ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp.65 - 72
doi: 10.18178/wcse.2017.06.011

A Progressive Group-Based Wear-Leveling Algorithm for Solid State Drives

Haizhen He [1], Lei Wen [1]and Rongming Luo[1 +]

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, P.R.China

Abstract. Flash-based SSD have the characteristics of non-volatile, low energy consumption, strong shock resistance, etc. However, due to the inherent defects of flash memory, the blocks composed the flash have a limited number of erase cycles. Once reached its erasing limit, the data on the storage blocks will become unreliable. In order to improve the endurance of flash memory, this paper proposed a *progressive group-based wear-leveling algorithm* which can notably reduce the consumption of RAM space required by wear leveling algorithm while maintaining the wear leveling performance. Our algorithm also provides a new way to identify the cold data more accurately which dramatically minimize the performance overheads due to data migration. The evaluation results reveal that our *progressive group-based wear-leveling algorithm* consumes only 8% of memory space compared to the previous scheme that manages per-block information, while showing roughly the same wear-leveling performance.

Keywords: memory consumption, endurance, wear-leveling, performance overheads

1. Introduction

Flash memory is a non-volatile storage medium, with a small size, low power consumption and strong vibration resistance. As technology advances and cost performance increases, solid state disks(SSDs) made of flash memory chips are utilized in many personal computers and enterprise servers. Compared with traditional magnetic hard disk drive (HDD), SSDs have many different characteristics. The architecture of SSD consists of lots of physical blocks, each block containing a certain number of pages. A block is the smallest unit of an erase operation while reads and writes are done in a page. In addition, a page cannot be reprogrammed unless the block it is in has been erased. Therefore, data can only be updated out-of-place.

With the characteristics of flash memory in out-place updates, data that are to be updated must be written to another free pages of flash memory instead of overwriting, where the original page of the data is marked as invalid. A block that contains invalid pages is recycling by the garbage collection algorithm while free blocks are less than a certain amount in the whole flash memory. The out-place-update feature results in wear-leveling issue over flash memory because if the data stored in block hasn't been updated for a long time, then the block will not be erased by recycling process, its erased times will be significantly less than the block which has frequently updated data. Due to the inherent defects of flash memory, the blocks composed the flash have a limited number of erase cycles and then the data stored in it becomes unreliable. Typically it is 100,000 for single-level cell (SLC) flash that has one bit in a cell, and 10,000 for multi-level cell (MLC) flash whose cell can store two or more bits. When a block is worn out, thereby causing the use of the flash memory system may be adversely affected, such as the loss of stored data or the inability to store data.

In order to improve the endurance of flash memory, various excellent wear-leveling algorithms have been proposed, e.g, [1, 2, 3, 4, 5, 6, 7]. These algorithms can be roughly divided into two categories, dynamic and static wear leveling. The dynamic wear leveling methods, greedily allocate data to free blocks with fewer erase count to prevent over-using some blocks excessively; Although dynamic wear leveling does

[+]Corresponding author. Tel.: +86 15575996999.
E-mail address: jt_hehaizhen@hnu.edu.cn.

have great improvement on wear leveling, the endurance improvement is stringently constrained by its nature: That is, blocks of cold data (infrequently updated data) are likely to stay intact, regardless of how updates of hot data (frequently updated data) wear out other blocks. In other words, updates and recycling of blocks/pages will only happen to blocks that are free or occupied by hot data, where cold data are infrequently updated data. On the other hand, the conception of the static wear leveling is migrating the cold data to blocks which have large erase cycles to prevent any cold data from staying at any block for a long period of time so that distribute the wear evenly in the whole SSD. But these methods require to store the erase information of every block, which need a large RAM space, and with the increase of SSD capacity, the required RAM space increase proportionally.

In terms of reducing RAM space consumption, Yuan-Hao Chang [1] algorithm do the best, which keeps a record of the BET (block erase table) instead of storing the erase information of every block in RAM, each bit of BET indicates a group consist of 2k blocks whether has blocks been erased. Even though the algorithm does well in RAM space consumption, the problem that cannot recognize cold data result in the wear-leveling issue because cold data blocks with valid pages could not be recycled by the garbage operation. Jung [2] proposed a group based wear-leveling which also can reduce the required RAM space. Algorithm groups logically adjacent blocks into a single group and only the summary information for each group is maintained. Since the update block only swap data with the group with the lowest average erase count which results in severely uneven in other groups.

In this paper, we proposed a progressive group-based wear-leveling algorithm which solved the problem of unrecognized cold data in Yuan-Hao Chang algorithm and uneven wear in Jung algorithm. Our algorithm divides blocks into a series of groups and ensures evenness by distributing wear leveling within the group and between groups.

The rest of this paper is organized as follows. Section 2 shows our proposed wear-leveling algorithm. Section 3 presents the test environment, performance and overhead of the proposed algorithm. Finally, Section 4 makes the conclusions for this paper.

2. A Progressive Group-Based Wear-Leveling Algorithm

2.1. Unevenness level

The basic principle of static wear-leveling is to swap the hot data and cold data to prevent the cold data staying at any block for a long period of time. Static wear-leveling algorithms are usually based on a threshold to determine whether the hot and cold data exchange, which required a large RAM space to store the erase information for every block. To reduce the RAM space consumption, we divide the whole SSD into different groups and keep only the summary information for each group in memory [2]. The group summary information is used to detect uneven wearing and to find specific blocks and groups for wear leveling. Suppose that a group summary shows that the group is severely uneven when a block was recycled by the garbage collection algorithm. It means that the recycled block is much older than other blocks and there are cold data in this group. Then, the recycled block is selected as a specific block for wear leveling, and the cold data will be migrated from young block to the specific block.

Since the proposed algorithm only keeps the group summary information instead of maintaining wear information for every block. And the group summary information only denotes the overall wear status of blocks that belong to the group, so the effectiveness of wear leveling relies on the accuracy of the summary information. In order to minimize negative impact on wear leveling, the summary should be designed to represent the wear status of a group as closely as possible.

We maintain two variables, namely a mean E and a variance Var, for each group. The mean E denotes the average erase count for all blocks in a group, while the variance Var refers to the deviation between the mean E and the erase count of blocks that are owned by the group. Once SSDs controller is initialized, the corresponding E and Var should be obtained by reading the erase information of each block which stored in blocks's spare area [2]. When a block is erased, the statistics need to be recalculated.

$$E' = E + 1/N \tag{1}$$

$$Var' = Var + 2(ec - E)/N + 1/N - 1/N^2 \quad (2)$$

where *ec*, *E*, *Var* represent the erase cycle, mean and variance before erase operation occurs, respectively. *N* is the number of blocks within this group.

The variance *Var* shows the deviation between the erase count of each block and the average erase cycles *E* in a group, so that the erasures of blocks are severely uneven when *Var* is significantly greater than a certain threshold. Thus, *Var* can be used to represent the unevenness level within a group.

The mean *E* represents the average erase count of each block inside one group. Over time, those *E* of groups of blocks with cold data is significantly smaller than groups of blocks with hot data. Thus, the difference of *E* can be used to represent the unevenness level between different groups.

2.2. Wear-Leveling policy inside a group

When a block was recycled by the garbage collection, the corresponding *Var* need to be recalculated. And the wear leveler will be invoked when the uneven level *Var* inside one group greater than the threshold. A wear-leveling operation inside a group is shown in Figure 1. The wear leveler would find a block which stores cold data and has a low erase count as a victim block, and then migrate the cold data contained in the victim block into the previously erased free block. Finally erase the victim block.

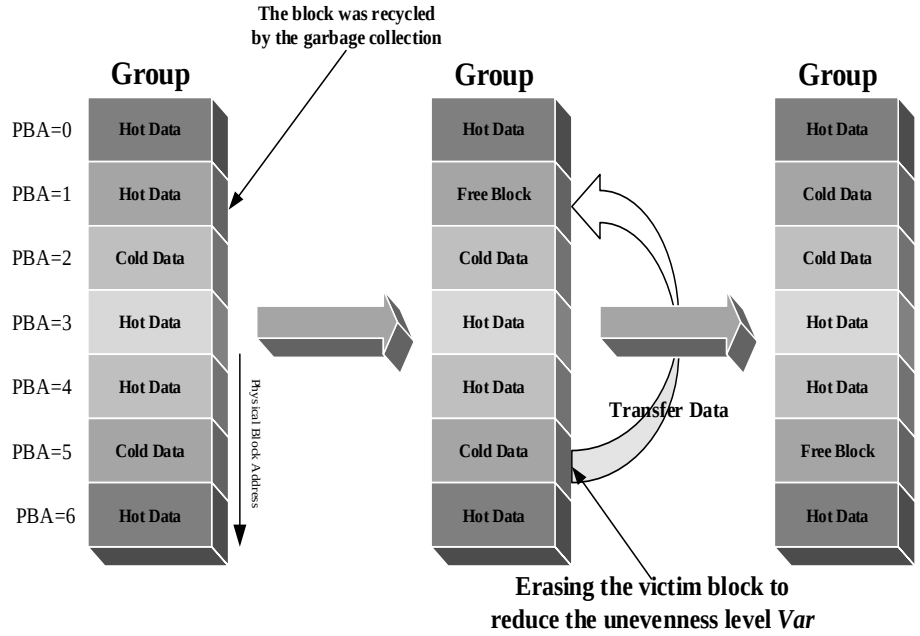

Fig. 1: Wear-Leveling Inside a Group

We can be clear from equation (2), the unevenness level *Var* uniquely associates with the erase frequency of the recycled block. In other words, when a block inside a group was erased, it's erased count uniquely determines the *Var* is increasing or decreasing. The wear leveler will be invoked when the uneven level *Var* of a group gradually increase than a certain threshold, which means the erase cycle of the recently erased block is much more than the average erase count of this group. Therefore, migrating cold data into that erased block is roughly equivalent to moving cold data into the excessively erased block. Furthermore, erasing the victim block can achieve the purpose which reduces the unevenness level *Var* since the erase frequency of the victim block is significantly less than the average erase count of this group.

In our proposed mechanism, the wear leveling operation is triggered by the erasing operation. And the higher the erase count of block selected to erase, the more easily lead to uneven erased. Accordingly, moving cold data into erased block can prevent the over erased block to erase again over a period of time. In the meantime, the victim block has lower erased frequency has the same write/erased probability with each free block in a group when it was erased. Therefore, after a period of time, the heat of hot block will gradually "cool down" and cold block will gradually "heat up", so as to achieve the purpose of wear leveling.

2.3. Wear-Leveling policy between groups

Although the use of wear leveling operation inside the group can achieve good evenness of block erases within the group, there are still some special situations may result in severely uneven between groups. Suppose that a group is completely filled with "cold" data. The average erase cycle of this group will be much smaller than the other groups after a period of time. Then, uneven wear will be happened.

By applying wear leveling operation inside the group, each block has the same wear status. Thus, we can use the average erase cycles E to express the wear condition of a group. Let E_{max} and E_{min} be the maximum and minimum average erase cycles of groups, respectively, and TH is a threshold decided by users. The group with E_{min} has the highest probability to contain cold data blocks, and the group with E_{max} has the highest probability to contain hot data blocks. We use the difference between E_{max} and E_{min} for representing unevenness level. The wear leveler will be invoked to exchange data in the groups with the maximum and minimum average erase cycles when E_{max}-E_{min}>TH is true. However, it is worth noting that take advantage of hot cold swap method may produce a large number of write amplification, so we should prevent wrong swapping as much as possible [4].

As be shown in Figure 2, the wear leveler exchanges data in the groups with maximum and minimum average erase cycle when E_{max}-E_{min}>TH is true, then after the wear leveling operation, the group with E_{max} still has the highest average erase cycle E_{max}+1. But this time the group is composed of cold blocks. When the next erase operation happens, E_{max}-E_{min}>TH is still true, and the wear leveler will work again to swap data in the groups with the maximum and minimum average erase cycles [4]. This condition causes the hot and cold blocks in the groups continuously exchange. Not only failed to achieve the purpose to prolong the life of SSDs, but also introducing a large number of additional erase operation which resulting in SSDs retirement in advance of its life.

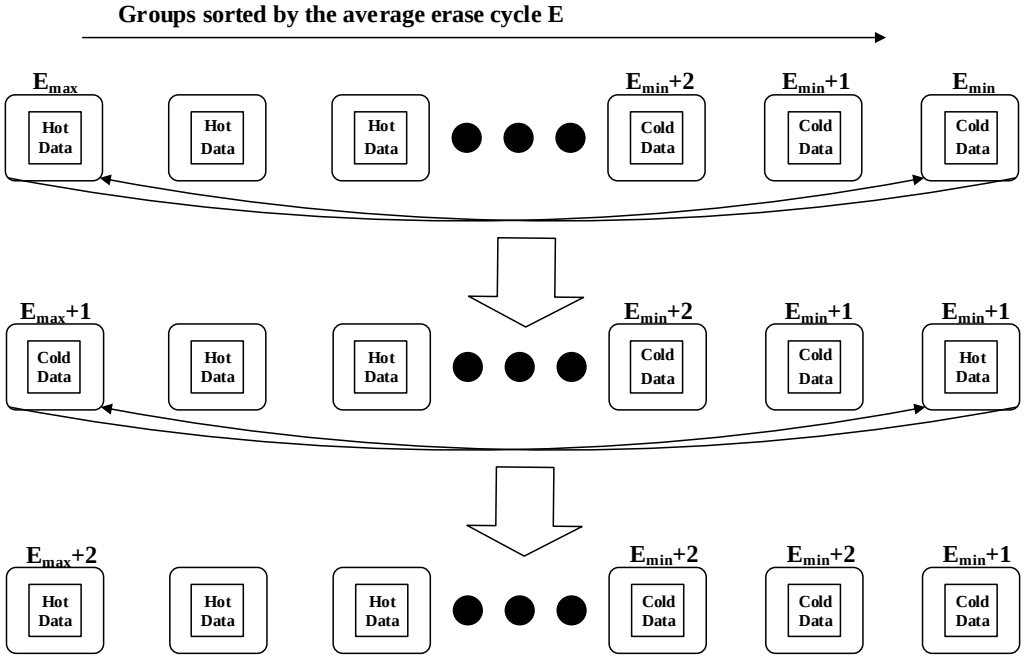

Fig. 2: Prevent false swapping

In order to avoid such meaningless data swapping, we determine whether the groups with hot and cold data blocks need to perform the exchanging operation is not only relying on the difference between E_{max} and E_{min}, but also makes full use of the distribution of hot and cold data within a group. We introduce hot cold ratio (*hcr*) to describe the proportion of the hot data blocks in a group. It aims to ensure that the groups are not swapping again in a period of time. Since the group with E_{max} has the highest probability to contain hot data blocks and the group with E_{min} has the highest probability to contain cold data blocks, the hot cold ratio of the group with E_{max} is obviously higher than the hot cold ratio of the group with E_{min}. If both E_{max}-E_{min}>TH and hcr_{max} * E_{max}-hcr_{min} * E_{min}> TH are true, then swapping the groups with E_{max} and E_{min}. When the exchanging operation is over, the hot cold ratio of the group with E_{max} will be lower, and the group with E_{min} will be higher, so it can prevent that groups are not swapping again in a period of time.

2.4. Cold data identification

To avoid moving hot data into the free block with higher erase count resulting in additional erase cycles, the cold data need to be accurately identified. In the proposed algorithm, when uneven wear occurs in the group, it means that the heat of data in each block is different. In other words, there are cold data blocks in the group. And the cold data are more likely to exist in the blocks that have the lowest erase cycles, or have never been erased in a recent period of time. Therefore, we maintain a Block Erasing Status Table (BEST) for each group to record whether a block is erased or not in a recent period of time. By combining BEST and the actual erase cycle of each block, we can easily and precisely locate the cold data blocks.

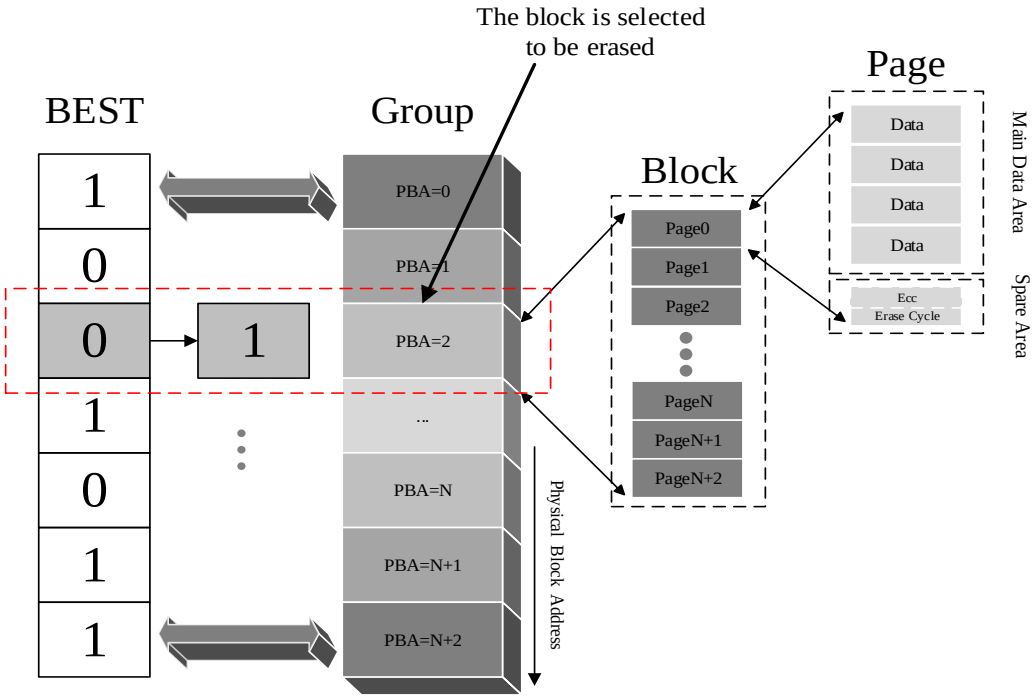

Fig. 3: Locating cold data

As be shown in Figure 3, BEST is a bit array, in which each bit corresponds to a physic block. Initially, the BEST is reset to have 0 for every bit. Whenever a block is selected to be erased by the cleaner, the corresponding bit will be set to 1. And when each bit of the BEST is set to 1, the BEST will be reset to initialization again.

According to the formula (2), if hot data is stored in the blocks with lower erase count, the unevenness level *Var* will gradually decrease in a period of time. Conversely, once the unevenness level sustained growth, it indicates that hot data exist in a over-erased blocks. In other words, the cold data store in the lower erase frequency blocks. However, we cannot simply rely on the erase cycles of each block to locate the cold block, but also need to make use of the corresponding state in BEST. Because a block with lower erase count also has a high probability to contain hot data when it was recycled by the wear leveler, and the erased state in BEST can avoid to identity those blocks as cold data blocks. Therefore, by making full use of BEST and the actual erase cycle of each block, we can easily and precisely find the cold data blocks.

3. Experimental Result

3.1. Simulation environment and workload setting

The purpose of this section is to evaluate the effectiveness of the proposed Progressive Group-based Wear-leveling algorithm. We established a SLC model [8] which is 2KB each page and 64 pages every block by using a trace driven simulator.

Four different traces are used for simulation. The first two traces, namely Financial1 and Financial2, were collected from an OLTP application hosted at a financial institution. These traces are made available courtesy of Ken Bates from HP, Bruce McNutt from IBM and the Storage Performance Council [9]. And the remaining traces are synthesized by the simulator.

We shall evaluate the effectiveness of PGW algorithm from two metrics. The typical performance metric for a wear leveling algorithm is the standard deviation that reflects the distribution of block erase. Another metric that is typically used to evaluate the performance of wear leveling is the average erase count that indicates the extra overhead due to data migrations.

To make a fair comparison we respectively implemented no wear-leveling, BET proposed by Yuan-Hao Chang [1], Dual-Pool algorithm proposed by Chang [7] and group based algorithm proposed by Jung [2].

3.2. Effectiveness of PGW

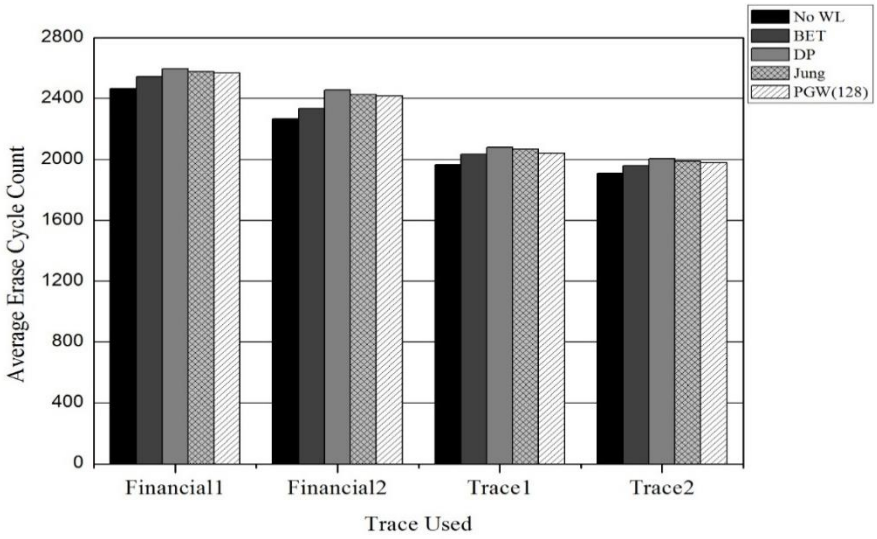

Fig 4: Average erase cycle count

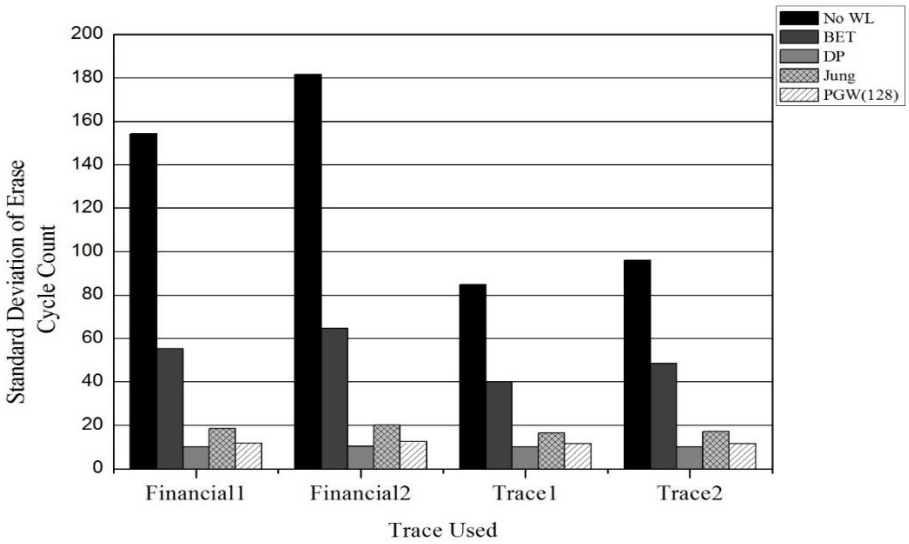

Fig. 5: Standard deviation of erase cycle count

Figure 4-5 show the overall effectiveness of five wear leveling algorithm under the different traces. According to the simulation results, DP has the best performance of wear leveling, but it also introduces a large number of extra overhead. The evenness performance of our proposed algorithm is very close to the DP, but the additional erase operation is less. This is because we migrate the cold data to the erased free block instead of swapping the hot and cold data in the blocks. Therefore, we will introduce fewer additional erase operation for each wear leveling operation.

3.3. The effect of group size

In the proposed algorithm, the group size is closely related to the extra overhead caused by positioning cold data because the standard deviation of erase cycle count can be artificially determined by users. The relationship between the cold data locating time and the group size is shown as Figure 6. As the group size

grows, the extra overhead increases because we need to traverse the erase count of each block in the group to pinpoint the cold data.

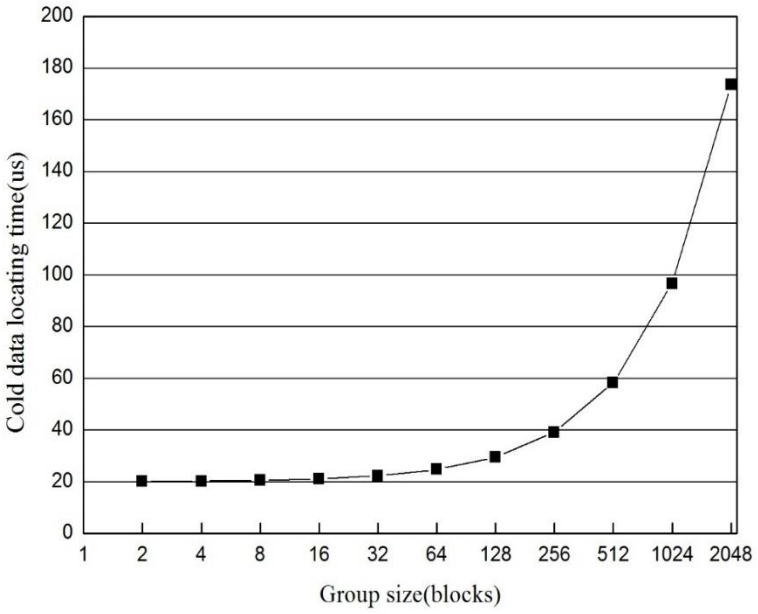

Fig. 6: Cold data locating time

3.4. Memory requirements

As show in Table 1, the memory requirements of the proposed algorithm are extremely low compared with traditional wear-leveling algorithms, who record each block's erase cycles across the lifetime. Assuming that the limit of erase cycle is about 100K, we need a 17-bit memory for each block at least. As to traditional algorithms, a mainly used 64GB SSD with 512K blocks would consume more than 1MB RAM space. The proposed algorithm is only storing the average erase count and block's erase variance of each group in memory instead of the erase cycle of each block. And the average erase count, block's erase variance respectively require 17-bit and 15-bit every group. In addition, the BEST that recording the block erase state is also occupies 1 bit per block. Therefore, when the capacity of SSD is 64 GB and the group size k=128, the total RAM consumption is only 80 KB, while DP algorithm consumes more than 4MB. Although the BET algorithm and the Jung algorithm also have a low RAM consumption, but it has a poor wear-leveling performance as shown in experiment results.

Table 1: The consumption of memory

Algorithm	BET(k=2)	DP	Jung(k=128)	This paper
Memory consumption(KB)	16	12800	33	80

4. Conclusion

This paper proposes a progressive group-based wear leveling algorithm. By calculating the variance of block's erase cycles inside a group and employing the Block Erasing Status Table to obtain the distribution of hot and cold data instead of recording the erase cycles of all blocks in memory. Therefore, the RAM consumption to realize the wear-leveling are extremely low compared with other wear leveling algorithms. By combining the BEST and block's erasure number can identify the cold data more accurately. The evaluation results show that the proposed algorithm can notably reduce the consumption of RAM space required by wear-leveling algorithm while maintaining the wear leveling performance.

5. Acknowledgements

This work is supported by the Hunan Province natural science foundation of China [NO.14JJ7026] and the Open Fund Project of innovation platform in Hunan Universities [No.13K015], and the authors would

like to thank the editors and anonymous reviewers for providing valuable comments which helped in improving the manuscript.

6. References

[1] Chang, Y. H., Hsieh, J. W., & Kuo, T. W. (2007, June). Endurance enhancement of flash-memory storage systems: an efficient static wear leveling design. *Proc. of the 44th annual Design Automation Conference.* 2007, pp. 212-217.

[2] D. Jung, Y. H. Chae ,H. Jo, J. S. Kim, and J. Lee. A group-based wear-leveling algorithm for large-capacity flash memory storage systems. *Proc. of the 2007 International Conference on Compilers, architecture, and synthesis for embedded systems.* 2007, pp. 160-164.

[3] A. Ban. Wear Leveling of Static Areas in Flash Memory. *United States Patent, 6,732,221*. M-systems, 2004.

[4] Hsieh W K, Ma H P. Conditional threshold wear-leveling algorithm for multi-channel NAND flash memory. *Proc. of 2010 International Symposium on VLSI Design Automation and Test (VLSI-DAT). IEEE*. 2010, pp. 147-150.

[5] Liu H, Nie H, Liu Q, et al. A Group-Based Hybrid Wear-Leveling Algorithm for Flash Memory Storage Systems. *Proc. of 2012 Third International Conference on Digital Manufacturing and Automation (ICDMA). IEEE*. 2012, pp. 58-61.

[6] Li-Pin Chang and T. W. Kuo. An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded systems. *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium.* 2002, pp. 187-196.

[7] Li-Pin Chang. On efficient wear leveling for large-scale flash-memory storage systems. *Proc. of the 2007 ACM Symposium on Applied Computing*. 2007, pp.1126-1130.

[8] Hu Y, Moore D. MLC vs. SLC NAND flash in embedded systems[J]. *Technical report*. 2009.

[9] "University of Massachusetts Amhesrst Storage Traces," *http://traces.cs.umass.edu/index.php/Storage/Storage.*