REVIEW ON DEVELOPMENTS IN NAND FLASH PAGE REPLACEMENT ALGORITHMS

Hitha Paulson
Research Scholar, Bharathiar University, Coimbatore, Tamilnadu, India
Assistant Professor, Little Flower College, Guruvayoor, Kerala, India
hitha@littleflowercollege.edu.in
https://littleflowercollege.edu.in/

Dr. Rajesh R
Associate Professor, Department of Computer Science,
CHRIST (Deemed to be University), Bangalore, India
r.rajesh@christuniversity.in
https://christuniversity.in/

Abstract - The non volatility, low power consumption and high density of NAND flash memories, made them an inevitable part of electronic industry. Due to the high wear out nature exhibited by flash systems, the algorithms used for page replacement in traditional memory systems are not suitable for flash page replacement. Along with the objective to maintain high hit rate, flash page replacement algorithms should aim at decreasing the page write count and maintain wear levelling. This paper presents major algorithms proposed for flash memory page replacement. The major contribution of this work is a relative study on various strategies, performance matrices and evaluation tools used for flash page replacement algorithm. The study shall help the researchers to identify the pros and cons of various flash page replacement algorithms, to identify the major gaps in between and to identify some commendable tools that can be used for flash page replacement algorithm evaluation. The gaps identified need to be addressed seriously in the near future.

***Keywords*:** Page replacement; NAND Flash; Write count; Hit ratio; LRU

1. Introduction

Recently more electronic equipments and even data centres are equipped with Non Volatile RAMs, especially NAND Flash memories due to its exceptional features like high density, light weight, high data access speed, low power consumption, increasing capacity and decreasing cost [1-4]. Thus Flash memories became a dominant memory component in consumer electronics. Even though these features of flash memories are highly appreciable for an efficient memory system, certain peculiarities of the Flash memory components needs to be addressed vigilantly. The life time of a flash memory is much shorter than a hard disk, so to wear level the memory cells we need to design programming components which reduces write operations to flash cells. The latency time of write operations in flash cells are much higher than read operations which affects the total efficiency of a memory system. This asymmetric nature is really a challenging issue before the researchers especially for the development of page replacement algorithms in memory swap system.

This paper is a comparative study on various page replacement algorithms developed for flash memory buffers, the outcome of the performance matrices addressed and various evaluation tools used for program trace generation and assessment. The paper is structured with the following sections. Background study, techniques in flash memory based page replacement algorithms, performance matrices appraisal of flash memory based page replacement algorithms, evaluation tools used for flash memory based page replacement algorithm design, research gaps and future trends and conclusion.

2. Background Study

For an effective implementation of support architectures for flash systems, a thorough understanding of characteristic differences in traditional hard disk drives and flash memory components are needed. Table 1 list out the features of Hard disk and Flash Memory system.

Table 1: Hard Disk vs Flash Memory

Hard Disk	Flash Memory
Read/Write performance is highly symmetric	Read/Write performance is highly asymmetric
The life is large unlimited	The life is largely limited
The disk latency is about 10s milli sec	The disk latency is about 10s – 100s micro sec
The addressing is sequence sector based	Direct addressing and byte addressable

Two different flash memory types are there. NAND flash memory and NOR flash memory [5]. NOR flash memory component is relatively fast and can execute application directly, but they are less dense and their manufacturing cost is so high. These features prevented its future prospects. Compared to NOR memory, NAND memory showed fast data accessibility, highly denser and comparatively cheaper in manufacturing cost [6].

NAND flash memory is organised by blocks and blocks composed of pages. Flash memory support three basic operations: Read, Write and Erase. Read and Write operation is page wise where as Erase operation is block wise. Since the write operation is preceded by erase operation always, write operation is highly costly than read operation. Flash devices make use of a Flash Translation Layer (FTL) as an interface between operating system interface and memory. This helps to achieve file system compatibility between operating system and storage unit [7]. Even at the presence of FTL layer traditional page replacement algorithms are not able to meet the peculiar characteristics of Flash systems.

The Least Recently Used (LRU) algorithm is known to be the widely recommended and implemented solution in page replacement problem related to main memory management. This algorithm focuses on increasing the hit ratio of pages and it gives preference to the pages which are very recently used, with an assumption that they will be referenced in the near future too. So the selection of victim for page replacement spotlights on a page which is least used in recent time. If we concentrate on this same LRU technique for flash memories, the selected victim will probably a dirty page and a large number write operations will be initiated in a NAND flash memory which leads to degradation in the life time of NAND flash components. Because of these concerns, the objectives of page replacement algorithms reframed for flash systems.

3. Techniques In Flash Memory Based Page Replacement Algorithms

As per the challenges discussed above, this section put forth a detailed analytical study about the evolution in page replacement algorithms related to flash memory implementations.

Clean First LRU

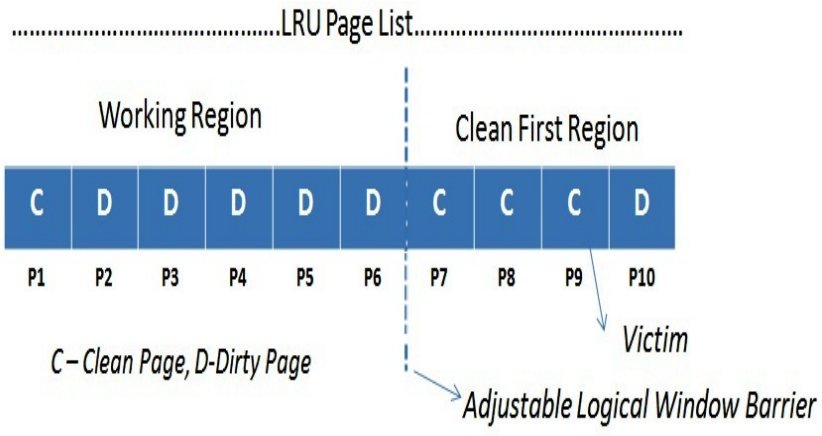

Fig 1: Symbolic representation of CFLRU algorithm

A LRU based algorithm, CFLRU (Clean First LRU) was proposed for NAND Flash based storage devices by Park et al [8]. This algorithm identifies pages as clean pages and dirty pages. The unmodified pages of reference are the clean pages and the modified ones which need write back to memory are called dirty pages. The corresponding page table entries keep watch on this clean/dirty parameter. CFLRU technique maintains a LRU page list and logically divides the list to working set region and clean first region set. The Most recently used end of the list belongs to working set region and the other end belongs to the clean first region. The algorithm selects a clean page victim for replacement from the clean first region for replacement and a dirty page from the LRU end in the absence of a clean page. If this region is empty, the LRU page in the working set will be replaced. The logical barrier gets adjusted appropriately as it leads to the increase/decrease in hit ratio. Various parameters like cost of flash write operation, cost of flash read operation, number of dirty pages and clean pages that have been evicted in the LRU order and probability of future reference of a clean page are considered for determining the window size[8].Figure 1 represents CFLRU algorithm symbolically.

CFLRU/C, CFLRU/E, DL-CFLRU/E

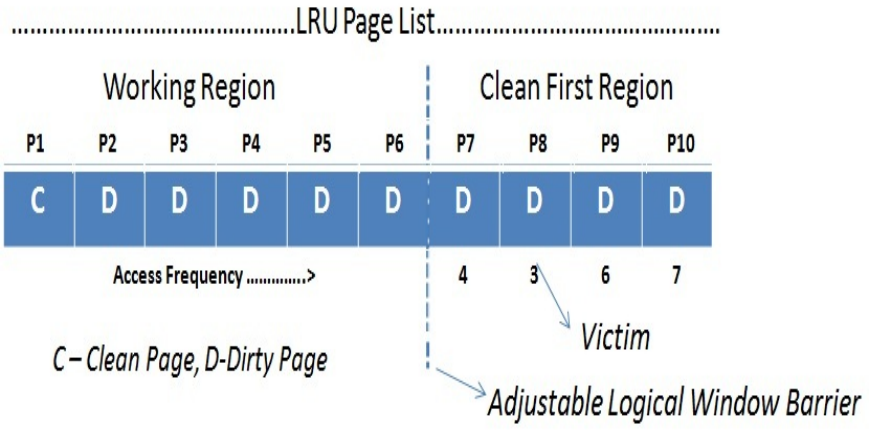

Fig 2.a: Symbolic representation of CFLRU/C algorithm

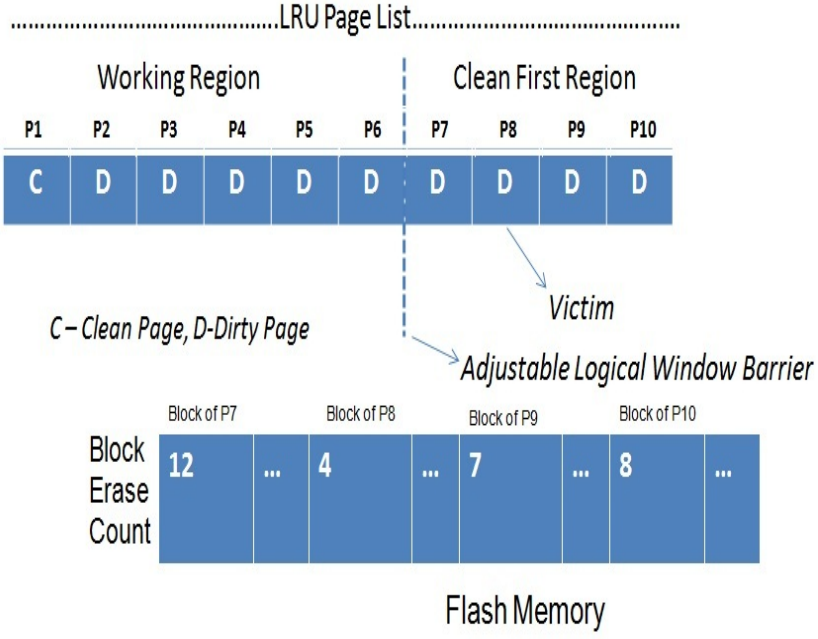

Fig 2.b: Symbolic representation of CFLRU/E algorithm

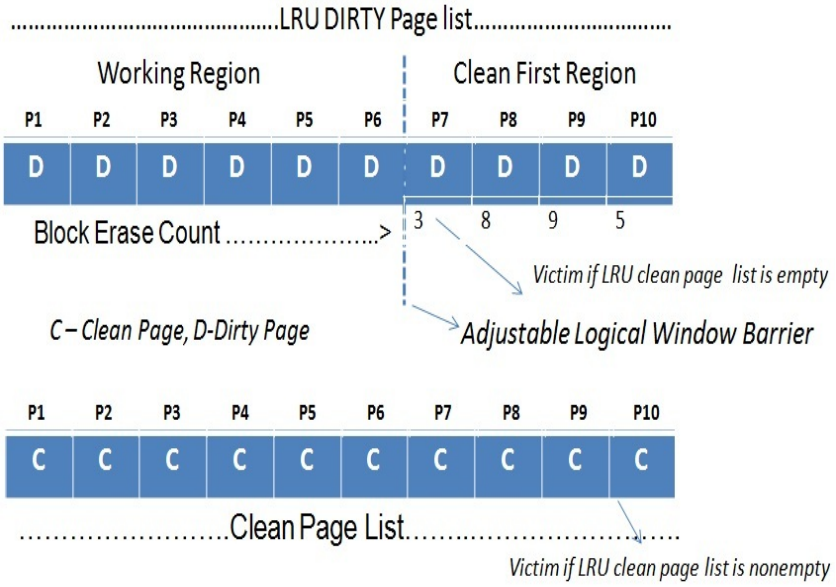

Fig 2.c: Symbolic representation of DL-CFLRU/E algorithm

CFLRU algorithm was later modified by Yoo et al and have proposed three algorithms called CFLRU/C (CFLRU/COUNT), CFLRU/E (CFLRU/ERASE) and DL-CFLRU/E (DOUBLE LIST CFLRU/ERASE) [9]. The first algorithm of the set works almost similar to CFLRU, but with a variation that when there is no clean page, rather than selecting the LRU dirty page, it selects dirty page with lowest access frequency as victim. The second algorithm of the set selects the dirty page of that block with lower erase count as victim, if no clean page available as victim in clean set region. The DL-CFLRU/E algorithm maintains two LRU sets, one for clean pages and other for dirty pages. The LRU page from the clean list gets evicted first. If there is no such victim available lowest erase count block page from dirty set is selected as victim. The main focus of the algorithm is on wear levelling of memory. Figure 2.a, 2.b, 2.c represents CFLRU/C, CFLRU/E, DL-CFLRU/E algorithms symbolically.

LRU-WSR

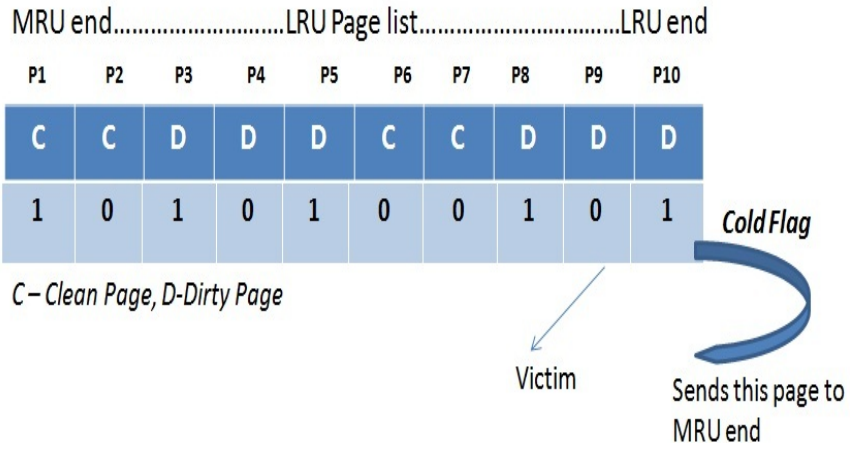

Fig 3: Symbolic representation of LRU-WSR algorithm

The write sequence reordering algorithm proposed by Jung et al is a modified version of LRU algorithm and is named as LRU-WSR [10]. A new concept of categorising pages as cold and hot pages is introduced in this algorithm. Cold pages are those pages which are referenced only once during its life time in flash memory. They raise a chance of no referencing in future. The algorithm maintains an LRU page list with a cold flag attached to each page. The cold flag indicates whether the clean/dirty page is a cold one or not. Pages for replacement are selected from the LRU end. If the page for eviction is a dirty page with a reset condition in cold flag, then the page is shifted to MRU position and the algorithm goes for another victim else it will be replaced.

If the victim page is a clean page it will be replaced without checking the cold flag. The main focus of the algorithm is to delay the wash out of dirty pages. Figure 3 represents LRU-WSR algorithm symbolically.

FARS Algorithm

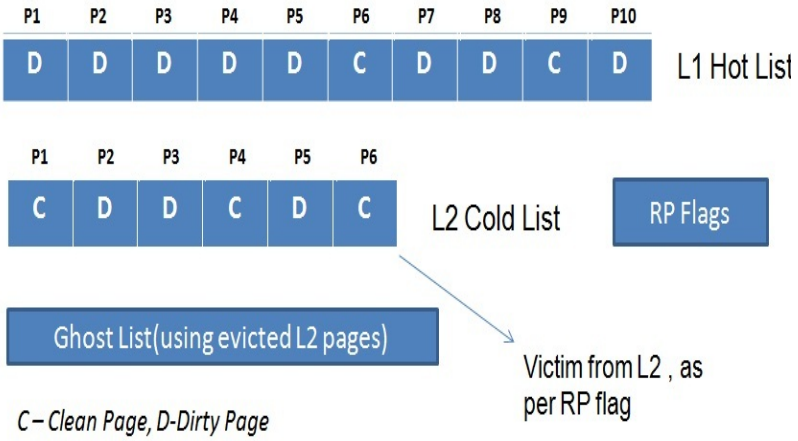

Fig 4: Symbolic representation of FARS algorithm

Ohhoon Kwon et al suggested an algorithm called FARS (Flash Aware Replacement Strategy) for NAND flash [11].This algorithm keeps 3 levels of LRU List. List L1 keeps hot pages and list L2 keeps cold pages. List L3 is a ghost list which keeps the metadata of the evicted pages from L2. When a page fault occurs, a suitable page from L2 list is selected as victim. If the new page to memory is from the ghost list, that indicates the hotness of that page and that page after memory fetch is placed in list L1. If it is a non ghost page, the page is placed in list L2. The suitable victim from list L2 is selected as per the Replacement Sequence Reassignment strategy. This strategy aimed to reduce the early replacement of clean pages merely for the sake of lessening write operations. The strategy tracks the write intensive and read intensive dirty pages using a flag called RP-Flag. Read intensive dirty pages are given more preference while eviction. The algorithm focuses on lessening the early replacement of clean pages. Figure 4 represents FARS algorithm symbolically.

CCF-LRU algorithm

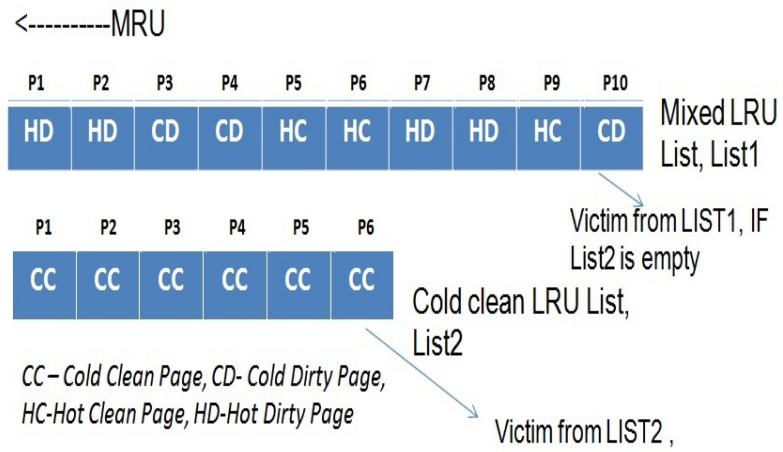

Fig 5: Symbolic representation of CCF-LRU algorithm

Li et al suggested a CCF-LRU algorithm for NAND Flash memory [12]. The algorithm keeps two LRU list. List1 is a mixed LRU list consisting of all dirty pages and hot clean pages. List2 keeps all cold clean pages. The pages from List2 moves to List1 upon their next reference and pages on List1 upon re-reference moves to MRU end of the same list. When a victim is needed for replacement, algorithm first search for pages in List2 and selects the LRU page if that list is non empty. If the List2 is empty, the LRU page from List1 is selected as replacement candidate. If that candidate is cold dirty page it is finalised as victim. If it is hot dirty, it is marked as cold dirty and moved to MRU end. If it is hot clean, it is marked as cold and moved to List2 MRU end. The process continues until a victim is finalised. The algorithm focuses on giving a second opportunity to many pages before they get evicted. Figure 5 represents CCF-LRU algorithm symbolically.

SEPL Algorithm

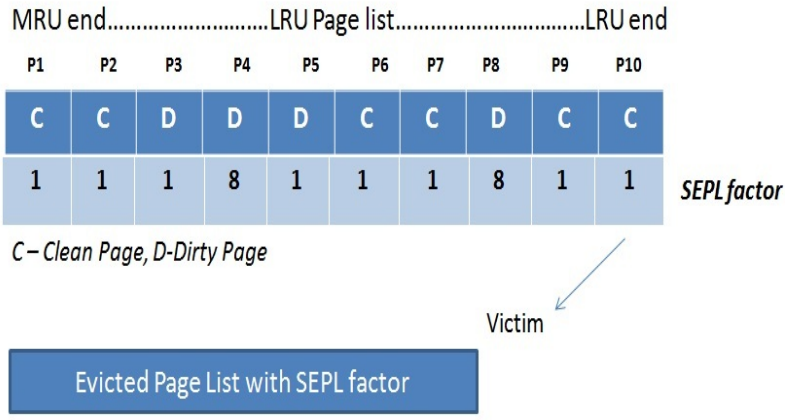

Fig 6: Symbolic representation of SEPL algorithm

The Smart Evicted Page List algorithm (SEPL) proposed by X. Jin et al [13] is a new approach to NAND Flash page replacement algorithms. An adjustable logical window is maintained in the normal LRU list. An evicted page list similar to ghost lists is also maintained for storing the meta data of the evicted pages. Each page is linked with a SEPL factor, which is a major factor in the decision making of victim page. Two constant factors, Readfactor and Writefactor are maintained with values 1 and 8 respectively. The values symbolises the multiplicative factor of write cost with respect to read cost. When a page hit occurs, the SEPL factor of the page gets incremented by the Readfactor/Writefactor parameter depending upon whether it is a read/write request to the page. If there is a miss and the page is there in the evicted list, the page is loaded to the MRU end of the Page list and its SEPL factor get incremented by 8, to reduce the replacement of frequently referenced pages. If the page is not in the eviction list and if it is 32 sectors away, the SEPL factor get incremented by 8 to keep the randomly accessed pages for a longer time. A replacement is performed when the buffer is full. The victim is selected from the LRU end. Clean page will be replaced first followed by dirty pages with low SEPL factor. Figure 6 represents SEPL algorithm symbolically.

AD-LRU algorithm

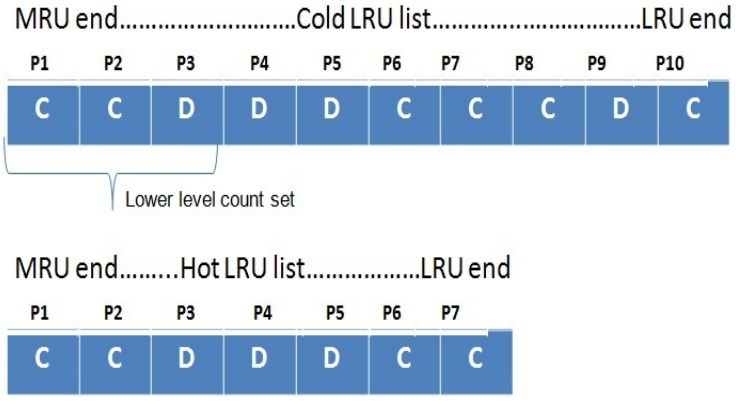

Fig 7: Symbolic representation of ADLRU algorithm

The Adaptive Double LRU (AD-LRU) proposed by P.Jin et al focuses on the eviction of least frequently and least recently used page during the need of page replacement [14].A cold LRU page queue and a Hot LRU page queue is maintained and the size of these LRU queues can be adjusted. Eviction happens from the LRU part of the Cold List. Clean pages from the LRU side get evicted first. In the absence of clean pages, dirty pages are selected following the strategy of second change existence. The main focus of the work is that a lower level count is set for the cold LRU queue at the MRU end, so that most recently and frequently pages are kept at the queue even though they are cold in nature. Pages are selected from the hot LRU list when the cold LRU list reaches its lower level count. Re referenced pages from cold LRU List moves to hot LRU list and re referenced pages from hot LRU list moves to the MRU end in the same list. Figure 7 represents ADLRU algorithm symbolically.

FAPRA algorithm

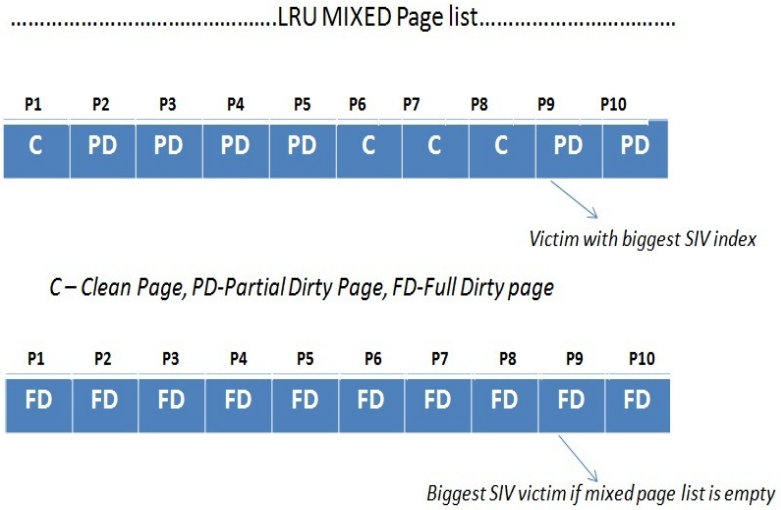

Fig 8: Symbolic representation of FAPRA algorithm

G.Xu et al suggested a Flash Aware Page Replacement Algorithm (FAPRA) in 2014 [15]. This algorithm introduced the concept of sub paging among the pages in the buffer. As per the degree of dirtiness among the subpages, the pages are categorised to clean pages, Partial dirty pages and full dirty pages. Partial dirty pages have clean and dirty subpages; where as full dirty pages have dirty subpages alone. The algorithm maintains two LRU lists. First list is a mixed page list of clean and partial dirty pages. Second list is a LRU list of full dirty pages. A benefit to cost ratio is calculated for each victim candidate page at the time of replacement. It indicates the percentage of dirty sub pages in each page. A selecting index value is also calculated from BCR by combining the frequency value of every replacement candidate and the erase count of the block of the page. The algorithm evicts the page with biggest SIV from mixed list and if the mixed list is empty full dirty page with biggest SIV is selected for eviction. The block erase property of NAND Flash is still a challenge to this algorithm. Figure 8 represents FAPRA algorithm symbolically.

History Aware –LRU Algorithm

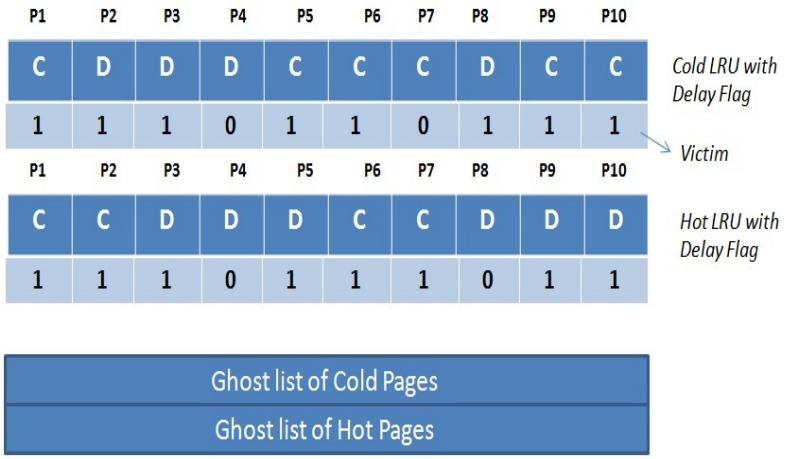

Fig 9: Symbolic representation of HA-LRU algorithm

The History Aware Page replacement algorithm (HA-LRU) proposed by M.Lin et al [16] aims at increasing page hit by giving a second chance to every dirty page to remain in main memory. The algorithm maintains four LRU lists, one for Cold Pages, one for hot pages another two are ghost lists for keeping the metadata of evicted cold and hot pages separately. A delay flag is associated with each page, which helps in delaying the eviction of dirty pages. If a page in a cold LRU list is referenced, it gets inserted to hot list. If the page in a hot list is referenced and it is a clean page it gets moved to MRU position. If that page is a dirty page, its delay flag get reset and it is moved to MRU position. If the references go to ghost pages, delay flag get reset and moves to MRU end of the hot list. At the time of replacement if it is a clean page at LRU end it will be selected as victim,

if it is a dirty page and its delay flag is one it will be given a second chance in the list by making its delay flag one. If the dirty pages delay flag is already one it is selected as victim. Figure 9 represents HA-LRU algorithm symbolically.

GASA ALGORITHM

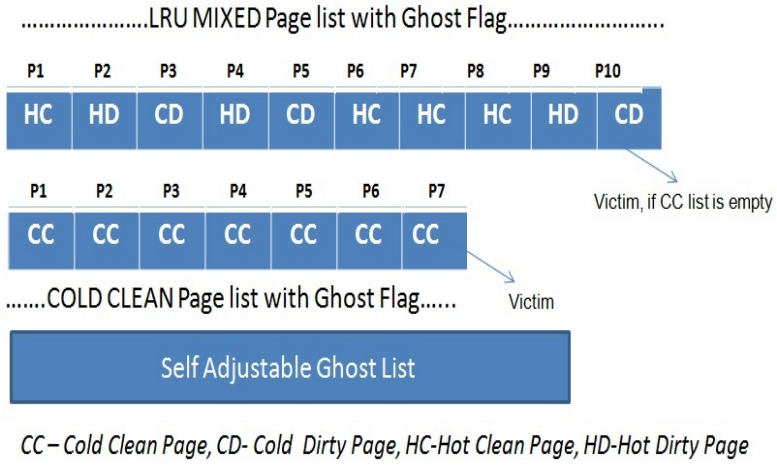

Fig 10: Symbolic representation of GASA algorithm

With a focus on maintaining a steady hit rate , a Ghost buffer Assisted and Self tuning Algorithm (GASA) was proposed by C.Li et al in the year 2016[17]. GASA algorithm makes use of three lists, Mixed List, Cold Clean List and Ghost List. Cold clean pages are stored in the second list and others in mixed list. Ghost data stores the Meta data of the ejected pages. Ghost buffer is an adjustable list. Cold clean list is scanned first for victim selection, if it is empty algorithm scans mixed list and searches for cold dirty pages. It is evicted and the Meta data is saved in ghost page. If a hot clean page is found it is converted to cold clean page and moved to cold clean list. Hot dirty pages are marked as cold dirty pages and moved to MRU of mixed list. The main attraction of the algorithm is its self adjustable ghost buffer. The ghost flag attached to the pages get set when it is referenced from a ghost list and the pages are placed at MRU end. If a ghost flag set page is referenced again algorithm identifies the hotness of the ghost list pages, increases the ghost list size and resets the ghost flag. If a victim page is found to have a ghost set flag that indicates the ghost list coldness and algorithm decrements the ghost list size. Figure 10 represents GASA algorithm symbolically.

AFAPRA Algorithm

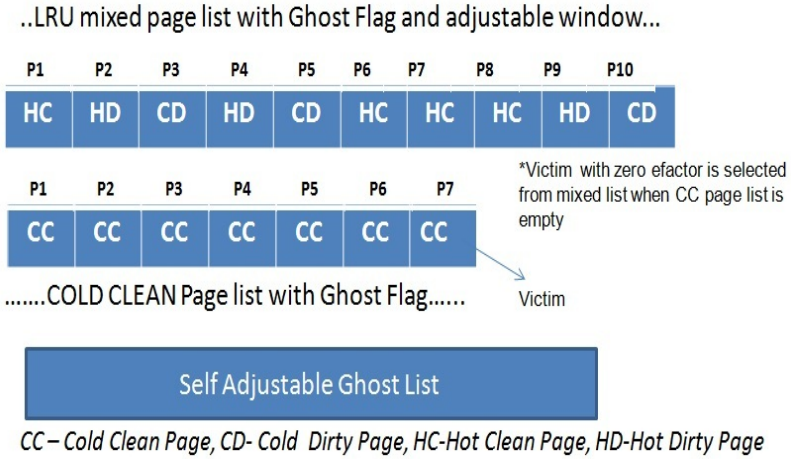

Fig 11: Symbolic representation of AFAPRA algorithm

L. Castellino et al proposed an Adaptive Flash Aware Page Replacement Algorithm (AFAPRA) [18]. The algorithm maintains three LRU Lists named Mixed List, Cold Clean List and Adjustable Ghost List which keeps the pages as the name indicates. The main feature of the algorithm is the implementation of an efactor count for the pages which indicates the pages priority and remains the main selecting criterion of the victim. The ghost flag of the pages identifies the need of adjusting the size of the Ghost Flag. Referenced pages are searched in the mixed list first and moved to MRU end if found. If the pages are referred for the write operation the efactor is

set to eight else one is added to the existing efactor. So the priority of write oriented pages increases by a factor of eight. The priority of the hot pages referred from Ghost list is increased by a factor of eight and that of the cold pages by a factor of four. Victim is selected from the cold clean list first, if it is empty algorithm checks the mixed list and page with efactor of zero is selected as victim. The nonzero efactor of traversed pages are decremented by one to adjust the priority. Figure 11 represents AFAPRA algorithm symbolically.

CF-ABR Algorithm

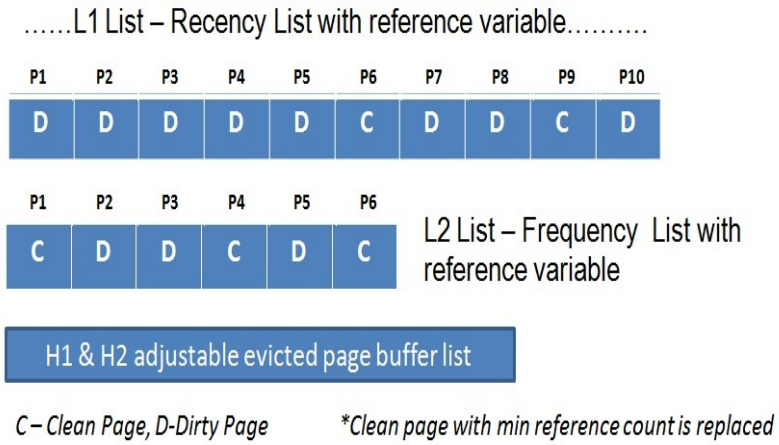

Fig 12: Symbolic representation of CF-ABR algorithm

Focusing on the importance of retaining the recently and frequently accessed pages of a flash buffer, Q. Huang proposed a Clean-First Adaptive Buffer Replacement Algorithm for NAND Flash (CF-ABR) [19]. This algorithm maintains two main lists (L1, L2) for recently accessed and frequently accessed pages and another two lists (H1, H2) for storing the pages evicted from the buffer. A reference variable is assigned to each page to keep the hit count and is initialised to zero first. If a referenced page is found in L1 or in L2its reference variable is increased by 1. The page which is referenced for the first time is kept at MRU end of L1 and upon their next reference they are moved to MRU end of L2. A variable called p is used to adjust the size of evicted page buffer. If pages are missing in L1 and L2 and are found from H1 list p is incremented else if it is found in H2 p is decremented. Clean page with min hit count is replaced preferably. Figure 12 represents CF-ABR algorithm symbolically.

4. Performance Matrices Appraisal of Flash Memory Based Page Replacement Algorithms

LRU is known to be the best algorithm in normal disk page replacement. The main performance metric of traditional page replacement algorithms is its hit ratio. The ratio indicates the possibility of finding the referenced page in the memory buffer itself during reference time. The high the ratio, less the effort needed to set pages to main memory. As the modern electronics goes on with Flash drives as memory component, more performance parameters are needed to be evaluated along with hit ratio, the prime performance parameter. The write cost of flash memory is around eight times higher than read cost. So various performance parameters like hit ratio, Memory Read count, Memory Write count, Wear levelling and Energy Consumption are to be evaluated for the better analysis of a Flash based algorithm.

To reduce the number of write operations to NAND Flash, we need to concentrate more on evicting clean pages while replacement. But mere replacement of clean pages adversely affects the hit ratio. So a balancing between hit ratio and number of page writes are expected. Recently developed Flash based page replacement algorithms even reported nearly 80% hit ratio [19] in all traces. For write most traces an average range of 130K to 250K write/read operations are reported in normal case [8-19].For read most traces an average range of 20K to 50K write/read operations are reported [8-19]. Wear levelling is the technique for extending the life time of a memory component. To wear level the flash drives, we need to ensure that the memory blocks of the flash are evenly erased. If the erase counts of the blocks are unbalanced, it results in the deterioration of certain blocks which affects the whole memory unit. Algorithms should concentrate on the block erase count of the pages to ensure wear levelling. CFLRU/E, FAPRA are some of the algorithms focused on wear levelling. The page hit ratio and write operation count are the main parameters that influence the energy consumption [16]. The energy consumption can be reduced by improving hit ratio and decreasing the write count of pages. An average energy consumption of 8700 mJ is reported by various algorithms in write oriented traces and an average of 3200 mJ is reported by various algorithms in read oriented caches.

As the number of buffers allocated to a program varies the performance also varies to a certain extend. Certain algorithms behave positively in increased buffer size and certain not. So algorithms are evaluated to various buffer size allocations, but determining an accurate buffer size for a process is still challenging.

5. Evaluation Tools Used For Flash Memory Based Page Replacement Algorithm Design

All the algorithms discussed above are tested upon the page trace either derived lively or made synthetically. Live traces are generated from the common programmes under use. Traces are categorised mainly under random set, write oriented set and read oriented set [8-19]. Each of the algorithms may behave differently on different category of traces but an algorithm which behaves evenly on all the category of traces can be considered as a superior one.

Various evaluation tools are used by the researchers in testing the algorithms proposed. Valgrind Toolset [20], Flash-DBSim [21] and DiskSim [22] are popular tools used to generate programme traces and to simulate the functioning of a disk. Valgrind is Linux oriented tool, whereas Flash-DBSim is Visual Studio based Windows oriented tool. Windows version of Valgrind and Linux version of Flash-DMSim are also available. WinValgrind is the Windows version of Valgrind and is popularly used as memory leak tracking tool for windows. A recently developed Linux version of Flash-DBSim [23] is a commendable one. Flash-DBSim is a trendy tool used to replicate the functioning of NAND Flash Disk. There is a provision to set the size of the disk, block and pages of the disk, which helps in real time simulation. Many pre-recorded traces are also available to research support in FlashDBSim. Valgrind is an efficient tool for dynamic analysis. DiskSim is a precise disk system simulator developed at the University of Michigan to support experiments in diverse aspects of storage systems. All these tools are built with modular architecture. So without disturbing the current structure set new tools can be developed easily in it. Algorithms can also be evaluated by replacing the open source code available for Linux based systems.

6. Research Gaps and Future Trends

The ever existed challenges of page replacement algorithm is to find the right victim to swap out during a page miss in a virtual memory system, the optimal allocation of pages to processes in main memory and to keep a consistent and high hit ratio. In a flash swap system few more challenges added up. Reducing the number of flash page writes, consistent maintenance of block wear levelling, reduced energy consumption by algorithms are the new challenges added.

To lessen the number of write operations many of the proposed algorithms focused on the eviction of clean pages but that inversely affected by reduced hit rate in many cases. By considering the frequency and recency of the pages dirty pages were also preferred over clean pages but in certain cases it affected the wear levelling in memory blocks. Still there are rooms for improvement.

- Even though number of write operations needs to be reduced, wear levelling is a major factor to be addressed. Very few algorithms considered the block erase count while finalising the victim. Important features in the proposed algorithms added with wear levelling solutions lead to powerful results.
- In order to achieve the major challenges rose above, many of the algorithms devised new methodologies which in affect increased the complexity in data structural implementations of algorithms. An optimisation in the data structural implementations of the flash page replacement algorithms was never addressed.

Future trends in this area are exciting. Hybrid Disk Buffer policy is really promising. Cooperative Buffer Management Policy and Write Pattern Aware Buffer Management [24] [25] considers hybrid buffers unlike existing disk buffers. Hybrid disk buffer consists of DRAM and NVRAM. The performances of these algorithms are hopeful even in write oriented traces. The supportive use of DRAM helps in reducing the major challenges in Flash memory writes.

7. Conclusions

This paper methodically analysed the major contributions in the flash memory page replacement algorithms. Evaluated the challenges and drifts happened in the Flash memory techniques. This evaluative study clearly reveals the effective research going on in the field of NAND flash memory algorithms and also spotlights the major gaps to be filled for more effective implementations. The policy makers shall utilise the performance evaluation and tool analysis explained in the papers and can contribute more to the gaps identified.

References

[1] R. Chen and M. Lin, “Energy-aware buffer management scheme for NAND and flash-based consumer electronics,” IEEE Trans.Consum. Electron, vol. 61, no. 4, pp. 484–490, Nov. 2015, doi: 10.1109/TCE.2015.7389803.

[2] 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.

[3] 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.

[4] 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.

[5] M. Lin, Z. Yao, T. J. O.-I. J. f. L. Huang, and E. Optics, "F-LRU: An efficient buffer replacement algorithm for NAND flash-based databases," vol. 127, no. 2, pp. 663-667, 2016.

[6] Y. Yuan, J. Zhang, G. Han, G. Jia, L. Yan, and W. J. I. A. Li, "DPWLRU: An Efficient Buffer Management Policy Based on Dynamic Page Weight for Flash Memory in Cyber-Physical Systems," vol. 7, pp. 58810-58821, 2019.

[7] Intel, Understanding the Flash Translation Layer (FTL) Specification, Viewpoints (December 1998).

[8] S.-Y. Park, D. Jung, J.-U. Kang, J.-S. Kim, and J. Lee, “CFLRU: A replacement algorithm for flash memory,” in Proc. International Conference on Compilers, Architecture and Synthesis for Embedded Systems, Seoul, Korea, pp. 234-241, Oct. 2006.

[9] Y.-S. Yoo, H. Lee, Y. Ryu, and H. Bahn, “Page replacement algorithms for NAND flash memory storages,” in Proc. International Conference on Computational Science and its Applications, Kuala Lumpur, Malaysia, pp. 201-212, Aug. 2007.

[10] H. Jung, H. Shim, S. Park, S. Kang, and J. Cha, “LRU-WSR: Integration of LRU and writes sequence reordering for flash memory,” IEEE Trans.Consum. Electron., vol. 54, no. 3, pp. 1215-1223, Aug. 2008.

[11] 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.

[12] Z. Li, P. Jin, X. Su, K. Cui, and L. Yue, “CCF-LRU: A new buffer replacement algorithm for flash memory,” IEEE Trans. Consum. Electron. vol. 55, no. 3, pp. 1351-1359, Aug. 2009.

[13] X. Jin, S. Jung, Y. H. Song, SEPL: Smart evicted page list buffer for NAND flash storage system, Proceedings - 10th IEEE International Conference on Computer and Information Technology, CIT-2010, 7th IEEE International Conference on Embedded Software and Systems, ICESS-2010, ScalCom-2010 (Cit) (2010) 1687–1694. http://dx.doi.org/10.1109/CIT.2010.297 doi:10.1109/CIT.2010.297.

[14] Jin, P., Ou, Y., Haerder, T., Li, Z.: ADLRU: An Efficient Buffer Replacement Algorithm for Flash-based Databases. Data and Knowledge Engineering (DKE) 72, 83–102 (2012)

[15] Guangxia Xu, Lingling Ren and Yanbing Liu, “Flash Aware Page Replacement Algorithm”, Mathematical Problems in Engineering, Hindawi Publishing Corporation, volume 2014.

[16] M. Lin, Z. Yao and J. Xiong, "History-aware page replacement algorithm for NAND flash-based consumer electronics," in IEEE Transactions on Consumer Electronics, vol. 62, no. 1, pp. 23-29, February 2016, doi: 10.1109/TCE.2016.7448559.

[17] C. Li, D. Feng, Y. Hua, W. Xia, F. Wang, Gasa: A new page replacement algorithm for NAND flash memory, 2016 IEEE International Conference on Networking Architecture and Storage, NAS 2016 - Proceedingshttp:// x.doi.org/10.1109/NAS.2016.7549403 doi:10.1109/NAS.2016.7549403.

[18] Lezwon Castellino and Joy Paulose, “AFAPRA: Adaptive Flash Aware Page Replacement Algorithm” in International Journal of Applied Engineering Research, ISSN 0973-4562 Volume 13, Number 5 (2018) pp. 3128–3138

[19] Q. Huang, R. Chen, M. Lin, C. Yang, Q. Chen and X. Li, "Clean-First Adaptive Buffer Replacement Algorithm for NAND Flash-Based Consumer Electronics," 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 2019, pp. 1217-1223, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00173.

[20] N. Nethercote and J. Seward, “Valgrind: a program supervision framework,” Electronic Notes in Theoretical Computer Science, vol. 89, no. 2, pp. 47–69, 2003.

[21] X. Su, P. Jin, X. Xiang, K. Cui, L. Yue, Flash- DBSim: A simulation tool for evaluating flash-based database algorithms, in: Proceedings - 2009 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009, 2009, pp. 185– 189. http://dx.doi.org/10.1109/ICCSIT.2009.5234967 doi:10.1109/ICCSIT.2009.5234967.

[22] John S. Bucy, Jiri Schindler, Steven W. Schlosser, Gregory R. Ganger,(2008).The DiskSim Simulation Environment Version 4.0 Reference Manual.

[23] Carniel, A. C.; Ciferri, R. R.; Ciferri, C. D. A. FESTIval: A versatile framework for conducting experimental evaluations of spatial indices. MethodsX, vol. 7, 2020, 100695, 2020.

[24] Q. Wei, C. Chen, and J. Yang, “CBM: A cooperative buffer management for SSD,” in Proc. Symp. Mass Stor. Syst. Technol. (MSST), San Jose, CA, USA, 2014, pp. 1–12, doi: 10.1109/MSST.2014.6855545.

[25] J. Choi, K. M. Kim and J. W. Kwak, "WPA: Write Pattern Aware Hybrid Disk Buffer Management for Improving Lifespan of NAND Flash Memory," in IEEE Transactions on Consumer Electronics, vol. 66, no. 2, pp. 193-202, May 2020, doi: 10.1109/TCE.2020.2981618.