Journal of Electrical and Electronic Engineering
2016; 4(3): 73-77
http://www.sciencepublishinggroup.com/j/jeee
doi: 10.11648/j.jeee.20160403.16
ISSN: 2329-1613 (Print); ISSN: 2329-1605 (Online)

Page Replacement Algorithm for NAND Flash Used in Mobile Devices

Hai Jun Zhang*, Wan Jun Yu

School of Computer Science & Information Engineering, Shanghai Institute of Technology, Shanghai, China

Email address:
13162575996@163.com (Hai Jun Zhang), 15216853935@163.com (Wan Jun Yu)
*Corresponding author

To cite this article:
Hai Jun Zhang, Wan Jun Yu. Page Replacement Algorithm for NAND Flash Used in Mobile Devices. *Journal of Electrical and Electronic Engineering*. Vol. 4, No. 3, 2016, pp. 73-77. doi: 10.11648/j.jeee.20160403.16

Received: May 22, 2016; **Accepted**: June 14, 2016; **Published**: June 17, 2016

Abstract: In modern society, intelligent devices equipped with flash memory are very popular. It has many wonderful characteristics, such as small, fast, little consumption, shock resistance and so on. Flash memory is divided into NOR memory and NAND memory. The NOR memory can be quickly read with byte data which is developed into data memory for code storage. A new algorithm is needed to optimize the performance of the flash memory. In this paper, we propose a new strategy for replacement to focus on reducing the execution time of the replacement cost and I / O, which is to improve the performance of the algorithm performance. Trace-driven method has a better performance than the existing algorithms in terms of cost and execution time.

Keywords: Page Replacement Algorithms, NAND Flash Memory, Embedded Systems

1. Introduction

Recently embedded systems are at a higher demand as mobile services are expanding. The capacity of storage increases and the price has fallen to a low level which is used widely. This is more suitable for making the NAND flash mobile embedded system. For example, a digital music player, PDA and wireless devices, embedded system is a higher demand for Mobile Service. Now, a variety of mobile embedded systems company uses Flash technology for their new products. Many intend to spend much money and time optimizing the performance in the near future. NAND Flash has many wonderful features such as small size, low power consumption, shock resistance which are very convenient and portable. In addition, the flash memory has a greater density than the non-volatile RAM, namely the same size can store much information. Embedded systems use flash memory as non-volatile data storage management. Different manufacturers use different technologies production for their flash memory.

Embedded systems such as mobile phones and digital music products contain DRAM, NAND flash memory and NOR flash memory. DRAM is used for main memory to store much information. NAND flash memory is used for data files, and NOR flash can process program codes. Mobile systems contain these three kinds of memory types, so the size of the phone is bigger. In order to scale down the size of the phone, we eliminate the NOR flash from the mobile phone and it can also scale down the cost of the mobile phone. If the mobile phones do not have the NOR flash memory, the code is needed to be copied to DRAM memory for processing the application programs when the program is running. This mechanism is called "shadowing". The mechanism has a good performance at running time because whole program codes are in the main memory and it doesn't need to spend time to exchange codes between two kinds of memory. However, it leads to another problem that the mechanism needs much time to copy a whole program code to the main memory. Besides, when shadowing is running, the mechanism will cost much main memory space. To solve the program about weakness of shadowing, we have exploited a "demand paging" for mobile embedded systems. Demand paging is a virtual memory technique. When code is needed, it will copy file from the secondary storage to main memory [1]. Thus, it doesn't occupy much main memory space and longer loading time than shadowing mechanism. When the space in the main memory is not enough and can't

load program because of its limited memory capacity, a new page need to be loaded in the demand paging mechanism to replace the page that is not used. That is called a replacement algorithm and it has significant importance when applied to I/O operations in embedded systems. The disk and the NAND flash medium have been studied for many years. Flash memory has different physical characteristics in contrast with traditional disks. The cost of write operation is higher than that of read operation in NAND flash, and each block has limited times of erase operations about 100,000 times. So it is very important to balance the times of each block erase operations. For these reasons, an effective replacement algorithm shall be made based on the embedded system for NAND flash memory. The newly replacement algorithm proposed for NAND flash attempts to reduce the number of write operations and reduce I/O execution time. The results of trace-driven simulation show that the performance of the replacement algorithm in cost and I/O execution time has a better than the existing algorithm about LRU and CFLU.

2. Related Works

This section describes some of the different types of flash about their physical characteristics, and analyzes some of the existing replacement algorithm.

2.1. Characteristics of Flash Memory

Flash memory is a non-volatile solid-state storage medium, and the density of storage has been improved, which is used as an auxiliary storage memory due to the physical characteristics. There are two kinds of memory, namely NOR flash and NAND flash memory. NOR flash can quickly randomly access the data which has a similarity with the DRAM memory. NOR flash can be used to execute the program codes. In contrast, NAND flash memory is widely used storing large amounts of data, such as multimedia files because it can't execute codes like NOR flash. However, NAND flash memory can store much information than NOR flash, and it has a characteristic of high density. NAND flash I/O operations are slightly expensive than traditional disk. As a result, manufacturers preferably used NAND flash memory as the storage system which is widely used in mobile devices and embedded system. In recent years, a new product called hybrid flash memory which RAM and NAND flash memory element are emerging is applied to computers and portable products. Hybrid flash memory composed of NAND flash and NOR flash increases rapidly. The hybrid storage medium widely used is accepted by many companies [2]. For example, Samsung's One-NAND which can not only store data but also execute code is one kind [10]. The characteristics of NOR, NAND and hybrid memory are listed in Table 1.

Table 1. *Performance comparison between NOR and NAND flash memory.*

	NOR multilevel cell (Mbytes/sec)	NAND 90-nm single- level cell (x8, large block) (Mbytes/sec)	Samsung One NAND 90-nm (Mbytes/sec)
Read	109	16.3	109
Write	0.14	6.9	8.3
Erase (single)	0.13	65	65

Flash memory is divided into a plurality of blocks, each block having a fixed number of pages. The working mechanism of the traditional disk and flash memory is different and flash memory has three different types of operations: read operation, write operation and block erase operation. Each operation has different performance characteristics. Comparing to the hard disk, Flash memory has two drawbacks involving I/O operations. Firstly, before the data in the flash memory is rewritten, the flash block needs to be erased for spare space. An erase operation costs a lot of time and energy than the other two operations, namely read operation and write operation. The second disadvantage is that the number of erase operations for each block is limited. Life becomes limited. This shortcoming hinders the development of flash memory used in embedded systems. To solve the problem of flash memory, a number of software layers mechanism proposed by people shall contain flash memory wear-leveling mechanism to balance the number of each block's operations [3]. Extending the lifetime of flash based NAND flash is a major challenge. However, the design of FTL (flash translation layer) can effectively optimize the performance. A good and reliable wear-leveling algorithm can not only extend the lifetime of the flash memory but also guarantee fast write and erase operations [4].

2.2. Existing Algorithms on Replacement

The demand paging technique for NAND flash code storage can reduce required main memory space. In Demand paging system, the purpose of a page replacement algorithm chooses the selection page which is less useful in the flash memory, then erases it for free to load a new page. When loaded, no free pages in the main memory, operating system will choose a victim page. If the selection page is clean, OS can remove the page from the main memory without exchanging data [5]. Hence, if the selection page is not clean, the data in the page will be copied to swap space in order to store the original data. There are some different kinds of replacement algorithms which are used for memory based on the system. LRU (least recently used) is widely used for page replacement on the traditional operating system, due to the simplicity of the algorithm. The main idea of LRU replacement algorithm is that the page cited more is more likely to be cited again later. LRU algorithm holds the page in the flash memory by the order of he cited time. In the flash memory, when more spare space needed, the operating system will choose the page which is used least as the victim page.

Megiddo proposed an algorithm called A Self-Tuning, Low

Overhead Replacement Cache [6]. ARC algorithm uses two lists. The first list includes the pages which are cited many times and the second list includes the pages only once used recently. Each kind of page fault occurred in each list determines the size of the two lists. The total of two lists is fixed, when the size of one list increases and the size of the other list decreases. Presentation above shows ARC algorithm is applicable to the page reference pattern.

Jiang proposed an efficient low inter-reference set replacement policy to enhance the performance [7]. LIRS algorithm replaces the page of hit with the inter-reference page. The LIRS algorithm also has an LRU stack to classify the total pages into two parts which are LIR pages and HIR pages. LRU stack maintains the LIR pages which are cited frequently when the HIR pages show that the pages are not cited frequently. To have a better performance, LIRS algorithm chooses the HIR pages as the objective pages. LIRS algorithm gets a better performance than LRU algorithm. However, when the size of buffer cache is larger than the size which is set before, the LIRS maybe have a bad performance and performs worse.

There is a fact that the cost of I/O operation is expensive, so some people proposed replacement algorithms to narrow down the page fault ration. The cost of a read operation is lower than the write operation which is based on the flash memory. Therefore, in order to reduce the cost, new replacement algorithms should try to improve the I/O performance and reducing the page fault.

Park proposed a Clean First LRU (CFLRU) replacement algorithm in the paper "Energy-aware demand paging on NAND flash-based embedded storages" [1]. The idea of CFLRU algorithm is that the I/O cost of the page read operation and page write operation is not the same in the flash memory. CFLRU algorithm remains the selection page in the LRU list, which has a similarity with the LRU page algorithm. However, different from the LRU algorithm, CFLRU list is divided into two areas, where the first area is clean-first area and the next area is the working area. The first area stores the victim pages which will be replaced preferably and the next area stores the pages used recently. In order to improve the I/O performance and reduce the cost of replacement, CFLRU algorithm removes first page in a clean-first area by the LRU algorithm order. CFLRU algorithm will remove dirty pages which are removed by LRU order, with no clean pages in the first area. In the working area, the principle of the CFLRU is as the same as the LRU algorithm.

3. Flash Aware Replacement Strategy

3.1. Replacement Strategy

In flash memory, the cost of the write operation and the read operation is not the same. The cost of the write operation is higher. In order to optimize the I/O performance and reduce the cost, it is more preferable to choose the clean page. However, it will have a problem of degrading the I/O performance because of the low match ratio. In order to not only maximize the match ratio but also reduce the cost, introducing three lists to achieve the two goals above. Figure 1 shows the three lists of FARS replacement algorithm.

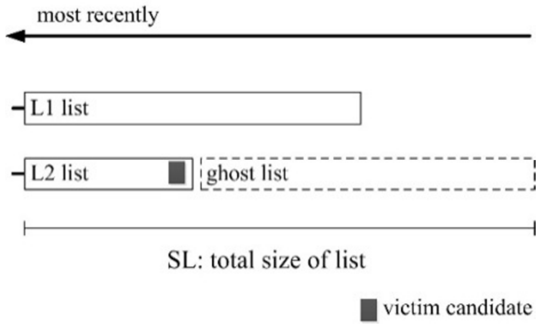


Figure 1. *Three level lists: L1, L2 and ghost list.*

The LRU lists are classified into two parts which are composed of L1 list and L2 list. SL1 is used to represent the volume of the L1 list which is frequently accessed by operating system. In addition, SL2 is used to represent the size of L2 list which is rarely used. The SL is the total of SL1 and SL2 which is a fixed number in the flash memory. Moreover, another list called L3 is used to represent the size of the volume called a ghost list which is used for storing the pages which are replaced from L2. The volume of SL2 and SL3 is equal to SL volume. L3 list does not store the page, but retains metadata of the pages deleted.

In the new algorithm, spare space is required for exchanging the page information when a page fault occurs. The algorithm chooses one page from the L2 list as an objective page out of the L2 list. And the L2 list will have space to be stored for a new page. In addition, the page will be stored in the L1 list when the new page is the ghost list before. Page in the L1 list will be accessed again. With the advantage of three lists, it is fast to remove a page which is not needed. With the ghost list, it is also fast to classify different types of pages.

3.2. Replacement Method

The Strategy of CFLRU algorithm is that dirty pages are stored for long periods in the DRAM memory which is mainly for reducing the overhead and improving I/O performance when dirty pages are removed. Cleaning the page early may cost the lower performance and affect the quality of the system running. In order to improve the hit ratio, FARS algorithm makes some optimizations. The main idea is as follows.

1. Determine whether the page is read or write intensively with the help of pattern detection algorithms
2. The priority of pages which will be removed is fixed, but when the dirty page is writing intensively, the operation to remove the page will be delayed.

In order to record the information of the dirty page and inspired by the classical shadow-paging method, new algorithm introduced a flag called "RP-flag" to record the status of the pages [8]. In the operating system, some kinds of pages like heap and stack pages are changed frequently. In contrast, a read intensive dirty page is modified once like data pages. To optimize the overall I/O performance, our proposed

algorithm maintains pages which are often written in the NAND flash memory as long as possible. The strategy is efficient to reduce the number of write operations. To a certain extent, it can improve the I/O performance. L2 list stores much information like read-intensive pages and write-intensive pages which can't store them for a fixed time set by people. If the page located in the L2 list for too long than the fixed time and it will remove the page from the list because it can make sure the consistency of data in the memory. That method addresses the program of the data missed when the power is off or the system is down. Fig. 2 illustrates the changes of the status with RP-flag.

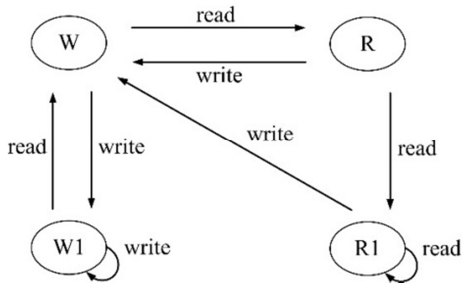


Figure 2. *The change of the status.*

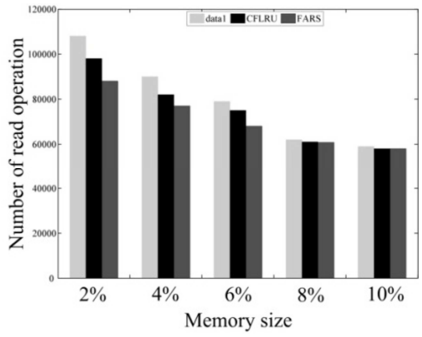
4. The Evaluation Performance

In this part, trace-driven simulations with real traces are implemented to evaluate the performance. On the comparison with other algorithms, the new algorithm is utilized to evaluate the performance. In order to compare the effectiveness of the FARS with LRU and CFLRU, the Val-grind which is a profiling tool of the Linux system can effectively evaluate the result. Val-grind can run with the Memcheck tool and it can detect a wide range of memory errors when running [9]. Table 2 shows the results of the characteristics.

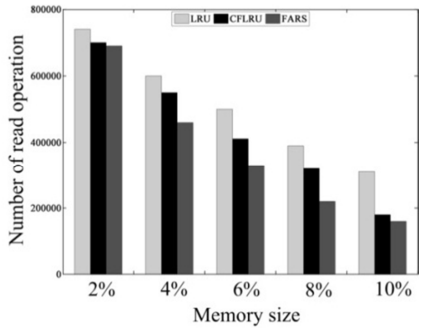
Val-grind tool can collect a single process about the trace of the memory reference and so trace-driven simulations are performed to assess the single process about the quality of I/O performance. Fig. 3 shows operation for the number of MP3 players tracking and the number of editing application performance results. In the results above, FARS algorithm shows the best performance, so it is effective. Fig. 4 shows that when the read operation is intensive, CFLRU has a better performance than the FARS algorithm because CFLRU keeps dirty pages as long as possible. However, comparing to the FARS algorithm, CFLRU algorithm costs much I/O execution time because the number of read times is bigger.

Table 2. *Characteristics of mp3 player and document editor.*

Workload	Memory used	Memory references			
		Total references	Instruction read	Data read	Data write
xmms (mp3 player)	11.09 MB	1,321,609	80,061 read: write=1:4.93	142,985	1,098,563
gedit (document editor)	12.85 MB	1,599,745	598,245 read: write=11.39:1	872,347	129,153

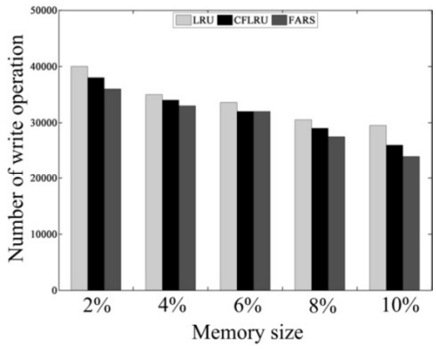


(a) xmms trace

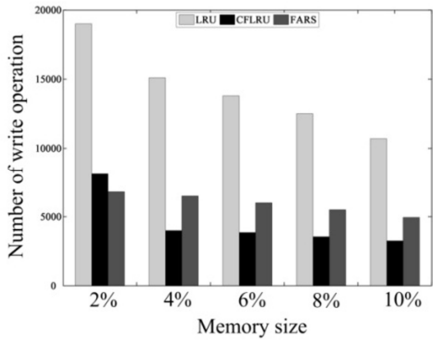


(b) gedit trace

Figure 3. *Number of read operations.*

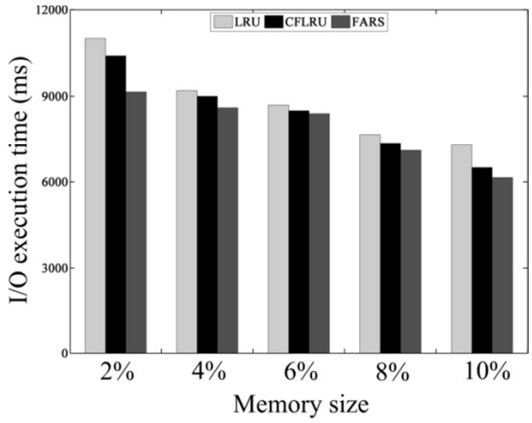


(a) xmms trace

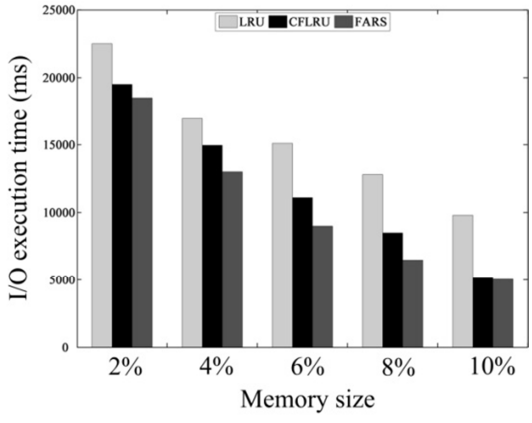


(b) gedit trace

Figure 4. *Number of write operation.*



(a) xmms trace



(b) gedit trace

Figure 5. *I/O execution time.*

Figure 5 shows the results about the execution time. In the performance results, FARS shows a better performance, because it reduces the read/write cycles. Trace-driven simulations show that the proposed algorithm performs better than the existing page replacement algorithms when they are performed on different kinds of NAND flash memories [10, 11].

5. Conclusion

In this paper, a new page algorithm is proposed which can reduce the execution time and replacement cost. In the new page algorithm, three lists are applied to improve I/O performance. In addition, a policy is proposed to detect whether the dirty pages are written or read more often. In terms of replacement cost and execution time, the result shows that proposed algorithm has a better performance than CFLRU algorithm or LRU algorithm. New algorithm mainly preserves LRU list. When a page is referenced, the list is responsible to move the page to a new position. However, what have done above will lead to another problem. The work not only costs much spending but it is also hard to implement in the flash memory. The next work is focusing on reducing the overhead of list and optimizing the algorithm.

References

[1] Pack, C, Kang, J, U, Park, S. Y, Kim, J. S, "Energy-aware demand paging on NAND flash-based embedded storages, "In: Proceedings of the 2004 International Symposium on Low Power Electronics and Design.2004.

[2] Chul. Lee, Sung Hoon Baek, Kyu Ho Park, "A Hybrid Flash File System Based on NOR and NAND Flash Memories for Embedded Devices," IEEE Translation on Computers, vol. 57, Issue. 7, July. 2008, pp. 102-1008.

[3] Baichuan Shen. "APRA: Adaptive Page Replacement Algorithm for NAND Flash Memory Storages", 2009 International Forum on Computer Science-Technology and Applications, 12/2009.

[4] T. Hoshi, K. Ootsu, T. Ohkawa and T. Yokota, "Runtime Overhead Reduction in Automated Parallel Processing System Using Valgrind," 2013 First International Symposium on Computing and Networking, Matsuyama, 2013, pp. 572-576. doi: 10.1109/CANDAR. 2013.102.

[5] W. Kim and D. Shin, "Non-preemptive demand paging technique for NAND flash-based real-time embedded systems," in IEEE Transactions on Consumer Electronics, vol. 56, no. 3, pp. 1516-1523, Aug. 2010.

[6] Megiddo, N, Modha, D," ARC:A Self-Tuning, Low Overhead Replacement Cache, "In the Proceedings of the 2nd USENIX Conference on File and Storage Technologies. 2003.

[7] Jiang, S, Zhang, X, "LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance," ACM SIGMENTRICS Performance Evaluation Review archive. 2002.

[8] S. T. On, J. Xu, B. Choi, H. Hu and B. He, "Flag Commit: Supporting Efficient Transaction Recovery in Flash-Based DBMSs," in IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 9, pp. 1624-1639, Sept. 2012.

[9] S. A. Hussain and A. Mansoor, "FLASH modelling for wear leveling algorithms," 8th International Conference on High-capacity Optical Networks and Emerging Technologies, Riyadh, 2011, pp. 267-272.

[10] J. Liu, S. Chen, G. Wang and T. Wu, "Page replacement algorithm based on counting bloom filter for NAND flash memory," in IEEE Transactions on Consumer Electronics, vol. 60, no. 4, pp. 636-643, Nov. 2014.

[11] M. Lin, S. Chen and Z. Zhou, "An efficient page replacement algorithm for NAND flash memory," in IEEE Transactions on Consumer Electronics, vol. 59, no. 4, pp. 779-785, November 2013.