IEICE TRANS. INF. & SYST., VOL.E101–D, NO.10 OCTOBER 2018

LETTER

RbWL: Recency-Based Static Wear Leveling for Lifetime Extension and Overhead Reduction in NAND Flash Memory Systems*

Sang-Ho HWANG[†], ***Nonmember and*** **Jong Wook KWAK**[††a)], ***Member***

SUMMARY In this letter, we propose a static wear leveling technique, called Recency-based Wear Leveling (RbWL). The basic idea of RbWL is to execute static wear leveling at minimum levels, because the frequent migrations of cold data by static wear leveling cause significant overhead in a NAND flash memory system. RbWL adjusts the execution frequency according to a threshold value that reflects the lifetime difference of the hot/cold blocks and the total lifetime of the NAND flash memory system. The evaluation results show that RbWL improves the lifetime of NAND flash memory systems by 52%, and it also reduces the overhead of wear leveling from 8% to 42% and from 13% to 51%, in terms of the number of erase operations and the number of page migrations of valid pages, respectively, compared with other algorithms.

key words: *NAND flash memory, static wear leveling, elapsed time, cold/hot data classification*

1. Introduction

NAND flash memory, which has characteristics of non-volatility, low power consumption, and high density, is widely used in devices such as embedded systems and solid state drives (SSD). However, NAND flash memory systems cannot be directly used in existing file systems, owing to their unique characteristics, such as erase-before-write and program/erase (P/E) endurance. To overcome the characteristics of NAND flash memory systems, Flash Translation Layer (FTL) applies an out-place-update policy, which updates data to other pages and then invalidates original pages. The invalid pages generated in this procedure have to be reclaimed to free up space via block erase operations, called garbage collection. Afterward, remaining valid pages of blocks selected as victim blocks in garbage collection must be moved to free spaces. This is the main overhead of garbage collection. NAND flash memory also has a limited lifetime, and it is critical to manage the lifetime of NAND flash memory effectively.

Various dynamic wear leveling techniques based on garbage collection have been researched to date, such as Cost Benefit (CB) [1] and Migration Cost Sensitive Garbage Collection (MCS_{GC}) [2]. These wear leveling techniques have low overhead owing to the sufficient use of blocks that contain many invalid pages for wear leveling. However, dynamic wear leveling techniques have a problem related to garbage collection in that the area which mainly stores read only files, such as a kernel OS code or media files, is not considered in the wear leveling algorithms. Because the main purpose of garbage collection used in dynamic wear leveling algorithms is to reclaim invalid pages, the blocks that have no invalid pages are not selected as victim blocks. Figure 1 shows the distribution of erase counts in whole blocks when a dynamic wear leveling algorithm is used in NAND flash systems. To observe the distribution of the erase counts according to the ratio of cold data, we conducted an experiment using various ratios of cold data created by Gaussian distribution. As shown in Fig. 1, an environment which has more cold data has a larger variation of erase counts than others. Furthermore, particular blocks (with no invalid pages) are not involved in the dynamic wear leveling procedure.

To solve this problem, static wear leveling such as Block Erasing Table (BET) [3] and Adaptive Wear Leveling (AWL) [4] were proposed to perform the migration of cold blocks, selected by the flag of the block erasing table or the number of erase operations, to other free or the most worn-out blocks. Unfortunately, static wear leveling techniques suffer from many write overheads in prolonging the lifetime of NAND flash memory because they only consider

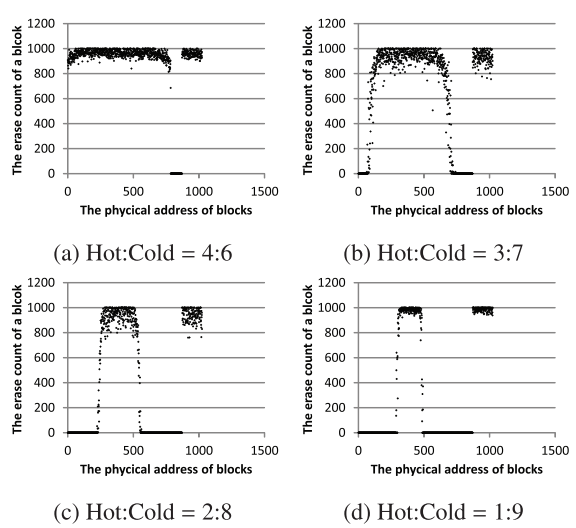


Fig. 1 The distribution of erase counts.

Manuscript received April 11, 2018.
Manuscript revised June 5, 2018.
Manuscript publicized July 9, 2018.

[†]The author is with the Wellness Convergence Research Center, DGIST (Daegu Gyeongbuk Institute of Science and Technology), Daegu, Korea.
[††]The author is with the Department of Computer Engineering, Yeungnam University, Gyeongsan, Korea.

*This work was supported by the 2018 Yeungnam University Research Grant.

a) E-mail: kwak@yu.ac.kr

DOI: 10.1587/transinf.2018EDL8076

Copyright © 2018 The Institute of Electronics, Information and Communication Engineers

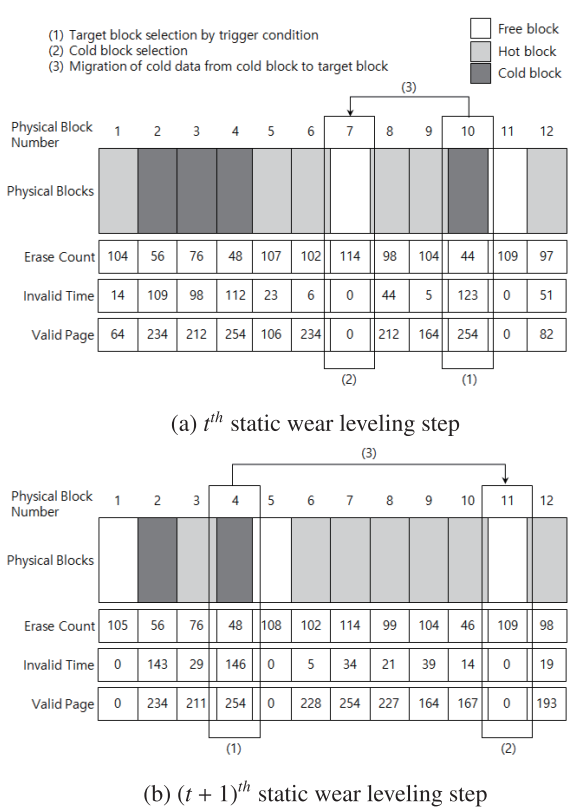
the balance of P/E in all blocks and neglect the cost of data migrations.

In this letter, we propose a static wear leveling technique, called Recency-based Wear Leveling (RbWL). The basic idea of RbWL is to execute static wear leveling at minimum levels, because the frequent migrations of cold data by static wear leveling cause excessive overhead in the NAND flash memory system. RbWL adjusts the execution frequency according to a threshold value that reflects the lifetime difference of the hot/cold blocks and the total lifetime of the NAND flash memory system. In this manner, RbWL further extends the lifetime compared to existing wear leveling techniques.

2. Recency-Based Wear Leveling (RbWL)

RbWL prevents blocks which have high P/E cycles from wearing out by migrating infrequently updated data (e.g., cold data). Figure 2 illustrates the main operations of RbWL. Erase count means the number of erases of the block, and invalid time is the elapsed time since the most recent invalidation of a page in the block. Valid page means the number of valid pages in the block.

In Fig. 2 (a), RbWL selects blocks 10 and 7 as the cold block and the target block, respectively. In this manner, block 10 with a high invalid time (123) becomes an empty block and block 7 with a high erase count (114) reduces the probability of further erase operations by storing cold data. In Fig. 2 (b), in the next wear leveling step, RbWL selects



(a) t^{th} static wear leveling step

(b) $(t+1)^{th}$ static wear leveling step

Fig. 2 Recency-based wear leveling (RbWL) senario.

blocks 4 and 11 as the cold block and the target block, respectively, as before. However, the previous technique selects a cold block based on erase count; thus, it would select block 10 with the minimum erase count (46) as a cold block. Because the data in block 10 have already been migrated to another block in the previous step, the data stored in block 10 are no longer considered cold data in RbWL, although it has the lowest erase count. Compared to the previous technique, RbWL uses recency time (RT) by exploiting invalid time, and it chooses block 4 with a high invalid time (146) as a cold block instead.

The frequent migrations of cold data by static wear leveling cause significant overhead in systems and have a negative impact on the lifetime of NAND flash systems. Therefore, it is important to execute static wear leveling at a minimum level. To do this, triggering RbWL is determined by Eq. (1):

$$\begin{aligned} TH &= EraseCount_{avg} \\ &+ (EraseCount_{Max} - EraseCount_{avg}) \times \frac{1}{\gamma} \\ \gamma &= EraseCountHB_{avg} - EraseCountCB_{avg} \\ &(if \quad \gamma < 1 \quad then \quad \gamma = 1) \end{aligned} \tag{1}$$

where $EraseCount_{avg}$ is the average erase count of all blocks and $EraseCount_{Max}$ is the maximum erase count for a set of given blocks. $EraseCountHB_{avg}$ is the average erase count of all hot blocks which satisfies $InvalidTime_i < InvalidTime_{avg}$. $InvalidTime_{avg}$ is the average elapsed time of page invalidations in all allocated blocks. $EraseCountCB_{avg}$ is the average erase count of all cold blocks which satisfies $InvalidTime_i >= InvalidTime_{avg}$.

If there are blocks that are worn out rapidly, cold data must be migrated to corresponding worn-out blocks before they become bad blocks. Migrated cold data are determined by Eq. (2):

$$\begin{aligned} value_i &= InvalidTime_i \times \frac{\#Validpage_i}{\#Totalpages_i} \\ InvalidTime_i &= Time_c - IT_i \end{aligned} \tag{2}$$

where $\#Validpage_i$ is the number of valid pages in block i and $\#Totalpages_i$ is the number of total pages in block i. $Time_c$ is the current time, and IT_i is the time of a page invalidation in block i. Therefore, $Time_c$ - IT_i is the invalid time of block i, as shown in Fig. 2. By this way, RbWL uses $value_i$ as recency information of blocks for selecting a victim block. RbWL has to consider the number of valid pages in selected cold blocks because of the possibility of selecting the block which has a few valid pages as a victim block in a subsequent garbage collection procedure. Algorithms 1 and 2 show the pseudo-codes of the RbWL algorithm.

RbWL can be used alone, but dynamic wear leveling can help to prolong the lifetime of NAND flash systems and reduce the overhead of data migrations and block erase operations. Figure 3 illustrates the overall procedure of RbWL

Algorithm 1: Recency-based wear leveling

```
input : block, ECT, ITT, VPT
/* ECT = Erase Count Table, ITT = Invalid Time
   Table, VPT = Valid Page Table                    */
/* block_pbn = physical block number for a
   corresponding block                              */
output: block
if ECT[block_pbn] > TH then
    coldBlock ← GetColdBlock(ITT, VPT)
    if exist coldBlock then
        migrate data from coldBlock to block
        Erase(coldBlock)
        block ← coldBlock
    end if
end if
return block
```

Algorithm 2: GetColdBlock

```
input : ITT, VPT
output: coldBlock
for n = 1 to N do
    if block_n is allocated then
        /* equation (2)                             */
        value ← evaluation(ITT[n],VPT[n])
        coldBlock ← find block_n with maximum value
    end if
end for
return coldBlock
```

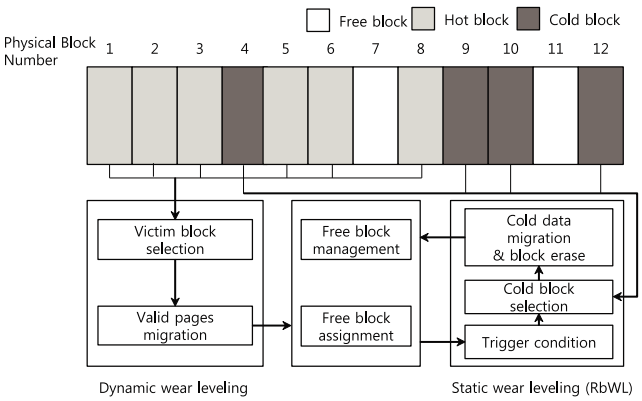


Fig. 3 The overall procedure of RbWL.

in NAND flash memory systems. As shown in Fig. 3, RbWL performs wear leveling in cold blocks as a static wear leveler. When one of the erase counts in free blocks reaches the threshold determined by Eq. (1), RbWL is triggered to balance the erase counts of cold blocks. Algorithm 3 shows the pseudo-code to trigger RbWL.

3. Performance Evaluation

To evaluate the performance of RbWL, we used the SSD extension for the Disksim simulator [5], which was developed by Microsoft. Table 1 shows the parameters of our simulation environment. For performance evaluation in various conditions, we used both financial trace files and a synthetically generated file. The financial trace files, named Financial1 and Financial2 [6], are traces from online transaction processing (OLTP) applications at two financial organizations. We also used FIU trace files including Homes, Online, and Webmail traces [7]. Note that these traces are widely used for performance measurements of flash-based storage systems. Furthermore, to assess the performance under high locality of references as a harsh condition, we used a synthetic trace, in which 90% of writes are performed on 10% of the data. For performance comparison, we evaluate RbWL compared with BET and AWL. Note that we used CB as a garbage collection policy to evaluate the efficiency of the static wear leveling in the environment where dynamic wear leveling works. The algorithm of CB exploits invalid time and invalid page ratio, and CB selects the block with the largest benefit/cost value as the victim block, where benefit = invalid time × invalid ratio and cost = 2 × (1 − invalid ratio) [1].

Figure 4 shows the normalized first failure time for write operations, which is directly related to the lifetime of

Algorithm 3: RbWL Call

```
input : ECT, ITT, VPT
/* call dynamic wear leveling such as MCS_GC        */
block_n ← VictimblockSelection()
ValidPageMigration(block_n)
for n = 1 to all victim blocks of dynamic wear leveling do
    Erase(block_n)
    FreeBlockPool(block_n)
    blk ← Get a free block which has the maximum erase count
    /* call static wear leveling                    */
    rtnBlk ← RbWL(blk, ECT, ITT, VPT)
    FreeBlockPool(rtnBlk)
end for
```

Table 1 Details of the experiment environment.

property	description
Chip size	2048 blocks
Block size	256 pages
Page size	4096 byte
Block erase time	5ms
Page write time	1500us
Page read time	211us
P/E cycles	1000
Over-provisioning ratio	15%
Garbage collection policy	Cost Benefit
Garbage collection threshold	5%
FTL scheme	page-level mapping

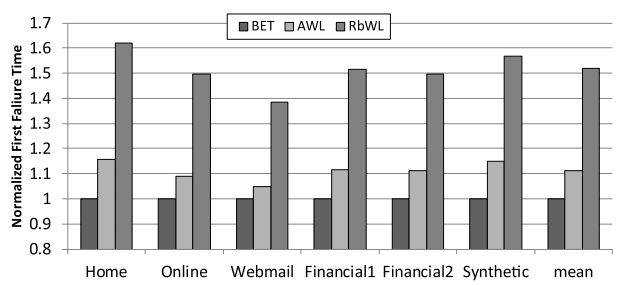


Fig. 4 Normalized first failure time for write operations.

NAND flash memory. As shown in Fig. 4, RbWL can prolong the lifetime to a greater extent than BET and AWL. RbWL reduces redundant migrations because it is based on the elapsed time of a page invalidation and the number of valid pages in a block, whereas previous techniques are based on the number of erasures when determining cold blocks to be migrated. On average, RbWL increased the lifetime of NAND flash memory by 52% and 31% compared with BET and AWL.

Figures 5, 6 and 7 show the overhead of each static wear leveling algorithm. Figure 5 shows the normalized erase operation ratio per write operation. Static wear leveling algorithms can cover whole blocks in the NAND flash system, which they have overhead for write operations and erase operations. As mentioned above, RbWL reduces redundant migrations, so it has lower erase overhead than other static wear leveling algorithms, resulting in flash memory lifetime enhancement. On average, RbWL reduced erase operations up to 8% and 42% compared with BET and AWL by preventing frequent and redundant cold block migrations.

Figure 6 shows the average number of page migrations for erased cold blocks in each wear leveling algorithm.

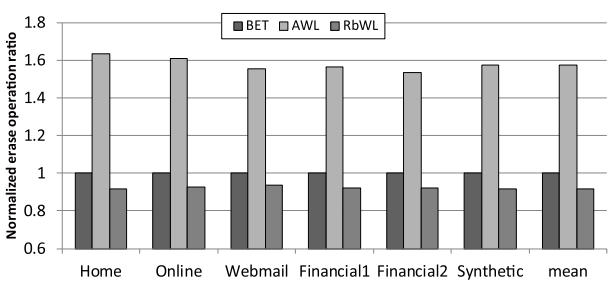


Fig. 5 Normalized erase operation ratio per write operation.

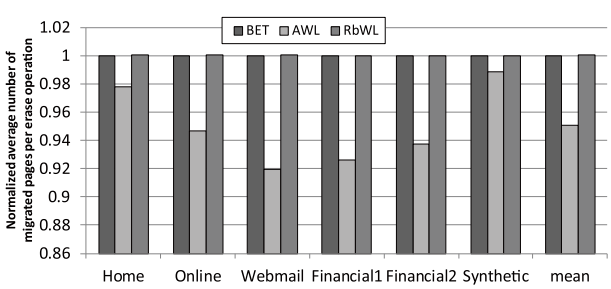


Fig. 6 Average number of migrations for erased block.

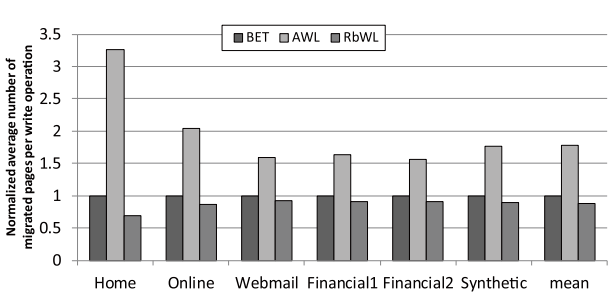


Fig. 7 Average number of migrations per write operation.

When the wear leveler selects the erase candidate blocks, it selects blocks with as many valid pages as possible among the cold blocks, which helps to prevent data from being selected and moved back to another location by garbage collection. Because RbWL considers the number of valid pages of a block when selecting a migration target block, the average number of page migrations for erased blocks is higher than other algorithms. Although BET exhibits similar ratios, AWL shows a 5.3% lower number of page migrations than RbWL, as it only considers the number of erase operations without considering the number of valid pages. In this manner, RbWL ensures that the worn block to which the cold data have been migrated are not erased by garbage collection or wear leveling. This overhead reduction ultimately provides the lifetime extension of NAND flash memory systems.

Figure 7 shows the average number of page migrations per write operation. Although RbWL has a large number of page migrations per block to be erased (as shown in Fig. 6), the overall cost is less than other algorithms because the number of blocks to be erased is low (as shown in Fig. 5). Therefore, as shown in Fig. 7, RbWL reduced the overhead of page migrations up to 13% and 51% compared with BET and AWL, respectively.

In our experiments using 2048 blocks, with 32-bit timestamps, the invalid time table (ITT) consumes 8 KB (4 B/block * 2048 blocks = 8 KB) for one NAND flash chip. However, in an environment where time-based garbage collection (such as CB or MCS_{GC}) is used as a base policy in FTL, there is no additional memory requirement, especially for RbWL, because the ITT and erase count table (ECT) used in RbWL are already embedded in FTL as part of the shared space of time-based garbage collection policies.

4. Conclusion

In this letter, we proposed Recency-based Wear Leveling (RbWL) to execute static wear leveling at minimum levels. RbWL adjusts the execution frequency according to a threshold value. In this manner, RbWL can solve an imbalance of erase counts between hot blocks and cold blocks and thus provide significant lifetime extension with low overhead. In experimental results, RbWL prolonged the lifetime of NAND flash systems from 31% to 52% compared with previous wear leveling algorithms. Our proposal also reduced the overhead of wear leveling from 8% to 42% and from 13% to 51% in terms of the number of erase operations and the number of page migrations of valid pages, respectively, compared with other algorithms.

References

[1] A. Kawaguchi, S. Nishioka, and H. Motoda, "A flash-memory based file system," USENIX, pp.155–164, 1995.

[2] S.-H. Hwang, J.H. Choi, and J.W. Kwak, "Migration cost sensitive garbage collection technique for non-volatile memory systems," IEICE Trans. Inf. & Syst., vol.E99-D, no.12, pp.3177–3180, Dec. 2016.

[3] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo, "Improving flash wear-leveling by proactively moving static data." IEEE Trans. Comput., vol.59, no.1, pp.53–65, 2010.

[4] J. Liao, F. Zhang, L. Li, and G. Xiao, "Adaptive wear-leveling in flash-based memory." IEEE Comput. Archit. Lett., vol.14, no.1, pp.1–4, Jan. 2014.

[5] V. Prabhakaran and T. Wobber, "SSD extension for DiskSim simulation environment," Microsoft Research, 2009.

[6] OLTP and Websearch Traces from UMass Trace Repository, http://traces.cs.umass.edu/index.php/Storage/Storage

[7] SNIA IOTTA Repository, http://iotta.snia.org/tracetypes/3