Survey on Different Page Replacement Algorithm for Flash-Aware Swap Systems

Liya Thomas
Department of Computer Science and Engineering
SCT College of Engineering
Trivandrum Kerala

Punya Peethambaran
Department of Computer Science and Engineering
SCT College of Engineering
Trivandrum Kerala

Lekshmi Radhakrishnan
Department of Computer Science and Engineering
SCT College of Engineering
Trivandrum Kerala

Dr. Jayasudha J. S.
Head of the Department
Department of Computer Science and Engineering
SCT College of Engineering
Trivandrum Kerala

***Abstract*—Flash memory is used widely nowadays. Being a non volatile storage technique with small size and no moving parts, they are of great use in small portable electronic equipments. The physical features of flash memory introduce limitations also. These distinguishing features of a flash memory make it important to have replacement algorithms specifically designed for them. Most of the traditionally followed replacement techniques aim at improving the hit ratio and not to reduce the cost of writing. In this paper, a study on different page replacement algorithms for flash aware swap systems is discussed.**

Keywords—Flash Memory; Page replacement; page eviction; flash aware swap system; page replacement cost.

I. INTRODUCTION

Nowadays, flash memory has become one of the best storage media for portable consumer electronics such as digital cameras, smart phones, music players and laptop computers. Flash memory is advantageous in several aspects when compared with the conventional magnetic hard disks. The attractive features of flash memory are faster data access speed, light weight and smaller dimensions, better shock resistance, low power consumption and high capacity. These attractive features as well as increase in capacity and decrease in price have made flash memory popular among mobile embedded systems as a non volatile storage [1]. With several gigabytes of capacity in a single chip, flash memory is replacing magnetic disks as a secondary storage of mobile computing devices.

Due to the limited memory resources of portable consumer electronics, they currently exploit efficient swap systems considering flash memory as swap space as a cost effective solution to extend the limited memory space [2]-[4]. Unlike hard disks, the write operations to the flash-based swap storage require relatively long latency compared to the read operations. Moreover increased number of write operations will be accompanied by even more costly erase operations. The life time of flash memory is shorter than that of a hard disk. In other words, only a limited number of erase operations can be performed safely to each memory cell, typically between 100,000 and 1,000,000 cycles [5].

All the write operations to flash-memory-based swap storage are requested during the page replacement algorithm to make free page frames for the requested swap in pages. Due to the out-of-place update scheme adopted to solve the erase-before-write constraint, intensive write operations could result in using up the flash-memory-based swap storage quickly and incurring frequent garbage collection operations with high energy consumption. In order to reduce energy consumption of battery-powered portable consumer electronics, the design principle for designing an efficient page replacement algorithm for flash-aware swap system is to reduce the number of flash page write operations.

In this paper, a study on different page replacement algorithms for flash aware swap systems is presented. The rest of the paper is organized as follows. Section II gives an overview of flash aware swap systems. Section III presents the study of different page replacement algorithms namely Clean First LRU (CF-LRU), LRU-Write Sequence Reordering (LRU-WSR), Flash Aware Replacement Strategy (FARS), Cold Clean First-LRU (CCF-LRU), Flash Aware Buffer management policy (FAB), Adaptive Cost Aware Buffer Replacement Algorithm (ACR), Adaptive Double-LRU (AD-LRU) and Greedy page replacement algorithm. Section IV presents a comparison of these algorithms and Section V contains the conclusion.

II. BACKGROUND

In this section, a brief overview of flash memory and flash-aware swap system is presented.

A. FLASH MEMORY

Flash memory is a type of non volatile computer storage chip. Flash memory can be erased very quickly by electrical means. The name 'FLASH' was chosen since it reminded its inventors of the speed of flash on a camera.

1) TYPES OF FLASH MEMORY

There are two types of flash memory, the NOR flash memory and the NAND flash memory. NOR performs slower sequential read and write operations than the NAND but achieves very fast random access. NOR flash memory is

suitable for code execution and storage. NAND type is most suitable for mass data storage [6]. NAND flash memory can store more data in smaller silicon area resulting in lower cost per bit. The NAND architecture is very compact.

2) PROPERTIES OF FLASH MEMORY

a) Endurance: Endurance describes the number of times the media can be erased and rewritten. Flash memory deteriorates over time. A proper flash management can considerably slow down this process.

b) Data Retention: Data retention describes how long the flash media can retain data before it becomes unreliable. Flash get used up over time after repeated writing and erasing operations shortening the data retention.

3) NAND FLASH MEMORY

NAND flash memory is organized by many blocks, and each block contains a fixed number of pages. As it can be seen in figure 3.1, each page consists of 512 bytes in the main area and 16 bytes in the spare area. The main area is usually used for storing data, while the spare area is often used to store management information and error correction code which corrects errors when reading and writing.

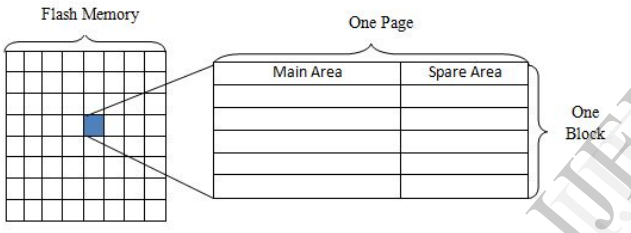

Figure 3.1 The architecture of flash memory

A NAND flash memory provides three basic operations: read, write, and erase. The read operation fetches data from a target page, while the write operation writes data to a page. The erase operation resets all the values of a target block to 1. Namely, the granularity of erase operation is a block, while the granularity of read/write operation is a page. However, flash memory exhibits a number of unique characteristics which might have a significant influence on energy consumption of traditional page replacement algorithms customized for magnetic disk directly implemented for flash-aware swap system. First, flash memory adopts out-of-place update scheme to solve the erase-before-write constraint of flash memory. Namely, once a page is written, it should be erased in advance before the subsequent write operation is performed on the same page. Second, the cost of flash page write operation is much higher than that of flash page read operation in terms of latency and energy consumption.

B. FLASH-AWARE SWAP SYSTEM

Almost every modern OS makes use of additional swap space in order to supplement the limited physical memory space. The portable consumer electronics currently exploit efficient swap systems considering flash memory as swap space as a cost effective solution to extend the limited memory space [7]. However, there are some limitations for flash memory. First, it does not support in-place update of pages, that is, a page in a flash memory cannot be overwritten before it is erased. Second, the number of erase operations allowed to be performed for each block is limited. Also it takes longer time to erase a block.

To relieve these drawbacks, one of the common practices used for flash memory based secondary storages is to employ a thin software layer called Flash Translation Layer (FTL). Since FTL emulates the conventional hard disks transparently, most systems do not require any modification to the kernel in order to build file systems or swap space on top of flash memory based secondary storages. FTL also performs garbage collection to reclaim free pages by erasing appropriate blocks. In addition, FTL should deal with power-off recovery because there could be unexpected power-outages any time in mobile embedded systems.

But the kernel or FTL can sometimes make bad decision because FTL has no access to kernel level information and kernel is not aware of flash memory states. For example, although there are many invalidated page slots in the swap area due to process termination, unmapped anonymous pages, and page slot release by the kernel, such information is not readily available to FTL. This will result in a situation where FTL uselessly copies those pages during garbage collection, considering them as valid.

Some functions of FTL are even unnecessary for swap system. For instance, the comprehensive power off recovery is not necessary for swap system, since the contents of page slots are meaningless after the system restarts. For the same reason, it is not required to preserve mapping information managed by FTL. This inefficiency can be avoided by turning off useless functions of FTL and providing communication channels between the kernel and FTL. However, this requires modification of both layers and incurs communication overhead. Moreover, the extra mapping performed inside FTL is redundant if the page table entry (PTE) points to the physical location of page slot directly. Hence a novel flash-aware swap system called FASS is introduced.

In FASS, the kernel manages NAND flash memory based swap space directly without the use of an intermediate layer such as FTL.

III. PAGE REPLACEMENT ALGORITHMS FOR FLASH AWARE SWAP SYSTEMS

A. Clean First LRU (CFLRU)

S.Y. Park and D. Jung in [8] proposed the Clean First LRU (CFLRU) replacement policy which takes into consideration the imbalance in the cost of read and write operations of the flash memory when replacing pages. The main idea of CFLRU is to keep a certain amount of dirty pages intentionally in page cache to reduce the number of flash write operations. Thus preventing the overall performance from being significantly affected due to the degraded cache hit rate. CFLRU is the first replacement algorithm proposed for flash memory.

CFLRU is a modification of the LRU algorithm. CFLRU divides the LRU list into two regions namely the working region and the clean-first region. The working region consists of recently used pages and most of cache hits are generated in this region. The clean-first region consists of pages which are candidates for eviction [8]. Inorder to save the flash write cost CFLRU selects a clean page from the clean-first region for eviction. In the absence of a clean page in this region, a dirty page from the end of the LRU list is selected for eviction.

The size of the clean-first region is called a window size, w. The window size should be properly adjusted as the hit rate depends on it. The hit rate may fall dramatically if the window size grows too large.

B. LRU-WRITE SEQUNCE REORDERING (LRU-WSR)

H. Jung et al. proposed LRU-WSR with the objective of reducing the number of flushes of dirty pages during page replacement [5]. The strategy used to achieve this objective is delaying evicting the page which is dirty and has high access frequency as possible [5]. Even though this strategy may reduce the hit ratio when compared to that of LRU, it effectively reduces the number of page writes and erases. As a result, it increases the overall performance of the flash memory based storage system.

Write Sequence Reordering (WSR) policy is used in this algorithm to overcome the limitations of CF-LRU. Basic scheme of WSR is as follows:

- Use cold-detection algorithm to judge whether the page is cold or not.
- Delays flushing dirty pages which are not regarded as cold.

LRU-WSR uses a page list and an additional flag known as cold flag. When a dirty page is chosen as a candidate for eviction, its cold flag is checked. In case it is not set, the page is moved to the MRU position of the buffer list after setting the flag and another candidate page is selected from the LRU position of the buffer list as victim. If the candidate page is dirty and its cold flag is set, the page is regarded as a cold dirty page, and is flushed into flash memory to avoid excessive decrement of the hit ratio. If a candidate page is clean, it is selected as a victim without checking the Cold flag. When a dirty page in a buffer list is referenced, the page is moved to the MRU position and its cold flag is cleared. WSR is heuristic algorithm and it effectively reduces the page writes and erases of flash memory without much degradation of hit-ratio.

C. Flash Aware Replacement Strategy(FARS)

Ohhoon Kwon. et al. proposed an algorithm called Flash Aware Replacement Strategy for NAND flash memory based embedded systems [9]. FARS exploits three levels of LRU lists for maximizing the hit ratio and minimizing the page replacement cost. In this method they divided the LRU list into three lists namely L1, L2 and L3 list. The size of L1, L2 and L3 are SL1, SL2 and SL3 respectively. The L1 list is used to maintain frequently accessed pages, and the L2 list is for pages not accessed frequently or pages accessed first.SL is the total number of pages in the memory which is the sum of SL1 and SL2. L3 is known as a ghost list which stores the information of pages replaced from L2. As it does not store the contents of pages but just the metadata of the evicted pages, it is called the ghost list. The sum of SL2 and SL3 is equal to SL.

When a page fault occurs, one of the pages in L2 is chosen as the victim and then a new page is stored in L2 if the page is accessed first. If the new page is resident in the ghost list, it is stored in the L1 list since the page is accessed again recently.

A Replacement Sequence Reassignment (RSR) strategy is used by FARS to reduce early eviction of clean pages. The basic idea of RSR is the following [9]:

1) Use a reference pattern detection algorithm to judge whether a page is read or write intensive.
2) If a dirty page is turned out to be write intensive, its eviction is delayed. Otherwise, the page is evicted by its ordinary priority.

To detect the reference pattern of a dirty page a flag called “RP-flag” is used. It changes when a read or a write reference occurs. The write intensive pages are kept in the main memory for long time in order to improve the performance. However, if there are no read or read-intensive pages, the write or write intensive pages are evicted from the L2 list. Also, if the dirty page resides in the L2 list over a predefined threshold time, the page should be evicted from the L2 list to guarantee the consistency of data in case of unexpected power off.

D. Cold-Clean-First LRU (CCFLRU)

Z. Li, P. Jin et.al introduced a new buffering algorithm which enhances the previous CFLRU and LRU-WSR methods by differentiating clean pages into cold and hot ones, and evicting cold clean pages first and delaying the eviction of hot clean pages [10]. In order to reduce the write count incurring in the replacement process, this algorithm tries to first evict clean pages with low access frequencies. If no such clean pages are there, it will evict the dirty pages with low access frequencies rather than evicting clean pages with high access frequencies. The cold-detection mechanism used is same as that of LRU-WSR.

The main idea of the CCF (Cold Clean First) strategy is as follows:

- It uses the cold-detection algorithm to judge whether the page is cold or hot.
- It evicts cold clean page preferentially as possible, especially for clean page accessed only once recently.
- If there is no cold clean page, cold dirty page is evicted instead of hot clean page.

The CCF-LRU algorithm maintains two LRU lists [10], namely mixed LRU list and cold clean LRU list. The mixed LRU list contains L1 pages and it maintains hot clean pages and dirty pages regardless of the status of its cold flag. The cold clean LRU list contains L2 pages and is used to maintain cold clean pages. If the buffer contains a total of L pages, the sizes of the two LRU lists are both from 0 to L. Moreover, the sum of L1 and L2 is L. By default the first referenced pages are regarded as cold are inserted into the cold clean LRU list with a cold flag set. When the page in the cold

clean LRU list is referenced again or becomes dirty, it will be moved from the cold clean LRU list to the MRU position in the mixed LRU list. When the page in the mixed LRU list is referenced, it will be moved to the MRU position of the mixed LRU list.

The CCF-LRU selects a victim page by the following rules in order:

- The LRU page in the cold clean LRU list is selected as the victim, if the cold clean LRU list is not empty.
- The LRU page in the mixed LRU list is chosen as the victim candidate, if the cold clean LRU list is empty. If the candidate is a cold dirty page, it is selected as the victim. If the candidate is a hot dirty page, it is labelled as cold and moved to the MRU position of the mixed LRU list. If the candidate is a hot clean page, it is set to cold and moved from the mixed LRU list to the MRU position of the cold clean LRU list and the LRU position in the mixed LRU list is checked continually. If there is no victim found after traversing the mixed LRU list, it needs to call the CCF-LRU algorithm one more time to select a victim.

E. Flash Aware Buffer Management Policy (FAB)

Joon Woon Lee et.al, introduces a flash aware buffer management technique that decreases the number of erase operations by deciding a victim based on its page utilization cost and not on the commonly used LRU policy [11]. This technique helps in decreasing the number of write and erase operations and thereby reducing the total execution time by 17% than the LRU policy.

This technique is particularly useful in portable media players (PMP). The media data in PMPs are accessed in long sequential order whereas the metadata of files are accessed with relatively short access. Thus in the case of PMPs, LRU policy cannot hold the data for short accesses in the buffer because the long sequential access pushes them away from the buffer. This technique also reduces the time required to search the data that is requested from the buffer.

The FAB on receiving a write request will search the buffer and if found (hit), that data will be overwritten. Else, the sector that is requested is written to a new slot that is allocated in the buffer. The data write on flash memory will happen only when the block is replaced. Here, the block which has the most pages in the buffer is identified as the victim block, when the buffer is full. This technique also considers whether the pages are dirty and if so, they are written back to the flash storage. This paper follows the LRU policy itself on a tie. Their concept of victim block rather than victim page adds on to the features of multimedia buffering.

Similarly, if there is a read request, if there is no hit and consequently, if the data has to be read from the flash storage to the buffer, then if the buffer is full, a victim block will be selected and evicted. The decision on whether the requested sector is in the buffer or not has to be taken more frequently, in [11] they have introduced two data structures for ease of search of the buffer such as block node list and page node list. Search Insert and replace operation have been designed exclusively for these data structures.

F. Adaptive Cost Aware BufferReplacement Algorithm (ACR)

Xian Tang and Xiaofeng Meng introduced an adaptive cost-aware replacement policy (ACR) which is using three cost-based heuristics in order to select the victim page [12]. Their technique works really well for various types of flash disks. They divided the buffer pages into two lists: clean and dirty lists. It is ensured that the frequently requested pages remain in the buffer for a longer time. The newly entered pages will be inserted at the middle and not at the MRU (Most Recently Used) position [12]. This technique that they have adopted made ACR a real adaptive buffer. This technique can be used in varying workloads. A ghost buffer is used as a buffer directory that maintains the list of once-requested pages that are recently removed from the buffer. If there are hits on the pages in the ghost buffer, they will be used to adaptively change the length of the buffer list and also adaptively identify the frequently accessed pages.

The basic idea that they have introduced is that the length of the LRU list containing clean pages and dirty pages should be proportional to the ratio of replacement cost of the pages to that of all the buffer pages based on the latest m requests, where m is half the buffer size. If the list of dirty pages is longer, then LRU page from that list should be paged out and vice versa.

The read and write operations have been discriminated as logical (Served in the buffer) and physical (Served in the disk). A conservative scheme that takes into account only the physical operations on pages and not the logical operations, an optimistic scheme that considers only the logical operations and a hybrid scheme that considers both physical and logical operations and thus combines the advantages of both the conservative and optimistic schemes have been defined for eviction based on costs. The ACR replacement policy adapts the length of the LRU list of clean pages and dirty pages based on the work load [8]. This paper introduced ACR replacement algorithms considering three cases:

- Hit request
- Miss request and the page is in the ghost buffer.
- Miss request and the page is not in the ghost buffer.

G. Adaptive Double LRU (AD-LRU)

Peiquan Jin, Yi Ou et.al introduced a buffer management technique for flash based databases which focuses on improving the overall runtime efficiency by reducing the number of write/erase operations and by retaining a high buffer hit ratio [13]. The AD-LRU concepts can be summarized as follows:

- Two LRU queues are used to capture both the recency and frequency of page references, among which one cold LRU queue stores the pages referenced only once and the hot LRU queue stores the pages referenced at least twice.
- Based on the changes in the reference patterns, the sizes of double LRU queues are dynamically adjusted.
- During the eviction procedure, the least-recently-used clean page from the cold LRU queue is selected as the

victim, for which a specific pointer FC is used. If clean pages do not exist in the cold LRU queue, a second-chance policy is used to select a dirty page as the victim.

The double LRU queues of AD-LRU separate the cold and hot pages in the buffer. When a page P is first referenced, it is put in the cold queue and set as the MRU element. When it is referenced again, it is moved into the hot queue and is set as the MRU element.

The page fetching algorithm [13] works as follows. If the requested page is found in the hot LRU queue, the page is moved to the MRU position. If the page is found in the cold LRU queue, the hot queue is enlarged; thereby automatically reducing the cold queue and the page is moved to the MRU position in the hot queue. If a page miss occurs and the buffer has free space, the size of the cold queue is increased and the fetched page is put into the cold queue. If the buffer is full, then a page has to be selected for replacement. The parameter min_lc sets the lowest limit for the size of the cold queue. If the cold LRU queue contains more than min_lc pages, the victim is evicted from the cold queue, otherwise, it is evicted from the hot queue.

When a page is referenced, its referenced bit is set to 1, which will be used in the SelectVictim routine to determine the victim based on the second-chance policy. When a victim is selected, the FC page (least-recently-used clean page) from the LRU queue is selected as the victim if FC is valid. If no clean pages exist in the queue, it selects a dirty page using the second-chance policy. The referenced bit of the buffer page under consideration is checked, if it is 1, the page is moved to the MRU position and the referenced bit is set to 0; this inspection is continued until the first page with referenced bit having 0 is located, which is then returned as the result.

H. Greedy Page Replacement Algorithm

M. Lin et al. proposed a greedy page replacement algorithm called GDLRU [14]. It maintains two page lists, namely clean page list and dirty page list. Each page in the dirty page list is divided into eight flash pages under the assumption that the sizes of page and flash page are 4KB and 512B, respectively [14]. Each flash page has a dirty flag. A dirty flash page is a flash page with dirty flag set.

If the number of free page frames in the main memory is lower than a threshold value, the greedy page replacement algorithm uses a clean-aware victim page selection (CPS) method to select a suitable page as victim. The CPS scans the clean page list and chooses the least recently referenced page as victim. If the clean page list is empty, the CPS scans the dirty page list and evicts the dirty page that minimizes the formula (1)

$$m / n \qquad (1)$$

Where m is the number of dirty flash pages in the selected victim dirty page and n is the number of all flash pages which the selected victim dirty page contains.

Once a page is selected as victim, the greedy page replacement algorithm uses a clean-aware victim page update (CPU) scheme in order to further reduce the number of flash page write operations. CPU checks whether the victim page is clean. If the victim page is clean, CPU just removes it from the physical memory to make the corresponding page frame free. If the victim page is dirty, CPU writes back only the dirty flash pages within the victim page into the flash-memory-based swap storage.

IV. COMPARISON

TABLE I. COMPARISON

Techniques	***Advantages***	***Disadvantages***
CLEAN FIRST LRU	• Helps reduce the write cost of flash memory when replacing pages. • Considers asymmetric I/O costs of read and write operation in flash memory. • Has better performance when compared to LRU.	• It needs to determine the window size dynamically. • It does not consider the access frequencies of data. • Dirty page selected as victim often contains clean data. • Not scan-resistant.
LRUWSR	• It considers the access frequencies of data while choosing a victim page. • Considers the cold/hot property of dirty pages. • Delays flushing cold dirty pages thus reducing the write request to the flash memory.	• Does not evict all clean pages before dirty pages. • Does not consider the clean data within the dirty page. • Does not consider the access frequency of clean pages. • Not scan-resistant and does not exploit frequency of references.
FLASH AWARE REPLACEMENT STRATEGY	• It reduces early eviction of clean pages by using RSR strategy. • It performs better than CFLRU.	• Does not consider the clean data within the dirty page.
CCFLRU	• Performs better than CFLRU and LRU-WSR by evicting cold clean pages first. • Differentiates both dirty and clean pages into cold ones and hot ones. • Focuses on the reference frequency of clean pages.	• No technique is used to control the length of the cold clean queue.
FAB	• Well suited for PMPs. • Decreases execution time by 17 % than LRU.	• Fragmentation and memory overhead is more for buffers with small size.
ACR	• Adapts to varying workloads and access	• Sequential writes may affect the

Techniques	*Advantages*	*Disadvantages*
	patterns • Scan Resistant • Loop Resistant	performance, only random writes are considered.
AD-LRU	• Self-tuning to respond to changes in reference patterns. • Takes into account the imbalance of read and write costs of flash memory when replacing pages. • Takes into account both the reference frequency of clean pages and dirty pages. • Has a mechanism to control the length of the cold queue. • Scan-resistant.	• Dirty pages evicted still contain clean data which is not considered.
GDLRU	• GDLRU incurs the least flash page read operations and consumes the least replacement cost. • It takes into account the clean data within a dirty page while replacement.	• Degrades the hit ratio in case of highly read intensive applications

V. CONCLUSION

In nutshell, the past and present of different flash aware swap systems are discussed. Device specific as well as device independent techniques can be devised. Flash memory page replacement cost can be further reduced by introducing more flash memory specific algorithms. Thus it has a rich scope for new research ideas and future works.

REFERENCES

[1] S. Park and S. Y. Ohm, "New techniques for real-time FAT file system in mobile multimedia devices," IEEE Transactions on Consumer Electronics, Vol. 52, No. 1, pp. 1-9, 2006.

[2] D. Jung, J. S. Kim, S. Y. Park, J. U. Kang and J. Lee, "A flash-aware swap system," Proc. of International Workshop on Software Support for Portable Storage, San Francisco,CA, USA, 2005.

[3] O. Kwon and K. Koh, "Swap space management technique for portable consumer electronics with NAND flash memory," IEEE Transactions on Consumer Electronics, Vol. 56, No. 3, pp. 1524-1531, 2010.

[4] S. Ko, S. Jun, Y. Ryu, O. Kwon and K. Koh, "A new Linux swap system for flash memory storage devices," Proc. of the International Conference on Computational Sciences and its Applications, p. 151-156, 2008.

[5] H. Jung, H. Shim, S. Park, S. Kang and J. Cha, "LRU-WSR: Integration of LRU and writes sequence reordering for flash memory," IEEE Transactions on Consumer Electronics, Vol. 54, No. 3, pp. 1215-1223, 2008.

[6] Arie Tal, "Two Technologies Compared: Nor vs. NAND" White Paper. http://www.m-sys.com/NR/rdonlyres/24795A9E-16F9-404A-857CC1DE21986D28/77/NOR_vs_NAND6.pdf

[7] D. Jung, J. S. Kim, S. Y. Park, J. U. Kang and J. Lee, "A flash-aware swap system," Proc. of International Workshop on software Support for Portable Storage, San Francisco,CA, USA, 2005.

[8] Y. Park and D. Jung, "CFLRU: A replacement algorithm for flash memory," Proc. of the 2006 International Conference on Compilers, Architecture and Synthesis for embedded systems, pp. 234-241, 2006.

[9] O. Kwon, B. Hyokyung and K. Kern, "FARS: A page replacement algorithm for NAND flash memory based embedded systems," Proc. of the IEEE 8th International Conference on Computer and Information Technology, pp. 218-223, 2008.

[10] Z. Li, P. Jin, X. Su, K. Cui, L. Yue, "CCF-LRU: A New Buffer Replacement Algorithm for Flash memory, " Transactions on Consumer Electronics, Vol. 55 , pp.1351-1359,2009.

[11] Heeseung Jo, Jeong-Uk Kang, Seon-Yeong Park, Jin-Soo Kim, and Joonwoon Lee, "FAB: Flash-Aware Buffer Management Policy for Portable Media Players," IEEE Transactions on Consumer Electronics, Vol. 52, No. 2, MAY 2006.

[12] Xian Tang, Xiaofeng Meng, "ACR: an Adaptive Cost-Aware Buffer Replacement Algorithm for Flash Storage Devices," Eleventh International Conference on Mobile Data Management, IEEE Computer Society, pp 33-42 , 2010.

[13] Peiquan Jin, Yi Ou, Theo Härder, Zhi Li, "AD-LRU: An efficient buffer replacement algorithm for flash-based databases," Data & Knowledge Engineering, 2012.

[14] Mingwei Lin, Shuyu Chen and Guiping Wang, "Greedy Page Replacement Algorithm for Flash-aware Swap System", 2012.