Hindawi
Scientific Programming
Volume 2018, Article ID 8171096, 10 pages
https://doi.org/10.1155/2018/8171096

Research Article

CRIM: Conditional Remapping to Improve the Reliability of Solid-State Drives with Minimizing Lifetime Loss

Youngpil Kim,[1] **Hyunchan Park**,[2] **Cheol-Ho Hong**,[3] **and Chuck Yoo**[1]

[1]*College of Informatics, Korea University, Seoul, Republic of Korea*
[2]*Division of Computer Science and Engineering, Chonbuk National University, Jeonju, Republic of Korea*
[3]*School of Electrical and Electronics Engineering, Chung-Ang University, Seoul, Republic of Korea*

Correspondence should be addressed to Chuck Yoo; chuckyoo@os.korea.ac.kr

Received 3 May 2018; Revised 21 July 2018; Accepted 17 August 2018; Published 20 September 2018

Academic Editor: Marco Aldinucci

Solid-state drive (SSD) becomes popular as the main storage device. However, over time, the reliability of SSD degrades due to bit errors, which poses a serious issue. The periodic remapping (PR) has been suggested to overcome the issue, but it still has a critical weakness as PR increases lifetime loss. Therefore, we propose the conditional remapping invocation method (CRIM) to sustain reliability without lifetime loss. CRIM uses a probability-based threshold to determine the condition of invoking remapping operation. We evaluate the effectiveness of CRIM using the real workload trace data. In our experiments, we show that CRIM can extend a lifetime of SSD more than PR by up to 12.6% to 17.9% of 5-year warranty time. In addition, we show that CRIM can reduce the bit error probability of SSD by up to 73 times in terms of typical bit error rate in comparison with PR.

1. Introduction

Solid-state drive (SSD) is a high-performance and energy-efficient storage device based on NAND flash memory. SSD has become increasingly popular in recent times, seeing usage from in personal computers to in high-performance servers [1, 2]. In particular, SSD can enhance the performance of cloud storage systems. Cloud storage systems in data centers should consider energy efficiency, performance, and maintenance costs [3, 4]. In large-scale cloud storage system, the replacement costs of storage devices can be concerned. SSDs can be the key solution because their performance is much faster than that of hard disk drives (HDDs), and their cost is much lower than that of static random access memory (SRAM).

However, SSD has reliability problems because NAND flash memory has bit errors. The bit errors degrade the performance and lifetime of SSD [5]. Especially, the degradation of lifetime is a serious problem because it reduces the amount of available blocks which are limited resources. Thus, enhancing lifetime is an important and urgent demand, so many studies have been proposed until recently [6–10].

Many SSDs handle bit errors by applying error correction code (ECC) mechanisms such as BCH [11]. However, ECC does not completely solve the bit error problem because the error correction capability of ECCs is not scalable. For example, BCH-512 can correct 7 bit errors with a 512-length code. BCH-32 k can correct 259 bit errors—the code length is 64 times longer than BCH-512 but the number of correctable errors is only 37 times more. Furthermore, the power consumption and additional space for metadata increase by multiples of 71 and 85, respectively [12].

After that, more advanced way to handle bit errors has been suggested: applying the periodic remapping (PR) of data blocks [12]. PR periodically replaces error-prone blocks into healthy ones. However, PR has a critical weakness in that PR reduces the overall lifetime of SSDs. Thus, we attempt to find a way to minimize the lifetime loss caused by PR.

Our key observation is that the cause of lifetime loss is invoking remapping operations too often in PR. As solution, we propose CRIM, the conditional invocation of the remapping operation based on failure likelihood. To reduce lifetime loss, CRIM invokes the remapping operation only when a certain condition is met. The condition is based on the probability of bit error occurrence, and it determines how much a target block is error prone.

For evaluation, we compare CRIM with PR with various workloads. We test how long a SSD can survive until all blocks run out, how many times a SSD can perform write operations when blocks are available, and how much a SSD reduces the bit error probability for each method. In our results, with CRIM, we show that SSD can survive more time than PR by maximum 230 days (12.6% of 5-year warranty time). CRIM also can endure more block writes than PR by 326.77, and this means that SSD can operate more time than PR by 17.9 % of 5-year warranty time when a workload with medium write ratio (Table 1) runs. Also, CRIM reduces the bit error probability by up to 182.23×10^{-8} over PR, and this means the 73-fold improvement of typical bit error rate 2.48×10^{-8} [13] in cloud data center.

Our work has two contributions. First, we suggest a novel SSD reliability approach that is based on prediction of bit error occurrence. Second, our work helps to reduce the overhead of reliability handling mechanism of SSDs in terms of lifetime loss.

In Section 2.1, we explain SSD background and review the previous work in bit error handling in SSD. Section 3 describes our problem of lifetime degradation. We suggest the main idea of CRIM in Section 1, and Section 5 presents experimental evaluation of CRIM and PR. Finally, we conclude in Section 7.

2. Background

2.1. SSD Basics. This section explains some key concepts of SSDs to aid the understanding of our work. Our work focuses on NAND flash memory-based SSDs.

2.1.1. Structure. A typical SSD storage device consists of multiple NAND flash memory *chips*. A chip consists of *dies*, and several *planes* are in a die. A plane is partitioned into *blocks*, and each block has a fixed number of *pages*. A page consists of multiple *bytes*, and it is an access unit for read and write operations using a loading/unloading page buffer. A basic unit for an erase operation is a *block*. Each block contains an array of *memory cells*. A memory cell includes one or more bits as encoding methods such as single-level cell (SLC), multilevel cell (MLC), and triple-level cell (TLC). A SLC has a bit data: 0 or 1. MLC and TLC have two bits and three bits data. It is known that many bits in a cell are prone to errors [14].

2.1.2. Write and Erase. A page physically cannot be overwritten. Thus, an erase operation must be done before each new write. Each cell in a block can only endure a limited number of erase operations, and the bit data in a cell cannot be guaranteed to be preserved when the operation count of the cell reaches the maximum count (*Maximum program-erase(P/E) cycles*).

2.1.3. Bit Errors. In SSDs, data in a cell can be corrupted by *bit errors* such as *read, write, and retention error*. *Read errors* occur when the read operations are performed. *Write errors* are bit errors when the data are written. *Retention errors* are bit errors that occur after the data is written. In SSD, every time when data are written, the SSD cell is recharged. However, the data may be corrupted over time due to natural leakage of charged electrons, and this is why retention errors occur. The time to retention error after data is written is *retention time*. The retention errors are known as the most dominant bit errors, and write errors are the second dominant one [12, 15]. Thus, we use the conventional bit error model considering both errors [5, 16]. The standard metric to evaluate SSD flash reliability is the raw bit error rate (RBER) of a SSD, defined as the number of corrupted bits per number of total bits read [13]. A recent study evaluates RBER of various SSD devices based on data of Google's data center, and the typical value of RBER in MLC SSD is reported as 2.48×10^{-8} [13].

2.1.4. Wear-Leveling. To avoid the "specific cell wear-out" described above, SSDs adopt a wear-leveling technique. This technique allows data writes to be evenly distributed over storage. The wear-leveling is achieved by an algorithm that the flash controller remaps logical block addresses to different physical block addresses in the SSD. The block addresses are checked for wear-leveling for every write request in the flash controller.

2.2. Related Work. Our work is mainly related to studies to enhance reliability of storage. In SSDs, there are two traditional approaches to enhance reliability: *reducing bit error occurrences* and *reducing the impact of bit errors*.

2.2.1. Reducing Bit Error Occurrences. To reduce *retention error* occurrences, periodic remapping (PR) [12] and retention relaxation (RR) [16] mechanisms have been suggested. PR [12] tried to reduce the impact of retention errors by periodically refreshing the stored data by mapping the data to a new block. However, PR has a limitation of periodic refreshing overhead which leads to additional lifetime loss. To overcome it, the authors in [12] suggest FCR. FCR has the same goals as CRIM but the differences are refresh condition and granularity. We cover these in more detail in Section 6.

RR [16] tried to relax the retention capability of NAND flash memory by shortening the refreshing time. However, RR has a limitation that the optimal refreshing time depends on specified datacenter workloads. CRIM tries to reduce the lifetime loss of PR and does not require physical modification of refreshing time. To reduce *write error* occurrences, disk-based write cache (DW) [17] is used in HDDs as a persistent write cache. In DW, all data written to SSDs are first appended in HDDs, and the data are migrated to SSDs later. However, this approach has a limitation that it requires HDD storage device additionally. CRIM does not require any additional HDDs.

Table 1: The summary of DWPD of test workloads.

Write ratio (WR)	Low			Medium			High		
Workload	MSR	Financial	OLTP	JEDES-client	Postmark	Cello99	JEDES-server-1	IOzone	JEDES-server-2
DWPD	0.005	0.05	0.14	1	2.8	5.5	10	20	30

2.2.2. Reducing Bit Error Occurrences. A solution is to make *data resilient*. Data differential redundant array of independent disks (Diff-RAID) [18] enhanced the data reliability by SSD RAID. Diff-RAID redundantly distributes parity blocks considering the age of storage device. However, this approach requires additional storage devices for the same amount of data; thus, it is costly. CRIM does not require additional storage.

Next solution is to *mitigate the programming interference*; two studies [19, 20] have been suggested. Dynamic voltage allocation (DV) [19] tried to reduce the interference by scaling the charging voltage. However, this approach requires additional storage capacity of device scaled by the number of cells. Progressive programming method (PP) [20] presented a single-level per cell- (SLC-) based progressive programming method to discover the lifetime of flash cell. By exploiting the gradual noise margin degradation, PP can sustain the bit data longer. However, this approach does not target modern SSDs such as MLC-based or TLC-based SSDs. CRIM does not require special programming method, nor targets a specific type of SSDs.

Our approach, CRIM, covers both approaches and also has a novelty. CRIM tries to reduce both retention errors and write errors by conditionally limiting invocation of remapping operations. Also, CRIM takes a new approach: prediction of bit error occurrences. In CRIM, a remapping operation is invoked only when a target block is error prone. Thus, CRIM can prevent the impact of bit errors effectively.

3. Problem: Lifetime Degradation due to Unconditional Remapping

This section explains our problem—lifetime degradation due to unconditional remapping operations. In order to inspect the impact of the problem, we analyze the amount of the lifetime degradation over a period of remapping operation (time between two remapping operations).

First, we need a metric to evaluate lifetime quantitatively in SSDs. It is difficult to know the exact lifetime of SSDs because it is to be measured through running workloads on the actual SSD until all blocks are unavailable. Instead of that, the estimated lifetime is used in many SSD studies [12, 17, 19, 20]. Commonly, the lifetime of SSD is estimated by endurable P/E cycles-the number of P/E cycles that memory cells could endure. For example, let us assume that there is a SSD with maximum 3 K P/E cycles, and all blocks in the SSD have 1 K P/E cycles. Then, the amount of endurable P/E cycles is 2 K. However, the endurable P/E cycle is not intuitive because this metric is not based on time. Thus, we use the extended estimated lifetime metric in a SSD vendor's white paper [14] which is as follows:

$$\text{Lifetime}_{\text{Years}} = \frac{(\text{Capacity}_{\text{GB}}) \times \left(\text{EndurablePE}_{\text{Cycles}}\right) \times (\text{Utilization}_{\%})}{(\text{UsagePerDay}_{\text{GB}}) \times (\text{WriteAmplification}_{\text{Ratio}}) \times \left(365_{\text{Days/Year}}\right)}. \tag{1}$$

This lifetime metric includes both endurable P/E cycles and *UsagePerDay*. Note that *UsagePerDay* can vary as the number of write operations of the given I/O workload. *WriteAmplification factor (WA)* is a constant for quantifying writing overhead. WA in [14] is 1.1. *Utilization* is a usage ratio of total block. Utilization of actively working SSD shows 95% in [14].

As the workload characteristics, a lifetime of SSD may appear differently. Especially, SSDs in enterprise servers assume active usage of 24 hours per day in JEDEC SSD standard [21]. Typically, SSD endurance is commonly described in terms of full *Drive Writes Per Day* (*DWPD*) for a certain warranty period (typically 3 or 5 years) [22]. Also, typical endurable P/E cycles for MLC SSD block are known as numbers range 3 K to 10 K in the recent literature [22]. For example, the estimated lifetime of SSD with endurable P/E cycles of 3 K, the capacity of 256 GB and the *UsagePerDay* of 256 GB is around 7.1 years. However, this assumes moderate SSD usage. The lifetime degrades by 10% if we assume the enterprise server of 10 DWPD; thus, it can seriously affect maintenance cost and reliability of server storage.

Second, we need to know the additional cost of a single remapping operation in PR study [12]. The steps of remapping operation are as follows: First, the data of a whole block are read out (*Read a block*). Second, if necessary, errors are corrected page by page (*Modify memory pages*). Third, the corrected block is written into another empty block (*Write a block*).

Finally, we can infer that the third step of remapping operation degrades the lifetime because the third step requires 'write a block' and the write operation decreases endurable P/E cycles in the above lifetime metric. Namely, if we perform remapping unconditionally, the problem of reduction lifetime can occur constantly. For solving that, our insight is to apply remapping operation conditionally.

4. Solution: Conditional Remapping Invocation Method

The basic idea of our solution is to invoke conditionally remapping operations. This operates like a filter against unconditionally flooding operations. We use a probability-based threshold and a raw bit error rate (RBER) model based on previous studies [5, 16] to determine the condition in which data blocks will be remapped. Figure 1 shows the overall architecture of CRIM. The white-filled boxes show traditional SSD components, and the black-filled boxes indicate new components for CRIM. CRIM requires two main software components in the flash controller, the *block-life-judge (BLJ)*, and the *block-life-saver (BLS)*. The BLJ decides when to apply the BLS, which performs remapping operations for the target block.

Figure 2 shows how CRIM works for incoming write and read operations from a host. Initially, CRIM waits for I/O operations from a host (state 0). In a typical SSD, ECC checking is performed whenever a read operation occurs (state 1). This is a conventional behavior of SSD [22]. When an ECC checking fails or a write operation occurs, BLJ is invoked to predict whether the target block is error prone or not (state 2). The details of BLJ are in Algorithm 1.

In Algorithm 1, RBER is expected to increase with the number of P/E cycles, and the growth was reported as exponential in the traditional literatures [5, 12]. However, in the recent literature [13], the authors have reported and verified the growth is close to a linear increase. Thus, we build a linear RBER model based on a-year-RBER data which is reported in [5]. The result is as follows:

$$\begin{aligned}\mathrm{RBER}(t,c) = {} & 9.991\times10^{-10}(c-1) + 1.0\times10^{-9} \\ & + 4.485\times10^{-4}\times t^{1.25},\end{aligned}\tag{2}$$

where t means retention time and c indicates the P/E cycles. We use the values that reflect the typical error rate of NAND flash memory that corresponds to a year-long data retention capability specified in JEDEC standard (SSD industrial standard [21]) under room temperature (e.g., 25°C).

Current is the amount of currently available P/E cycles, and Expected is the amount of changed P/E cycles. We can easily get the Current via wear-leveling management scheme with a timestamp; thus, we can represent it as time-series. Then, we obtain the Expected by a moving average filter through the feedback control way [23]. Moving average filter calculates the average value based on recent incoming data; therefore, it can reflect dynamically changing P/E cycles consumptions of workloads. Also, feedback control is a promising way to calculate expected values from time-series data. Using both techniques, BLJ can predict Expected adaptively for the dynamic workloads.

The ABER represents a criterion to classify two states: *error-prone* and *normal*. Acceptable raw bit error rate (ABER) is the maximum RBER that SSD can tolerate without any workloads under the given SSD device configuration (such as ECC code-size). SSD device vendors or SSD standards generally specify their ABER. For example, the uncorrectable bit error rate under typical workload execution should be less than 10^{-15} [24]. The RBER can increase during a year without additional operations such as read or writes by retention error. In CRIM, we get the ABER by calculating maximum retention error rate during a year. Workload execution affects the increment RBER because bit errors are accumulated by write or read operations. If there exist some workloads to execute, the RBER will reach ABER faster than no workloads. Therefore, CRIM will invoke remapping operations conditionally when the expected RBER on next prediction exceeds ABER. The specific value of ABER can be changed depending on the flash chip type and error probability aspects. However, ABER is not dependent to a specific workload.

If there are no workloads to execute for a long time, retention errors can be accumulated by data ages (= retention time). Because retention errors depend on time, we need to estimate the deadline time to mitigate it in advance. We calculate the deadline time by solving the equation of $\mathrm{RBER}(t,c) = \mathrm{ABER}$ for t. We explain how it is solved in Appendix, and the solution for obtaining t is as follows:

$$t = e^{\log\left(\mathrm{ABER}-\left(9.991\times10^{-10}\cdot(c-1)+\left(1.0\times10^{-9}\right)\right)/\left(4.4485\times10^{-4}\right)\right)/1.25}.\tag{3}$$

We name the estimated time as the "expected retention time (ERT)." When BLJ predicts "error-prone," CRIM also instantly checks the ERT value (state 4 in Figure 2). If the ERT value does not expire, CRIM checks the ERT value lazily in a garbage collection (GC) task which is invoked when the SSD is idle (state 3 in Figure 2).

BLS is invoked when the ERT expires (state 5 in Figure 2). Then, BLS performs remapping according to 2.

The BLS sets the target block status as error prone and starts to search for a new free block in the free block pool (state 5 in Figure 2). Then, the BLS remaps the logical address to the physical address of the new block. When the free block pool is empty, current block mapping relation is maintained. After that, if a block error occurs, it is marked as a bad block and becomes unavailable by a bad block manager in SSD devices [25] (state 7 in Figure 2). This does not affect the conventional SSD structure because we utilize it only.

When BLS finishes remapping successfully, or BLJ predicts "normal," the ERT value is recalculated for preventing further retention errors (state 6 in Figure 2).

5. Evaluation

The purpose of our evaluation is to show that CRIM performs better than PR in terms of reliability and lifetime loss prevention. All evaluation data used to support the findings of this study have been deposited in the github repository (https://github.com/saintgodkyp/CRIM.git).

Evaluation points are two. First, we conduct the reliability analysis of CRIM and PR (PR-day and PR-week). Second, we evaluate the remapping costs of CRIM and PR.

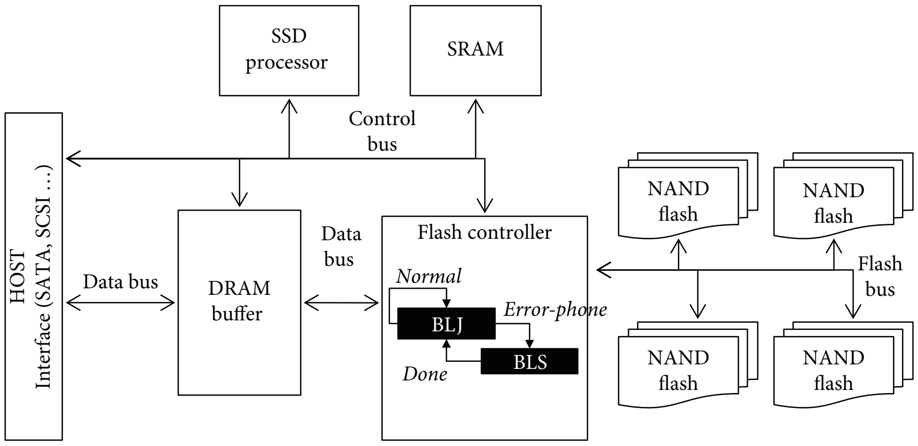

Figure 1: The overall architecture of CRIM.

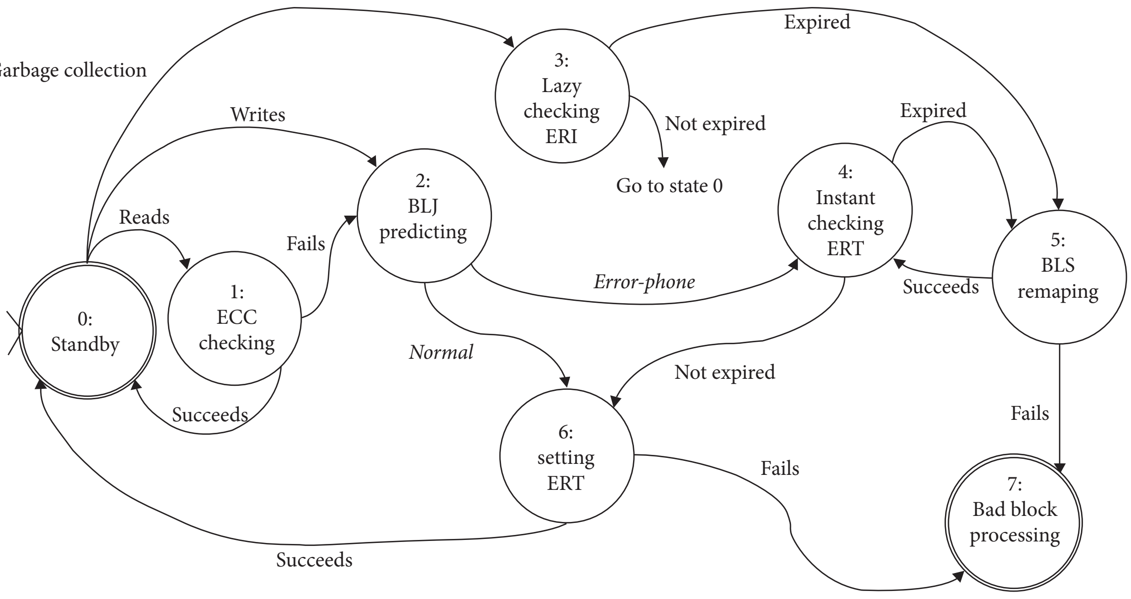

Figure 2: The state diagram of CRIM.

5.1. SSD Configurations and Workloads Characteristics. Our evaluation method is an analytic model validation method based on parameters obtained from simulations on realistic workloads. We collect workload characteristics using DiskSim 4.0 simulator with SSD extensions, which is commonly used in SSD studies. We set the SSD configuration as 256 GB with 2 bit-MLC (4 channels, 8 chips per channel, and 8192 blocks per chip) which is a typical SSD configuration and is used in PR [12]. We use various real workload trace data used in PR. The trace data include Financial, Cello99, Postmark, and MSR-Cambridge (MSR), and their read ratios are 23%, 38%, 52%, and 80%, respectively. From the trace data, we extract all DWPDs of the workload. Also, we consider application classes in JEDEC standard [21]. There are two classes: client and enterprise server. In terms of DWPD, we assume that a client (JEDES-client) has 1 DWPD and two enterprise severs (JEDES-server-1,2) have from 10 to 30 DWPD. We categorize our workloads into three classes as write ratio (WR), and summarize our workload characteristics in terms of DWPD in the following Table 1.

5.2. Evaluation Results

5.2.1. Reliability Analysis. To evaluate SSD's reliability, we simulate typical remapping methods (PR-day and PR-week) and CRIM. Our simulation program is written by using Python code (version 3.5.2). We assume that our SSD device has 256 GB storage and the endurable P/E cycles of each block are 3 K, and simulation time is 5 years, which is a typical SSD configuration used in PR [12].

We apply workload characteristics into them and conduct reliability analysis. To analyze SSD reliability, we measure "mean time to failures (MTTF)," "remained P/E cycles," and "last RBER." First, "MTTF" is a classic reliability analysis method [26] that is commonly used in evaluating system reliability. High MTTF means more reliable. In SSD context, we consider a failure to completely run out of available SSD blocks.

Table 2 shows the result of MTTF and remained P/E cycles. We categorize our results into three workload classes

```
**Data**: Current P/E cycles of target block (Current), Expected increment value of P/E cycle (Expected), Acceptable bit error rate (ABER), Retention time (t)
**Result**: "error-prone", "normal."
if (RBER($t$, (Current + Expected)) ≥ ABER) then
    return "error-prone";
end
else
    return "normal";
end
```

Algorithm 1: Block-life-judge.

```
Data: the logical address of target block (Logical), physical address of target block (Physical)
Result: "done"
SetErrorPronePhysicalBlock(Physical);
NewPhysical = GetFreePhysicalBlock();
Remap(Logical, NewPhysical);
return "done";
```

Algorithm 2: Block-life-saver.

Table 2: Results of mttf and remained P/E cycles for warranty period of 5 years.

Write ratio	Low		Medium		High	
Used metrics	MTTF	Cycles	MTTF	Cycles	MTTF	Cycles
Methods						
PR-day	1825	206.36	917.33	0	171	0
PR-week	1825	486.55	1125.66	0.72	181.66	0
CRIM-best	1825	533.13	1147.66	0.93	183.33	0

in terms of write ratio: low, medium, and high. The column "MTTF" shows the average MTTF of each workload class. For low WR, all methods can satisfy the five-year warranty period; namely, MTTF of all methods are five years (1825 days). In this case, some available blocks remain because total consumed P/E cycles based on DWPD of low WR is less than endurable P/E cycles. For example, DWPD of OLTP workload is 0.14 in Table 1. If we consider purely consumed P/E cycles during five years by OLTP workload, the whole P/E cycles are about 256 (= 1825 × 0.14). The P/E cycles are much lower than the endurable P/E cycles (3 K); thus, 2744 available blocks remain. However, for medium WR, all methods cannot reach the warranty period. MTTF of PR-day in medium WR shows 50.26% of 5 years, and one of PR-week is 61.68%. CRIM in medium WR shows 62.89%, and this is the longest MTTF. CRIM can survive 230 days or more than PR-day, and 22 days or more than PR-week. In worst cases, for high WR, all methods except CRIM cannot reach 10% of 5 years warranty period. In PR-day, SSD has been unavailable when the time reaches 9.37% of warranty period. PR-week shows MTTF of 9.95% of warranty period. CRIM shows the longest MTTF of 10.05% of warranty period, and this result is 12 days and 2 days longer MTTF than PR-day and PR-week. These several days are precious time for handling SSD device failures; thus, CRIM can save the time.

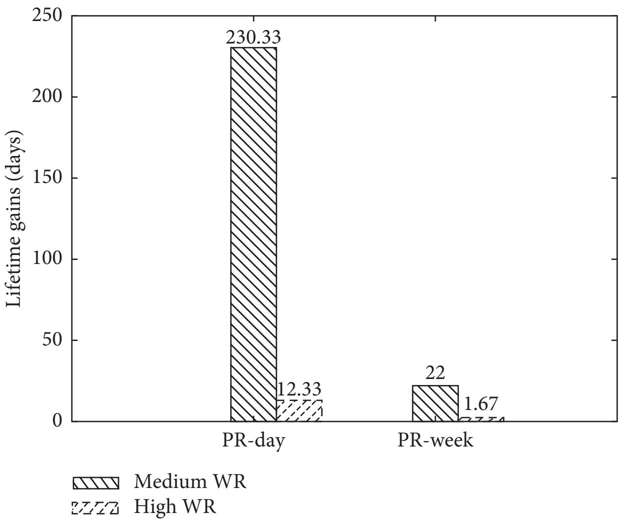

Figure 3: Average lifetime gain of CRIM against PR.

To clearly show the effectiveness of CRIM, we calculate the average lifetime gain against PR-day and PR-week in Figure 3. From the result, we observe that CRIM can extend a lifetime of SSD more than PR by maximum 230 days, which means 12.6% of 5-year warranty period.

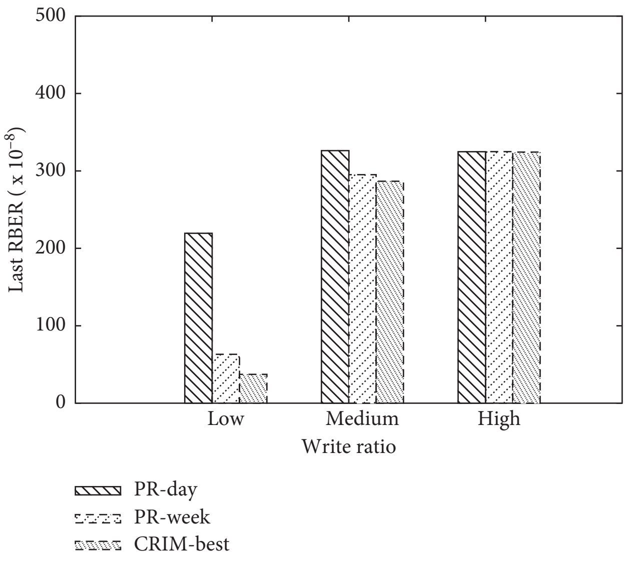

Figure 4: Average last RBER.

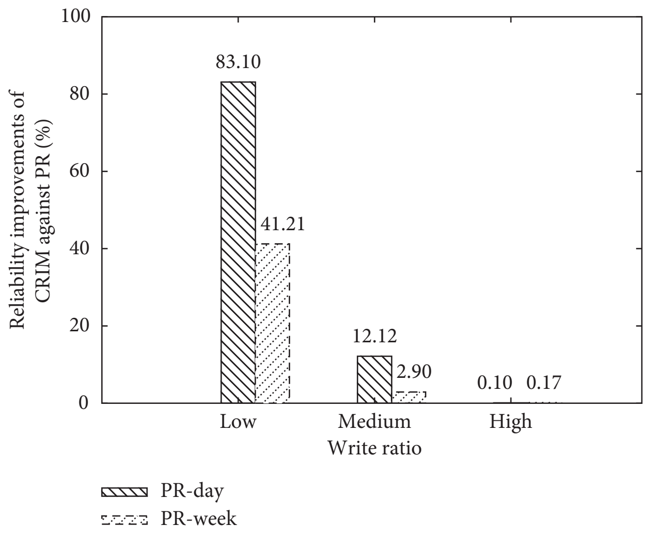

Figure 5: Reliability improvements against PR-day and PR-week.

Second, the "remained P/E cycles" indicates how long we will use SSD in the future, namely, currently available lifetime. Larger remained P/E cycles are better. The column "cycles" in Table 2 shows the result of remained P/E cycles after running each workload. For low WR, PR-day and PR-week show the remained P/E cycles on average of 206.36 and 486.55. CRIM shows the best result of 533.13. For medium WR, all workloads that DWPD of Postmark or more run out of available P/E cycles. PR-day shows a worst result of the remained P/E cycle of zero. PR-week shows 0.72, but CRIM shows the largest result of 0.93. For high WR, all methods cannot survive during 5-year warranty period; thus, all methods show zero of remained P/E cycles. From the best result, CRIM saves the remained P/E cycles up to maximum 326.77, and with the remained P/E cycles, a workload of medium write ratio (e.g., JEDES-client in Table 1) can keep running more time than PR by 17.9% of 5-year warranty.

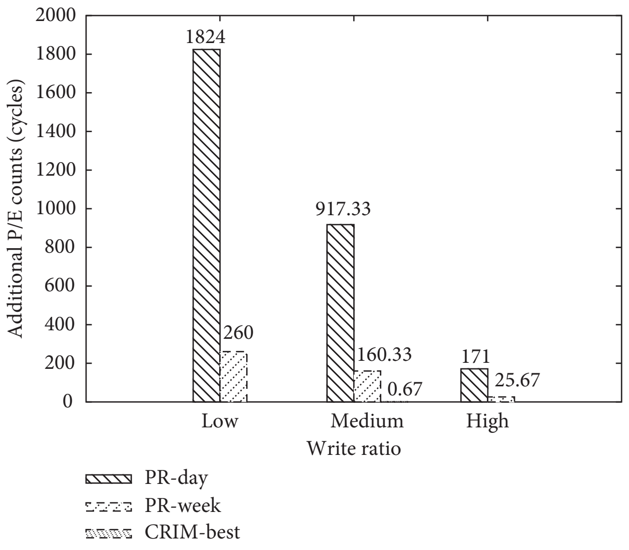

Figure 6: Comparison of remapping cost.

Third, "last RBER" means that how much bit errors can occur when SSD operations perform to target block. Low last RBER shows better reliability. In short, for better reliability, a target method should have a higher MTTF, larger remained P/E cycles and lower last RBER. We summarize average last RBER in Figure 4. For each WR, we measure last RBER after completing all block writes, and the values are scaled by 10^{-8}. For low WR, last RBER of PR-day and PR-week show 219.29 and 63.03 on average. CRIM shows the best RBER of 37.06; thus, the error rate is lowest. For medium WR, last RBER of PR-day and PR-week show 326.14 and 295.19 on average. CRIM shows the best RBER of 286.61; thus, the error rate is the lowest in three methods. For high WR, CRIM also shows the lowest last RBER, but all methods have high RBER of 324 or more. This is because workloads in high WR consume available P/E cycles fast, and write errors affect the entire bit errors seriously rather than retention errors.

We summarize CRIM's gains of reliability in Figure 5. The reliability improvement in Figure 5 indicates how much CRIM reduces average last RBER against PR methods (PR-day and PR-week) relatively. The calculation formula is as follows:

$$\frac{\text{Avg. last RBER}_{\text{PR}} - \text{Avg. last RBER}_{\text{CRIM}}}{\text{Avg. last RBER}_{\text{PR}}}. \tag{4}$$

For workloads in low and medium WR, CRIM can enhance reliability of SSD block access as much as maximum 83% on average. Workloads in high WR, CRIM does not show effectiveness because of heavy write errors. From the best result, CRIM reduces the bit error probability more than PR by up to 182.23×10^{-8}. Considering the typical bit error rate-2.48×10^{-8} [13], CRIM shows the 73-fold improvements more than PR.

5.2.2. Remapping Cost. To analyze the remapping cost that reduces available lifetime of SSD, we take each average of additional P/E cycles of low, medium and high WR workloads with PR-day, PR-week and CRIM. Namely, more additional P/E cycles indicate more lifetime loss. In Figure 6, we can see that the average lifetime loss of CRIM-best is only maximum 0.67, and it is the smallest in the all methods. This results show that CRIM can effectively remove unconditionally invoking hundreds or more remapping operations (maximum 1824 and 260 operations in PR-day and PR-week). Note that CRIM shows better lifetime gains in Figure 3, and this is the reason of it.

6. Discussion

The state of the art of PR is Adaptive-rate FCR suggested in [12]. It has the same goals as CRIM but also has differences. Thus, we discuss two questions: what are different points and what are CRIM's benefits. Unfortunately, Adaptive-rate FCR is not open source software; thus, it is difficult to compare it accurately with CRIM in running code level. Instead, we try to compare CRIM with the logic of adjusting the refresh period in Adaptive-rate FCR.

6.1. Common and Different Points. Commonly, both CRIM and the Adaptive-rate FCR have the same goal of mitigating refresh overhead and enhancing lifetime. The significant differences are two: refresh condition and granularity. The refresh condition determines when refresh invokes. The refresh granularity indicates how precisely a refresh is invoked.

With regard to the refresh condition in CRIM, when an SSD block failure is highly predictive, a refresh task is invoked. CRIM predicts a block access failure through a prediction model and performs conditional remapping according to the prediction result. In Adaptive-rate FCR, at the refresh period adjusted by the refresh rate, a refresh action is invoked. Adaptive-rate FCR adjusts this refresh rate from low to high as P/E cycles degrade.

The refresh granularity of CRIM is fine-grained as frequently as a write interval. The CRIM's block-life-judge (BLJ) is triggered when a write request is made. Regardless of the time interval length between writes, at the next write time, BLJ determines the error-prone state considering both the current bit error rates and available P/E cycles. The refresh granularity of Adaptive-rate FCR is limited to a daily refresh at the highest refresh rate. Adaptive-rate FCR can change the refresh rate from no-refresh to a daily refresh; it does not provide a more precise rate than the daily rate [12].

In short, CRIM mitigates bit errors by predictively refreshing on writes, and Adaptive-rate FCR mitigates bit errors by periodically refreshing on the adjusted rate. Also, CRIM provides a fine-grained refresh of using write intervals, and Adaptive-rate FCR provides a coarse-grained refresh (from a daily refresh to no refresh) than CRIM.

6.2. Benefits of CRIM. From the above two differences, CRIM has two benefits than Adaptive-rate FCR. First, CRIM can handle more bit errors with a fine-grained manner in a modern OS than Adaptive-rate FCR. From the comparison with CRIM and Adaptive-rate FCR, we identify that there can be bit errors that Adaptive-rate FCR cannot handle. These unhandled bit errors can deteriorate consistency in the file system of a modern OS. Modern OSs maintain the consistency between in-memory and on-disk data of the file system by typically using a journaling-based file system (e.g., Linux Ext4 and BSD log-structured file system (LFS)). These file systems periodically generate many write requests to synchronize their journals to SSD storage at every checkpoint (a periodic time to flush, typically seconds: e.g., five seconds in Linux Ext4 (https://www.kernel.org/doc/Documentation/filesystems/ext4.txt)). Because writes to blocks can occur per second, the bit error rate can increase per second. However, Adaptive-rate FCR cannot refresh them because its minimum refresh rate is daily (24 hours). Because CRIM does not miss any writes, bit errors on writes can be refreshed when it predicts an error-prone state.

Second, CRIM can extend lifetime more than Adaptive-rate FCR in a modern OS. The refresh rate of Adaptive-rate FCR increases as the available P/E cycles reduce. When the highest rate daily refresh is applied, Adaptive-rate FCR invokes a refresh every 24 hours. If the refresh rate is daily, it means that the available P/E cycles are low. However, within the 24 hours, bit errors can occur with a high probability. Bit errors under low available P/E cycles produce a block access failure, and then its MTTF is determined at the block access failure time. However, CRIM can handle bit errors regardless of the period; thus it can extend MTTF of the block because CRIM has already remapped it in advance at the previous write time.

7. Conclusion

The reliability of SSDs can be enhanced if blocks are remapped periodically (PR). However, it has a limitation that PR causes additional lifetime loss. We propose a new method, CRIM, in order to reduce the additional lifetime loss by conditionally invoking remapping operations. In our experiments, we show that CRIM can extend a lifetime of SSD more than PR by up to 12.6% to 17.9% of 5-year warranty time. Also, we show that CRIM can reduce the bit error probability of SSD by up to 73 times in terms of typical bit error rate in comparison with PR.

Appendix

Solving Expected Retention Time Equation

Original RBER equation is as follows:

$$\begin{aligned}\mathrm{RBER}(t,c) = 9.991\times10^{-10}(c-1) + 1.0\times10^{-9}\\ + 4.485\times10^{-4}\times t^{1.25}.\end{aligned}\tag{A.1}$$

Our goal is to solve the equation $\mathrm{RBER}(t,c) = \mathrm{ABER}$ for t, and the target equation is as follows:

$$\begin{aligned}9.991\times10^{-10}(c-1) + 1.0\times10^{-9}\\ + 4.485\times10^{-4}\times t^{1.25} = \mathrm{ABER}.\end{aligned}\tag{A.2}$$

Let us define k and r as

$$k = 9.991 \times 10^{-10}(c-1) + 1.0 \times 10^{-9},$$
$$r = 4.485 \times 10^{-4}. \tag{A.3}$$

Then, the original equation will be

$$k + r \cdot t^{1.25} = \text{ABER}. \tag{A.4}$$

Now, we solve this for t as the following steps. First, subtract k term on the same side of "=" sign, and we get

$$r \cdot t^{1.25} = \text{ABER} - k. \tag{A.5}$$

Second, divide r term on the same side of "=" sign; note that r is not zero, and we get

$$t^{1.25} = \frac{(\text{ABER} - k)}{r}. \tag{A.6}$$

Here, let us define a constant p as $p = (\text{ABER} - k)/r$, and we get

$$t^{1.25} = p. \tag{A.7}$$

Next, we apply logarithm with base e on the same side of "=" sign, and we get

$$\log\left(t^{1.25}\right) = \log(p). \tag{A.8}$$

Then, we get t equation by arranging exponents as follows:

$$1.25 \times \log(t) = \log(p),$$
$$\log(t) = \frac{\log(p)}{1.25}, \tag{A.9}$$
$$t = e^{\log(p)/1.25}.$$

Finally, we get predicted t equation by substitution of all constants k, r and p as follows:

$$t = e^{\left(\log\left(\text{ABER}-\left(9.991\times10^{-10}\cdot(c-1)+1.0\times10^{-9}\right)\right)/\left(4.4485\times10^{-4}\right)\right)/1.25}. \tag{A.10}$$

Data Availability

Data used in this study can be availed from the link https://github.com/saintgodkyp/CRIM or from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the SW Starlab support program (IITP-2018-2015-0-00280) and supervised by the IITP (Institute for Information and Communications Technology Promotion). This work was also partly supported by IITP grant funded by the Korea government (MSIT) (no. 2015-0-00288, Research of Network Virtualization Platform and Service for SDN 2.0 Realization). This work was also supported by a Korea University Grant.

References

[1] B. K. Debnath, S. Sengupta, and J. Li, "Chunkstash: Speeding up inline storage deduplication using flash memory," in *Proceedings of USENIX Annual Technical Conference*, Boston, MA, USA, June 2010.

[2] B. Debnath, S. Sengupta, and J. Li, "Flashstore: high throughput persistent key-value store," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1414–1425, 2010.

[3] A. Beloglazov, R. Buyya, Y. C. Lee et al., "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, no. 2, pp. 47–111, 2011.

[4] A. Elzeiny, A. A. Elfetouh, and A. Riad, "Cloud storage: a survey," *International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)*, vol. 2, no. 4, pp. 342–349, 2013.

[5] N. Mielke, T. Marquart, N. Wu et al., "Bit error rate in NAND flash memories," in *Proceedings of 2008 IEEE International Reliability Physics Symposium (IRPS 2008)*, pp. 9–19, IEEE, Phoenix, AZ, USA, May 2008.

[6] G. Yadgar, E. Yaakobi, and A. Schuster, "Write once, get 50% free: saving SSD erase costs using WOM codes," in *Proceedings of 13th USENIX Conference on File and Storage Technologies (FAST 15)*, pp. 257–271, Santa Clara, CA, USA, February 2015.

[7] Y. Liang, Y. Chai, N. Bao, H. Chen, and Y. Liu, "Elastic queue: a universal SSD lifetime extension plug-in for cache replacement algorithms," in *Proceedings of the 9th ACM International on Systems and Storage Conference*, p. 5, ACM, Haifa, Israel, June 2016.

[8] T. Kim, S. Lee, J. Park, and J. Kim, "Efficient lifetime management of SSD-based raids using dedup-assisted partial stripe writes," in *Proceedings of 2016 5th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–6, IEEE, June 2016.

[9] S. Moon and A. Reddy, "Does raid improve lifetime of SSD arrays?," *ACM Transactions on Storage (TOS)*, vol. 12, no. 3, p. 11, 2016.

[10] C. Wang and S. Baskiyar, "Extending flash lifetime in secondary storage," *Microprocessors and Microsystems*, vol. 39, no. 3, pp. 167–180, 2015.

[11] H. Imai, *Essentials of Error-Control Coding Techniques*, Academic Press, Cambridge, MA, USA, 2014.

[12] Y. Cai, G. Yalcin, O. Mutlu et al., "Flash correct-and-refresh: retention-aware error management for increased flash memory lifetime," in *Proceedings of 2012 IEEE 30th International Conference on Computer Design (ICCD)*, pp. 94–101, Montreal, Canada, September 2012.

[13] B. Schroeder, R. Lagisetty, and A. Merchant, "Flash reliability in production: the expected and the unexpected," in *Proceedings of 14th USENIX Conference on. File and Storage Technologies (FAST '16)*, pp. 67–80, Santa Clara, CA, USA, February 2016.

[14] A. R. Olson and D. J. Langlois, "Solid state drives data reliability and lifetime," *Imation White Paper*, pp. 1–27, 2008.

[15] 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 *Proceedings of IEEE Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 204–206, IEEE, San Francisco, CA, USA, February 2011.

[16] R.-S. Liu, C.-L. Yang, and W. Wu, "Optimizing NAND flash-based SSDs via retention relaxation," in *Proceedings of 10th USENIX Conference on File and Storage Technologies*, USENIX Association, p. 11, San Jose, CA, USA, February 2012.

[17] G. Soundararajan, V. Prabhakaran, M. Balakrishnan, and T. Wobber, "Extending SSD lifetimes with disk-based write caches," in *Proceedings of FAST'10*, pp. 101–114, San Jose, CA, USA, February 2010.

[18] M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi, "Differential raid: rethinking raid for SSD reliability," *ACM Transactions on Storage (TOS)*, vol. 6, no. 2, p. 4, 2010.

[19] T.-Y. Chen, A. R. Williamson, and R. D. Wesel, "Increasing flash memory lifetime by dynamic voltage allocation for constant mutual information," in *Proceedings of 2014 Information Theory and Applications Workshop (ITA)*, pp. 1–5, IEEE, February 2014.

[20] G. Dong, Y. Pan, and T. Zhang, "Using lifetime-aware progressive programming to improve SLC NAND flash memory write endurance," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 6, pp. 1270–1280, 2014.

[21] JSST Association, *Solid-State Drive (SSD) Requirements and Endurance Test Method*, JEDEC STANDARD, JESD218B.01, Japan, pp. 1–28, 2016.

[22] R. W. Kaese, "Flash memory basics for SSD users," in *Proceedings of OpenZFS european conference*, pp. 1–15, OpenZFS, Paris, France, June 2015.

[23] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*, John Wiley & Sons, Hoboken, NJ, USA, 2004.

[24] S. Tanakamaru, M. Fukuda, K. Higuchi et al., "Post-manufacturing, 17-times acceptable raw bit error rate enhancement, dynamic codeword transition ECC scheme for highly reliable solid-state drives, SSDs," *Solid-State Electronics*, vol. 58, no. 1, pp. 2–10, 2011.

[25] JSST Association, *Bad Block Management in NAND Flash Memories*, ST Microelectronics, APPLICATION NOTE, AN 1819, Geneva, Switzerland pp. 1–7, 2004.

[26] C. Heising, *IEEE Recommended Practice for the Design of Reliable Industrial and Commercial Power Systems*, IEEE, New York, NY, USA, 1991.