LETTER

HaWL: Hidden Cold Block-Aware Wear Leveling Using Bit-Set Threshold for NAND Flash Memory

Seon Hwan KIM†, Ju Hee CHOI††, Nonmembers, and Jong Wook KW Ak†, Member

SUMMARY In this letter, we propose a novel wear leveling technique we call Hidden cold block-aware Wear Leveling (HaWL) using a bit-set threshold. HaWL prolongs the lifetime of flash memory devices by using a bit array table in wear leveling. The bit array table saves the histories of block erasures for a period and distinguishes cold blocks from all blocks. In addition, HaWL can reduce the size of the bit array table by using a one-to-many mode, where one bit is related to many blocks. Moreover, to prevent degradation of wear leveling in the one-to-many mode, HaWL uses bit-set threshold (BST) and increases the accuracy of the cold block information. The performance results illustrate that HaWL prolongs the lifetime of flash memory by up to 48% compared with previous wear leveling techniques in our experiments.

key words: wear leveling, NAND flash memory, bit array table, hidden cold block problem, bit-set threshold

1. Introduction

Flash memory has advantages in that it provides fast access speed, low power, and a low price. Therefore, it is widely used in many sectors of the electronics industry. However, flash memory has a weak point, which is the limited number of program/erase (P/E) cycles.

Many wear leveling techniques have been studied to prolong the lifetime of flash memory by equalizing the P/E cycles of the blocks. Wear leveling uses information tables, which obtain the erase count of the blocks, the elapsed time of accesses, and so on [1], [2]. The cost benefit (CB) technique measures the elapsed time of invalid pages in each block and selects a victim block for wear leveling [3]. The cost age time (CAT) technique uses the erase count of the blocks and the elapsed time of invalid pages [4]. However, they do not involve all blocks in wear leveling. Therefore, the limited number of blocks participates in wear leveling and the performance of lifetime extension can be low for flash memory.

To involve all blocks in wear leveling, static wear leveling (SWL) using a block erase table (BET), which is composed of a bit array, was studied. BET saves the histories of the erase operations in each block [5]. If there are no erased blocks during a given time period, SWL determines that the blocks are cold blocks. Subsequently, SWL migrates the cold blocks to free blocks to fulfill wear leveling. SWL can also map one bit to many blocks to reduce the size of a BET. However, if one bit is related to two or more blocks, the performance of wear leveling degrades. Information about cold blocks disappears with the erase operation of a hot block, even though cold blocks exist in related blocks using one bit. We call this the hidden cold block problem. Figure 1 shows the hidden cold block problem in SWL.

Fig. 1 The distribution of the number of block erases by increasing the values of $k$ in SWL.
• HaWL defines and identifies the hidden cold block problem.
• In page mapping-based FTL (pFTL), HaWL prolongs the lifetime of flash memory without additional overhead, compared with SWL and other techniques.
• In block mapping-based FTL (NFTL), HaWL provides best lifetime increase for all cases, compared with SWL, and reduces memory requirement, compared with other techniques (CB and CAT).

2. The Proposed HaWL Technique

HaWL is based on the SWL technique and it adopts BET, wear leveling trigger conditions, and parameters $T$ and $k$ [5]. In addition, HaWL also has the advantages of SWL by using a bit array table and low cost operations for wear leveling. Compared with SWL, the differences with HaWL are to set bits of a BET by using the BST and to exploit a dedicated block for migrations during wear leveling. Consequently, HaWL increases the lifetime of flash memory more than SWL without additional memory overhead.

2.1 Bit-Set Threshold

We propose BST to solve the hidden cold block problem. The key ideas of BST are as follows.

• The hidden cold block problem is occurred by the loss of cold block information as the size of BET is reduced.
• In BST, cold blocks can be identified by using invalid page counts without additional memory usage, whereas SWL cannot.
• BST saves a threshold value which is determined by invalid page counts to increase the accuracy of detecting cold block.
• The value of BST is calculated by proposed formulas, which use the deviation of invalid page counts for each group of blocks by erase operations, to consider the dynamic state of each group.

BST is calculated by the number of invalid pages of blocks related in one bit and prevents setting the related bit of a BET to 1 by the erasure of hot blocks when cold blocks exist in the blocks that are related to the same bit in the BET. If the erase operation of blocks occurs, BST is calculated with Eq. (1).

$$BST(f_{index}) = IPC(b_{index}) - AVG_IPC(f_{index})$$

(1)

$IPC$ means invalid page count. $f_{index}$ is a bit index of the BET. $b_{index}$ is an erasing block index. $AVG_IPC$ is the average number of invalid pages for the blocks related to the bit $f_{index}$. $f_{index}$ and $AVG_IPC$ are calculated with Eq. (2) to Eq. (4).

$$f_{index} = b_{index}/2^k$$

(2)

$$AVG_IPC(f_{index}) = \frac{1}{ValidBlockCount} \sum_{n=f_{index} \times 2^k}^{(f_{index}+1) \times 2^k-1} IPC(n)$$

(3)

$$AvgBlockCount = \begin{cases} 2^k & \text{initially} \\ \text{decrease } 1 & \text{if } n \text{ is a free block} \\ 1 \leq ValidBlockCount \leq 2^k & \text{end if} \end{cases}$$

(4)

In Eq. (1), $b_{index}$ means a victim block erased by garbage collection, and HaWL is based on garbage collection using a greedy algorithm (GA) because of the similarity with SWL. $IPC(b_{index})$ has the number of invalid pages of $b_{index}$ before the erased $b_{index}$. Therefore, $IPC(b_{index})$ is equal to the maximum number of invalid pages, and BST has a maximum deviation compared with the average invalid page count of blocks related to $f_{index}$. Totally, if the minimum invalid page count of blocks ($MIN_IPC$) is lower than BST, HaWL determines that cold blocks exist in the blocks related to $f_{index}$, and prevents setting the bit of $f_{index}$ to 1. The condition for bit setting of the BET is as shown in Eq. (5).

$$BET[f_{index}] = \begin{cases} \text{not changed} & \text{if } BST(f_{index}) > MIN_IPC(f_{index}) \\ 1 & \text{otherwise} \end{cases}$$

(5)

Algorithm 1 shows the update of the BET using BST. BST can be used in the update of the BET when Algorithm 1 is called by garbage collection in line 2. $e_{cnt}$ and $f_{cnt}$ are used for a wear leveling trigger, because HaWL adopts a wear leveling trigger and some parameters from SWL. When wear leveling is triggered, HaWL selects a victim block that has the minimum invalid page count in blocks related to a zero bit, and it migrates the victim block for wear leveling.

2.2 Block Allocation Policy

In flash translation layer (FTL) using page or block mapping algorithms, the units for logical-to-physical address mapping is pages [1]. A unit of cold data can be a page unit,
or a block unit as a cold page, or a cold block in a physical address view. Therefore, some cold pages can be migrated to a different block from a resident block in page mapping. Moreover, cold data and hot data are mixed by wear leveling and garbage collection. As a result, the average number of invalid pages of victim blocks is low, and thus garbage collection is frequently performed. Therefore, the number of block erases is increased and the lifetime of flash memory is shortened.

To solve this problem, in HaWL, a dedicated block is only used for wear leveling in block migration operations and can be used in page mapping. A role of the dedicated block is that migration pages shift to the dedicated block during wear leveling.

3. Performance Evaluation

We implemented a simulator for our experiments. Table 1 shows details of the experiments. In the experiments, the ratio of cold data to hot data is 7:3, and the update of the files used a normal distribution. The lifetime of flash memory is defined as the first time to occur any bad block. Besides, time units of CAT for the measurement of elapsed time table (ETT) are based on the number of sequential writes, which can be used as relative time frame in experiments.

HaWL was compared with several previously used wear leveling techniques; GA, CB, CAT, and SWL, as shown in Fig. 2. pFTL means FTL using page mapping and NAND flash translation layer (NFTL) is based on a block mapping algorithm [1].

In Fig. 2 (a) and 2 (b), the standard deviation of HaWL is reduced by up to 81%, compared with other wear leveling techniques. Moreover, in comparison of flash memory lifetime, HaWL prolonged the lifetime of flash memory by up to 49%, as seen in Fig. 2 (c) and 2 (d). In addition, the lifetime of HaWL in NFTL was not reduced, even though $k$ is times, HaWL prolonged the lifetime of flash memory by up to 49%, as seen in Fig. 2 (c) and 2 (d). In addition, the lifetime of HaWL in NFTL was not reduced, even though $k$ is

### Table 1: Details of experiments

| Items                  | Values and Descriptions |
|------------------------|-------------------------|
| block count            | 2048                    |
| page count per block   | 128                     |
| page size              | 4KB                     |
| each file size         | 222 pages (888KB)       |
| total file count       | 1000                    |
| garbage collection trigger | $e_{cnt}/f_{cnt} < 10$  |
| wear leveling trigger  | $w_{cnt} < 1000$        |
| initial free space     | 15%                     |
| bad block              | erase operation count > 1000 times. |

### Table 2: The memory allocation of wear leveling tables and address mapping tables

| Classification     | Types               | Formulas for allocation size |
|--------------------|---------------------|-----------------------------|
| wear leveling      | GA                  | ICT×BC                      |
|                    | CB                  | (ICT + ETT)×BC              |
|                    | CAT                 | ICT×BTC                    |
|                    | SWL                 | ICT×BTC + BC/2              |
|                    | HaWL                | ICT×BTC + BC/4              |
| mapping algorithms | pFTL                | BC×PC×ES, ES = 18 bits††     |
|                    | NFTL                | BC×ES, ES = 11 bits††        |

ICT†: invalid count table (7 bits), BC: block count, ETT: elapsed time table (32 bits), ECT††: erase count table (10 bits), PC: page count per block, ES: entry size of address mapping table.

†An entry of address mapping table is represented by a range of address unit for mapping algorithms. Therefore, in pFTL, the size of an entry is 18 bits (BC×PC), so is NFTL 11 bits (BC).

††The number of ICT entry is 7 bits, because page count per block is 128 (2^7) at maximum.

†††The number of ECT entry is 10 bits, because the limited number of erase operations per block is 1000 ($< 2^{10}$) at maximum.
Table 3  The allocated memory size for each technique in experiments (unit: KB)

| Techniques | pFTL | NFTL |
|------------|------|------|
|            | AMT  | WLT  | Total | AMT  | WLT  | Total |
| GA         | 576  | 2.8  | 578.8 | 1.8  | 4.6  |
| CB         | 576  | 9.8  | 585.8 | 2.8  | 12.6 |
| CAT        | 576  | 12.3 | 588.3 | 2.8  | 15.1 |
| SWL or HaWL (k=0) | 576  | 2.3  | 578.3 | 2.8  | 5.1  |
| SWL or HaWL (k=1) | 576  | 2.8  | 578  | 2.8  | 5.6  |
| SWL or HaWL (k=2) | 576  | 1.9  | 577.9 | 2.8  | 4.7  |

AMT: address mapping table, WLT: wear leveling table increased. When HaWL is further compared with SWL on pFTL and NFTL in Fig. 2 (e), the lifetime of flash memory using HaWL was steadily higher than SWL. In particular, HaWL retained the balanced performance of wear leveling, although the size of the bit array table is reduced by the values of $k$.

Table 2 and Table 3 show the allocation of memory for each wear leveling technique, and Fig. 2 (f) shows the memory requirement for each wear leveling technique. In Fig. 2 (d), Fig. 2 (f), and Table 3, although the lifetime of CB and CAT is higher 14% on average than HaWL, the memory requirement of HaWL is reduced by up to 85%, compared with CB and CAT, and HaWL provided best lifetime increase for all cases with same hardware requirement, compared with SWL. In addition, HaWL in pFTL prolongs the lifetime without additional overhead compared with all other techniques.

4. Conclusion

In this letter, we proposed a novel wear leveling technique, we called Hidden cold block-aware Wear Leveling (HaWL) using a bit-set threshold (BST). HaWL uses a bit array table for wear leveling, and additionally, HaWL exploits a bit-set threshold to prevent the performance degradation of wear leveling in one-to-many modes. In addition, for FTL using page mapping algorithms, HaWL uses a dedicated block for wear leveling to increase the performance of wear leveling. In our experiments, HaWL increased the lifetime of flash memory by up to 49%, compared with previous wear leveling techniques.

References

[1] D. Ma, J. Feng, and G. Li, “A survey of address translation technologies for flash memories,” ACM Computing Surveys (CSUR), vol.46, no.3, pp.1–39, Jan. 2014.
[2] M.-C. Yang, Y.-M. Chang, C.-W. Tsao, P.-C. Huang, Y.-H. Chang, and T.-W. Kuo, “Garbage collection and wear leveling for flash memory: past and future,” 2014 International Conference on Smart Computing (SMARTCOMP), pp.66–73, Nov. 2014.
[3] A. Kawaguchi, S. Nishioka, and H. Motoda, “A flash memory based file system,” Proc. 1995 USENIX Technical Conference, pp.155–164, New Orleans, LA, USA, 1995.
[4] M.-L. Chiang and R.-C. Chang, “Cleaning policies in mobile computers using flash memory,” Journal of Systems and Software, vol.48, no.3, pp.213–231, 1999.
[5] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo, “Improving flash wear-leveling by proactively moving static data,” IEEE Transactions, Computers, vol.59, no.1, pp.53–65, Jan. 2010.