International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 1 (2016) pp 578-585
© Research India Publications. http://www.ripublication.com

Prediction of Elapsed Time based Wear Leveling for NAND Flash Memory in Embedded Systems

Sung Ho Kim
Ph. D. Candidate, Department of Computer Engineering,
Yeungnam University, 214-1 Dae-dong, Gyeongsan-si,
Republic of Korea.
E-mail: boca123@ynu.ac.kr

Jong Wook Kwak*
Associate Professor, Department of Computer Engineering
Yeungnam University, 214-1 Dae-dong, Gyeongsan-si,
Republic of Korea.
E-mail: kwak@yu.ac.kr

Abstract

In recent years, many storage systems use NAND flash memory increasingly as their secondary storages. NAND flash memory has non-volatile memory characteristics with low power, low latency and high reliability. On the other hand, NAND flash memory has different issue, compared to existing secondary storages, which is the characteristics such as erase-before-write, low endurance and different operation unit. These problems can be solved by using address translation table, garbage collection and wear leveling techniques. Unfortunately, previous works are difficult to directly be applied to embedded systems because they did not consider the memory requirement. In this paper, we propose PET-WL, which stands for Prediction of Elapsed Time based Wear Leveling, and it can be efficiently applied to embedded systems. Our policy has the characteristics to predict the elapsed time of each block using the difference of the number of invalid pages. In experimental results, PET-WL prolonged the lifetime of NAND flash memory up to 430% and reduced the page migration cost up to 39%, compared to other techniques.

Keywords: NAND flash memory, wear leveling, garbage collection, embedded systems, elapsed time.

Introduction

Magnetic disks have been used as secondary storages for the last decades. As the increase of processor and main memory performance, secondary storages also demand the performance improvement. As reflecting this trend, many storage systems use NAND flash memory increasingly as their secondary storage, which is called SSD (Solid State Disk). NAND flash memory is a non-volatile memory with the characteristics such as low power, low access latency and high reliability. In addition, NAND flash memory can make high capacity with inexpensive cost. Because of these characteristics, NAND flash memory is adopted in variety devices like as tablets, smart-phones and embedded systems and so on.

On the other hand, NAND flash memory has very different three issues, compared to existing secondary storages. First, NAND flash memory consists of two different units: page and block. The page is the minimum unit for an operation for data, which correspond to the unit of program (i.e., write) or read operation. The block is the set of pages, which correspond to the unit of erase operation. In addition, NAND flash memory must perform the erase operation before the program operation for overwriting the page, and this feature is called erase-before-write. As shown in TABLE.1, the erase latency is slower up to from least 2 times to most 28 times, compared to program and read latencies. Therefore, the erase-before-write problem degrades the performance of overall NAND flash memory systems significantly.

Table 1: Performance Comparison of NAND Flash Memory Operations

Parameters	SLC	MLC
Read latency	25us	60us
Program latency	200us	800us
Erase latency	700us	1.5ms
The limited number of erases per block	10^5	10^4

Second, as the capacity of recent applications scales up, secondary storages demand more capacity as well. As reflecting this trend, NAND flash memory has tried to divide the number of bits per cell, which is called SLC (Single-Level Cell) and MLC (Multi-Level Cell) respectively. SLC has 1-bit of one cell, which has an endurance of about 100,000 of the limited number of erases per block. MLC has 2-bit of one cell, which has an endurance of about 10,000 of the limited number of erases per block. Unfortunately, NAND flash memory has a trade-off relation between the number of bits per cell and the endurance [1-2].

Third, NAND flash memory has a lower endurance from 10 times to 100 times as the limited number of erases per block, compared to conventional magnetic disks. To solve this endurance problem, many previous works try to prolong the lifetime of NAND flash memory, especially by garbage collection and wear leveling. However, previous works are difficult to be applied to embedded systems, which usually have memory constraints, because they did not consider the memory requirements in their implementations [3-10].

In this paper, we propose PET-WL (Prediction of Elapsed Time-based Wear Leveling) where can be applied to the embedded system efficiently which has memory requirement constraints. PET-WL has the characteristics to predict the elapsed time of each block using the difference of the number of invalid pages. As performing wear leveling, the elapsed time of each block can be obtained by calculating the number of invalid blocks for a given period of time and it can be used to identify the hot blocks or cold blocks for wear leveling.

This paper is organized as follows. Section 2 presents background of system architecture for general NAND flash memory and analyzes representative previous works about wear leveling. In section 3, we present a wear leveling model of PET-WL for NAND flash memory. Section 4 evaluates the performance of our proposal and analyzes the memory requirement, compared to other techniques. Finally, Section 5 concludes this paper.

Background and Related Works

A. Background

NAND flash memory has special characteristics such as erase-before-write, lower endurance and different operation unit and therefore, it cannot use a conventional file system of magnetic disk based secondary storages. Instead, NAND flash memory systems mainly use FTL (Flash Translation Layer) and by adopting FTL, NAND flash memory solves its own problems. Generally, FTL consists of three modules: allocator, wear leveler, and cleaner, as shown in Fig.1. Each module has the following characteristics [11].

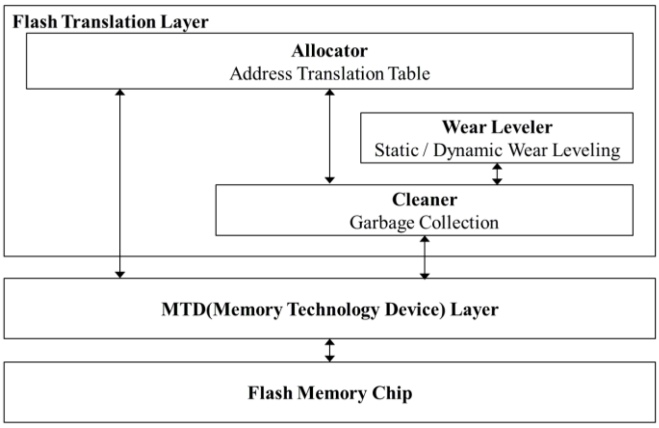

Flash Translation Layer
Allocator
Address Translation Table
Wear Leveler
Static / Dynamic Wear Leveling
Cleaner
Garbage Collection
MTD(Memory Technology Device) Layer
Flash Memory Chip

Figure 1: System Architecture of FTL

First, erase-before-write of NAND flash memory degrades overall system performance, which causes many problems such as long operation latency and unbalanced block endurance of NAND flash memory systems. To solve these problems, address translation table, that is the allocator, performs the program operation in an empty page of physical address (actual address of NAND flash memory), and then the physical address is mapped to logical address (virtual address defined by address translation table) by the address translation table. The main role of address translation table is for overwriting pages of NAND flash memory systems. Generally, address translation table is classified into page mapping, block mapping and hybrid mapping, according to the mapping unit and operations.

Second, as performing continual program operations, free empty page does not exist anymore in NAND flash memory systems. In this case, NAND flash memory has many invalid pages (i.e., dirty pages), which needs to erase some blocks among the allocated blocks. For the support of this operation, garbage collection module, i.e., the cleaner, copies all valid pages of allocated blocks selected by a garbage collection policy, and then it performs the erase operation of copied blocks. Garbage collection sets a minimal number of free blocks (erased block by garbage collection) through system configuration, which are generally about 5% of the total blocks.

Third, NAND flash memory has shorter lifetime, compared to magnetic disks. To prolong the lifetime of NAND flash memory, previous works try to solve this problem by balancing the endurance, which is called wear leveling by wear leveler. Wear leveling can be further classified into dynamic wear leveling and static wear leveling. Dynamic wear leveling is performed to frequently accessed blocks and the target of dynamic wear leveling is hot block only. On the other hand, static wear leveling is performed to all blocks and the target is all memory blocks in the flash memory systems.

B. Related Works

Previous works tried to prolong the lifetime through garbage collection and wear leveling [12-15], and the followings show the representative techniques with their features.

GA (Greedy Algorithm) is the representative technique of garbage collection, which selects a victim block (a selected block by garbage collection) using the number of invalid pages. GA performs garbage collection by considering the number of invalid pages, which select a victim block by only current time. Therefore, GA has the advantage for unconstraint memory requirement system only and it does not have any wear leveling policy [16].

DP (Dual Pool) proposed the wear leveling technique to identify hot or cold blocks, which has the set of hot blocks and the set of cold blocks, using hot pool and cold pool respectively. In its policy, a block is arbitrarily distributed to one of pools initially. If the maximum hot pool and the minimum cold pool exceed a user-configurable parameter TH, DP copies all valid pages in the maximum hot pool from the minimum cold pool, which is called as the operation DS (Dirty Swap). The use of DS has the sensitive problem in the changes of access patterns. Although DP tried to solve the DS problem by using an erasure cycle time (a time of last DS), the erasure cycle is not effective due to the time based on the erase count [17].

BET (Block Erase Table) was proposed to be applied to embedded systems. BET has a bit array in which each bit corresponds to a set of 2^k contiguous blocks where k is an integer that equals or exceeds 0. In BET, one flag is used in the tracking bit whether any one of the corresponding 2^k blocks is erased. In BET, e_{cnt} and f_{cnt} is the total erase count and flag erase count respectively. The ratio of e_{cnt} and f_{cnt} equals or exceeds a given threshold T, BET is invoked to trigger the cleaner to do garbage collection over selected blocks. BET is somewhat efficient in memory requirement but BET does not consider the migration cost of the cold blocks because the wear leveling using one flag is simply performed [18].

As shown in the techniques above, the problems of previous techniques do not consider a time period in the changes of write patterns as well as memory requirements for their implementations. In this paper, we use the information of time in the changes of write patterns and it can be used to identify hot blocks and cold blocks more efficiently with reasonable memory overhead.

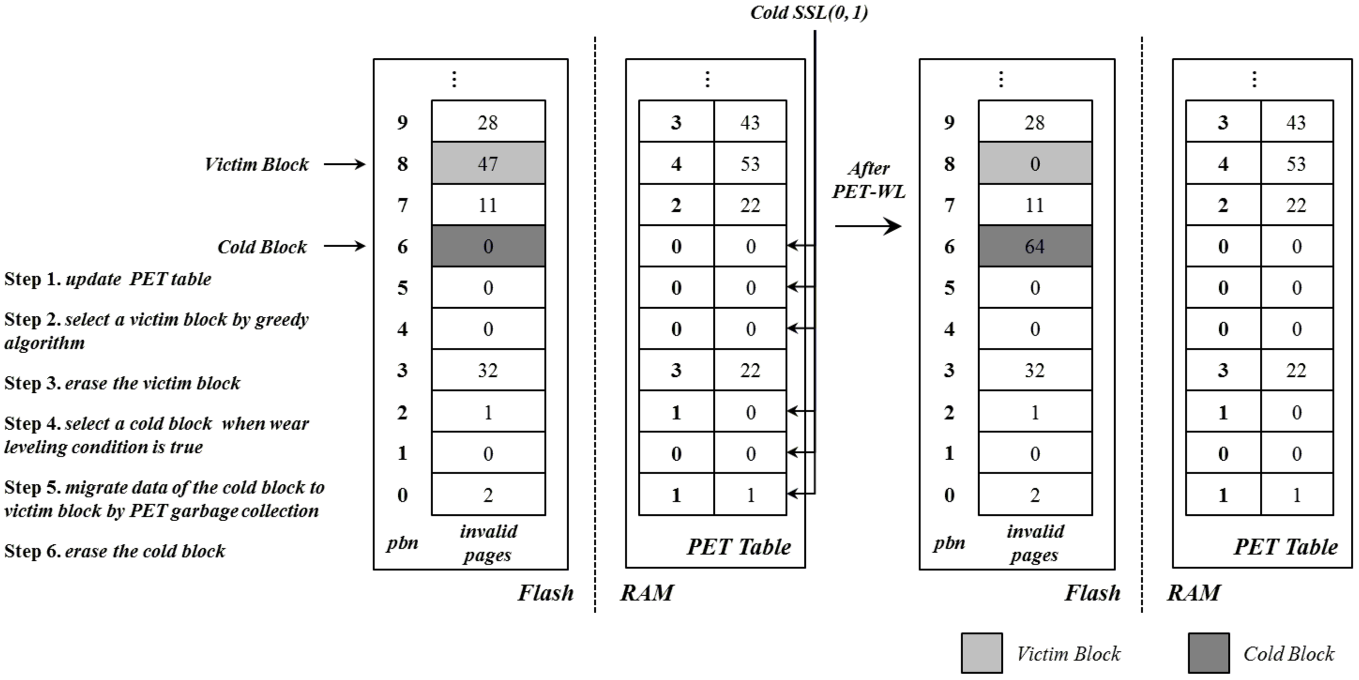
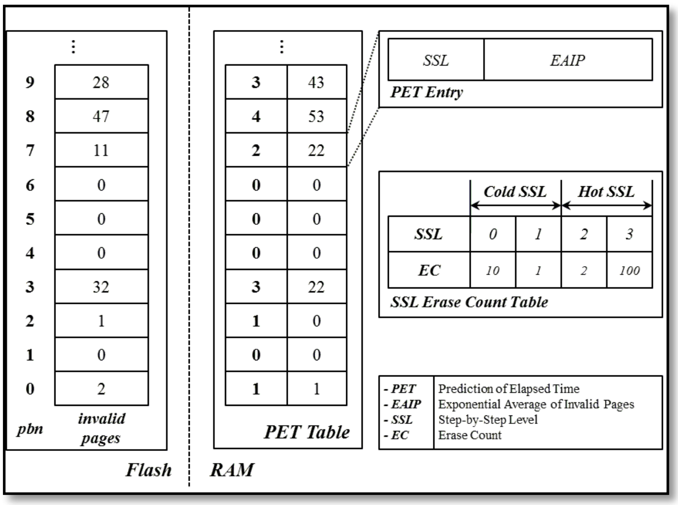

Cold SSL(0, 1)
Victim Block
Cold Block
Step 1. update PET table
Step 2. select a victim block by greedy algorithm
Step 3. erase the victim block
Step 4. select a cold block when wear leveling condition is true
Step 5. migrate data of the cold block to victim block by PET garbage collection
Step 6. erase the cold block
pbn
invalid pages
PET Table
Flash
RAM
After PET-WL
Victim Block
Cold Block

Figure 3: Procedure for PET Wear Leveling

Prediction of Elapsed Time-based Wear Leveling

A. Overview

This section describes the wear leveling technique by considering the changes of write patterns in a given time on embedded systems, which is called PET-WL (Prediction of Elapsed Time-based Wear Leveling). PET-WL can predict the elapsed time using a difference between the number of past invalid pages and the number of current invalid pages during a given time of last erase operation. In our proposal, the number of invalid pages to predict the elapsed time is a key component. In addition, the memory requirement of invalid page table in PET-WL is almost trivial, which is an advantageous feature to be applied to embedded systems.



SSL	0	1	2	3
EC	10	1	2	100

Figure 2: PET Wear Leveling Modules

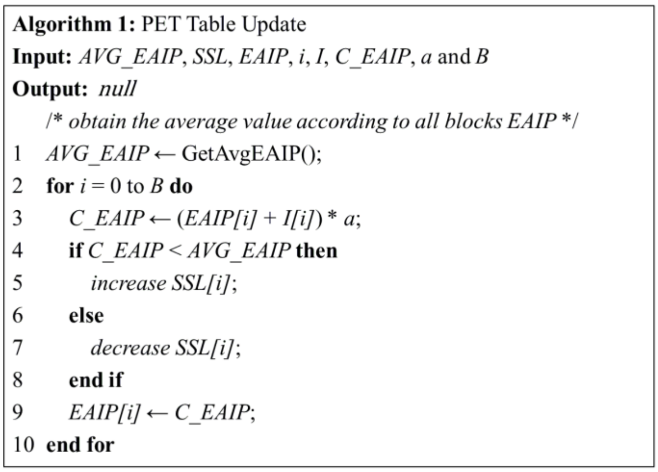
PET-WL presents two methods to measure the elapsed time using the number of invalid pages: EAIP (Exponential Average of Invalid Page) and SSL (Step-by-Step Level).

EAIP has the cumulative average from the number of past invalid pages to the number of recent invalid pages, which is used to identify hot blocks or cold blocks by calculating a difference of the cumulative average. If the difference of EAIP is more than the average of all blocks, the block becomes a hot block. Otherwise, the block becomes a cold block. Therefore, EAIP has a role to identify the hot blocks or cold blocks.

SSL is for identifying the accurate hot blocks or cold blocks and SSL classifies the blocks as four levels. If a level of SSL is close to zero, the block is very cold block. The very cold block means that the block is accessed very infrequently, such as kernel data area in operating system. On the other hand, very hot block means that the block is accessed very frequently, such as temporary local data. Therefore, SSL uses four levels to identify very hot, hot, cold and very cold blocks to measure the changes of write patterns during a given time of last erase operation.

PET Table is composed of PET Entry and each entry consists of the set of EAIP and SSL. PET Table is the mapping table of one-to-one mapping mechanism between the block and the PET Entry of all blocks according to NAND flash memory systems, as shown in Fig.2.

Besides, SSL Erase Count Table is used to prevent the frequent page migrations by PET-WL. SSL Erase Count Table has the number of erase count of each SSL according to an erased block by garbage collection and PET-WL, which is used as the condition to perform PET-WL, as shown in Fig. 2.



```
Algorithm 1: PET Table Update
Input: AVG_EAIP, SSL, EAIP, i, I, C_EAIP, a and B
Output: null
    /* obtain the average value according to all blocks EAIP */
1   AVG_EAIP ← GetAvgEAIP();
2   for i = 0 to B do
3       C_EAIP ← (EAIP[i] + I[i]) * a;
4       if C_EAIP < AVG_EAIP then
5           increase SSL[i];
6       else
7           decrease SSL[i];
8       end if
9       EAIP[i] ← C_EAIP;
10  end for
```

Figure 4: Pseudo-code of PET Table Update

B. PET Wear Leveling Algorithm

PET-WL preforms wear leveling through the following procedure when the condition of wear leveling is true. Fig.3 shows the procedure for the PET wear leveling and PET Table Update Algorithm in Fig.4 updates the EAIP and the SSL of PET Table whenever it performs garbage collection. The EAIP and the SSL measure the elapsed time through a difference in the number of invalid pages (Step 1).

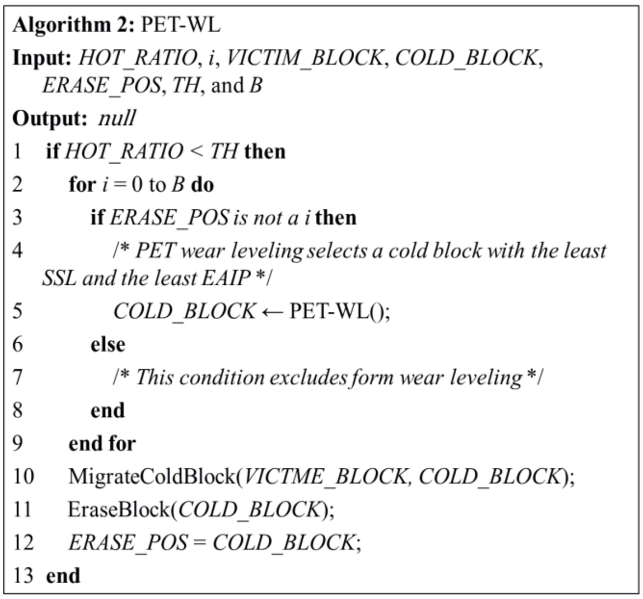
Algorithm 1 in Fig.4 illustrates the pseudo code of PET Table Update. To identify the hot blocks and cold blocks, PET-WL calculates the average (AVG_EAIP) of all blocks (Line 1). As repeating in each block, the current EAIP is obtained by equation (1), which is given by

$$\mathrm{C\_EAIP}_i = \mathrm{B}_{(\mathrm{I+EAIP})i} \times \alpha \qquad (1)$$

where B is the block, i is the block index, I is the number of invalid pages in the current block, EAIP is the past value of EAIP, α is the weight, and C_EAIP is the current value of EAIP. We use the number of both the past EAIP and the current EAIP. Therefore, we use 0.5 of α value. If C_EAIP is less than the value compared to the average EAIP (the obtained average EAIP by Line 1), the SSL is decreased. Otherwise, the SSL is increased (Lines 3-8). Then, EAIP is updated using C_EAIP to maintain the modified EAIP value (Line 9).

Then, garbage collection selects a victim block by GA (the representative technique of garbage collection). The GA copies all valid pages in the victim block to free blocks, which performs the erase operation at the victim block (Steps 2-3).

After GA operation, PET-WL performs wear leveling when the hot ratio of SSL Erase Count Table is more than TH (the defined hot ratio by user-configuration). The Optimal TH will be shown through experimental results in Section 4 later (Step 4).



```
Algorithm 2: PET-WL
Input: HOT_RATIO, i, VICTIM_BLOCK, COLD_BLOCK,
    ERASE_POS, TH, and B
Output: null
1   if HOT_RATIO < TH then
2       for i = 0 to B do
3           if ERASE_POS is not a i then
4               /* PET wear leveling selects a cold block with the least
    SSL and the least EAIP */
5               COLD_BLOCK ← PET-WL();
6           else
7               /* This condition excludes form wear leveling */
8           end
9       end for
10      MigrateColdBlock(VICTME_BLOCK, COLD_BLOCK);
11      EraseBlock(COLD_BLOCK);
12      ERASE_POS = COLD_BLOCK;
13  end
```

Figure 5: Pseudo-code of PET Wear Leveling

Algorithm 2 in Fig.5 illustrates the pseudo code of PET-WL. As repeating each block, PET-WL selects the corresponding block of the least SSL and the least EAIP, which is called a cold block as selected by PET-WL. In the selection procedure of the cold block, we exclude the selected cold block by the prior PET-WL, which is a procedure to prevent the frequent selection of the cold block (Lines 2-9). PET-WL copies all valid pages in the cold block to a victim block, which is the balanced procedure for both hot pages and cold pages, and then the cold block performs the erase operation (Lines 10-11). Finally, PET-WL updates the block position (i.e., the block index) according to the selected cold block (Line 12).

Fig.3 shows the PET-WL procedure with an example. Garbage collection module selects block 8 by GA, which copies all valid pages in block 8 to free blocks, and then erase operation is performed at block 8 (Steps 1-3). PET-WL checks whether the wear leveling condition is true or not. If the wear leveling condition is not true, PET-WL completes the procedure of garbage collection. Otherwise, PET-WL selects block 6 by a cold block selection policy (Step 4), and PET-WL copies all valid pages in block 6 to block 8. Then, PET-WL erase block 6 and finally, PET-WL updates the position of block 6 (Steps 5-6).

Table 2: Parameters of Simulation Environment

Parameter	Value and Description
Total capacity	8GB
Reserved free blocks	15%
Garbage Collection Trigger	the number of free blocks performed under 5%
Flash chip elements	1
Planes per elements	1
Blocks per plane	2048
Pages per block	64
Page size	8KB
Page read latency	60us
Page program latency	800us
Block erase latency	1.5ms
Lifetime per block	10^4

Performance Evaluation

A. Experimental Environment

In this section, we evaluate the performance of PET-WL using the SSD model, based on DiskSim Simulation 4.0 [19-20]. We use the parameters of simulation environment of the MLC NAND flash memory specification developed by Samsung Electronics, as shown in TABLE.2.

In our experiments, the datasets consist of three trace files. As shown in TABLE.3, each trace is used the following hot and cold ratio.

TABLE.3. A Dataset of Simulation Environment

Trace	Hot ratio (%)	Cold ratio (%)
case1	10	90
case2	20	80
case3	30	70

The detailed pattern of trace files is based on the distribution by equation (2), which is given by

$$p = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-u)^2/2\sigma^2} \qquad (2)$$

where u is the mean of the distribution. The parameter σ is its standard deviation. As shown in Fig.6, the parameter σ of the distribution is the percentage of cold ratio.

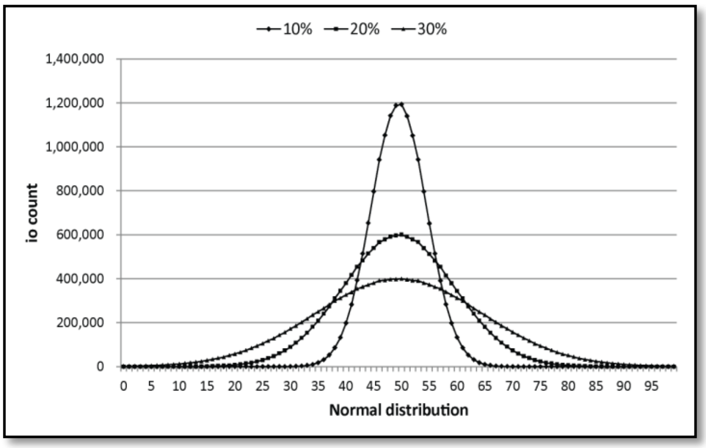


Figure 6: Normal Distribution of Each Case

B. Performance Analysis

We evaluated the performance of PET-WL through two experiment results: the first bad time and the page migration overhead during the garbage collection and the wear leveling. The first bad time is critical performance indicator, which shows the system failure time. The system failure means that NAND flash memory cannot be performed as a correct secondary storage. In addition, wear leveling module improves the lifetime through the forced migration of cold data. However, this migration generates additional page overhead. Therefore, we also have to consider migration overhead to reduce the system overhead as well as lifetime extension. Finally, we analyzed the memory requirement of PET-WL and showed the efficiency of our proposal to be applied to embedded systems.

i. Experimental Results and Discussion

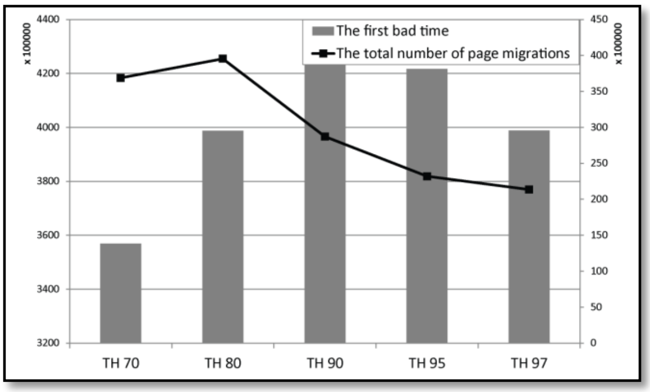


Figure 7: Performance Evaluations according to Each Threshold TH of PET-WL

We first evaluated the performance of PET-WL according to each threshold TH. By this simulation, we can identify an optimal TH value and then, we evaluated the performance of our proposal using the optimal TH of PET-WL, compared with GA, DP and BET techniques. In order to obtain the optimal TH, Fig.7 presented the optimal TH through five cases. Compared to other TH values, TH 90 and TH 95 are the optimal results in PET-WL. The value shows optimal performance between the first bad time and the total number of page migrations and therefore, our following experiments use these two TH values.

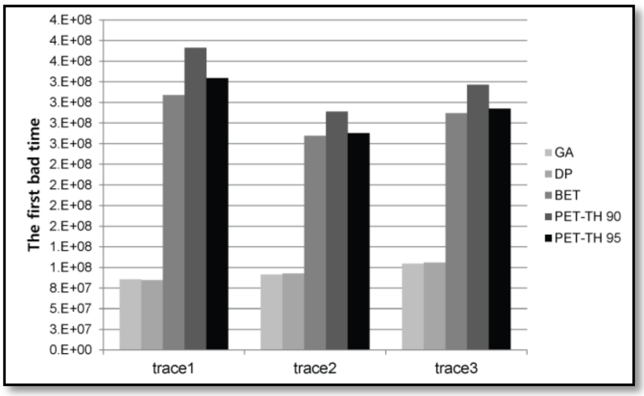


Figure 8: The First Bad Time

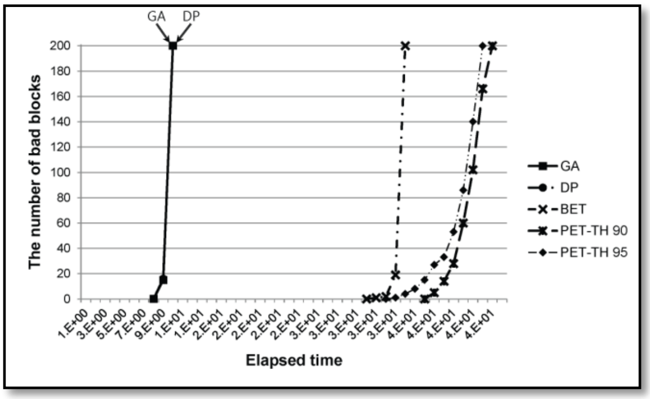


Figure 9: The Number of Worn-out Blocks as Time Progresses

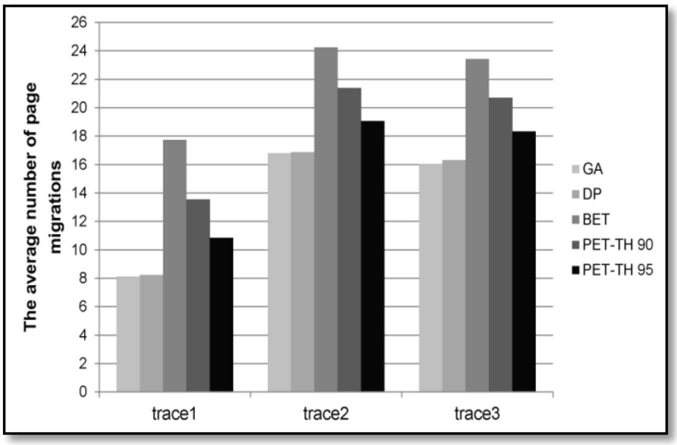


Figure 10: The Average Number of Page Migrations

Next simulation shows the first bad time. We measured the performance of the first bad time through two assumptions. First assumption is that the endurance of blocks is 10^3. Second, if NAND flash memory system has the number of bad blocks more than 10% among the whole blocks, it is assumed that the system is a failure condition.

Fig.8 shows the first bad time as a key performance indicator. The TH 90 of PET-WL prolonged the lifetime performance up to 429%, 430% and 68%, and the TH 95 of PET-WL prolonged the lifetime performance up to 386%, 387% and 25%, compared with GA, DP and BET.

Fig.9 shows the elapsed time for the system failure after the first bad time. PET-WL showed the slow progress trend after the first bad time, compared to other techniques. As this characteristic of PET-WL, it means that PET-WL performs wear leveling more accurately by measuring precise elapsed time to select the cold blocks unlike other techniques.

Next, to measure the migration overhead of PET-WL, we measured the performance of page migration overhead through two assumptions during the garbage collection and the wear leveling. We assumed 10^4 as the endurance of each block and we measure the overhead of migration by performing 10^8 programming operation in NAND flash memory systems.

Fig.10 shows the page overhead during the garbage collection and the wear leveling. GA does not consider the additional page overhead and it does not perform wear leveling either. Although DP performs wear leveling, it shows lower performance and more overhead than GA. DP performs wear leveling to migrate the frequently accessed blocks only. The TH 90 of PET-WL reduced the page migration overhead up to 24% and the TH 95 of PET-WL reduced the page migration overhead up to 39%, compared to BET.

In addition, the page migration overhead of wear leveling affects the response time of overall NAND flash memory systems. As shown in Fig.11, the TH 90 of PET-WL increased the response time about 40% and the TH 95 of PET-WL increased the response time about 20% on average respectively, compared to GA and DP. However, GA and DP suffer from the occurrence of severe first bad time as shown in Fig.8, so this comparison is meaningless. Compared to BET, the TH 90 of PET-WL reduced the response time up to 24% and the TH 95 of PET-WL reduced the response time up to 39%.

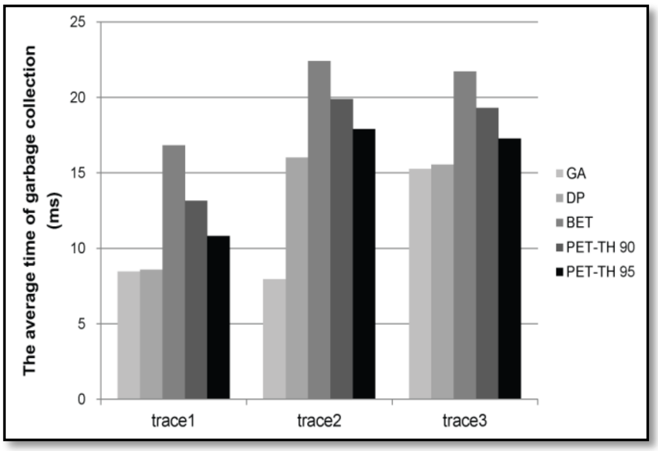


Figure 11: The Average Time of Garbage Collection

Table 4: Parameters of Simulation Environment

Technique	Memory Requirement
GA	No consideration
DP	ptr (word) : the pool address of each blocks elapsed time (double) : the last time of DS (Dirty Swap) mem = B * (ptr + elapsed time)
BET	b (bit) : the bit flag of each blocks mem = B * b
PET	entry (byte) : the PET entry erase pos (word) : the erase position of past cold blocks SSL EC (word) : the erase count of four SSL mem = (B * u) + (SSL EC * 4) + erase pos

B : The number of blocks, mem : memory requirement

ii. Memory Requirement

In this section, we analyze the memory requirement of our proposal to apply efficiently to embedded systems. As shown in TABLE.4, we discuss the memory requirement using the number of blocks in experiment setup. GA does not require additional memory, because it just checks the number of invalid pages and it does not have wear leveler module. DP requires the memory space of the pool address and the elapsed time table corresponding to all blocks, which consumes about 24Kbyte. BET requires the memory space of one bit flag corresponding to predefined block groups, which consumes about 256byte. In PET-WL, PET Table requires the memory of the PET entry corresponding to all blocks. In addition, SSL Erase Count Table requires the memory of four levels of SSL and the erase position corresponding to cold blocks. Therefore, PET-WL consumes about 2Kbyte.

In recent years, embedded systems are increasing the memory capacity. Compared to GA and BET, PET-WL requires more memory, but the requirement of PET-WL memory is little about 2Kbyte. With this slightly increased memory overhead, on the other hand, PET-WL improved the performance up to 430% of the overall NAND flash memory systems. Compared to DP, PET-WL reduced the memory requirement up to 16 times and improved the overall system performance. Therefore, our proposal PET-WL is efficient solution to be applied to the embedded systems because PET-WL performs the efficient wear leveling using the reasonable memory requirement.

Conclusion

In recent years, many storage systems use NAND flash memory increasingly as their secondary storages. NAND flash memory has non-volatile memory characteristics with low power, low latency and high reliability. On the other hand, NAND flash memory has different issue, compared to existing secondary storages, which is the characteristics such as erase-before-write, low endurance and different operation unit. These problems can be solved by using address translation table, garbage collection and wear leveling techniques. However, previous works are difficult to directly be applied to the embedded system because they did not consider the memory requirement.

In this paper, we proposed PET-WL, and it can be efficiently applied to embedded systems. Our policy has the characteristics to predict the elapsed time of each block using the difference in the number of invalid pages and the number of current invalid pages during the time of last erase operation. In our policy, the number of invalid pages is used to predict the elapsed time during a given time in the changes of write patterns. Our PET-WL is composed of two modules: PET Table and PET Erase Count Table. PET Table has the values to identify hot blocks or cold blocks, and PET Erase Count Table performs the wear leveling to reduce page migration costs. In addition, the memory requirement of invalid page table in our proposal is trivial, which is advantageous to be applied to embedded systems. In experimental results, PET-WL prolonged the lifetime of NAND flash memory up to 430% and reduced the page migration cost up to 39%, compared to previous technique, such as GA, DP and BET.

Acknowledgement

Second author(*) is a corresponding author and this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2014R1A1A2057146).

References

[1] MT29F4G08ABADAWP 8Gbit SLC NAND Flash Memory Data Sheet, http://www.micron.com, Micro Technology, 2009

[2] K9GAG08U0M 2G x 8bit MLC NAND Flash Memory Data Sheet, Samsung Electronics, https://www.samsung.com, Feb, 2007

[3] Wang, Chundong, and Weng-Fai Wong. "Observational wear leveling: an efficient algorithm for flash memory management." Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE. IEEE, 2012.

[4] Lee, Junghee, et al. "A semi-preemptive garbage collector for solid state drives." Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on. IEEE, 2011

[5] Chang, Li-Pin, and Li-Chun Huang. "A low-cost wear-leveling algorithm for block-mapping solid-state disks." ACM SIGPLAN Notices. Vol. 46. No. 5. ACM, 2011.

[6] Lin, M. W., and Song Yan Chen. "Swap time-aware garbage collection policy for NAND flash-based swap system." Electronics Letters 49.24 (2013): 1525-1526.

[7] Lim, Seung-Ho, and Kyu-Ho Park. "An efficient NAND flash file system for flash memory storage." Computers, IEEE Transactions on 55.7 (2006): 906-912.

[8] Ji, Seunggu, and Dongkun Shin. "Locality and duplication-aware garbage collection for flash memory-based virtual memory systems." Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on. IEEE, 2010.

[9] Murugan, Muthukumar, and David HC Du. "Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead." Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on. IEEE, 2011.

[10] Liao, Jilong, et al. "Adaptive Wear-leveling in Flash-based Memory." (2014).

[11] Ma, Dongzhe, Jianhua Feng, and Guoliang Li. "A survey of address translation technologies for flash memories." ACM Computing Surveys (CSUR) 46.3 (2014): 36.

[12] Yang, Ming-Chang, et al. "Garbage collection and wear leveling for flash memory: Past and future." Smart Computing (SMARTCOMP), 2014 International Conference on. IEEE, 2014.

[13] Jeong, Jaehyeong, and Yong Ho Song. "A technique to improve garbage collection performance for NAND flash-based storage systems." Consumer Electronics, IEEE Transactions on 58.2 (2012): 470-478.

[14] Chung, Ching-Che, Duo Sheng, and Ning-Mi Hsueh. "A high-performance wear-leveling algorithm for flash memory system." IEICE Electronics Express 9.24 (2012): 1874-1880.

[15] Yang, Ming-Chang, et al. "New ERA: new efficient reliability-aware wear leveling for endurance enhancement of flash storage devices." Proceedings of the 50th Annual Design Automation Conference. ACM, 2013.

[16] Kwon, Ohhoon, et al. "FeGC: An efficient garbage collection scheme for flash memory based storage systems." Journal of Systems and Software 84.9 (2011): 1507-1523.

[17] Chang, Li-Pin. "On efficient wear leveling for large-scale flash-memory storage systems." Proceedings of the 2007 ACM symposium on Applied computing. ACM, 2007.

[18] Chang, Yuan-Hao, Jen-Wei Hsieh, and Tei-Wei Kuo. "Improving flash wear-leveling by proactively moving static data." Computers, IEEE Transactions on 59.1 (2010): 53-65.

[19] V. Prabhakaran and T. Wobber, “SSD Extension for DiskSim Simulation Environment”, 2010.

[20] BUCY, J., SCHINDLER, J., SCHLOSSER, S., AND GANGER, G. “The DiskSim Simulation Environment Version 4.0” http://www.pdl.cmu.edu/DiskSim, 2010