Swap-aware Garbage Collection for NAND Flash Memory Based Embedded Systems

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

Abstract

Embedded systems use NAND flash memory as a secondary storage device because it has many attractive features such as small size, fast access speeds, shock resistance, and light weight. NAND flash memory based embedded systems exploit a "demand paging" to run applications and also use a "swapping" to extend a limited main memory space. Because the embedded systems use NAND flash memory as swap space, it should perform garbage collection, which is a time-consuming operation. Besides, the number of the erase operations allowed to each block is also limited. In this paper, we propose a new garbage collection policy for embedded systems with the swap system. The proposed garbage collection policy focuses on minimizing the garbage collection time and even 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.

1. Introduction

In recent years, flash memory has become an important storage device in embedded systems because it has many attractive features such as small size, shock resistance, high reliability, low power consumption, and lightweight.

A typical embedded system such as cellular phone and MP3 player contains DRAM, NOR flash and NAND flash memory. DRAM is used for a main memory, NOR flash memory is used for a program code, and NAND flash memory is used for user data. Because the typical embedded systems contain three kinds of memory, it is difficult to cut down the cost of hardware and reduce the size of embedded systems. In order to reduce the cost and size, it has been attempted to eliminate NOR flash memory from embedded systems. If the embedded system does not contain NOR flash memory, the application program code needs to be copied from NAND flash memory to the main memory during running the application. This processing mechanism is called "shadowing." The shadowing shows the best performance at runtime because the whole program codes reside in the main memory. However this mechanism needs a longer loading time since the program code should be copied to the main memory. Besides, the main memory should become large because the application codes become large in recent years. To address the weakness of the shadowing, "demand paging" is exploited for NAND flash memory based embedded systems. Demand paging is a virtual memory technique that code or data is loaded from the secondary storage only when needed by a process. Thus, it requires a less main memory capacity and a shorter loading time. Furthermore, the embedded systems using demand paging exploit a "swapping" mechanism to extend a limited main memory space. Since the swapping frequently performs write and read operations to NAND flash memory, there are many invalid pages in NAND flash memory and thus the embedded system should often perform the garbage collection operation to translate invalid pages to free pages. In NAND flash memory, the erase operation cost is higher than the read and write operation according to table 1, and the number of erase operation is limited to about 100,000 times. Because of these reasons, we should develop an

Table 1. Performance of flash memory [11]

	Page Read (2KB)	Page Write (2KB)	Block Erase (128KB)
Performance (μs)	25	200	2000

0-7695-2983-6/07 $25.00 © 2007 IEEE
DOI 10.1109/CIT.2007.76

efficient garbage collection policy for NAND flash memory based swap system. This paper proposes a new garbage collection policy for NAND flash memory based swap system. The proposed garbage collection policy focuses on minimizing the garbage collection time and extending a lifetime of NAND flash memory to improve a performance of embedded systems.

Trace-driven simulations show that the 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. Section 2 reviews the physical characteristics of flash memory and then analyzes existing works on garbage collection. Section 3 presents a new garbage collection policy for NAND flash memory based swap system. Then, performance evaluation results of the proposed policy are given in Section 4. Finally, Section 5 concludes the paper.

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 whose density and I/O performance have improved to a level at which it can be used as an auxiliary storage for embedded systems. 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. The performance and energy consumption of the three kinds of operations are summarized in Table I. As shown in the table, each operation needs significantly different time and energy consumption. With these characteristics, flash memory has two drawbacks related to I/O operations. First, blocks of flash memory need to be erased before they are rewritten. An erase operation needs more time and energy than a read or a write operation. The second drawback is that the number of erase operations allowed to each block is limited. This drawback becomes an obstacle to develop a reliable flash memory based storage systems. To relieve this problem, the software layer of a flash memory device usually contains the wear-leveling mechanism that controls the erase count of all flash blocks as evenly as possible.

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 dose 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. Swap-aware Garbage Collection

In this paper, we propose the new garbage collection policy, which extends the Greedy policy for NAND flash memory based swap system. Thus, our proposed garbage collection policy is named 'S-Greedy'. In flash memory, an erase operation is even

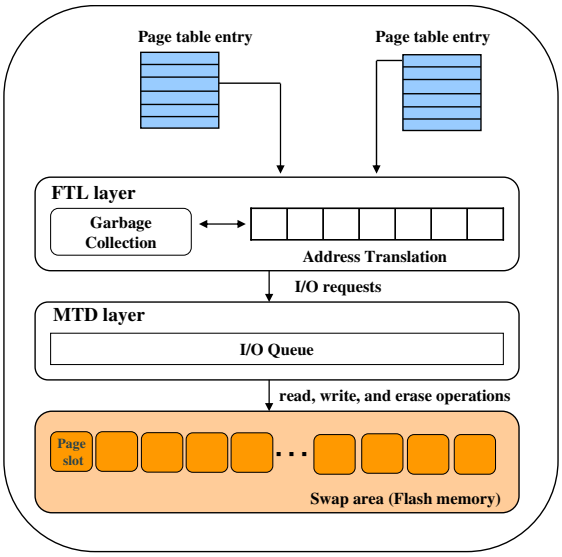

Figure 1. The architecture of NAND flash memory based swap system

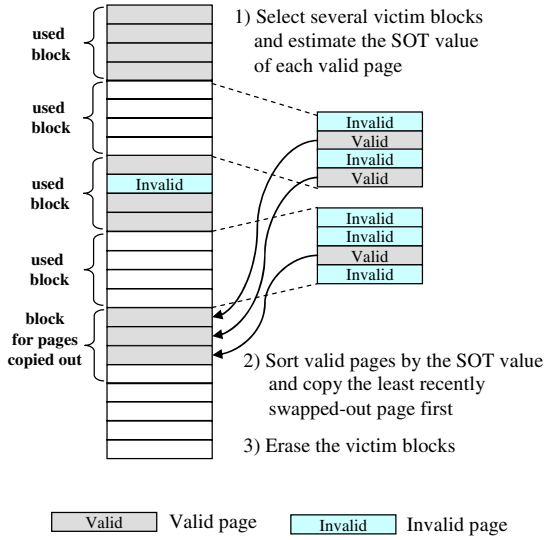

Figure 2. The redistribution of the valid pages

slower than a read and write operation. Thus, the erase operation is dominant to the performance of the NAND flash memory based swap system. In order to improve the performance, the proposed garbage collection policy tries to reduce the number of the erase operations and also considers the wear-leveling for the endurance of NAND flash memory.

3.1 NAND Flash Memory Based Swap System

Fig. 1 shows the architecture of the NAND flash memory based swap system. The swap area consists of a sequence of page slots, which is used to store a page swapped out from the main memory. When a page is swapped out, the location of the swapped-out page is stored in the corresponding page table entry (PTE). The location information in the PTE is used to find the correct swap slot in the swap area when the page is swapped in. Unlike a hard disk based swap system, the NAND flash memory based swap system has the Flash Translation Layer (FTL) and the Memory Technology Device (MTD) layer. FTL provides a transparent access to the NAND flash memory based swap system. If there are not enough free blocks in the swap area, the swap system should perform garbage collection. Garbage collection is also handled in FTL [13]. The MTD layer handles read, write, and erase operations for the NAND flash memory based swap system [14].

3.2 Block Recycling Scheme

In NAND 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. Therefore, we should minimize the garbage collection time to improve the performance of the NAND flash memory based swap system. 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 the number of valid pages in blocks and chooses the block with the least utilization as the victim block, we can minimize the garbage collection time. However it dose not consider wear-leveling and was shown 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 time of the pages in the blocks and the number of the erase operation of the blocks.

Fig. 2 shows the redistribution of the valid pages during garbage collection. 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 consider the Swapped-Out Time (SOT) of the valid page. The Swapped-Out Time (SOT) is the time when

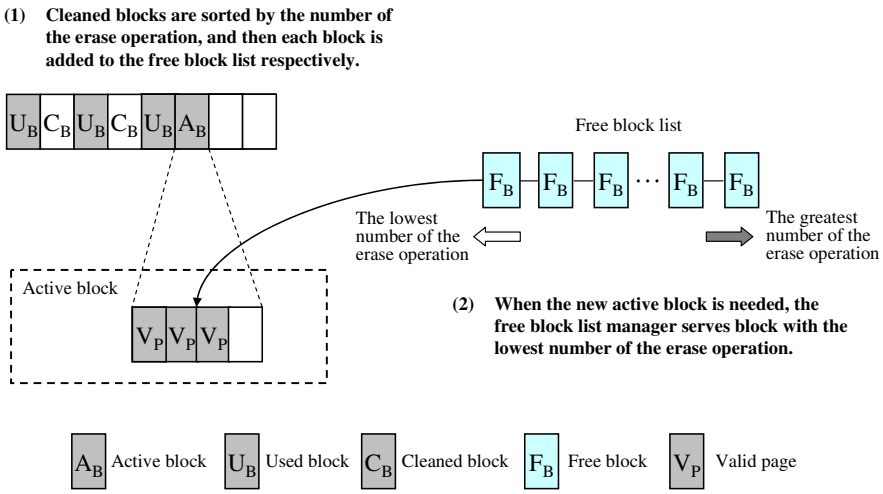

Figure 3. The efficient free block list management for wear-leveling

the page is swapped out from the main memory. Because the current operating systems use the round-robin based process scheduling scheme, the least recently swapped-out page is likely to swap in the main memory in the near futures [12]. Hence, we can classify the least recently swapped-out page as hot page. Since we calculate the SOT of the valid pages and sort the valid pages by the SOT value, and then copy the least recently swapped-out page first, we can get hot valid pages together into a free block during redistributing.

3.3 Efficient Free Block List Management

NAND flash memory used as the swap area should be controlled to evenly wear out all blocks since wearing out specific blocks could limit the usefulness of the whole flash memory based swap system. 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 the flash memory based swap system, we propose an efficient free block list management scheme for wear-leveling on the flash memory based swap system. In our proposed policy, we use the sorted free block list. After cleaning the victim blocks, we calculate the number of the erase operation of the block, and then the block is added to the free block list. The free block in the 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 operation to valid pages, and could evenly wear out. Fig. 3 shows the efficient free block list management 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 synthetic 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. We used the synthetic trace to assess the performance of the flash memory based swap system. Since the operating systems swap out many pages in a short period of time, we consider this access pattern to generate the synthetic trace.

To evaluate the performance, when the size of free block is fewer than 10% 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 20% of the total size of flash memory. Fig. 4 and Fig. 5 show the performance results of the number of erase operation and pages copied out for the four garbage collection policies. Because garbage collection performs a lot of page write and block erase operations, we should reduce the number of erase operation and pages copied out to improve the performance of the flash memory swap based system. Our proposed policy, S-Greedy shows better performance in these performance results, and these

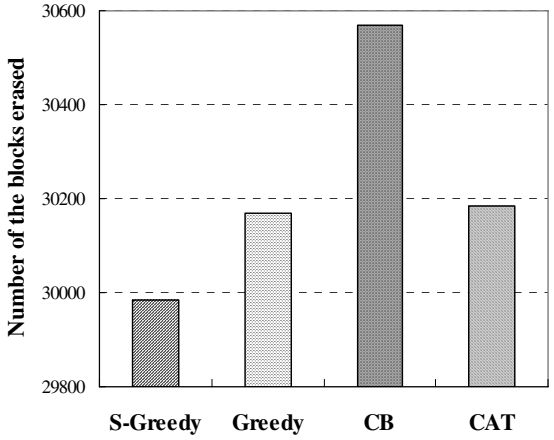

Figure 4. Number of the erase operations

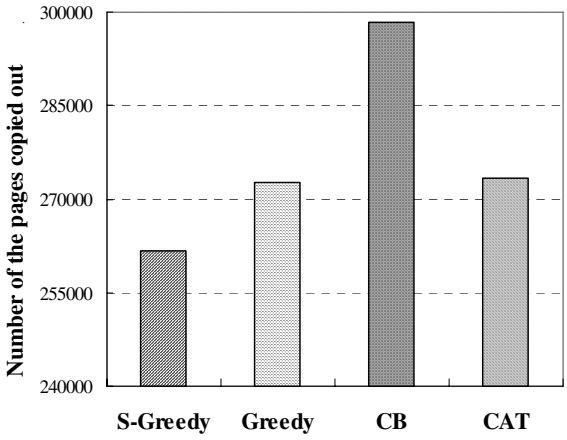

Figure 5. Number of the page copied out

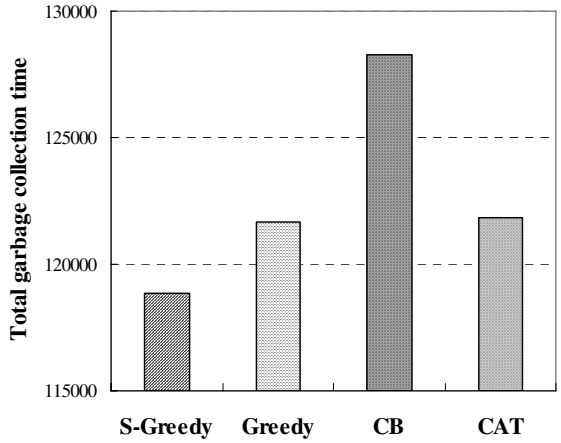

Figure 6. Total garbage collection time

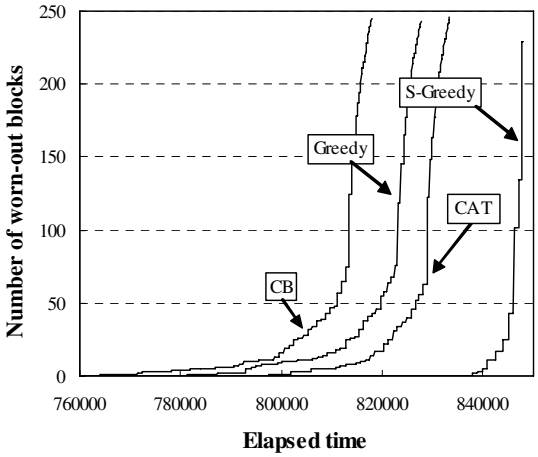

Figure 7. Number of worn-out blocks

results affect the performances of the garbage collection time and the number of worn-out blocks.

Fig. 6 shows the performance result of the garbage collection time. The S-Greedy policy shows better performance than other garbage collection policies. This is because the S-Greedy policy just considers the utilization of each block to minimize the garbage collection time unlike other policies. Furthermore, our proposed policy performs better than the original Greedy policy because it consider the Swapped-Out Time (SOT) of each page and exploits the SOT value to redistribute the valid pages.

Finally, Fig. 7 shows the performance results of the number of worn-out blocks. In this result, our proposed policy, the S-Greedy policy shows the best performance result than other garbage collection policies due to the efficient free block list management. This result means that our proposed policy guarantees the long endurance of NAND flash memory.

5. Conclusion

In this paper, we proposed the new garbage collection policy for the NAND flash memory based swap system. In order to minimize the garbage collection time and extend the lifetime of the NAND flash memory based swap system, we extended the Greedy policy by considering the different swapped-out time of the pages and the efficient free block lists management. As a result, the proposed garbage collection policy performs better than other existing garbage collection policies in terms of the number of erase operations, the garbage collection time, and the endurance of flash memory.

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. Dobb’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] D. P. Bovet and M. Cesati, “Understanding the Linux Kernel” O’Reilly, third edition, 2006.
[13] Intel Corporation, “Understanding the Flash Translation Layer (FTL) Specification”.
[14]http://www.linux-mtd.infradead.org