Journal of Engineering and Applied Sciences 14 (15): 5072-5077, 2019
ISSN: 1816-949X

Hot/Cold Based Replacement Algorithm for Flash Memory Buffer Management

Jeong-Joon Kim
Department of Computer Science and Engineering, Korea Polytechnic University,
15073 Gyeonggi-do, Siheung-si, Korea
jeong-joon kim +82-31-8041-0532

Abstract: To improve buffer performance, an existing buffer replacement algorithm based on hard disk drive is proposed and many studies about buffer replacement algorithm based on flash memory which consider characteristic of flash memory have been conducted recently. When existing buffer replacement algorithm based on flash memory select victim page, it considers reference status and don't considers reference frequency or when it takes into account reference recency, there are disadvantages that operate speed is slow because it consider elapsed time. So, to solve the problem that has existing buffer replacement algorithm this thesis divide page into 6 groups and when proposed algorithm select victim page, it consider reference page frequency and page recency.

Key words: Buffer replacement algorithm, flash memory, hot/cold page, AD-LRU, elapsed time, speed

INTRODUCTION

The buffer replacement algorithm for hard disks wanted to maintain the high buffer hit ratio because the speed of the read operation and write operation was equal (Jiang *et al*., 2005; Jiang and Zhang, 2002; Shasha and Johnson, 1994; O'neil *et al*., 1993). However, the buffer replacement algorithm for flash memory should additionally account for not only the hit ratio but also the writing operation, owing to the difference in reading computation and write computation (O'neil *et al*., 1993; Jung *et al*., 2008; Li *et al*., 2009; Park *et al*., 2006). Applying buffer replacement algorithms for hard disks to flash memory is not a desirable method. In addition, the buffer replacement algorithm for the existing flash memory is selected by selecting or choosing a reference time information, referring to the referenced time information for page cleaning and the latest page of the page when selecting the replacement page (Lin *et al*., 2014; Li *et al*., 2008). However, this information alone is likely to result in incorrect replacement of the replacement page and there is also a possibility that the hit ratio will decrease.

Thus, in this thesis when selecting a replacement target page, the algorithm presented to the specialized algorithms relied on the hardware properties of the flash memory rather than the existing algorithm, taking advantage of the number of pages referenced in the pages of the page as well as the existing algorithm which was used to refer to the pages of the referenced page as well as the existing algorithm.

Literature review

Buffer replacement algorithms based on flash memory: The buffer cache stores a portion of the entire disk block to reduce the physical I/O request. Since, the size of the buffer cache is relatively small relative to the entire disk, frequent replacement of data occurs and the buffer replacement algorithm is needed to efficiently utilize it.

A buffer replacement algorithm based on the hard disk that has traditionally been used as a storage device has the same buffer performance as the hard disk speed and the high-write logic of the hard disk, many hard disk based buffers replacement algorithms have been proposed to replace pages based on the latest (recency) or frequency (frequency) of page references in the buffer. On the other hand, flash memory can replace pages at lower cost because the write operation speed is slower than the read operation because the data in the buffer is replaced by replacing the data in the buffer rather than replacing the data in the buffer that causes the write operation to change. As such flash memory has been proposed for buffering algorithms that consider the cost of writing.

AD-LRU; Adaptive-Double LRU: The AD-LRU divides the buffer cache into two lists of hot LRUs and cold LRUs to dynamically adjust the size of the buffer list to manage the replacement pages (Jin *et al*., 2012). Figure 1 shows an example of selecting a replacement page for AD.

The LRU used represents the oldest page area and MRU represents the most recently used page area. In AD- LRU, the cold LRU and the hot LRU are dynamically managed. If a page within the cold LRU is referenced by a processor, the reference bit changes to true, the cold LRU size decreases and the hot LRU size increases. The corresponding page is then inserted into the hot LRU of the hot LRU. Additionally, if a page within a hot LRU is

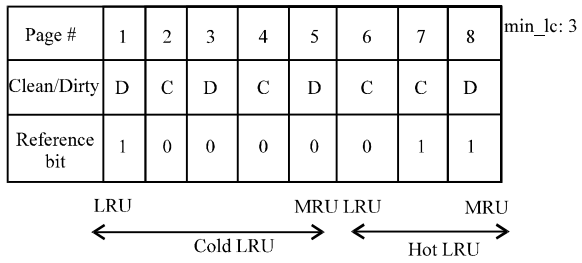

Fig. 1: Example of AD-LRU

referenced by a processor, the referenced page changes to true and is inserted into the hot LRU of the hot LRU. If the cold LRU list is larger than the lc value, if the cold LRU list is larger than the lc value, the cold LRU list is selected for the cold LRU list and the cold LRU list size smaller than the min value of the cold LRU list selects the replacement page for the hot LRU list. If the current lc value is 3 and the cold LRU list size is 5, select the replacement target page for the cold LRU list.

If a clean page exists on the selected list, select the oldest clean page in the buffer list as the replacement page. If the clean page does not exist, select the oldest dirty page as the replacement candidate page. If the reference bit on this page is changed to false, then change to false, then move the old dirty pages to the MRU page and then select the replacement page for the replacement page. If the reference bit on the candidate page is false, finally, select the replacement target page as the replacement target page. Dirty pages selected as replacement pages produce write operations when replacing pages and insert new pages into empty pages. If the replacement page was present in the cold LRU list, insert the new page into the hot LRU list and if present, decrease the hot LRU to the hot LRU list and increase the size of the cold LRU list to insert the new page in the cold LRU list of the cold LRU list.

Because the AD-LRU takes account of both reference times as well as reference times, it has the advantage of choosing a replacement page based on more information than other buffer replacement algorithms considering conventional reference times. However, if the page referenced to the referenced page is referenced only 2 times, the disadvantage is that it is not possible to calculate the correct frequencies because the hot LRU manages the page.

MATERIALS AND METHODS

HCSA (Hot-Cold Separation Algorithm): The efficient buffer replacement algorithm proposed in this study classified the pages in the buffer into six groups according to three criteria in order to select replacement pages. The three criteria for classifying pages are as follows. The first criterion is the clean page dirty page of the page. If the data in the page in the buffer is not changed by the processor, it is regarded as a clean page and if it is changed, it is regarded as a dirty page. Next, the second criterion is subdivided into two types of dirty pages, namely, a full dirty page and a partial dirty page.

Finally, it is the page's hot page. Pages in the buffer that are frequently referenced by the processor are called hot pages and pages that are not frequently referenced are called cold pages. Hot pages and cold pages are distinguished using a hot cold-sorting algorithm. In this study, according to the above three criteria, the pages are classified into 6 categories as hot full dirty, hot partial dirty, hot clean, cold full dirty, cold partial dirty and cold clean (Fig. 2-4).

According to the replacement priority, the first priority is to determine whether the page is cold hot, since, the first to third replacement priority is a cold page and the fourth to sixth are hot pages. Next, we distinguish between clean and dirty pages. Finally, if the page is a dirty page, we divide the dirty page into two types of pulsatile pages and partial dirty pages.

Therefore, it is most important to determine whether or not to select a hot key for selecting a replacement target page. In this study, in order to determine whether cached hot, the reference time and the reference count are considered together with the time remaining after the page is loaded into memory and the reload frequency.

The normalized value for a reference time of an arbitrary page is obtained by dividing the difference obtained by subtracting the minimum value of the pages in the buffer by the difference between the maximum value and the minimum value of the pages in the buffer. For example, Fig. 5, the normalized value of p2 is calculated by subtracting 12:10 from 12:40 min the minimum value of 12:10 min the 12:50 min 4 = 0.75.

The memory duration refers to the time that the page was held in memory for the duration of the program execution. For example, suppose a p2 page is loaded three times in memory. If it is loaded for 2:10 min on the first load, 1:40 min on the second load and 1:20 min on the third load, the total sum is 5:10 min It counts as lasting in memory.

The consideration of memory duration is also calculated by normalizing the duration which is the same as the reference time. For example, the normalized value of p2 has a duration of 5:10 which is the shortest duration of the pages in the buffer at 6:00. The value 0.75 divided by the value obtained by subtracting 2:40 is the normalized value of the page p2.

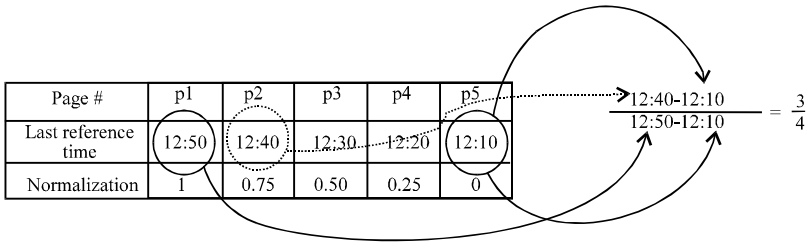

Fig. 2: Example of reference time normalization

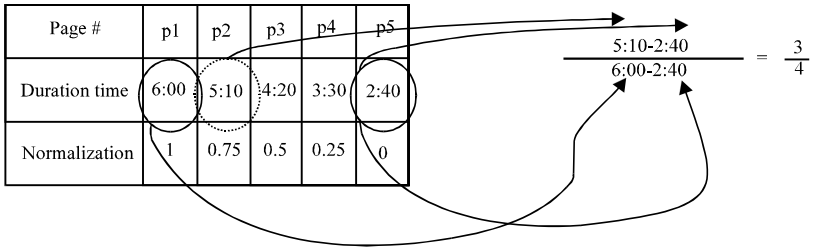

Fig. 3: Example of duration time normalization

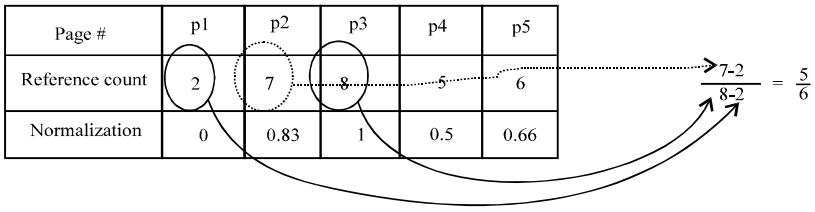

Fig. 4: Example of reference count normalization

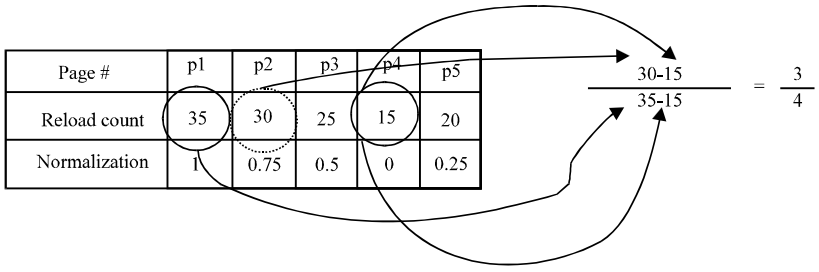

Fig. 5: Example of reload count normalization

The consideration of the reference frequency is also calculated by normalizing the reference frequency which is the same method as the reference time. For example, the normalized value of p2 is a value obtained by subtracting 2 which is the reference number of the pages in the buffer from reference No. 7 of p2 and subtracting 2 which is the smallest number of reference times from 8, 0.83 is the normalized value of the page p2 (Fig. 5).

The number of memory reloads refers to the number of times a page has been reloaded into memory during the program execution period. The consideration of the number of memory reloads is also calculated by normalizing the number of reloads which is the same as the reference time. For example, the normalization value of p2 is 15 which is the smallest number of reloads in the buffer at the number of reloads of 30 at p2. The value 0.75 divided by the subtracted value becomes the normalized value of the page p2.

In this study, we propose a method that uses a reference count, a reference time, a duration time and a reload count to calculate a normalized value using a reference time of a specific page, a normalized value using a reference count of a specific page. And the number of times the page was re-entered into the memory can be expressed as a formula and the terminology for describing the page is shown in Table 1. The equation for obtaining

Page #	p1	p2	p3	p4	p5
T_{pi}	1	0.75	0.5	0.25	0
C_{pi}	0	0.83	1	0.50	0.66
D_{pi}	1	0.75	0.5	0.25	0
R_{pi}	1	0.75	0.5	0	0.25
0.25 (T_{pi}+C_{pi}+D_{pi}+R_{pi})	0.75	0.77	0.63	0.25	0.23
HOT/COLD	Hot	Hot	Hot	Cold	Cold

Avg: $\frac{0.75+0.77+0.63+0.25+0.23}{5} = 0.526$

T_{p2}+C_{p2}+D_{p2}+R_{p2} is larger than avg. value, so, it considerd hot

T_{p2}+C_{p2}+D_{p2}+R_{p2} is less than avg. value, so, it considerd cold

Fig. 6: Example the reference time, the reference count, the duration and the reload counts togather

Table 1: Terminology for describing the hot cold classification algorithm

Terms	Descriptions
P_1	Specific page
$P = \{p_1, p_2, ..., p_n\}$	Full set of pages within the buffer list
t_{p1}	Last reference time for specific page P1
t_p	$\{t_{p1}, t_{p2}, ..., t_{pn}\}$
T_{p1}	The value of the reference time for a particular page on page P1
	$\{T_{p1}, T_{p2}, ..., T_{pn}\}$
c_{p1}	Reference count for specific page P1
c_p	$\{c_{p1}, c_{p2}, ..., c_{pn}\}$
C_{p1}	Values that have a normalized reference count on a specific page P1
C_P	$\{C_{p1}, C_{p2}, ..., C_{pn}\}$
d_{p1}	Duration time for specific page P1
d_p	$\{d_{p1}, d_{p2}, ..., d_{pn}\}$
D_{p1}	The value of the duration time for a particular page on page P1
D_p	$\{D_{p1}, D_{p2}, ..., D_{pn}\}$
r_{p1}	Reload count for specific page P1
R_p	$\{r_{p1}, r_{p2}, ..., r_{pn}\}$
R_{p1}	Values that have a normalized reload count on a specific page P1
R_p	$\{Rr_{p1}, R_{p2}, ..., R_{pn}\}$

the normalization value of the reference time of a specific page based on the term used in Table 1 can be expressed as Eq. 1:

$$T_{pr} = \frac{t_{pr} - \min(t_p)}{\max(t_p) - \min(t_p)} \quad (1)$$

Next, the Eq. 2 for obtaining the normalized value of the reference count of a specific page can be expressed as Eq. 2:

$$C_{pr} = \frac{c_{pr} - \min(c_p)}{\max(c_p) - \min(c_p)} \quad (2)$$

Next, the Eq. 3 for obtaining the normalized value of the duration time of a specific page can be expressed as Eq. 3:

$$D_{pi} = \frac{d_{pi} - \min(d_p)}{\max(d_p) - \min(d_p)} \quad (3)$$

Next, the Eq. 4 for obtaining the normalized value of the reload count of a specific page can be expressed as Eq. 4:

$$R_{pi} = \frac{r_{pi} - \min(r_p)}{\max(r_p) - \min(r_p)} \quad (4)$$

Next, we introduce the reference time, the reference count, the duration and the number of reloads together in Fig. 6.

The normalized values of a specific page p_i in the buffer may be given weight w and the weight at that time may be selected according to the importance of the user. The sum of all weights is set to one. The equation for the hot cold classification algorithm that takes into account the weight values can be expressed as Eq. 5:

$$w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi} \quad (5)$$

If the normalized values of p1 are 1, 0, 1 and 1, respectively as shown in Fig. 5, if we assume that all w values are arbitrarily assigned to 0.25, we apply 0.25×1+ 0.25×0+ 0.25×1+0.25×1 = 0.75.

When the hot-cold sorting algorithm of pages is equal to Eq. 5. when the average value of each page in the buffer calculated using (Eq. 5) is average the value of $w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi}$ compare. If the average value is larger than $w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi}$ the value of a certain page, it is considered as a hot page. If the average value is smaller than $w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi}$ the value of an arbitrary page, it is regarded as a cold page. On the basis of this, it is possible to express (Eq. 5) that a random page is regarded as a cold page and (Eq. 6) which regards a random page as a hot page:

$$w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi} \leq \frac{\sum_{i=1}^{n} w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi}}{n} \quad (6)$$

This Eq. 2 implies a cold page and average can be expressed as:

$$\frac{\sum_{i=1}^{n} w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi}}{n}$$

$$w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi} > \frac{\sum_{i=1}^{n} w_1 \times T_{pi} + w_2 \times C_{pi} + w_3 \times D_{pi} + w_4 \times R_{pi}}{n} \quad (7)$$

This Eq. 7 implies a hot page: As shown in Fig. 5, the buffers of all pages can be normalized by using Eq. 5-7, so that, hot pages and cold pages can be distinguished from each other and the pages to be replaced first can be selected.

RESULTS AND DISCUSSION

Performance evaluation: In this study, we show the performance evaluation results of the AD-LRU and the construction environment to verify the performance of the hot cold separate algorithm proposed in this study.

Performance evaluation environments: Table 2 shows the flash memory specifications for performance evaluation of this study. As shown in the table, the page size is 4, 096 bytes, the block size is 64 pages, the page read speed is 25 μsec per page, the page write speed is 220 μsec per page and the block erase speed is 1.5 msec per block.

Comparative performance evaluation: In this study, the performance is verified by comparison with the AD-LRU discussed in the related study. The test environment is shown in Table 3. Performance evaluation items include buffer hit rate, number of write operations and replacement time. Figure 6 shows the buffer hit ratio.

As shown in Fig. 6, the buffer hit rate is improved about 20% over the AD-LRU in all environments and the superiority of the hot cold separate algorithm has been verified. Figure 7 shows the buffer hit ratio.

As shown in Fig 7, the HCSA shows better performance in test 3, although, it does not show a large performance difference in the write operation. It can be interpreted as a result of reducing the total number of operations in consideration of pages to be replaced. Figure 8 shows the replacement time. As shown in Fig. 9, it can be seen that the replacement takes longer than the AD-LRU. This can be interpreted as the computational process for determining the page to be replaced is more complex than the AD-LRU.

Table 2: Flash memory specifications

Specifications	Values
Page size	4,096 byte
Block size	64 page
Page read speed	25 μsec/page
Page write speed	220 μsec/page
Block delete speed	1.5 msec/block

Table 3: Test environment

Test No.	Operation	Read/write ratio (%)
1	1,000,000	50/50
2	1,000,000	90/10
3	1,000,000	10/90

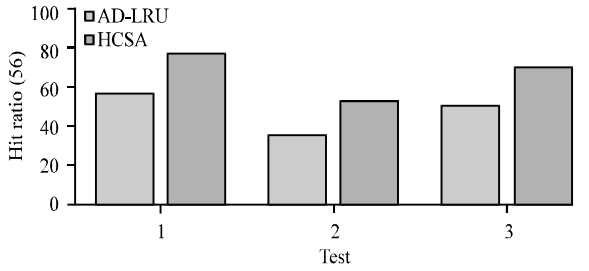

Fig. 7: Buffer hit rate

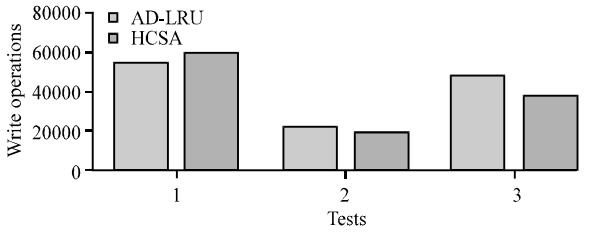

Fig. 8: Write operations

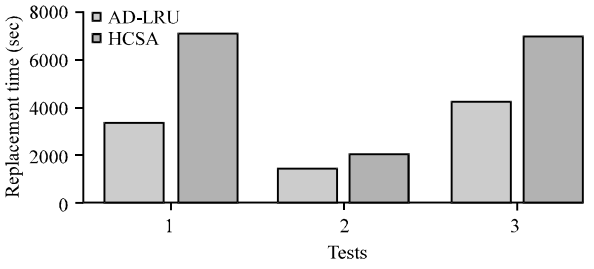

Fig. 9: Replacement time

CONCLUSION

A buffer is intended to store a large number of pages to reduce the gap between CPU and storage between CPU and storage. Buffer replacement algorithms have been proposed to improve the performance of buffers and the existing buffer replacement algorithm has been proposed on the same hard disk as the same hard disk. Thus, many studies have been conducted on buffer replacement

algorithms that take into account these flash memory characteristics, since, the existing algorithms are not suitable for other flash memory. Thus, this thesis divided the pages into six groups of studies, grouped into more detailed ones and presented the reference number and reference times together with reference times. Given the limited lifetime of flash memory, the candidate was chosen to replace the replacement page based on the number of erased.

ACKNOWLEDGEMENT

This research was supported by the National Resaerch Foundation of Korea (NRF) grant funded by the Korea Government (MSIP) (No. 2017RIA2B4011243).

REFERENCES

Jiang, S. and X. Zhang, 2002. LIRS: An efficient low inter-reference recency set replacement policy to improve buffer cache performance. Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '02) Vol. 30, June 15-19, 2002, ACM, Marina Del Rey, California, USA., pp: 31-42.

Jiang, S., F. Chen and X. Zhang, 2005. CLOCK-Pro: An effective improvement of the clock replacement. Proceedings of the USENIX Annual Conference on Technical (ATEC '05), April 10-15, 2005, USENIX Association Berkeley, California, USA., pp: 323-336.

Jin, P., Y. Ou, T. Harder and Z. Li, 2012. AD-LRU: An efficient buffer replacement algorithm for flash-based databases. Data Knowl. Eng., 72: 83-102.

Jung, H., H. Shim, S. Park, S. Kang and J. Cha, 2008. LRU-WSR: Integration of LRU and writes sequence reordering for flash memory. IEEE. Trans. Consum. Electron., 54: 1215-1223.

Li, H.L., C.L. Yang and H.W. Tseng, 2008. Energy-aware flash memory management in virtual memory system. IEEE. Trans. Very Large Scale Integr. Syst., 16: 952-964.

Li, Z., P. Jin, X. Su, K. Cui and L. Yue, 2009. CCF-LRU: A new buffer replacement algorithm for flash memory. IEEE. Trans. Consum. Electron., 55: 1351-1359.

Lin, M., S. Chen, G. Wang and T. Wu, 2014. HDC: An adaptive buffer replacement algorithm for NAND flash memory-based databases. Opt., 125: 1167-1173.

O'neil, E.J., P.E. O'neil and G. Weikum, 1993. The LRU-K page replacement algorithm for database disk buffering. ACM. Sigmod Rec., 22: 297-306.

Park, S.Y., D. Jung, J.U. Kang, J.S. Kim and J. Lee, 2006. CFLRU: A replacement algorithm for flash memory. Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, October 22-25, 2006, ACM, New York, USA., pp: 234-241.

Shasha, D. and T. Johnson, 1994. 2Q: A low overhead high performance buffer management replacement algoritm. Proceedings of the 20th International Conference on Very Large Databases, September 12-15, 1994, ACM, Santiago, Chile, pp: 439-450.