EF-Greedy: A Novel Garbage Collection Policy for Flash Memory Based Embedded Systems

Ohhoon Kwon, Jaewoo Lee, and Kern Koh

School of Computer Science and Engineering, Seoul National University
{ohkwon, jwlee, kernkoh}@oslab.snu.ac.kr

Abstract. Flash memory is becoming increasingly important for embedded systems because it has many attractive features such as small size, fast access speeds, shock resistance, and light weight. Although flash memory has attractive features, it should perform garbage collection, which includes erase operations. The erase operations are very slow, and usually decrease the performance of the system. Besides, the number of the erase operations allowed to each block is also limited. To minimize the garbage collection time and evenly wear out, our proposed garbage collection policy focuses on minimizing the garbage collection time and wear-leveling. Trace-driven simulations show that the proposed policy performs better than existing garbage collection policies in terms of the garbage collection time and the endurance of flash memory. Specifically, we have shown that the performance improvement of our proposed policy against the greedy policy in terms of the endurance of flash memory is as much as 90.6%.

Keywords: Flash memory, Garbage collection, Embedded systems.

1 Introduction

Flash memory is a non-volatile solid state memory, which is becoming increasingly important for digital computing devices because it has many attractive features such as small size, fast access speeds, shock resistance, high reliability, and light weight. Because of these features, flash memory will be widely used in various computing systems such as embedded systems, mobile computers, and consumer electronics.

Although flash memory has attractive features, it has a critical drawback, which is an inefficiency of in-place-update operation. When we update data, we can not write new data directly at same address due to physical characteristics of flash memory. All data in the block must first be copied to a system buffer and then updated. Then, after the block has been erased, all data must be written back from system buffer to the block. Therefore, updating even one byte data requires one slow erase and several write operations. Besides, if the block is a hot spot, it will soon be worn out.

To address the problem of the in-place-update operation, the out-place-update operation is exploited in many flash memory based systems [6-8]. When the data is updated, the out-place-update operation writes new data at new place, and then the obsolete data are left as garbage. When there are not enough free spaces in flash memory, we should collect the garbage space and translate a free space. This operation

Table 1. Performance of NAND flash memory [11]

	Page Read (2K bytes)	Page Write (2K bytes)	Block Erase (128K bytes)
Performance (μ s)	25(Max.)	200(Typ.)	2000(Typ.)

is a garbage collection, which consists of the write operations and the erase operations. In the flash memory, the erase operations are very slow, and usually decrease the performance of the system. Besides, the number of the erase operations allowed to each block is also limited. In order to minimize the garbage collection time and evenly wear out flash memory, we propose a novel garbage collection policy in this paper. Our proposed garbage collection policy focuses on minimizing the garbage collection time as well as reducing the number of the erase operations and wear-leveling. Trace-driven simulations with HP traces show that our proposed policy performs better than the greedy, the Cost-Benefit (CB), and the Cost Age Time (CAT) policies in terms of the garbage collection time, the number of erase operations, and the endurance of flash memory.

The remainder of this paper is organized as follows. We review characteristics of flash memory and existing works on garbage collection in Section 2. Section 3 presents a new garbage collection policy for flash memory. We evaluate the performance of the proposed policy in Section 4. Finally, we conclude this paper in Section 5.

2 Related Works

In this section, we describe characteristics of flash memory and existing works on garbage collection.

2.1 Characteristics of Flash Memory

Flash memory is a non-volatile solid state memory, which has many attractive features such as small size, fast access speeds, shock resistance, high reliability, and light weight. Furthermore, its density and I/O performance have improved to a level at which it can be used as an auxiliary storage for mobile computing devices such as PDA and laptop computer. Flash memory is partitioned into blocks and each block has a fixed number of pages. Unlike hard disks, flash memory has three kinds of operations: page read, page write, and block erase operations. They have difference performances, and the performances of three kinds of operations are summarized in Table 1.

As aforementioned, flash memory has many attractive features. Flash memory, however, has two drawbacks. First, blocks of flash memory need to be erased before they are rewritten. The erase operation needs more time than read or write operation. The second drawback is that the number of erase operations allowed to each block is limited. This drawback becomes an obstacle to developing a reliable flash memory-based embedded system. Due to this drawback, the flash memory based embedded systems are required to wear down all blocks as evenly as possible, which is called wear-leveling.

2.2 Existing Works on Garbage Collection

Rosenblum et al. proposed the Log-Structured File System (LFS) and garbage collection policies have long been discussed in log-based disk storage systems [1-4]. Fortunately, the Log-Structured File System can be applied to flash memory based storage systems and the garbage collection policies in log-based disk storage also can be applied to flash memory based storage systems. Wu et al. proposed the greedy policy for garbage collection [5]. The greedy policy considers only valid data pages in blocks to reduce write cost and chooses the block with the least utilization. However it does not consider wear-leveling. Therefore, it was shown to perform well for random localities of reference, but it was shown to perform poorly for high localities of reference. Kawaguchi et al. proposed the cost-benefit policy [6]. The cost-benefit policy evaluates the cost benefit of all blocks in flash memory using $((a*(1-u))/2u)$ method, where a is the elapsed time from the last data invalidation on the block, and u is the percentage of fullness of the block. After evaluating the all blocks, it chooses the victim block that has a maximum cost benefit value. Chiang et al. proposed the Cost Age Time (CAT) policy [7]. The CAT policy focuses on reducing the number of the erase operation. To reduce the number of the erase operations, they use a data redistribution method that uses a fine-grained method to separate cold and hot data. The method is similar to the cost-benefit policy but operates at the granularity of pages. Furthermore, the CAT policy considers wear-leveling. To perform even-leveling, the CAT chooses the victim block according to cleaning cost, ages of data in blocks, and the number of the erase operations. Kim et al. proposed the cleaning cost policy, which focuses on lowering cleaning cost and evenly utilizing flash memory blocks. In this policy, they dynamically separates cold data and hot data and periodically move valid data among blocks so that blocks have more even life times [9]. Chang et al. proposed the real-time garbage collection policy, which provides a guaranteed performance for hard real-time systems [10]. They also resolved the endurance problem by the wear-leveling method.

3 EF-Greedy: A Novel Garbage Collection Policy

In flash memory, the erase operation is even slower than the read and write operation. Thus, the erase operation is dominant to the performance of the flash memory based system. As mentioned in Section 2, to improve the performance, existing works for garbage collection tried to reduce the number of the erase operations. They also considered the wear-leveling for the endurance of flash memory.

In this paper, we propose the new garbage collection policy, which extends the greedy policy and focuses on minimizing the garbage collection time as well as wear-leveling. Thus, our proposed garbage collection policy is named ‘EF-Greedy’, which stands for ‘*Endurant and Fast Greedy*’.

3.1 Block Recycling Scheme

In flash memory, if there are not enough free blocks, the system should perform garbage collection. During garbage collection, we should wait and do not perform any operations such as read and write operations until the garbage collection finishes. To improve the performance of the flash memory based system, we should minimize the

garbage collection time. In this paper, we use the greedy policy to make a decision which block should be erased during garbage collection. Because the greedy policy considers only valid pages in blocks and chooses the block with the least utilization, we can minimize the garbage collection time. However it does not consider wear-leveling and was shown to perform poorly for high localities of reference. To address the problems of the greedy policy, we extend the greedy policy by considering the different update interval of the pages in the blocks and the number of the erase operation of the blocks.

Fig. 1. The classification of the pages and the redistribution of the valid pages

When we perform garbage collection, we select several victim blocks with the least utilization, and then copy valid pages in the victim blocks to the free block before we clean the block. For the redistribution of valid pages, we should classify data pages as hot and cold pages. Hot valid pages are updated frequently at recent time, and cold valid pages are not updated frequently. During redistributing, hot valid pages are redistributed to hot blocks, while cold valid pages are redistributed to cold blocks. To classify data pages, we estimate the update interval of each page based on past update behavior, and then use this update interval information in deciding whether the page is hot or cold in the EF-Greedy policy. Expression (1) represents the calculation of the *predicted inter-update time (PIU)* based on past update times. The *predicted inter-update time* denotes the interval between future update time and latest update time on an identical page. Let I_k be the (k) th inter-update time, I_{k-1} be the $(k-1)$ th inter-update time, and I_{k-2} be the $(k-2)$ th inter-update time. Then, the k th predicted inter-update time PIU_k is computed as

$$PIU_k = \frac{\sum_{j=k-n+1}^k I_j}{n} \quad (1)$$

where n is the number of the past inter-update time. We, hence, just consider the last inter-update time if n is 1, and consider the whole inter-update times if n is k . EF-Greedy sets the default values of n as 3. If the *PIU* value of the page is greater than the average *PIU* value of all pages, the page is classified the cold page, otherwise the page is classified the hot page. Fig. 1 shows the classification of the pages and the redistribution of the valid pages.

3.2 Efficient Free Block Lists Management Scheme

Flash memory should be controlled to evenly wear out all blocks since wearing out specific blocks could limit the usefulness of the whole flash memory. Thus, most of the existing works considered wear-leveling of flash memory when the victim block is selected. In contrast, our proposed policy does not consider wear-leveling similar to the greedy policy when the victim block is selected. In order to guarantee the long endurance of flash memory, we propose an efficient free block lists management scheme for wear-leveling on flash memory. In our proposed policy, we use two free lists: hot free block list and cold free block list. After cleaning the victim blocks, if the number of the erase operation of the block is greater than the average number of the

Fig. 2. The efficient two free block lists management scheme for wear-leveling

erase operation of all free blocks, the block is added to the cold free block list. Otherwise, the block is added to the hot free block list. And then the free blocks in each free block list are sorted by the number of the erase operation of the block. Hence, during copying out, we could allocate the block with the minimum number of the erase operations to valid pages, and could evenly wear out. Fig. 2 shows the efficient free block lists management scheme for wear-leveling.

4 Performance Evaluation

In this section, we present the performance evaluation results for various garbage collection policies to assess the effectiveness of our proposed policy. We conducted trace-driven simulations with the HP traces to compare the performance of our proposed policy with those of the greedy, the Cost-benefit (CB), and the Cost Age Time (CAT) policies.

The HP traces are disk-level traces of personal workstation (hplajw) at Hewlett Packard laboratories, which were used for document editing and electronic mail. They also exhibit high locality of reference that 71.2% of writes were to metadata [12]. Since the size of the disks used in hplajw does not equal to the size of the flash memory used in simulations, the traces were preprocessed to map flash memory spaces before simulation. To evaluate the performance, when the size of free block is fewer than 5% of the total size of flash memory, garbage collection is started. And garbage collection is stopped when the size of free block is larger than 10% of the total size of flash memory. Fig. 3 shows the garbage collection time for the four policies as a function of the flash memory size. Our proposed policy shows better performance in terms of the average and maximum garbage collection time. This is because the EF-Greedy policy just considers the utilization of each block to minimize the garbage collection time unlike other three policies. Furthermore, our proposed policy performs better than the original greedy policy because it estimates the predicted inter-update time (PIU) of each page and exploits the PIU value to redistribute pages.

Fig. 4 shows the performance results of the number of erase operation and worn-out blocks. In these results, our proposed policy shows the best performance in terms of the number of worn-out blocks due to the efficient free block lists management scheme. This result means that our proposed policy guarantees the long endurance of flash memory. Specifically, the performance improvement of EF-Greedy against the greedy policy is as much as 90.6% in terms of the endurance of flash memory. The result of the number of erase operation also shows better performance in our proposed policy.

Fig. 3. The performance results of the garbage collection time

Fig. 4. The performance results of the number of erase operation and worn-out blocks

5 Conclusion

In this paper, we presented the novel garbage collection policy for flash memory based embedded systems. To minimize the garbage collection time and guarantee the endurance of flash memory, we extended the greedy by considering the different update interval of the pages and the efficient free block lists management scheme. As a result, EF-Greedy performs better than other existing garbage collection policies in terms of the garbage collection time, the number of erase operations, and the endurance of flash memory. Specifically, we have shown that the performance improvement of EF-Greedy policy against the greedy policy in terms of the endurance of flash memory is as much as 90.6%.

Acknowledgments. The authors thank Hewlett Packard Laboratories for making their I/O traces available.

References

1. Rosenblum, M., Ousterhout, J. K.: The Design and Implementation of a Log-Structured FileSystem. *ACM Transactions on Computer Systems*, Vol. 10, No. 1 (1992)
2. Blackwell, T., Harris, J., Seltzer, M.: Heuristic Cleaning Algorithms in Log-Structured File Systems. *Proceedings of the 1995 USENIX Technical Conference*, Jan. (1995)
3. Matthews, J. N., Roselli, D., Costello, A. M., Wang, R. Y., Anderson, T. E.: Improving the Performance of Log-Structured File Systems with Adaptive Methods. *Proceedings of the Sixteenth ACM Symposium on Operating System Principles* (1997)
4. Seltzer, M., Bostic, K., McKusick, M. K., Staelin, C.: An Implementation of a Log-Structured File System for UNIX. *Proceedings of the 1993 Winter USENIX* (1993).
5. Wu, M., Zwaenepoel, W.: eNVy: A Non-Volatile, Main Memory Storage System. *Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems* (1994)

6. Kawaguchi, A., Nishioka, S., and Motoda, H.: A Flash-Memory Based File System. Proceedings of USENIX Technical Conference (1995)
7. Mei-Ling Chiang, Paul C. H. Lee, Ruei-Chuan Chang: Cleaning policies in mobile computers using flash memory. *Journal of Systems and Software*, Vol. 48, (1999)
8. Torelli, P.: The Microsoft Flash File System. *Dr. Dobbs's Journal*, Feb. (1995)
9. Hanjoon Kim, Sanggoo Lee, S. G.: A new flash memory management for flash storage system. Proceedings of the Computer Software and Applications Conference (1999)
10. Li-Pin Chang, Tei-Wei Kuo, Shi-Wu Lo: Real-time garbage collection for flash-memory storage systems of real-time embedded systems. *ACM Transactions on Embedded Computing Systems*, Vol. 3. (2004)
11. Samsung Electronics: 128M x 8 Bit NAND Flash Memory. <http://www.samsung.com>
12. Ruemmler, C., Wilkes, J.: UNIX Disk Access Patterns. Proceedings of the 1993 Winter USENIX (1993)