sources: write errors mainly correspond to the tail at the high-V_{th} side; retention errors correspond to the tail at the low-V_{th} side. In Section 3.1, we model NAND Flash considering these error characteristics.

A common approach to handle NAND Flash errors is to adopt ECCs (error correction codes). ECCs supplement user data with redundant parity bits to form codewords. With ECC protection, a codeword with a certain amount of bits corrupted can be reconstructed. Therefore, ECCs can greatly reduce the bit error rate. We refer to the bit error rate after applying ECCs as the uncorrectable bit error rate (UBER). The following equation gives the relationship between UBER and RBER [27]:



(a) Measured $RBER(t)$ and fitting to power-law trends for 63–72 nm 2-bit MLC NAND Flash. Data are aligned to $t = 0$.



(b) Write errors and retention errors

Figure 2: Bit error rate in NAND Flash memories

$$UBER = \frac{\sum_{n=t+1}^{N_{CW}} \binom{N_{CW}}{n} \cdot RBER^n \cdot (1-RBER)^{(N_{CW}-n)}}{N_{User}} \quad (1)$$

Here, N_{CW} is the number of bits per codeword, N_{User} is the number of user data bits per codeword, and t is the maximum number of error bits the ECC code can correct per codeword.

UBER is an important reliability metric for storage systems and is typically required to be under 10^{-13} to 10^{-16} [27]. As mentioned earlier, NAND Flash's RBER increases with time due to retention errors. Therefore, to satisfy both the retention and reliability specifications in storage systems, ECCs must be strong enough to tolerate not only write errors presenting in the beginning but also retention errors accumulating over time.

3 Retention Relaxation for NAND Flash

The key observation we make in this paper is that since retention errors increase over time, if we could relax the retention capability of NAND Flash memories, fewer retention errors need to be tolerated. These error margins can then be utilized to improve other performance metrics. In this section, we first present a V_{th} distribution modeling methodology which captures the RBER of NAND Flash. Based on the model, we elaborate on the strategies to exploit the benefits of retention relaxation in detail.

3.1 Modeling Methodology

We first present the base V_{th} distribution model for NAND Flash. Then we present how we extend the model to capture the characteristics of different error causes. Last, we determine the parameters of the model by fitting the model to the error-rate behavior of NAND Flash.

3.1.1 Base V_{th} Distribution Model

The V_{th} distribution is critical to NAND Flash. It describes the probability density function (PDF) of V_{th} for each data state. Given a V_{th} distribution, one can evaluate the corresponding RBER by calculating the probability that a cell contains incorrect V_{th}, i.e., V_{th} higher or lower than the intended level.

V_{th} distributions have been modeled using bell-shape functions in previous studies [23, 42]. For MLC NAND Flash memories with q states per cell, q bell-shape functions, $P_k(v)$ where $0 \le k \le (q-1)$, are employed in the model as follows.

First, the V_{th} distribution of the erased state is modeled as a Gaussian function, $P_0(v)$:

$$P_0(v) = \alpha_0 \cdot e^{-\frac{(v-\mu_0)^2}{2\sigma_0{}^2}} \quad (2)$$

Here, σ_0 is the standard deviation of the distribution and μ_0 is the mean. Because data are assumed to be in one of the q states with equal probability, a normalization coefficient, α_0, is employed so that $\int_v P_0(v) = \frac{1}{q}$.

Furthermore, the V_{th} distribution of each non-erased state (i.e., $1 \le k \le (q-1)$) is modeled as a combination of a uniform distribution with width equal to ΔV_P in the middle and two identical Gaussian tails on both sides:

$$P_k(v) = \begin{cases} \alpha \cdot e^{-\frac{(v-\mu_k+0.5\Delta V_P)^2}{2\sigma^2}}, & v < \mu_k - \frac{\Delta V_P}{2} \\ \alpha \cdot e^{-\frac{(v-\mu_k-0.5\Delta V_P)^2}{2\sigma^2}}, & v > \mu_k + \frac{\Delta V_P}{2} \\ \alpha, & \text{otherwise} \end{cases} \quad (3)$$

Here, ΔV_P is the voltage increment in ISPP, μ_k is the mean of each state, σ is the standard deviation of the two Gaussian tails, and α is again the normalization coefficient to satisfy the condition that $\int_v P_k(v) = \frac{1}{q}$ for the $k-1$ states.

Given the V_{th} distribution, the RBER can be evaluated by calculating the probability that a cell contains incorrect V_{th}, i.e., V_{th} higher or lower than the intended read voltage levels, using the following equation:

$$RBER = \sum_{k=0}^{q-1} \left(\underbrace{\int_{-\infty}^{V_{R,k}} P_k(v)dv}_{V_{th}\text{lower than intended}} + \underbrace{\int_{V_{R,(k+1)}}^{\infty} P_k(v)dv}_{V_{th}\text{higher than intended}} \right) \quad (4)$$

Here, $V_{R,k}$ is the lower bound of the correct read voltage for the k^{th} state and $V_{R,k+1}$ is the upper bound as shown in Figure 3.

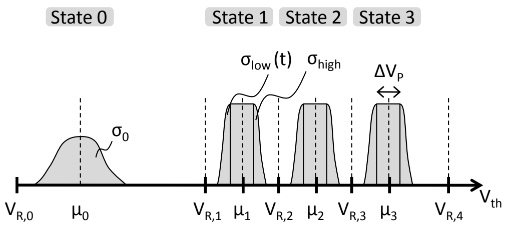


Figure 3: Illustration of model parameters

3.1.2 Model Extension

As mentioned in Section 2, the two tails of a V_{th} distribution are from different causes. The high-V_{th} tail of a distribution is mainly caused by V_{th} over-programming (i.e., write errors); the low-V_{th} tail is mainly due to charge losses over time (i.e., retention errors) [27]. Therefore, the two Gaussian tails may not be identical. To capture this difference, we extend the base model by setting different standard deviations to the two tails as shown in Figure 3.

The two standard deviations are set based on the observation in the previous study on Flash's retention process [7]. Under room temperature[1], a small portion of cells have a much larger charge-loss rate than others. As such charge losses accumulate over time, the distribution tends to form a wider tail at the low-V_{th} side. Therefore, we extend the base model by setting the standard deviation of the low-V_{th} tail to be a time-increasing function, $\sigma_{low}(t)$, but keeping σ_{high} time-independent. The extended model is as follows:

$$P_k(v,t) = \begin{cases} \alpha(t) \cdot e^{-\frac{(v-\mu_k+0.5\Delta V_P)^2}{2\sigma_{low}(t)^2}}, & v < \mu_k - \frac{\Delta V_P}{2} \\ \alpha(t) \cdot e^{-\frac{(v-\mu_k-0.5\Delta V_P)^2}{2\sigma_{high}^2}}, & v > \mu_k + \frac{\Delta V_P}{2} \\ \alpha(t), & \text{otherwise} \end{cases} \quad (5)$$

Here, the normalization term becomes a function of time, $\alpha(t)$, to keep $\int_v P_k(v,t) = \frac{1}{q}$.

[1]According to the previous study [32], in datacenters, HDDs' average temperatures range between 18–51°C and stay around 26–30°C most of the time. Since SSDs do not contain motors and actuators, we expect SSDs should be in lower temperature than HDDs. Therefore, we only consider room temperature in our current model.

We should note that keeping σ_{high} time-independent does not imply that cells with high V_{th} are time-independent and never leak charge. Since the integral of PDF for each data state remains $\frac{1}{q}$, the probability that a cell belongs to the high-V_{th} tail drops as the low-V_{th} tail widens over time. The same phenomenon happens to the middle part, too.

Given the V_{th} distribution in the extended model, $RBER(t)$ can be evaluated using the following formula:

$$RBER(t) = \sum_{k=0}^{q-1} \left(\underbrace{\int_{-\infty}^{V_{R,k}} P_k(v,t)dv}_{V_{th}\text{lower than intended}} + \underbrace{\int_{V_{R,(k+1)}}^{\infty} P_k(v,t)dv}_{V_{th}\text{higher than intended}} \right) \quad (6)$$

3.1.3 Model Parameter Fitting

In the proposed V_{th} distribution model, ΔV_P, $V_{R,k}$, μ_k, and σ_0 are set to the values shown in Figure 4 according to [9]. The two new parameters in the extended model, σ_{high} and $\sigma_{low}(t)$, are determined through parameter fitting such that the resulting $RBER(t)$ follows the error-rate behavior of NAND Flash. Below we describe the parameter fitting procedure.

We adopt the power-law model [20] to describe the error-rate behavior of NAND Flash:

$$RBER(t) = RBER_{write} + RBER_{retention} \times t^m \quad (7)$$

Here, t is time, m is a coefficient, $1 \leq m \leq 2$, $RBER_{write}$ corresponds to the error rate at $t = 0$ (i.e., write errors), and $RBER_{retention}$ is the incremental error rate per unit of time due to retention errors.

We determine m in the power-law model based on the curve-fitting values shown in Figure 2(a). In the figure, the power-law curves fit the empirical error-rate data very well with m equal to 1.08, 1.25, and 1.33. We consider 1.25 as the typical case of m and consider the other two values as the corner cases.

The other two coefficients in the power-law model, $RBER_{write}$ and $RBER_{retention}$, can be solved given RBER at $t = 0$ and RBER at the maximum retention time, t_{max}. According to the JEDEC standard JESD47G.01 [19], NAND Flash blocks cycled to the maximum specified endurance have to retain data for 1 year, so we set t_{max} to 1 year. Moreover, recent NAND Flash requires 24-bit error correction for 1080-byte data [4, 28]. Assuming that the target $UBER(t_{max})$ requirement is 10^{-16}, by Equation (1), we have:

$$RBER(t_{max}) = 4.5 \times 10^{-4} \quad (8)$$

As shown in Figure 2(a), $RBER(0)$ is typically orders of magnitude lower than $RBER(t_{max})$. Tanakamaru *et al.* [41] also show that write errors are between 150× to 450× fewer than retention errors. This is because retention errors accumulate over time and eventually dominate. Therefore, we set $RBER_{write}$ accordingly:

$$RBER_{write} = RBER(0) = \frac{RBER(t_{max})}{C_{write}} \quad (9)$$

Parameter	Value (V)
$V_{R,0}$	$-\infty$
$V_{R,1}$	0
$V_{R,2}$	1.2
$V_{R,3}$	2.4
$V_{R,4}$	6
ΔV_P	0.2
μ_0	-3
μ_1	0.6
μ_2	1.8
μ_3	3
σ_0	0.48
σ_{high}	0.1119
$\sigma_{low}(0)$	0.1039

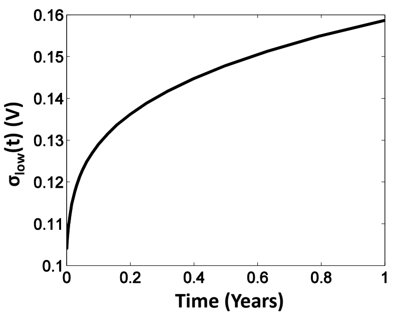


Figure 4: Modeled 2-bit MLC NAND Flash

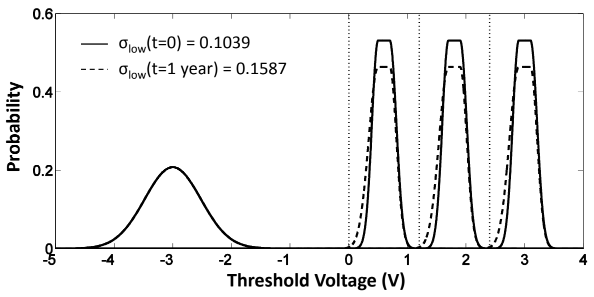


Figure 5: Modeling results

Here, C_{write} is the ratio of $RBER(t_{max})$ to $RBER(0)$. We set C_{write} to 150, 300, and 450, where 300 is considered as the typical case and the other two are considered as the corner cases.

We also have $RBER_{retention}$ as follows:

$$RBER_{retention} = \frac{(RBER(t_{max}) - RBER(0))}{t_{max}{}^{m}} \tag{10}$$

We note that among write errors (i.e., $RBER_{write}$), a major fraction of them correspond to cells with higher V_{th} than intended. This is because the root causes of write errors tend to make V_{th} over-programmed. Mielke *et al.* [27] show that this fraction is between 62% to about 100% for the NAND Flash devices in their experiments. Therefore, we give the following equations:

$$RBER_{write\_high} = RBER_{write} \times C_{write\_high} \tag{11}$$

$$RBER_{write\_low} = RBER_{write} \times (1 - C_{write\_high}) \tag{12}$$

Here, $RBER_{write\_high}$ and $RBER_{write\_low}$ correspond to cells with V_{th} higher and lower than intended levels, respectively. C_{write_high} stands for the ratio of total write errors to write errors which are higher than the intended levels. We set C_{write_high} to 62%, 81%, and 99%, where 81% is considered as the typical case and the other two are considered as the corner cases.

Now we have the error-rate behavior of NAND Flash. σ_{high} and $\sigma_{low}(0)$ are first determined so that the error rate for V_{th} being higher and lower than intended equals $RBER_{write\_high}$ and $RBER_{write\_low}$, respectively. Then, $\sigma_{low}(t)$ is determined by matching the $RBER(t)$ derived from the V_{th} model with NAND Flash's error-rate behavior described in Equations (7) to (10) at a fine time step.

Figure 5 shows the modeling results of the V_{th} distribution for the typical-case NAND Flash. In this figure, the solid line stands for the V_{th} distribution at $t = 0$; the dashed line stands for the V_{th} distribution at $t = 1$ year. We can see that the 1-year distribution is flatter than the distribution at $t = 0$. We can also see that as the low-V_{th} tail widens over a year, the probability of both the middle part and the high-V_{th} tail drops correspondingly.

3.2 Benefits of Retention Relaxation

In this section, we elaborate on the benefits of retention relaxation from two perspectives — improving write speed and improving ECCs' cost and performance. The analysis is based on NAND Flash memories cycled to the maximum specified endurance (i.e., 100% wear-out) with data retention capability set to 1 year [19]. Since NAND Flash's reliability typically degrades monotonically in terms of P/E cycles, considering such an extreme case is conservative for the following benefit evaluation. In other words, NAND Flash in its early lifespan has more head room for optimization.

3.2.1 Improving Write Speed

As presented earlier, NAND Flash memories use the ISPP scheme to incrementally program memory cells. The V_{th} step increment, ΔV_P, directly affects write speed and data retention. Write speed is proportional to ΔV_P because with larger ΔV_P, less steps are required during the ISPP procedure. On the other hand, data retention decreases as ΔV_P gets larger because large ΔV_P widens V_{th} distributions and reduces the margin for tolerating retention errors.

Algorithm 1 shows the procedure to quantitatively evaluate the write speedup if data retention time requirements are reduced. The analysis is based on the extended NAND Flash model presented in Section 3.1. For all the typical and corner cases we consider, we first enlarge ΔV_P by various ratios between 1× to 3×, thereby, speeding up NAND Flash writes proportionately. For each ratio, we test $RBER(t)$ at different retention time from 0 to 1 year to find the maximum t such that $RBER(t)$ is within the base ECC strength.

Figure 6 shows the write speedup vs. data retention. Both the typical case (black line) and the corner cases (gray dashed lines) we consider in Section 3.1 are shown. For the typical case, if data retention is relaxed to 10 weeks, 1.86× speedup for NAND Flash page write is achievable; if data retention is relaxed to 2 weeks, the speedup is 2.33×. Furthermore, the speedup for the corner cases are close to the typical case. This means the speedup numbers are not very sensitive to the values of the parameters we obtain using parameter fitting.

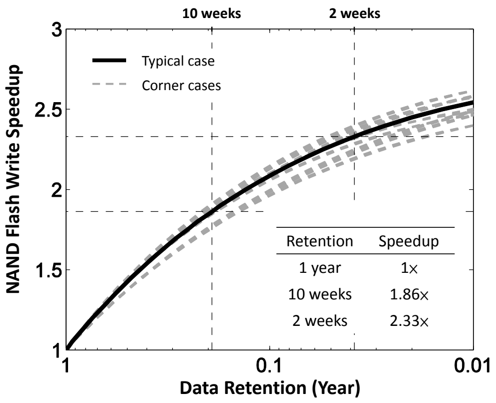


Figure 6: NAND page write speedup vs. data retention

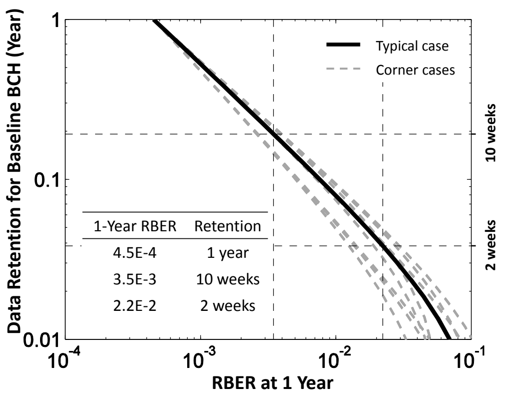


Figure 7: Data retention capability of 24 error correction per 1080 bytes BCH codes

Algorithm 1 Write speedup vs. data retention

```
1:  $C_{BCH} = 4.5 \times 10^{-4}$
2:  for all typical and corner cases do
3:      for $VoltageRatio = 1$ to 3 step = 0.01 do
4:          Enlarge $\Delta V_P$ by $VoltageRatio$ times
5:          $WriteSpeedUp = VoltageRatio$
6:          for $Time\ t = 0$ to 1 $year$ step = $\delta$ do
7:              Find $RBER(t)$ according to $\sigma_{low}(t)$ and $\alpha(t)$
8:          end for
9:          $DataRetention = max\{t{:}RBER(t) \leq C_{BCH}\}$
10:         plot $(DataRetention, WriteSpeedUp)$
11:     end for
12: end for
```

3.2.2 Improving ECCs' Cost and Performance

ECC design is emerging as a critical issue in SSDs. Nowadays, NAND Flash-based systems heavily rely on BCH codes to tolerate RBER. Unfortunately, BCH degrades memory storage efficiency significantly once the RBER of NAND Flash reaches 10^{-3} [22]. Recent NAND Flash has RBER around 4.5×10^{-4}. As the density of NAND Flash memories continues to increase, RBER will exceed the BCH limitation inevitably. Therefore, BCH codes will become inapplicable in the near future.

LDPC codes are promising ECCs for future NAND Flash memories [13, 28, 41]. The main advantage of LDPC is that they can provide correction performance very close to the theoretical limits. However, LDPC incurs much higher encoding complexity than BCH does [21, 25]. For example, an optimized LDPC encoder [44] consumes 3.9 M bits of memory and 11.4 k FPGA Logic Elements to offer 45 MB/s throughput. To sustain write throughput of high-performance SSDs, e.g., 1 GB/s ones [1], high-throughput LDPC encoders are required, otherwise the LDPC encoders may become the throughput bottleneck. This leads to high hardware cost because hardware parallelization is one basic approach to increase the throughput of LDPC encoders [24]. In this paper, we exploit retention relaxation to alleviate such cost and performance dilemma. That is, with retention relaxation, fewer retention errors need to be tolerated; therefore, BCH codes could be still strong enough to protect data even if NAND Flash's 1-year RBER soars.

Algorithm 2 analyzes the achievable data retention time of BCH codes with 24 bits per 1080 bytes error-correction capability under different NAND Flash $RBER(1\ year)$ values. Here we assume that $RBER(t)$ follows the power-law trend described in Section 3.1.3. We vary $RBER(1\ year)$ from 4.5×10^{-4} to 1×10^{-1}, and derive the corresponding write error rate ($RBER_{write}$) and retention error increment per unit of time ($RBER_{retention}$). The achievable data retention time of the BCH codes is the time when RBER exceeds the capability of the BCH codes (i.e., 4.5×10^{-4}).

Algorithm 2 Data retention vs. maximum RBER for BCH (24-bit per 1080 bytes)

```
1:  $t_{max} = 1\ year$
2:  $C_{BCH} = 4.5 \times 10^{-4}$
3:  for all typical and corner cases do
4:      for $RBER(t_{max}) = 4.5 \times 10^{-4}$ to $1 \times 10^{-1}$ step = $\delta$ do
5:          $RBER_{write} = \frac{RBER(t_{max})}{C_{write}}$
6:          $RBER_{retention} = \frac{(RBER(t_{max}) - RBER_{write})}{t_{max}{}^{m}}$
7:          $RetentionTime = (\frac{C_{BCH} - RBER_{write}}{RBER_{retention}})^{\frac{1}{m}}$
8:          plot $(RBER(t_{max}), RetentionTime)$
9:      end for
10: end for
```

Figure 7 shows the achievable data retention time of the BCH code given different $RBER(1\ year)$ values. The black line stands for the typical case and the gray dashed lines stand for the corner cases. As can be seen, for the typical case, the baseline BCH code can retain data for

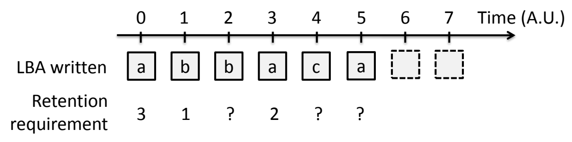


Figure 8: Data retention requirements in a write stream

10 weeks even if $RBER(1\ year)$ reach 3.5×10^{-3}. Even if $RBER(1\ year)$ reaches 2.2×10^{-2}, the baseline BCH code can still retain data for 2 weeks. We can also see similar trends for the corner cases.

4 Data Retention Requirements Analysis

In this section, we first analyze real-world traces from enterprise datacenters to show that many writes into storage require days or even shorter retention time. Since I/O traces are usually gathered in days or weeks, to estimate the percentage of writes with retention requirements beyond the trace period, we present a retention-time projection method based on two characteristics obtained from the traces, the write amount and the write working set size.

4.1 Real Workload Analysis

In this subsection, we analyze real disk traces to understand the data retention requirements of real-world applications. The data retention requirement of a sector written into a disk is defined as: *the interval from the time the sector is written to the time the sector is overwritten.* Let's take Figure 8 for example. The disk is written by the address stream **a, b, b, a, c, a, ...** and so on. The first write is to address **a** at time 0, and the same address is overwritten at time 3; therefore, the data retention requirement for the first write is $(3-0) = 3$. Usually disk traces only cover a limited period of time, for those writes whose next write does not appear before the observation ends, the retention requirements cannot be determined. For example, for the write to address **b** at time 2, the overwritten time is unknown. We denote its retention requirement with '?' as a conservative estimation. It is important to note that we are focusing on data retention requirements for data blocks *in write streams* rather than that *in the entire disk*.

Table 1 shows the three sets of traces we analyze. The first is from an enterprise datacenter in Microsoft Research Cambridge (MSRC) [29]. This set covers 36 volumes from various servers and we select 12 of them which have the largest write amounts. These traces span 1 week and 7 hours. We skip the first 7 hours which do not form a complete day and use the remaining 1-week part. The second set of traces is MapReduce which has been shown to benefit from the increased bandwidth and reduced latency of NAND Flash-based SSDs [11]. We use Hadoop [2] to run the MapReduce benchmark on a cluster of two Core-i7 machines each of which has

Table 1: Workload summary

Category	Name	Description	Span
MSRC	prn_0 proj_0, proj_2 prxy_0, prxy_1 src1_0, src1_2 src2_2 usr_1, usr_2	Print server Project directories Web proxy Source control Source control User home directories	1 week
MapReduce	hd1 hd2	WordCount benchmark	1 day
TPC-C	tpcc1 tpcc2	OLTP benchmark	1 day

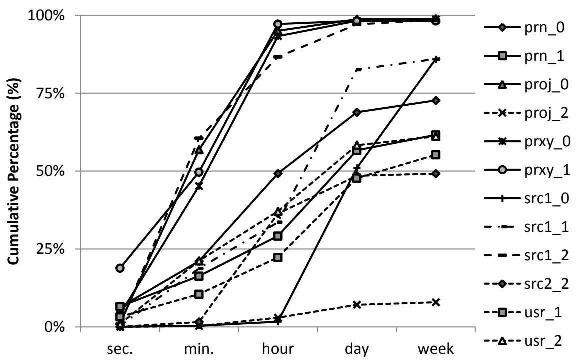
8 GB RAM and a SATA hard disk and runs 64-bit Linux 2.6.35 with the ext4 filesystem. We test two MapReduce usage models. In the first model, we repeatedly replace 140 GB text data in the Hadoop cluster and invoke word counting jobs. In the second model, we interleave performing word counting jobs on two sets of 140 GB text data which have been pre-loaded in the cluster. The third workload is the TPC-C benchmark. We use Hammerora [3] to generate the TPC-C workload on a MySql server which has a Core-i7 CPU, 12 GB RAM, and a SATA SSD and runs 64-bit Linux 2.6.32 with the ext4 filesystem. We configure the benchmarks as having 40 and 80 warehouses. Each warehouse has 10 users with keying and thinking time. Both MapReduce and TPC-C workloads span 1 day.

For each trace, we analyze the data retention requirement of every sector written into the disk. Figure 9 shows the cumulative percentage of data retention requirements less than or equal to the following values — a second, a minute, an hour, a day, and a week. As can be seen, the data retention requirements of the workloads are usually low. For example, more than 95% of sectors written into the disk for proj_0, prxy_1, tpcc1, and tpcc2 need less than 1-hour data retention. Furthermore, for all the traces except proj_2, 49–99.2% of sectors written into the disk need less than 1-week data retention. For tpcc2, up to 44% of writes require less than 1-second retention. This is because MySql's storage engine, InnoDB, writes data to a fixed-size log, called the doublewrite buffer, before writing to the data file to guard against partial page writes; therefore, all writes to the doublewrite buffer are overwritten very quickly.

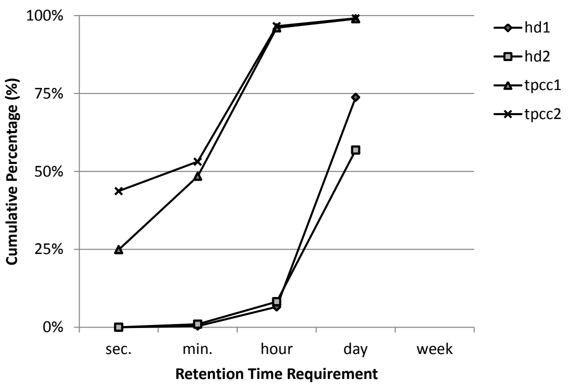
4.2 Retention Requirement Projection

The main challenge of retention time characterization for real-world workloads is that I/O traces are usually gathered in a short period of time, e.g., days or weeks. To estimate the percentage of writes with retention requirements beyond the trace period, we derive a projection method based on two characteristics obtained from the traces, the write amount and the write working set size.

We denote the percentage of writes with retention time



(a) MSRC workloads



(b) MapReduce and TPC-C workloads

Figure 9: Data retention requirement distribution

requirements less than X within a time period of Y as $S_{X,Y}\%$. We first formulate $S_{T_1,T_1}\%$ in the write amount and the write working set size, where T_1 is the time span of the trace. Let N be the amount of data sectors written into the disk during T_1 and W be the write working set size (i.e., the number of distinct sector addresses being written) during T_1. We have the following formula (the proof is similar to the pigeonhole principle):

$$S_{T_1,T_1}\% = \frac{N-W}{N} = 1 - \frac{W}{N} \tag{13}$$

With this formula, the first projection we make is the percentage of writes that have retention time requirements less than T_1 in an observation period of T_2, where $T_2 = k \times T_1$, $k \in \mathbb{N}$. The projection is based on the assumption that for each T_1 period, the write amount and the write working set size remain N and W, respectively. We derive the lower bound on $S_{T_1,T_2}\%$ as follows:

$$S_{T_1,T_2}\% = \frac{k(N-W) + \sum_{i=1}^{k-1} u_i}{kN} \geq S_{T_1,T_1}\% \tag{14}$$

where u_i is the number of sectors whose lifetime is across two periods and their retention time requirements are less than T_1. Equation (14) implies that we do not overestimate the percentage of writes that have retention time requirements less than T_1 by characterizing a trace gathered in a T_1 period.

With the first projection, we can then derive the lower bound on $S_{T_2,T_2}\%$. Clearly, $S_{T_2,T_2}\% \geq S_{T_1,T_2}\%$. Combined with Equation (14), we have:

$$S_{T_2,T_2}\% \geq S_{T_1,T_2}\% \geq S_{T_1,T_1}\% \tag{15}$$

The lower bound on $S_{T_2,T_2}\%$ also depends on disk capacity, A. During T_2, the write amount is equal to $k \times N$, and the write working set size must be less than or equal to the disk capacity, i.e, $k \times W \leq A$. By with Equation (13), we have:

$$S_{T_2,T_2}\% \geq \frac{kN-A}{kN} = 1 - \frac{A}{kN} \tag{16}$$

Combining Equation (15) and (16), the lower bound on $S_{T_2,T_2}\%$ is given by:

$$S_{T_2,T_2}\% \geq max(1 - \frac{A}{kN}, S_{T_1,T_1}\%) \tag{17}$$

Table 2 shows the data retention requirements analysis using the above equations. First, we can see that the $S_{T_1,T_1}\%$ obtained from Equation (13) matches Figure 9. Let's take hd2 for example. There are a total of 726 GB of writes in 1 day whose write working set size is 313 GB. According to Equation (13), 57% of the writes whose retention time requirements are less than 1 day. This is the case shown in Figure 9. Furthermore, if we can observe the hd_2 workload for 1 week, more than 86% of writes whose retention time requirements are expected to be less than 1 weeks. This again shows the gap between the specification guarantee and actual programs' needs in terms of data retention.

5 System Design

5.1 Retention-Aware FTL (Flash Translation Layer)

In this section, we present the SSD design which leverages retention relaxation for improving either write speed or ECCs' cost and performance. Specifically, in the proposed SSD, data written into NAND Flash memories could occur in variable write latencies or be encoded by different ECC codes, which provide different levels of retention guarantees. We refer to the data written by these different methods as in different "modes". In our design, data in a physical NAND Flash block are in the same mode. To correctly retrieve data from NAND Flash, we need to record the mode of each physical block. Furthermore, to avoid data losses due to a

Table 2: Data retention requirements analysis

Volume Name	Disk Capacity (A)	Write Amount (N)	Write Working Set (W)	$S_{1d,1d}$	$S_{1w,1w}$	$S_{5w,5w}$
	GB	GB	GB	%	%	%
prn_0	66.3	44.2	12.1		72.6	≧72.6
prn_1	385.2	28.8	11.1		61.6	≧61.6
proj_0	16.2	143.8	1.6		98.9	≧98.9
proj_2	816.2	168.4	155.1		7.9	≧7.9
prxy_0	20.7	52.7	0.7		98.7	≧98.7
prxy_1	67.8	695.3	12.5		98.2	≧98.2
src1_0	273.5	808.6	114.1		85.9	≧93.2
src1_1	273.5	29.6	4.2		85.9	≧85.9
src1_2	8.0	43.4	0.7		98.5	≧98.5
src2_2	169.6	39.3	20.0		49.2	≧49.2
usr_1	820.3	54.8	24.5		55.2	≧55.2
usr_2	530.4	25.6	10.0		61.1	≧61.1
hd1	737.6	1564.9	410.1	73.8	≧93.3	≧98.7
hd2	737.6	726.3	313.3	56.9	≧85.5	≧97.1
tpcc1	149	310.3	3.1	99.0	≧99.0	≧99.0
tpcc2	149	692.8	6.0	99.1	≧99.1	≧99.4

[1] The disk capacity of the MSRC traces are estimated using their maximum R/W address. The estimation results conform to the previous study [30].

[2] 1d, 1w, and 5w stand for a day, a week, and 5 weeks, respectively.

[3] GB stands for 2^{30} bytes.

shortage of data retention capability, we have to monitor the remaining retention capability of each NAND Flash block. We implement the proposed retention-aware design in the Flash Translation Layer (FTL) in SSDs rather than in OSes. FTL-based implementation requires minimum OS/application modification, which we think is important for easy deployment and wide adoption of the proposed scheme.

Figure 10 shows the block diagram of the proposed FTL. The proposed FTL is based on the page-level FTL [5] with two additional components, Mode Selector (MS) and Retention Tracker (RT). For writes, MS sends different write commands to NAND Flash chips or invokes different ECC encoders. As discussed in Section 3.2.1, write speed could be improved by adopting larger ΔV_P. In current Flash chips, only one write command is supported. To support the proposed mechanism, NAND Flash chips need to provide multiple write commands with different ΔV_P values. MS keeps the mode of each NAND Flash block in memories so that during reads, it can invoke the right ECC decoder to retrieve data. RT is responsible for ensuring that every NAND Flash block in the SSD does not run out of its retention capability. RT uses one counter per NAND Flash block to keep track of its remaining retention time. When the first page of a block is written, the retention capability of this write is stored in the counter. These retention counters are periodically updated. If a block is found to approach its data retention limit, RT schedules background operations to move valid data in this block to another new block and then invalidates the old one.

One main parameter in the proposed SSD design is how many write modes we should employ in the SSD. The optimal setting depends on retention time varia-

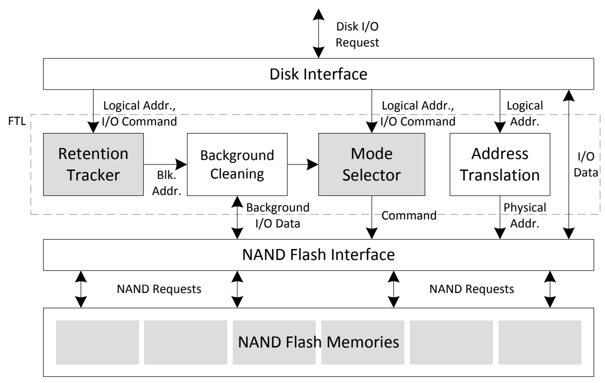


Figure 10: Proposed retention-aware FTL

tion in workloads and the cost for supporting multiple write modes. In this work, we present a coarse-grained management method. There are two kinds of NAND Flash writes in SSD systems: host writes and background writes. Host writes correspond to write requests sent from the host to the SSDs; background writes comprise cleaning, wear-leveling, and data movement internal to the SSDs. Performance is usually important to the host writes. Moreover, host writes usually require short data retention as shown in Section 4. In contrast, background writes are less sensitive to performance and usually involve data which have been stored in the storage for a long time; therefore, their data are expected to remain for a long time in the future (commonly referred to as cold data). Based on this observation, we propose to employ two levels of retention guarantees for the two kinds of writes. For host writes, retention-relaxed writes are used to exploit their high probability of short retention requirements and gain performance benefits; for background writes, normal writes are employed to preserve the retention guarantee.

In the proposed two-level framework, to optimize write performance, host writes occur in fast write speed with reduced retention capability. If data are not overwritten within their retention guarantee, background writes with normal write speed are issued. To optimize ECCs' cost and performance, a new ECC architecture is proposed. As mentioned earlier, NAND Flash RBER will soon exceed BCH's limitation (i.e., RBER $\geq 10^{-3}$); therefore, advanced ECC designs will be required for future SSDs. Figure 11 shows such an advanced ECC design for future SSDs which employs multi-layer ECCs with code concatenations: the inner code is BCH, and the outer code is LDPC. Concatenating BCH and LDPC exploits the advantages of both [43]: LDPC greatly improves the maximum correcting capability, while BCH complements LDPC for eliminating LDPC's error floor. The main issue with this design is since every write needs to be encoded in LDPC, a high-throughput LDPC encoder is required to prevent the LDPC encoder from be-

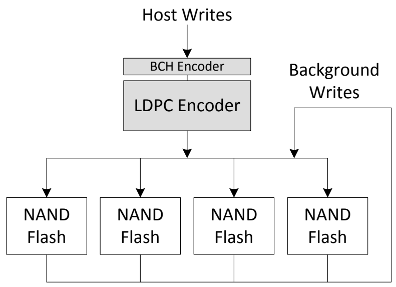


Figure 11: Baseline concatenated BCH-LDPC codes in an SSD

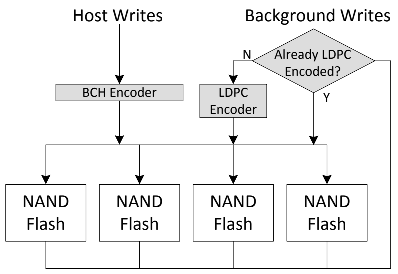


Figure 12: Proposed ECC architecture leveraging retention relaxation in an SSD

ing the bottleneck. In the proposed ECC architecture shown in Figure 12, host writes are protected by BCH only since they tend to have short retention requirements. If data are not overwritten within the retention guarantee provided by BCH, background writes are issued. All background writes are protected by LDPC. In this way, the LDPC encoder is kept out of the critical performance path. Its benefits are two-fold. First, write performance is improved since host writes do not go through time-consuming LDPC encoding. Second, since BCH filters out short-lifetime data and LDPC encoding can be amortized in the background, the throughput requirements of LDPC are less than the baseline design. Therefore, the LDPC hardware cost can be reduced.

We present two specific implementation of retention relaxation. The first one relaxes the retention capability of host writes to 10 weeks and periodically checks the remaining retention capability of each NAND Flash block at the end of every 5 weeks. Therefore, FTL always has another 5 weeks at least to reprogram those data which have not been overwritten in the past period and can amortize the re-programming task in the background over the 5 weeks without causing burst writes. We set the period of invoking the reprogramming tasks to 100 ms. The second one is similar to the first one except that the retention capability and checking period are 2 weeks and 1 week, respectively. These two designs are referred to as **RR-10week** and **RR-2week** in this paper.

5.2 Overhead Analysis

Memory Overhead

The proposed mechanism requires extra memory resources to store write modes and retention time information for each block. Since we only have two write modes, i.e., the normal mode and the retention-relaxed one, each block requires only a 1-bit flag to record its write mode. As for the size of the counter for keeping track of the remaining retention time, both RR-2week and RR-10week require only a 1-bit counter per block because all retention-relaxed blocks written in the n^{th} period are reprogrammed during the $(n+1)^{th}$ period. For

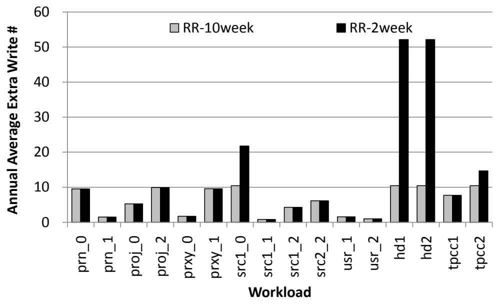


Figure 15: Wear-out overhead of retention relaxation

an SSD having 128 GB NAND Flash with 2 MB block size, the memory overhead is 16 KB .

Reprogramming Overhead

In the proposed schemes, data that are not overwritten in the guaranteed retention time need to be reprogrammed. These extra writes affect both the performance and the life time of SSDs. To analyze its performance impact, we estimate *reprogramming amounts per unit of time* based on the projection method described in Section 4.2. Here, we let T_2 be the checking period in the proposed schemes. For example, for RR-10week, T_2 equals 5 weeks. Therefore, at the end of each period, the total write amount is kN, the percentage of writes which require reprogramming is at most $(1 - S_{T_2,T_2}\%)$, and the reprogramming tasks can be amortized over the upcoming period of T_2. The reprogramming amounts per unit of time are as follows:

$$\frac{(1 - S_{T_2,T_2}\%) \times k \times N}{T_2} \tag{18}$$

The results show that the amount of reprogramming tasks range between 1.13 kB/s to 1.25 MB/s for RR-2week, and between 1.13 kB/s to 0.26 MB/s for RR-10week. Since each NAND Flash plane can provide 6.2 MB/s write throughput (i.e., writing a 8 kB page in 1.3 ms), we anticipate that reprogramming does not lead to high performance overhead. In Section 6, we evaluate its actual performance impact.

To quantify the wear-out effect caused by reprogramming, we show *extra writes per cell per year* assuming

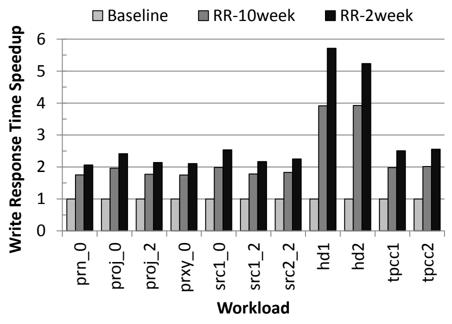


Figure 13: SSD write response time speedup

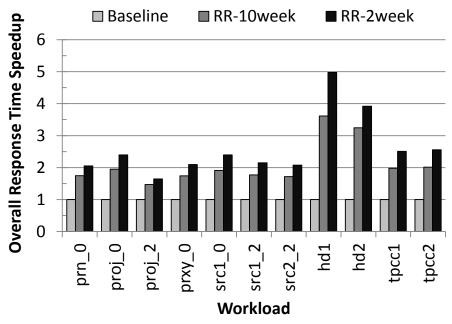


Figure 14: SSD overall response time speedup

perfect wear-leveling. We first give the upper bound on this metric. Let's take RR-2week for example. In the extreme case, RR-2week reprograms the entire disk every week, which leads to 52.1 extra writes per cell per year. Similarly, RR-10week causes 10.4 extra writes per cell per year at most. These extra writes are not significant compared to NAND Flash's endurance which is usually a few thousands P/E cycles. Therefore, even in the worst case, the proposed mechanism does not cause significant wear-out effect. For real workloads, the wear-out overhead is usually smaller than the worst case as shown in Figure 15. The wear-out overhead for each workload is evaluated based on the disk capacity and the reprogramming amounts per unit of time presented above.

Table 3: NAND Flash and SSD configurations

Parameter	Value
Over-provisioning	15%
Cleaning threshold	5%
Page size	8 KB
Pages per block	256
Blocks per plane	2000
Planes per die	2
Dies per channel	1~8
Number of channel	16
Mapping policy	Full stripe

Parameter	Value
Page read latency	75 µs
Page write latency	1.3 ms
Block erase latency	3.8 ms
NAND bus bandwidth	200 MB/s

Trace Name	Dies per Disk	Exported Capacity (GB)
prn_0, proj_0, prxy_0, src1_2	16	106
src2_2	32	212
src1_0	64	423
proj_2, hd1, hd2, tpcc1, tpcc2	128	847

6 System Evaluation

We conduct simulation-based experiments using SSDsim [5] and Disksim-4.0 [10] to evaluate the RR-10week and RR-2week designs. SSDs are configured to have 16 channels. Detailed configurations and parameters are listed in Table 3. Eleven of the 16 traces listed in Table 2 are used and simulated for the whole trace. We omit prxy_1 because the simulated SSD can not sustain its load, and prn_1, src1_1, usr_1, usr_2 are also omitted because they contain write amounts less than 15% of the total raw NAND Flash capacity. SSD write speedup and ECCs' cost and performance improvement are evaluated separately. The reprogramming overhead described in Section 5.2 are considered in the experiments.

Figure 13 shows the speedup of write response time for different workloads if we leverage retention relaxation to improve write speed. We can see that RR-10week and RR-2week typically achieve 1.8–2.6$\times$ write response time speedup. hd1 and hd2 show up to 3.9–5.7$\times$ speedup. These two workloads have high queuing delay due to high I/O throughput. With retention relaxation, the queuing time is greatly reduced, between 3.7$\times$ to 6.1$\times$. Moreover, for all workloads, RR-2week gives about 20% extra performance gain over RR-10week. Figure 14 shows the speedup in terms of overall response time. The overall response time is mainly determined by write requests due to the significant amount of write requests in the tested workloads and the long write latency. Therefore, we can see that the speedup trend is similar to that of write response time.

To show how retention relaxation benefits ECC design in future SSDs, we consider SSDs comprising NAND Flash whose 1-year RBER approaches 2.2×10^{-2}. We compare the proposed RR-2week design with the baseline design which employs concatenated BCH-LDPC codes. The LDPC encoder is modeled as a FIFO and its throughput is chosen among 5, 10, 20, 40, 80, 160, 320, and ∞ MB/s. Since the I/O queue of the simulated SSDs could saturate if LDPC's throughput is insufficient, we first report the minimum required throughput configurations without causing saturation in Figure 16. As can be seen, for the baseline ECC architecture, throughput up to 160 MB/s is required. In contrast, for RR-2week, the lowest throughput configuration (i.e., 5MB/s) is enough to sustain the write rates in all tested workloads. Figure 17 shows the response time of the baseline and RR-2week under various LDPC throughput configurations. The response time reported in this figure is the average of the response time normalized to that with unlimited LDPC throughput:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{ResponseTime_i}{IdealResponseTime_i}\right) \tag{19}$$

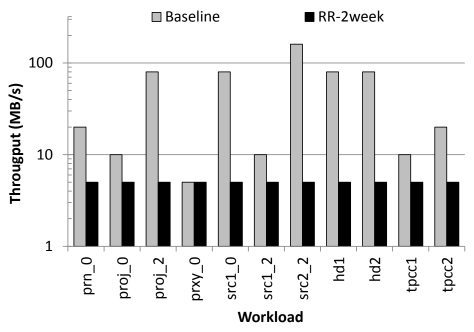


Figure 16: Minimum required LDPC throughput configurations without causing I/O queue saturation

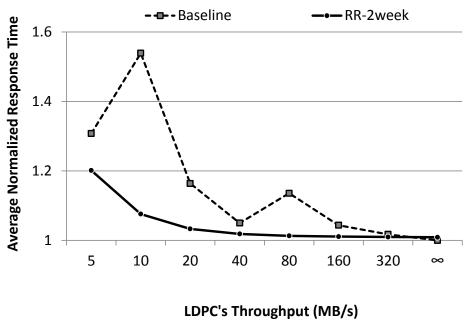


Figure 17: Average normalized response time given various LDPC throughput configurations

where N is the number of workloads which do not incur I/O queue saturation given specific LDPC throughput. In the figure, the curve of the baseline presents a zigzag appearance between 5 MB/s to 80 MB/s because several traces are excluded due to the saturation in the I/O queue. This may inflate the performance of the baseline. Even so, we see RR-2week outperforms the baseline significantly with the same LDPC throughput configuration. For example, with 10MB/s throughput, RR-2week performs 43% better than the baseline. Only when the LDPC throughput approaches infinite does RR-2week perform a bit worse than the baseline due to reprogramming overhead. We can also see that with a 20 MB/s LDPC, RR-2week already approaches the performance of unlimited LDPC throughput, while the baseline requires 160 MB/s to achieve the similar level. Because hardware parallelization is one basic approach to increase the throughput of a LDPC encoder [24], in this point of view, retention relaxation can reduce the hardware cost of LDPC encoders by $8\times$.

7 Related Work

Access frequencies are usually considered in storage optimization. Chiang *et al.* [12] propose to cluster data with similar write frequencies together to increase SSDs' cleaning efficiency. Pritchett and Thottethodi [33] observe the skewness of disk access frequencies in datacenters and propose novel ensemble-level SSD-based disk caches. In contrast, we focus on the time interval between two successive writes to the same address which defines the data retention requirement.

Several device-aware optimizations for NAND Flash-based SSDs were proposed recently. Grupp *et al.* [17] exploit the variation in page write speed in MLC NAND Flash to improve SSDs' responsiveness. Tanakamaru *et al.* [40] propose wear-out-aware ECC schemes to improve the ECC capability. Xie *et al.* [42] improve write speed through compressing user data and employing stronger ECC codes. Pan *et al.* [31] improve write speed and defect tolerance using wear-out-aware policies. Our work considers the retention requirements of real workloads and relaxes NAND Flash's data retention to optimize SSDs, which is orthogonal to the above device-aware optimization.

Smullen *et al.* [36] and Sun *et al.* [38] improve energy and latency of STTRAM-based CPU caches through redesigning STTRAM cells with relaxed non-volatility. In contrast, we focus on NAND Flash memories used in storage systems.

8 Conclusions

We present the first work on optimizing SSDs via relaxing NAND Flash's data retention capability. We develop a NAND Flash model to evaluate the benefits if NAND Flash's original multi-year data retention can be reduced. We also demonstrate that in real systems, write requests usually require days or even shorter retention times. To optimize the write speed and ECCs' cost and performance, we design SSD systems which handle host writes with shortened retention time while handling background writes as usual and present corresponding retention tracking schemes to guarantee that no data loss happens due to a shortage of retention capability. Simulation results show that the proposed SSDs achieve 1.8–5.7$\times$ write response time speedup. We also show that for future SSDs, retention relaxation can bring both performance and cost benefits to the ECC architecture. We leave simultaneously optimizing write speed and ECCs as our future work.

Acknowledgements

We would like to thank our program committee shepherd Bill Bolosky and the anonymous reviewers for their insightful comments and constructive suggestions. This research is supported in part by research grants from ROC National Science Council 100-2219-E-002-027, 100-2219-E-002-030, 100-2220-E-002-015; Excellent Research Projects of National Taiwan University 10R80919-2; Macronix International Co., Ltd. 99-S-C25; and Etron Technology Inc., 10R70152, 10R70156.

References

[1] Fusion-IO ioDrive Duo. http://www.fusionio.com/products/iodrive-duo/.

[2] Hadoop. http://hadoop.apache.org/.

[3] Hammerora. http://hammerora.sourceforge.net/.

[4] NAND Flash support table, July 2011. http://www.linux-mtd.infradead.org/nand-data/nanddata.html.

[5] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for SSD performance. In *Proc. 2008 USENIX Annual Technical Conference (USENIX '08)*, June 2008.

[6] D. G. Andersen and S. Swanson. Rethinking Flash in the data center. *IEEE Micro*, 30:52–54, July 2010.

[7] F. Arai, T. Maruyama, and R. Shirota. Extended data retention process technology for highly reliable Flash EEPROMs of 10^6 to 10^7 W/E cycles. In *Proc. 1998 IEEE International Reliability Physics Symposium (IRPS '98)*, April 1998.

[8] R. C. Bose and D. K. R. Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, March 1960.

[9] J. Brewer and M. Gill. *Nonvolatile Memory Technologies with Emphasis on Flash: A Comprehensive Guide to Understanding and Using Flash Memory Devices*. Wiley-IEEE Press, 2008.

[10] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger. The DiskSim simulation environment version 4.0 reference manual reference manual (CMU-PDL-08-101). Technical report, Parallel Data Laboratory, 2008.

[11] A. M. Caulfield, L. M. Grupp, and S. Swanson. Gordon: using Flash memory to build fast, power-efficient clusters for data-intensive applications. In *Proc. 14^{th} International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '09)*, March 2009.

[12] M.-L. Chiang, P. C. H. Lee, and R.-C. Chang. Using data clustering to improve cleaning performance for Flash memory. *Softw. Pract. Exper.*, 29:267–290, March 1999.

[13] G. Dong, N. Xie, and T. Zhang. On the use of soft-decision error-correction codes in NAND Flash memory. *IEEE Trans. Circuits Syst. Regul. Pap.*, 58(2):429 –439, February 2011.

[14] D. Floyer. Flash pricing trends disrupt storage. Technical report, May 2010.

[15] R. Gallager. Low-density parity-check codes. *IRE Trans. Inf. Theory*, 8(1):21 –28, January 1962.

[16] J. Gray. Tape is dead. Disk is tape. Flash is disk. RAM locality is king. Presented at the CIDR Gong Show, January 2007.

[17] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf. Characterizing Flash memory: anomalies, observations, and applications. In *Proc.* 42^{nd} *IEEE/ACM International Symposium on Microarchitecture (MICRO '09)*, December 2009.

[18] A. Hocquenghem. Codes correcteurs d'Erreurs. *Chiffres (paris)*, 2:147–156, September 1959.

[19] JEDEC Solid State Technology Association. *Stress-Test-Driven Qualification of Integrated Circuits, JESD47G.01*, April 2010. http://www.jedec.org/.

[20] JEDEC Solid State Technology Association. *Failure Mechanisms and Models for Semiconductor Devices, JEP122G*, October 2011. http://www.jedec.org/.

[21] S. Kopparthi and D. Gruenbacher. Implementation of a flexible encoder for structured low-density parity-check codes. In *Proc. 2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'07)*, Auguest 2007.

[22] S. Li and T. Zhang. Approaching the information theoretical bound of multi-level NAND Flash memory storage efficiency. In *Proc. 2009 IEEE International Memory Workshop (IMW '09)*, May 2009.

[23] S. Li and T. Zhang. Improving multi-level NAND Flash memory storage reliability using concatenated BCH-TCM coding. *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, 18(10):1412 –1420, October 2010.

[24] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong. Efficient encoding of quasi-cyclic low-density parity-check codes. *IEEE Trans. Commun.*, 54(1):71 – 81, January 2006.

[25] C.-Y. Lin, C.-C. Wei, and M.-K. Ku. Efficient encoding for dual-diagonal structured LDPC codes based on parity bit prediction and correction. In *Proc. 2008 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS '08)*, December 2008.

[26] R. Micheloni, L. Crippa, and A. Marelli. *Inside NAND Flash Memories*. Springer, 2010.

[27] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. Nevill. Bit error rate in NAND Flash memories. In *Proc. 2008 IEEE International Reliability Physics Symposium (IRPS '08)*, May 2008.

[28] R. Motwani, Z. Kwok, and S. Nelson. Low density parity check (LDPC) codes and the need for stronger ECC. Presented at the Flash Memory Summit, August 2011.

[29] D. Narayanan, A. Donnelly, and A. Rowstron. Write off-loading: Practical power management for enterprise storage. *Trans. Storage*, 4:10:1–10:23, November 2008.

[30] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating server storage to SSDs: analysis of tradeoffs. In *Proc. 4th ACM European Conference on Computer Systems (EuroSys '09)*, April 2009.

[31] Y. Pan, G. Dong, and T. Zhang. Exploiting memory device wear-out dynamics to improve NAND Flash memory system performance. In *Proc. 9th USENIX Conference on File and Stroage Technologies (FAST '11)*, February 2011.

[32] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *Proc. 5th USENIX Conference on File and Stroage Technologies (FAST '07)*, February 2007.

[33] T. Pritchett and M. Thottethodi. SieveStore: a highly-selective, ensemble-level disk cache for cost-performance. In *Proc. 37th annual International Symposium on Computer Architecture (ISCA '10)*, June 2010.

[34] Samsung Electronics. *K9F8G08UXM Flash memory datasheet*, March 2007.

[35] J.-Y. Shin, Z.-L. Xia, N.-Y. Xu, R. Gao, X.-F. Cai, S. Maeng, and F.-H. Hsu. FTL design exploration in reconfigurable high-performance SSD for server applications. In *Proc. 23rd International Conference on Supercomputing (ICS '09)*, 2009.

[36] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. Stan. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *Proc. 17th IEEE International Symposium on High Performance Computer Architecture (HPCA '11)*, February 2011.

[37] 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 programming scheme. *IEEE J. Solid-State Circuits*, 30(11):1149 –1156, November 1995.

[38] Z. Sun, X. Bi, H. L. nad Weng-Fai Wong, Z. liang Ong, X. Zhu, and W. Wu. Multi retention level STT-RAM cache designs with a dynamic refresh scheme. In *Proc. 44th IEEE/ACM International Symposium on Microarchitecture (MICRO '11)*, December 2011.

[39] S. Swanson. Flash memory overview. cse240a: Graduate Computer Architecture, University of California, San Diego, November 2011. http://cseweb.ucsd.edu/classes/fa11/cse240A-a/Slides1/18-FlashOverview.pdf.

[40] S. Tanakamaru, A. Esumi, M. Ito, K. Li, and K. Takeuchi. Post-manufacturing, 17-times acceptable raw bit error rate enhancement, dynamic codeword transition ECC scheme for highly reliable solid-state drives, SSDs. In *Proc. 2010 IEEE International Memory Workshop (IMW '10)*, May 2010.

[41] S. Tanakamaru, C. Hung, A. Esumi, M. Ito, K. Li, and K. Takeuchi. 95%-lower-BER 43%-lower-power intelligent solid-state drive (SSD) with asymmetric coding and stripe pattern elimination algorithm. In *Proc. 2011 IEEE International Solid-State Circuits Conference (ISSCC '11)*, February 2011.

[42] N. Xie, G. Dong, and T. Zhang. Using lossless data compression in data storage systems: Not for saving space. *IEEE Trans. Comput.*, 60:335–345, 2011.

[43] N. Xie, W. Xu, T. Zhang, E. Haratsch, and J. Moon. Concatenated low-density parity-check and BCH coding system for magnetic recording read channel with 4 kB sector format. *IEEE Trans. Magn.*, 44(12):4784 –4789, December 2008.

[44] H. Yasotharan and A. Carusone. A flexible hardware encoder for systematic low-density parity-check codes. In *Proc. 52nd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS '09)*, Auguest 2009.