International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 05, May-2015

# CLRU : A Page Replacemnet Algorithm for NAND Flash based Electronic Devices

Datta T
Department of ECE
SJBIT, Bangalore
Karnataka, India

Assoc. Prof. Srividya P
Department of ECE
SJBIT, Bangalore
Karnataka, India

***Abstract*— The use of NAND flash memories has increased steadily in personal computers and several consumer products alike. However the data management part of the PC or other device still use management techniques developed for older magnetic disks. The change hasn't been brought forth to accommodate this new memory type.**

**In this paper, a page replacement algorithm called CLRU is proposed and implemented. This is designed primarily for devices using the flash based memory systems as a secondary storage. The CLRU is designed keeping in mind the capabilities of flash memory, owing to the improvement of data retention and reliability of the storage module. This algorithm essentially reduces the number of read and write cycles on the secondary storage while also improving the page hit ratio. This is done by dividing the pages into cold and hot based on the frequency of referencing and preferentially evicting the page with least probability of being referenced again.**

***Keywords*— *CLRU; page; algorithm; NAND Flash memory.***

## I. INTRODUCTION

The increase in use of electronic gadget's in the recent scenario has led to improvements in memory modules. In previous days the basic memory is nothing but Magnetic disk drives. The severe drawback of magnetic disk drives is due to its mechanical system, high latency and more power consumption.

Flash memory overcomes this drawback by having less latency, less power consumption and no mechanical system. Flash memory is faster than the traditional magnetic disk drives.

Existing memory management algorithms today are designed for magnetic disk. They work with good performance for a magnetic disk based device. But these data management schemes cannot be used for flash memories due to is varied characteristics.

NAND flash memory has numerous appealing highlights, for example, low power utilization, light weight, quick information access speed, high capacity, ease, etc. In addition, the limit of flash memories are expanding and its cost is diminishing.

The algorithm of the magnetic disk is designed keeping in mind that the access times are slow on the disk as there is mechanical movement of the head on the disk for both read and write. The number of hit ratios are increased while writes on the disk are more and aren't essentially damaging the disk. But in a NAND flash memory the I/O operations are not symmetric like the magnetic disk. The read latency is lesser than the write latency. So the older page replacement techniques cannot be used for flash memory.

NAND flash memory uses the out-of-place plan to manage the exceptional equipment limitation as far as read and write is concerned. In the event that the pages swapped out from principle memory to NAND flash memory are dirty, the free space of NAND flash based storage will be utilized up rapidly and garbage accumulation with high power consumption and increased latency. Also, the write operation on the flash memory is slower than read. So, the write operations on the device should be reduced while not tampering or reducing the page hit ratio.

The CLRU algorithm proposed in this paper is designed specifically for devices using NAND flash based as secondary storage. The CLRU considerably reduces the write operations by carefully considering the correct cold pages for eviction. This is called preferential eviction scheme which increases the page hit ratio.

This CLRU algorithm is compared with older existing page replacement algorithms by performing several experiments. The results show that CLRU performs better than the existing algorithms with accord to page hit ratio and number of write operations on the NAND flash memory.

The rest of the paper is organized as follows. Section II briefly reviews the hardware characteristics of NAND flash memory and compares NAND flash memory with magnetic disk. The motivation of this work is also presented in this section. Section III then discusses existing page replacement algorithms designed for NAND flash memory. The details of the proposed CLRU algorithm are presented in Section IV. Experimental results are described in Section V.

## II. BACKGROUND

### *A. Motivation*

The magnetic disk uses mechanical parts like a actuator arm to move the head throughout the disk. The page replacement algorithms today increase the page hit ratio and reduce both read and write cycles performed on the disk storage. So these algorithms work in well coherence with the magnetic disk. But the NAND flash memory architecture,

operation and performance is far different than the traditional disk. It does not have any mechanical parts, it has no seek time and read write operations have asymmetric latencies. Therefore the old algorithms cannot be used and the use of NAND flash memories demands the design of the algorithms used specifically for it as well.

### *B. NAND Flash Memory*

Flash memory characteristics while compared to magnetic mechanical disks:

(1) No seek for time. As a sort of immaculate electronic hardware, NAND flash memory will not possess any kind of mechanical segments and a long seeking time is spared.

(2) Asymmetric I/O operations. NAND flash memory shows diverse I/O exhibitions for their write, read, and erase operations. On the other hand magnetic plate indicates practically the same I/O execution for read and write operations. The input output timing performance difference on the both are shown in table.

| Basic Operation | Magnetic Disk | NAND Flash Memory |
|---|---|---|
| Read operation (2 KB) | 12.7ms | 80µs |
| Write operation (2 KB) | 13.7ms | 200µs |
| Erase operation (128 KB) | N/A | 1.5ms |

TABLE I: I/O Performance comparison between NAND flash memory and Magnetic Disk

(3) Out-of-place update. Flash memory ought to perform erase operation to erase the entire block before overwriting the information in this block. Because of the set number of erase operations permitted to every block and high vitality utilization of erase operation, repeated erase operations to the same individual block may corrupt the framework execution of NAND flash based memory. They is the reason it has the strange type of redesign plan for invalidating the first primary data, then later writing the newly arrived data to a free page.

(4) The predetermined limitation on the number of erase operations permitted to each block on the memory cell. In a scenario where the set number of erase operations of a block is surpassed beyond expectations, that particular block will experience the ill effects of several write errors.

(5) Low power utilization. Contrasted to the magnetic disk, the NAND flash memory consumes lesser power. The development of the NAND flash memory gives effective backing for building green data centers and database management frameworks with low power consumption.

There are two sorts of flash memory, as illustrated in Fig. 1. NOR flash memory and NAND flash memory. NOR flash memory and NAND flash memory are very different in operation. Because of the low irregular access latency and execute-in place highlight, NOR flash memory is regularly used to store system code and execute program directly. Since NAND flash memory is capable of providing the substantial capacity with lower price, numerous customer gadgets are furnished with NAND flash memory as secondary storage device instead of using traditional magnetic disks.

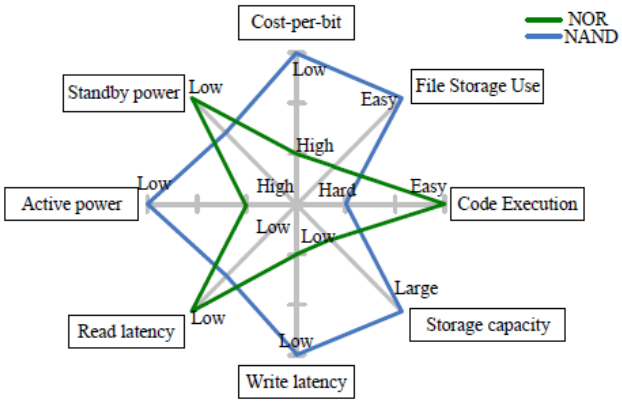


Fig. 1. NOR and NAND flash memory cost and performance comparison.

As represented in Fig. 2, every NAND flash memory chip is generally included a majority of blocks and every block is made out of a few pages. Also refer Fig.3. For example, a 128 MB NAND flash memory chip contains 1024 blocks and every block contains 8 pages. Every page has two sections, which are the main area and the spare area, separately. The main area is 2 KB which holds user data, while the span of spare area is 64 B and it is utilized for putting away data about storage checksum and page addresses. NAND flash memory gives three fundamental operations, which are write, read, and erase, individually. The read and write operation can be performed on a page individually, while the erase operation can be performed on an entire block.

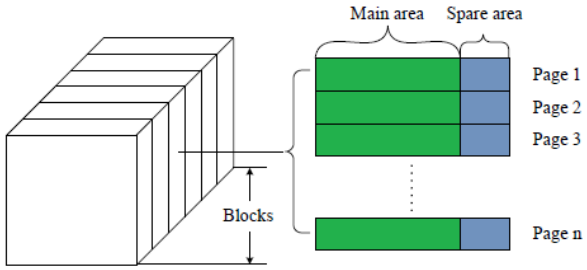


Fig. 2. The architecture of NAND flash memory.

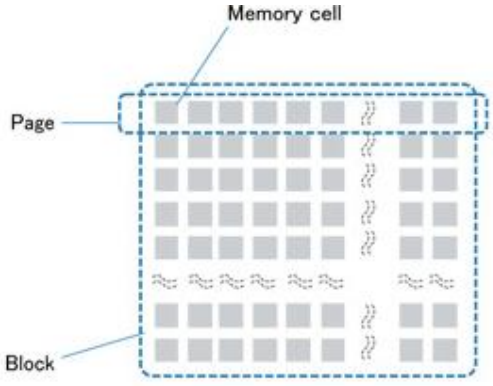


Fig. 3. NAND flash pages and blocks representation

The NAND type flash memory performs writing in page units as shown in the fig. It erases in block units. Because erasing affects a wide range of data in one quick operation, the memory is called "flash" memory.

## III. Related Work

Today flash memory is not a very tremendously new thing in the market. More and more consumer electronic devices are being designed with flash memory incorporated. Hence there should also be certain primitive page replacement algorithms created for it. Although not very mature in operation, we will look at the few LRU modified examples made for NAND flash made for today's Operating Systems, whilst giving credit to the people who designed them.

1. CFLRU ( Clean First Least Recently Used) by Park et al - This the first page replacement algorithm ever proposed for the NAND flash memory. This was initially designed in consequence of the unique hardware constraints posed by the NAND flash storage memory.

This algorithm is based off of the original LRU algorithm and in enhanced to fit the needs of the NAND flash device. This introduces the concept cold and hot pages on the existing Clean and Hot pages.

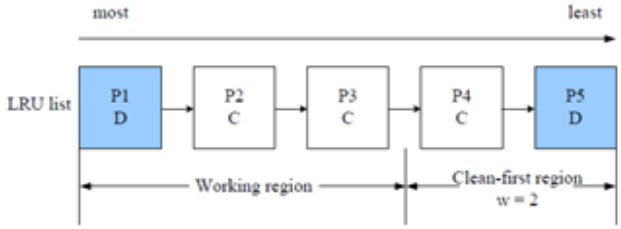


Fig. 4. The example of CFLRU.

This also introduces the concept of preferential eviction of clean pages over dirty pages apart from LRU order. Referring to the Fig. 4. The CFLRU holds a page list. This is arranged in the LRU order. Here 'C' refers to clean page and 'D' refers to dirty page. The entire page list is divided onto to portions, which being clean-first and working. This is done to make way for preferential eviction and prevent the degradation of the page hit ratio while retaining the integrity of the memory. The Working region holds the latest data which is the most recently used or referenced and which may be referenced sooner in the future. ON the other hand the cold-first region holds the older data which is least recently used and which may not referenced anywhere in the near future. The window size of the clean-first region will depend on the regulation based on the page hit ratio.

The working goes as follows. Every new data is entered in the LRU format as explain in the previous section. The first page frame P1 will always hold the latest data and the last page frame will hold the oldest referenced data. Now, once the page list is full, if the next reference is present in the page list all of the data will get rearranged. If the new reference is not present in the page list and the page list is already full , then eviction has to be done.

The eviction will start from the clean-first region but not the LRUs page frame. Although the page frame P4 has higher priority than the page frame 5 which has dirty data, the algorithm will first evict P4 instead of P5 and if P5's data is not referenced again then it is omitted. This is done to reduce the number of evictions as eviction will result in a write operation the NAND flash memory.

2. Three modified versions of CFLRU by Yoo et al -

The existing concept of CLRU was modified by Yoo et al. He proposed 3 efficient modifications on the CFLRU to further improve its hold on operation on NAND flash memory.

The first one CFLRU/C -

This algorithm in its essence works exactly similar to the original CFLRU algorithm as per the LRU order within the clean-first region. It first looks for any clean pages existing in the clean first region and evicts it. But if there are no clean pages in the first region then it chooses a dirty page within this region itself which has the lowest frequency of access as the victim for eviction.

- The second one CFLRU/E -

This algorithm also works just like other the original CFLRU and CFLRU/C algorithm as per the LRU order within the clean-first region. It scans for clean pages in the first region. If no clean pages are found it will select a dirty page belonging to the block or page frame which has the lease erase count as the victim for that round of eviction.

- The third DL-CFLRU/E -

In this algorithm there are two page lists clean page list and dirty page list. The DL here refers to Dirty List. the dirty page list is further divided into two regions. Firstly the DL-CFLRU/E checks the clean page list for eviction in the LRU order. If the clean page list is empty, it goes to the dirty page list second region being the window at the least recently used position. Here within the window the block which has the lowest erase count is chosen for eviction. Finally if that window is also empty then the rest of the dirty pages are evicted in the LRU order.

Referring to the Fig. 5. P1 to P5 are clean, P8 to P12 are dirty within which P11 and P12 form the window. First P1 to P5 are checked for pages. If empty then the window is checked. Here both P11 and P12 are filled P11 is chosen as victim for eviction even though P11 has higher priority to be referenced in the future than P12. This is done to extend the lifetime of NAND flash memory.

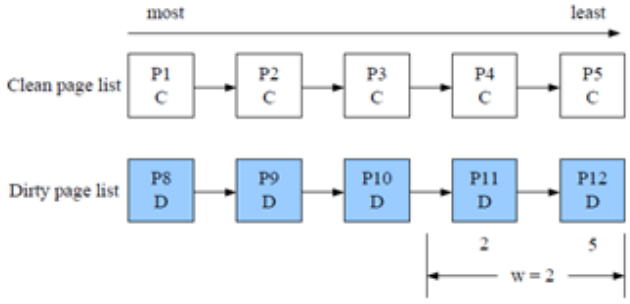


Fig. 5. The example of DL-CFLRU/E.

3. LRU-WSR (Least Recently Used Write Sequence Reordering) by Jung et al -

This is a modified version of the LRU introduced with Write Sequence Reordering (WSR) strategy for NAND flash memory. The LRU-WSR algorithm maintains a single page list in the LRU order similar to the original LRU algorithm. But any dirty pages in the list will have a cold flag which may or may not be set as per the situational operation. The identification of Cold or Hot of a dirty page is taken care of by a cold detection algorithm. A dirty page may be cold, if that is the least referenced page in the list and its cold flag is set. And a dirty page may be hot if that page is page is referenced again and the cold flag on it is cleared.

```
Algorithm LRU-WSR
Input:LRU list = the page list linking the pages by the LRU order;
Output:The victim page for evicting
        for (each page in the LRU list)
                if (the LRU page is a clean page)
                        victim page = the LRU page;
                end if
                else if (the cold flag of LRU page is set)
                        victim page = the LRU page;
                end else if
                else
                        set the cold flag of LRU page and move it to
the MRU position of LRU list;
                end else
        end for
        return victim page;
End LRU-WSR
```

Fig. 6. The pseudo-code of LRU-WSR.

The Fig. 6. shows the pseudo-code for the LRU-WSR algorithm. During operation the algorithm kicks in, if there is no free frame left in the list. The first if code checks for the least referenced clean page and evicts it unconditionally. If the page is dirty then it proceeds further to check for the cold flag on it. If the cold flag on it is set, then it is considered as a cold page and is evicted. On the other hand if the cold flag is not set then it is considered a hot page and is left alone. This is later moved to the MRU position and later setting its cold flag and gives it a second chance to be i the main memory. In this way the clean pages need not always evicted heavily which may degraded the page hit ratio. Using this avoids the degradation.

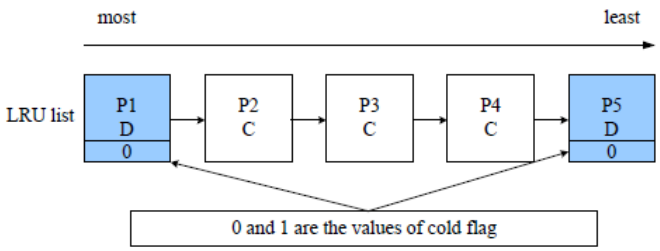


Fig. 7. The example of LRU-WSR.

The Fig. 7. shows the example of LRU-WSR algorithm. First the Least recently used page is checked. In this case the page P5, but it's cold flag is not set and hence this will be moved to the MRU position of the page list and later the flag on it is set to indicate it as cold. Then the next least recently used page P4 is selected and check if dirty or clean, in this case being clean it is evicted without issues. In this way Cold flags are used to delay the eviction on the dirty pages to reduce the unwanted evictions which will affect the NAND flash memory.

## IV. The Design And Implementation of CLRU

In this section, the system architecture of NAND flash based consumer electronics and the proposed CLRU algorithm will be presented.

### *A. System Architecture of NAND Flash-based Consumer Electronics*

As illustrated in Fig. 8, NAND flash-based client electronics carries with it 3 elements, that are the OS, flash translation layer, and NAND nonvolatile storage chip. so as to cover the various hardware characteristics of NAND nonvolatile storage from magnetic disk, a flash translation layer is deployed between NAND nonvolatile storage chip and traditional in operation systems. Flash translation layer emulates the NAND nonvolatile storage chip as a standard block device and traditional in operation systems will operate the NAND nonvolatile storage chip swimmingly.

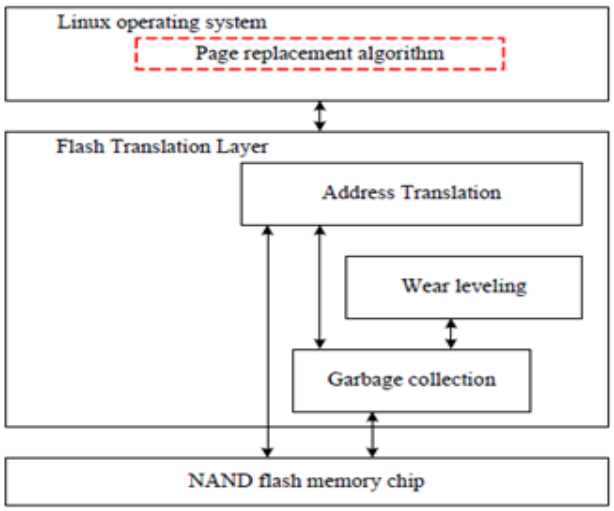


Fig.8. System architecture of NAND flash-based consumer electronics.

The page substitution calculation is an essential plan in the working frameworks. In this framework building design, the page substitution calculation is intended to decrease the quantity of write operations to NAND flash based capacity gadget and hold a high page hit ratio.

### *B. The Proposed CLRU Algorithm*

The basic design analysis of the NAND flash memory shows that the damage on wearing out of the m memory is more on every write cycle rather than a read cycle. The energy consumption and latency of the write operation is also more than read operation. If more intense write cycles are performed on the memory then garbage collection will kick in to free up the space which give rise to even more energy wastage.

The existing page replacement algorithms that we have discussed earlier are all based on the clean first strategy. They evict the clean regions preferentially. But evicting the clean pages isn't always a good thing. This could cause a lot of unwanted avoidable page faults. Thus excessive eviction of clean pages will hurt the page hit ratio to a great extent. This in-turn will cause excess latency in the working of the Operating System and the response times go downhill. In light of this awkward situation the page replacement algorithm named CLRU was proposed for all the electronic consumer products which incorporate a NAND flash memory storage as a secondary storage device.

Let us now dive into the concept itself. CLRU is again a modification of the LRU algorithm. But it is far more different in operation and highly efficient when compared to the others as far s needless eviction, page hit ratio and write cycles go.

CLRU holds and maintains two page lists. They are the Clean and the Dirty page list respectively. These are again divided into two separate regions, the cold-first region and the hot region. Remember that this is cold-first and not the older clean-first which far different in meaning and terms of operation. The cold region will contain pages that are older and least recently referenced while the Hot region contains the pages that will most likely be referenced in the near future of operation. The sizes of the region's windows are arbitrary and the cold detection algorithm decides alignment or transitions from the hot to cold region.

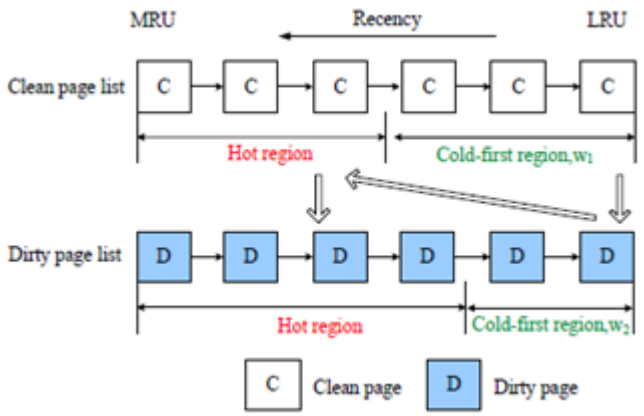


Fig. 9. The example of CLRU.

Consider the Fig. 9. which shows the example of the organization in the Cold Least Recently Used algorithm. The two lists clean list and dirty list have a cold-first window. The size of the cold-first window of the clean region is W1. Similarly the size of the cold-first window of the dirty region is W2.

There are a total of 4 regions defined here- Cold clean, Cold dirty, Hot clean and Hot dirty. All of which are ordered in the LRU scheme.

The cold detection algorithm is the one which decides whether a page is to be left hot or mode into cold. The cold detection algorithm will decide the coldness of a page of sorts via the normalized values of the elapsed times. Let us now consider the example of the clean page list to implement the cold detection algorithm. The cold detection algorithm calculates the normalized values of the elapsed times since it was being referenced the last time. This is done for each page. Let us assume two pages as 'i' and 'j'. The calculation of normalized value of elapsed time goes as shown in the equation:

$$NVET_i^{'} = \frac{NVET_i - \min_{j \in m}^{m}(NVET_j)}{\max_{j \in m}^{m}(NVET_j) - \min_{j \in m}^{m}(NVET_j)}$$

Here $NVET_i$ is the normalized value of the elapsed time of page i since it was referenced last time. And also $NVET_j$ is the normalized value of the elapsed time of page j since it was referenced last time. Where 'm' being the total number of pages in the page list.

If the normalized value of elapsed time page i since it was referenced last time is bigger than or equivalent to the average normalized value of the elapsed times of every other the page, it will be referred to as a cold page and it will be moved and held in the cold-first region of the clean page list. Similarly If the normalized value of elapsed time page i since it was referenced last time is lesser than that of the average normalized value of the elapsed times of every other the page, it will be referred to as a hot page and it will be held in the hot region of the clean page list itself and will not be touched for now.

It will remain in the hot region till its NVET has reached a peak value. The Cold detection calculation on the dirty page list will also work in the similar fashion. The algorithm may be designed with NVET formulation on cold only while keeping the Dirty list in LRU so reduce cycles of operation, this is a optimization choice. But we still have to keep in mind that all of them or LRU ordered and reordered as per referencing and re-referencing.

```
Algorithm CLRU
Input:Clist = clean page list;
      Dlist = dirty page list;
      w1 = window size for cold-first region in the clean page list;
      w2 = window size for cold-first region in the dirty page list;
      L1 = size of clean page list;
      L2 = size of dirty page list;
Output:The victim page for evicting
      for (int i = 0; i <= w1; i++)
            victim page = the page with the lowest access frequency in
cold-first region in the Clist;
            return victim page;
      end for
      for (int i = 0; i <= w2; i++)
            victim page = the page with the lowest access frequency in
cold-first region in the Dlist;
            return victim page;
      end for
      for (int i = 0; i <= L1-w1; i++)
            victim page = the least recently referenced page in host
region in the Clist;
            return victim page;
      end for
      for (int i = 0; i <= L2-w2; i++)
            victim page = the least recently referenced page in host
region in the Dlist;
            return victim page;
      end for
End CLRU
```

Fig. 10. The pseudo-code of CLRU

The pseudo-code for CLRU is shown in the Fig. 10. The victim selection by CLRU goes as follows.

i. As shown in the pseudo code, at first the CLRU will scan the cold-first region of the clean page list. It will evict the page which is LRU of the lot. If this is empty it will jump to scan the next region.

ii. Li et al showed that the eviction of a hot clean page has greater adverse effect than eviction of a cold dirty page. So the second priority of page eviction would be form the cold-first region of the dirty page list which be the LRU order as well. The algorithm enters this region only when the clean cold-first region is empty.

iii. Next the dirty cold region is empty, the third priority would be the hot region of the clean page list and selects the least recently referenced page as a victim page.

iv. Finally if there are not pages even in the hot clean region CLRU switches to scan the hot dirty region which is the fourth and the last region of priority. These are the dirty pages whose eviction is delayed so they may be omitted. All of these follow the least referenced strategy.

In this way the division of regions based on the cold detection algorithm and preferentially evicting the pages by delaying unwanted hot dirty page evictions, the number of write operations on the NAND flash memory can be reduced while also increasing the page hit ratio of the algorithm.

## V. PERFORMANCE EVALUATION

In order to investigate the effectiveness of the proposed CLRU algorithm, the performance of the proposed CLRU algorithm is compared with those of LRU, CFLRU, DLCFLRU/ E, and LRU-WSR. For experiments, a reference board is used. It is equipped with a 533 MHz microprocessor, 96 KB SRAM, 128 MB SDRAM, and 256 MB NAND flash memory chip. A series of experiments are conducted by executing two representative applications under the Linux running on the reference board, which are type 1 and 2, respectively. Type 1 is a popular mp3 player for playing music and type 2 is a document editor for processing text. The characteristics of type 1 and 2 are illustrated in TABLE II. The data in TABLE II shows that type 1 is a write intensive application while type 2 is a read intensive application .

TABLE II : The Characteristics of Type 1 and Type 2

THE CHARACTERISTICS OF TYPE 1 AND 2

| Workload | Memory Footprint | Memory References |
|---|---|---|
| Type 1 (mp3 player) | 10.58MB | read : write = 1 : 5.02 |
| Type 2 (document editor) | 13.01MB | read : write = 12.15 : 1 |

The execution measurements that the trials are performed to test are the page hit ratio and the no. of write operations. It is accepted that the buffer size is 4 MB. Since the extent of window in the page rundown has an impact on the page hit ratios of CFLRU, and DL-CFLRU/E, the window sizes of CFLRU and DL-CFLRU/E are situated to 0.2.

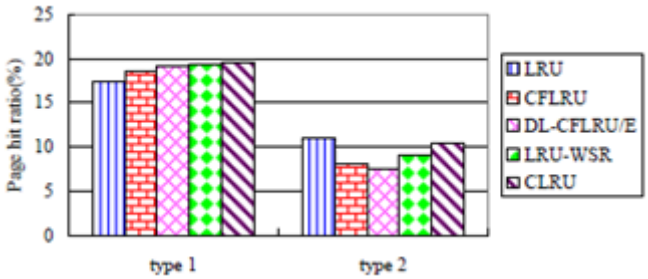


Fig. 11. Page hit ratios of several page replacement algorithms

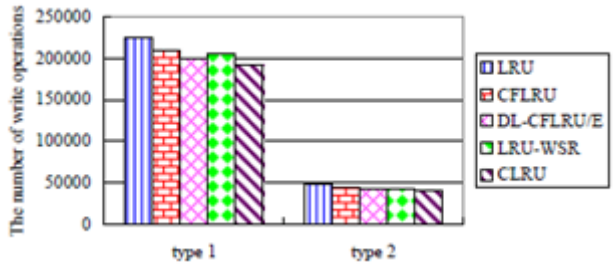


Fig. 12. The number of write operations for several page replacement algorithms.

The page hit ratios of five page substitution calculations are demonstrated in Fig.11. For type 1, the page hit ratios of the page trade calculations intended for NAND flash memory are higher than the first LRU calculation that is streamlined for attractive circle. The reason is that type 1 is a write escalated application and all the page substitution calculations aside from the first LRU calculation defer the eviction of dirty pages. The expense of evicting a hot clean page is higher than that of evicting a cold dirty page and CLRU evicts the cold

dirty pages rather than the hot clean pages when there are no cold pages inside of the cold-first region of the clean page list. Also, CLRU considers the entrance recurrence of every page inside of the cold-first regions of the clean page rundown and dirty page list. Subsequently, the proposed CLRU algorithm demonstrates the most noteworthy page hit ratio. For type 2, all the page swap calculations intended for NAND flash memory demonstrate the lower page hit ratios than the first LRU calculation, The reason is that type 2 is a read concentrated application and all the page trade calculations intended for NAND flash memory evicts the clean pages specially. Be that as it may, CLRU demonstrates the higher page hit ratio than other page swap calculations intended for NAND flash memory. Fig. 12 demonstrates the quantity of write operations to NAND flash memory when performing the five page substitution calculations. Since the proposed CLRU calculation defers the eviction of cold dirty pages, it can be seen that CLRU causes the minimum write operations to the NAND flash-based customer hardware.

## VI. CONCLUSION

This paper introduces an effective page replacement algorithm called CLRU, for NAND flash-based consumer hardware. CLRU keeps up to two page records by the LRU request, specifically the clean page list and the dirty page list. The two are separated into the cold-first region and hot region. CLRU decreases the quantity of write operations by deferring the eviction of cold dirty pages properly and also enhances the page hit ratio by evicting the cold pages in the driving cold first regions specially and considering the entrance recurrence of cold pages in the cold first regions. Through different tests, trial results demonstrate that the proposed CLRU calculation is superior to anything existing page trade calculations intended for NAND flash memory as far as page hit ratio and the quantity of write operations.

## REFERENCES

[1] G. Lawton, “Improved flash memory grows in popularity,” Computer, vol. 39, no. 1, pp. 16-18, Jan. 2006.

[2] H.-L. Li, C.-L. Yang, and H.-W. Tseng, “Energy-Aware Flash memory management in virtual memory system,” IEEE Trans. VLSI Syst., vol. 16, no. 8, Aug. 2008.

[3] S. Park and S. Y. Ohm, “New techniques for real-time FAT file system in mobile multimedia devices,” IEEE Trans. Consum. Electron., vol. 52, no. 1, pp. 1-9, Feb. 2006.

[4] S. Ahn, S. Hyun, T. Kim, and H. Bahn, “A compressed file system manager for flash memory based consumer electronics devices,” IEEE Trans. Consum. Electro., vol. 59, no. 3, pp. 544-549, Aug. 2013.

[5] E.-J. O’Neil, P.-E. O’Neil, and G. Weikum, “The LRU-K page replacement algorithm for database disk buffering,” SIGMOD Rec., vol. 22, no. 2, pp. 297-306, Jun. 1993.

[6] Debabrata Swain, Bancha Nidhi and GITA Bhubaneswa, "Analysis and Predictability of Page Replacement Techniques towards Optimized Performance," IRCTITCS., vol. 16,pp 13-15, 2011.

[7] S. Jiang and X. D. Zhang, “LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance,” Perform Eval. Rev., vol. 30, no. 1, pp. 31-42, Jun. 2002. G. Xu et al.: CLRU: A New Page Replacement Algorithm for NAND Flash-based Consumer Electronics 43

[8] T. Johnson and D. Shasha, “2Q: A low overhead high performance buffer management replacement algorithm,” in Proc. 20th International Conference on Very Large Data Bases, Santiago, USA, pp. 439, Sep. 1994.

[9] Y. Zhou and J. F. Philbin, “The multi-queue replacement algorithm for second level buffer caches,” in Proc. 2001 USENIX Annual Technical Conference, Boston, USA, pp. 91-104, Jun.2001.

[10] 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.

[11] Kaveh Samiee, "A Replacement Algorithm Based on Weighting and Ranking Cache Objects," International Journal of Hybrid Information Technology, Vol.2, No.2, April, 2009.

[12] 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.

[13] 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.

[14] Hyejeong Lee and Hyokyung Bahn, "Page Replacement for Write References in NAND Flash Based Virtual Memory Systems," Journal of Computing Science and Engineering,Vol. 8, No. 3, September 2014.

[15] J. Kim, J. M. Kim, S.-H. Noh, S.-L. Min, and Y. Cho, “A space-efficient flash translation layer for compact flash systems,” IEEE Trans. Consum. Electron., vol. 48, no. 2, pp. 366-375, May 2002.

[16] A. Gupta, Y. Kim, and B. Urgaonkar, “DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings,” ACM SIGPLAN Not., vol. 44, no. 3, pp. 229-240, Mar. 2009.

[17] Anvita Saxena, "A Study of Page Replacement Algorithms," International Journal of Engineering Research and General Science Volume 2, Issue 4, June-July, 2014.

[18] D. Liu, W. Yi, Z. Qin, Z. Shao, and Y. Guan, “A space reuse strategy for flash translation layers in SLC NAND flash memory storage systems,” IEEE Trans. VLSI Syst., vol. 20, no. 6, pp. 1094-1107, May 2012.

[19] H. Jo, J.-U. Kang, S.-Y. Park, J.-S. Kim, and J. Lee, “FAB: flash-aware buffer management policy for portable media players,” IEEE Trans. Consum. Electron., vol. 52, no. 2, pp. 485-493, May 2006.