J Inf Process Syst, Vol.14, No.3, pp.631~644, June 2018
https://doi.org/10.3745/JIPS.04.0073

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Development of Flash Memory Page Management Techniques

Jeong-Joon Kim*

Abstract
Many studies on flash memory-based buffer replacement algorithms that consider the characteristics of flash memory have recently been developed. Conventional flash memory-based buffer replacement algorithms have the disadvantage that the operation speed slows down, because only the reference is checked when selecting a replacement target page and either the reference count is not considered, or when the reference time is considered, the elapsed time is considered. Therefore, this paper seeks to solve the problem of conventional flash memory-based buffer replacement algorithm by dividing pages into groups and considering the reference frequency and reference time when selecting the replacement target page. In addition, because flash memory has a limited lifespan, candidates for replacement pages are selected based on the number of deletions.

Keywords
Flash Memory, Page Replacement Algorithm, SSD

1. Introduction

Flash memory has distinct features from hard disks. Since the speed of read and write operations differ in flash memory and overwrite is impossible, a deletion operation is added to solve this problem. In addition, since flash memory has a limited number of erasure times, “Wear-Leveling” has to be considered [1,2].

“Buffer” refers to the area in which a part of the disk block is stored in the main memory to improve the system’s performance by reducing the disk’s number of I/O operations. The buffer is used to hold the data of frequently accessed storage devices. If the buffer has no free space, the replacement target page can be selected through the buffer replacement algorithm [3].

The buffer replacement algorithm for the hard disk tries to maintain the high buffer hit ratio because the read and write operations have the same speed. However, since the buffer replacement algorithm for flash memory has different speeds for read and write operations, both write operations and the Hit Ratio must be considered. Thus, applying a buffer replacement algorithm for transferring from a hard disk to flash memory is an undesirable method. In addition, the buffer replacement algorithm for existing flash memory is selected based on whether the page is clean, reference, and the reference time

Manuscript received April 12, 2017; first revision July 26, 2017; accepted September 8, 2017.
Corresponding Author: Jeong-Joon Kim (jjkim@kpu.ac.kr)
* Dept. of Computer Science and Engineering, Korea Polytechnic University, Siheung, Korea (jjkim@kpu.ac.kr)

information indicating the page's recency [4-7]. However, this information alone can result in the wrong selection of a replacement page, and there is the possibility that the Hit Ratio will be lowered [8-10].

Therefore, when selecting a replacement target page, unlike in the existing algorithm that checks whether a page is clean or dirty, the reference time information, and whether the page is referenced in binary form, we propose an algorithm that considers the number of deletion operations, which is a feature of flash memory, besides the number of references, which will take account of the flash memory's hardware properties more than existing algorithms.

2. Related Research

The buffer cache stores a portion of the entire disk block to reduce physical I/O requests. Since the buffer cache is relatively small compared to the entire disk, data is frequently replaced; a buffer replacement algorithm is required to efficiently utilize this.

Since the buffer replacement algorithm based on using a hard disk as a conventional storage device has the same read/write operation speed as the hard disk, it can be said that the higher the buffer "Hit Ratio" that is maintained, the better the performance buffer. Therefore, many hard disk-based buffer replacement algorithms that change page based on the Recency or Frequency of page references in the buffer have been proposed.

However, since the write operation speed of the flash memory is 10 times slower than that of the read operation, replacing the clean page and inducing the read operation by leaving the data unchanged in the buffer costs less than replacing the dirty page inducing the write operation by changing the data in the buffer. Therefore, buffer replacement algorithms that consider the write cost are proposed for flash memory.

2.1 CLRU

The CLRU distinguishes the page from the clean page list and the dirty page list, and divides the page into four types based on the reference time to select the replacement target page. Fig. 1 is an example of selecting the replacement target page of CLRU [11].

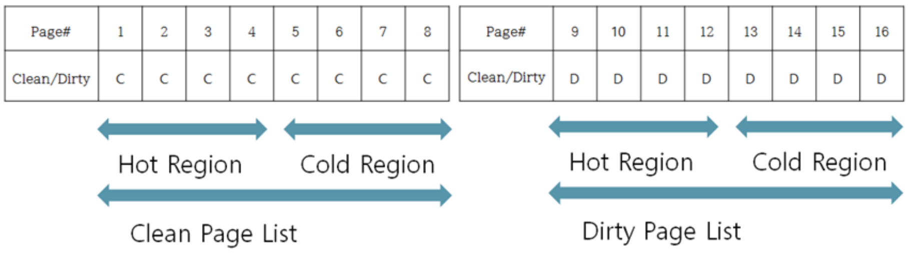


Fig. 1. Example of CLRU.

As shown in Fig. 1, the CLRU is divided into a clean page list and a hard page list. Each list is divided into a Cold Area and a Hot Area, and the size of the Cold Area can be represented by a normalized expression that considers the reference time.

$$NVET'_i = \frac{NVET_i - min^{m}_{j\in m}(NVET_j)}{max^{m}_{j\in m}(NVET_j) - min^{m}_{j\in m}(NVET_j)} \tag{1}$$

As shown in Eq. (1), $NVET_i$ represents the elapsed time of page i, $NVET'_i$ is the normalized value of the elapsed time for page i, and m represents the total number of pages in the page list. If $NVET_i$ is less than the average value for $NVET_i$, the page is included as a hot page in the hot area of the page list, and if $NVET_i$ is greater than or equal to the average, the page is included as a cold page in the cold section of the page list.

CLRU selects the page with the lowest access frequency in the Cold Area of the clean page list as the replacement target page. If there are no pages in the Cold Area of the clean page list, the page with the lowest access frequency in the Cold Area of the dirty page list is selected. If there are no Cold Area pages in the dirty page list, the page with the lowest access frequency in the Hot Area of the clean page list is selected as the replacement target, and if there is no Hot Area page in the clean page list, the Hot Area of the dirty page list is selected as the replacement target page.

CLRU reduces the write operation of the flash memory while delaying the replacement of cold dirty pages rather than the cold clean page, and increases the accuracy rate of pages by replacing cold pages ahead of hot pages. However, when selecting a page to replace, the page only considers the time referenced to the processor; the number of references is not considered. Therefore, if the number of references is taken into account when selecting the page to be replaced, an appropriate replacement target page can be selected based on more information.

2.2 HDC

In addition to the write operation delay that the existing flash memory based buffer replacement algorithm uses to select the replacement page, HDC considers distinguishing the buffer cache from the clean and dirty page list and the largest weight value of the page assigned for use in the sub-paging technique when selecting the replacement target page [12]. Fig. 2 shows the sub-paging technique.

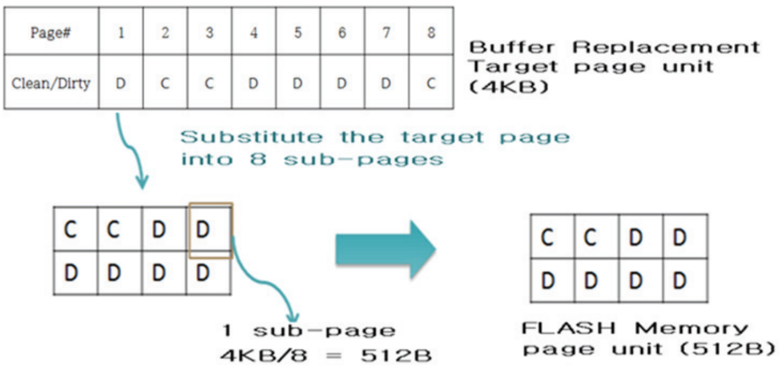

Page#
1
2
3
4
5
6
7
8
Clean/Dirty
D
C
C
D
D
D
D
C
Buffer Replacement Target page unit (4KB)
Substitute the target page into 8 sub-pages
C C D D
D D D D
1 sub-page
4KB/8 = 512B
C C D D
D D D D
FLASH Memory page unit (512B)

Fig. 2. Sub paging method.

The conventional flash memory-based buffer replacement algorithm causes a write operation on the page when a dirty page replacement occurs. At this time, the size of one page of memory buffer cache is 4 kB and the size of one page of flash memory is 512 B. Therefore, eight write operations occur in total when a dirty replacement page is selected in the buffer. However, recent research on the fact that the page in the buffer contains clean data in the dirty replacement page has been carried out, and some of the eight write operations include read operations caused by clean data. Therefore, write operations on some clean data must be removed, since they cause more write operations than are actually done. HDC proposed a sub-paging technique to address this problem. As shown in Fig. 2, the sub-paging technique is proposed to divide pages in a buffer into flash memory units to prevent excessive write operations. The page P1 in the buffer has already generated eight write operations. However, if the sub-paging technique is used, P1 is divided into eight sub-pages, and then whether the divided sub-pages are clean or dirty is checked. P1 generates six write operations in total because there are two clean sub-pages and six dirty sub-pages.

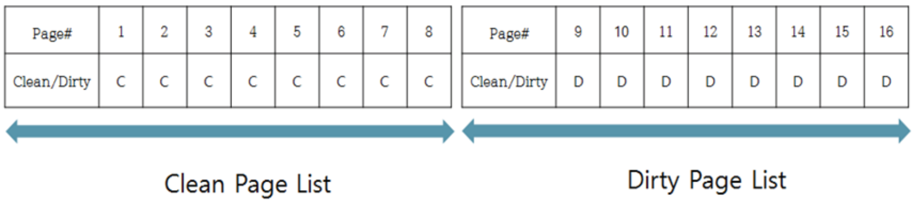


Fig. 3. Example of HDC.

Fig. 3 shows an example of selecting the replacement target page from HDC. The HDC is divided into two lists, clean pages and dirty pages. The weight value is imposed based on CostofWriting, which is the ratio of dirty subpages for each page in the list, HotDegree, which is the time the page was referenced to the processor, and λ, which represents the ratio of read and write operation rates. The weight value RI(P) can be expressed by the following equation.

$$RI(\rho) = \lambda \times HotDegree(\rho) + (1 - \lambda) \times CostofWriting(\rho) \tag{2}$$

The clean page and dirty page lists are sorted in ascending order based on their weight value. The page with the largest weight value in each list is selected as the replacement candidate page candidate and the page that has the smallest weight value among the candidates is replaced with the replacement target page.

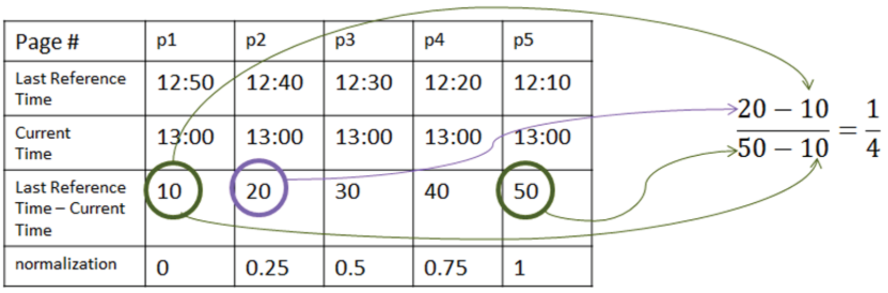
The HDC selects replacement pages considering the reference time considered by the existing flash memory algorithm, the ratio of the dirty subpages using the sub-paging technique, and the ratio between the flash memory's read and write operation speeds. This has the advantage that it reduces the write operation for the flash memory, but there is the disadvantage that the limited number of erase operations in the flash memory is not considered.

3. Page Replacement Algorithm

3.1 Hot and Cold Classification Algorithm

The Hot and Cold Classification Algorithm is an algorithm that determines whether a page in a buffer is hot or cold.

The existing algorithm only considers the reference time in hot page and cold page discrimination, but in the Hot and Cold Classification Algorithm proposed in this paper, the reference time and reference frequency are considered together. Fig. 4 shows an example that considers the reference time.

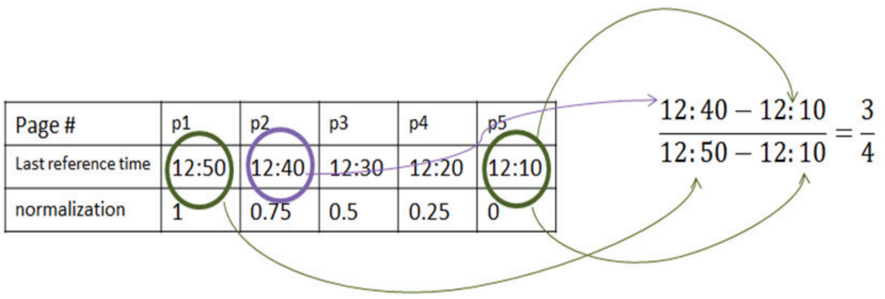


Page #	p1	p2	p3	p4	p5
Last Reference Time	12:50	12:40	12:30	12:20	12:10
Current Time	13:00	13:00	13:00	13:00	13:00
Last Reference Time – Current Time	10	20	30	40	50
normalization	0	0.25	0.5	0.75	1

$$\frac{20-10}{50-10}=\frac{1}{4}$$

Fig. 4. The existing research contains an example that considers the reference time.

Previous research studies have shown the normalized value of each page by dividing the difference between the elapsed time value of an arbitrary page and the maximum elapsed time value of pages in the buffer into the difference between the maximum elapsed time and the minimum elapsed time of pages in the buffer based on the elapsed time value, which is the difference between the last time the page was referenced and the current time. For example, the elapsed time value of p2 in Fig. 4 is 20, the maximum value of the elapsed time value in the buffer is 50, and its minimum value is 10. Therefore, the normalized value of p2 is obtained as (20 – 10) / (50 – 10) = 1/4 = 0.25, and normalization values in all buffers can be derived.



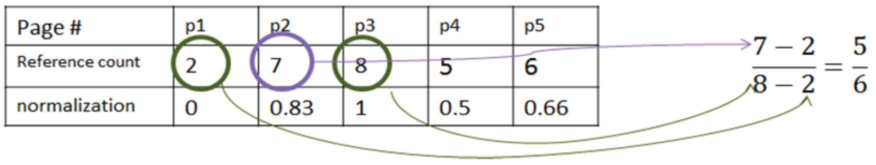
Page #	p1	p2	p3	p4	p5
Last reference time	12:50	12:40	12:30	12:20	12:10
normalization	1	0.75	0.5	0.25	0

$$\frac{12:40-12:10}{12:50-12:10}=\frac{3}{4}$$

Fig. 5. This paper considers the proposed reference time.

However, this paper divides the difference between the values of arbitrary pages and the maximum values of pages in the buffer by the difference between the maximum and minimum values. At this time, the value of an arbitrary page is normalized based on the last reference time value. For example, in Fig. 5, the normalized value of p2 is 3/4 = 0.75, which is attained by dividing the difference between 12:40 and 12:10 into the difference between the maximum value of 12:50 in the buffer and the minimum value of 12:10. This reduces the computation cost by deriving the normalized value that only considers the last reference time, compared to the existing related method that considers the reference time as the elapsed time difference between the current time and the last reference time.

Next, Fig. 6 shows the consideration of the number of references.



Page #	p1	p2	p3	p4	p5
Reference count	2	7	8	5	6
normalization	0	0.83	1	0.5	0.66

$$\frac{7-2}{8-2}=\frac{5}{6}$$

Fig. 6. Example considering reference operations.

As Fig. 6 shows, reference counts are normalized in the same way as reference times, and the number of references is considered. For example, the normalization value of p2 is calculated by dividing the difference between the reference count 7 of p2 and the minimum reference count 2 of the pages in the buffer into the difference between the maximum reference count of eight pages and the minimum reference count of 2 in the buffer. In this manner, the normalized value of the reference count can be derived for every page in the buffer.

We can express the proposed method of this paper in the equations, which considers the normalized value of the reference time and the reference frequency of a specific page, using the reference frequency and reference time and the terminology used for describing this as shown in Table 1.

Table 1. Terminology to describe the hot/cold segmentation algorithm

Terms	Explanation
p_i	A specific page
$p = \{p_1, p_2, \cdots, p_n\}$	The entire set of pages in the buffer list
t_{p_i}	The last reference time for a specific page p_i
t_P	$\{t_{p_1}, t_{p_2}, \cdots, t_{p_n}\}$
T_{p_i}	The reference time for a particular page, p_i normalized value
T_P	$\{T_{p_1}, T_{p_2}, \cdots, T_{p_n}\}$
C_{p_i}	Number of referrals for a specific page p_i
C_P	$\{C_{p_1}, C_{p_2}, \cdots, C_{p_n}\}$
C_{P_i}	The normalized values refer to a specific page number of p_i
C_P	$\{C_{p_1}, C_{p_2}, \cdots, C_{p_n}\}$

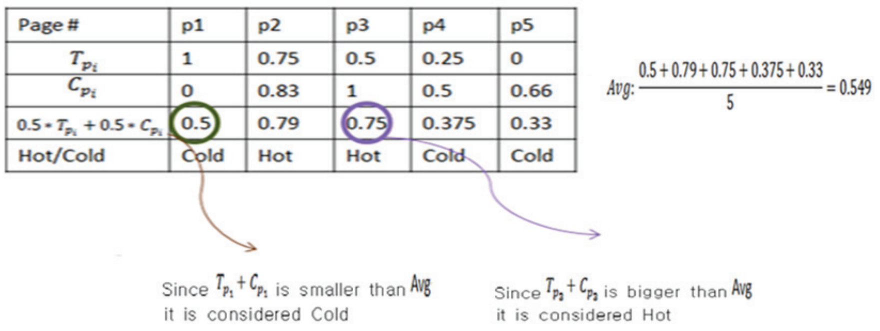
Based on terminology used in Table 1, the formula for obtaining the normalized value of the reference time of a specific page can be expressed as Eq. (3).

$$T_{p_i} = \frac{t_{p_i} - \min(t_P)}{\max(t_P) - \min(t_P)} \tag{3}$$

Next, the formula for calculating the normalized value of a specific page's reference count can be expressed as Eq. (4).

$$C_{p_i} = \frac{c_{P_i} - \min(c_P)}{\max(c_P) - \min(c_P)} \tag{4}$$

Next, we introduce a method that considers the reference time and count together in Fig. 7.



Page #	p1	p2	p3	p4	p5
T_{p_i}	1	0.75	0.5	0.25	0
C_{p_i}	0	0.83	1	0.5	0.66
$0.5 * T_{p_i} + 0.5 * C_{p_i}$	0.5	0.79	0.75	0.375	0.33
Hot/Cold	Cold	Hot	Hot	Cold	Cold

$$Avg: \frac{0.5+0.79+0.75+0.375+0.33}{5} = 0.549$$

Since $T_{p_1}+C_{p_1}$ is smaller than Avg it is considered Cold

Since $T_{p_3}+C_{p_3}$ is bigger than Avg it is considered Hot

Fig. 7. An example in which reference times and reference times are considered together.

Hot and Cold pages are identified by assigning weights ω and 1- ω to the normalized value T_{p_i} of the reference count and the normalized value C_{P_i} of the reference time of the specific page P_i in the buffer, respectively. The weight ω is calculated by assigning a weight value to the reference time and reference frequency according to the importance set by the user. ω denotes a weight value for the reference time and has a value in the range 0–1; 1- ω denotes a weight value for reference count and its value depends on ω.

The formula showing the Hot and Cold Classification Algorithm that considering the weight values can be expressed as Eq. (5).

$$\omega \times T_{p_i} + (1-\omega) \times C_{p_i} \tag{5}$$

As shown in Fig. 7, if the normalized value T_{p_i} of the reference time of p1 is 1, the normalized value C_{P_i} of the reference number is 0 and if the normalized value T_{p_i} of the reference time of p3 is 0.5, the normalized value of reference number C_{P_i} is 1. The ω value is given the same weight value of 0.5 for the reference frequency and reference time. Eq. (3), p1 has 0.5 and p3 has 0.75.

When the Hot and Cold Classification Algorithm of the pages is equal to Eq. (5), the average value of each page in the buffer calculated using Eq. (5) is represented as Avg, and the $\omega \times T_{p_i} + (1-\omega) \times C_{p_i}$ value of each page and Avg compare the values. If the value of $\omega \times T_{p_i} + (1-\omega) \times C_{p_i}$ is larger than that Avg for a certain page, it is regarded as a hot page, and if $\omega \times T_{p_i} + (1-\omega) \times C_{p_i}$ is smaller than Avg, it is regarded as a cold page.

Based on this, it can be shown how an arbitrary page can be regarded as a cold page (Eq. (6)) or hot page (Eq. (7)).

$$\omega \times T_{p_i} + (1-\omega) \times C_{p_i} \le \frac{1}{n}\sum_{i=1}^{n}(\omega \times T_{p_i} + (1-\omega) \times C_{p_i} \tag{6}$$

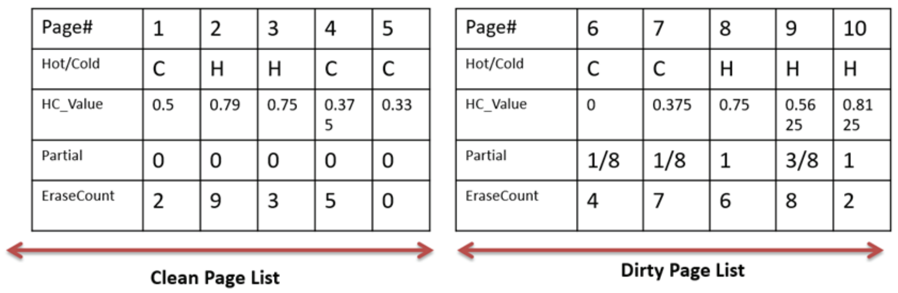
This expression is a cold page, and Avg can be expressed as $\frac{1}{n}\sum_{i=1}^{n}(\omega \times T_{p_i} + (1-\omega) \times C_{p_i}$.

$$\omega \times T_{p_i} + (1-\omega) \times C_{p_i} > \frac{1}{n}\sum_{i=1}^{n}(\omega \times T_{p_i} + (1-\omega) \times C_{p_i} \tag{7}$$

As shown in Fig. 7, the value of $0.5 \times T_{p_i} \times 0.5 \times C_{p_i}$ in p1 is 0.5 when Avg is 0.549, which is the average value of pages in the buffer, calculated using (7). Therefore, p1 is a cold page because it has a value smaller than Avg. The value of $0.5 \times T_{p_i} \times 0.5 \times C_{p_i}$ for p3 is 0.75, making p3 a hot page because it has a value larger than Avg. In this manner, it is possible to determine whether pages in the buffer are hot or cold.

3.2 Algorithm

The mIBRA algorithm proposed in this paper consists of a clean page list and a dirty page list. Fig. 8 shows the structure of the buffer list for the IBRA algorithm.



Page#	1	2	3	4	5
Hot/Cold	C	H	H	C	C
HC_Value	0.5	0.79	0.75	0.375	0.33
Partial	0	0	0	0	0
EraseCount	2	9	3	5	0

Clean Page List

Page#	6	7	8	9	10
Hot/Cold	C	C	H	H	H
HC_Value	0	0.375	0.75	0.5625	0.8125
Partial	1/8	1/8	1	3/8	1
EraseCount	4	7	6	8	2

Dirty Page List

Fig. 8. Buffer list structure of the IBRA algorithm.

As shown in Fig. 8, the HC_Value, which is calculated by the Hot/Cold Detection algorithm to check whether a page is hot or cold, consists of a HC_Value with a calculated result, Partial, which can be known as partial or as a page of clean and dirty pages full partials, and EraseCount, which carries delete operation count information.

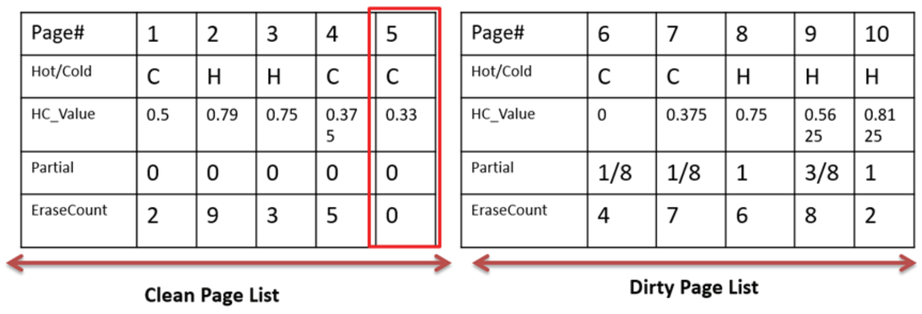
The replacement target page is selected in the order: Cold Clean Page, Cold Partial Dirty Page, Cold Full Dirty Page, Hot Clean Page, Hot Partial Dirty Page, and then Hot Full Dirty Page according to the priority of the replacement page selection proposed in this paper.

If there are many Cold Clean Pages in the buffer, replace the Cold Clean Page with the lowest HC_Value value first. If there are no Cold Clean Pages in the buffer, the next priority, Cold Partial Dirty Page, is selected for replacement. If there are many Cold Partial Dirty Pages in the buffer, the page with the smallest product of the HC_Value and the Partial is substituted first. If there are no Cold Partial Dirty Pages in the buffer, the next priority Cold Full Dirty Page is selected as the replacement target. If there are many Cold Full Dirty Pages in the buffer, replace the Cold Full Dirty Page with the one with the lowest HC_Value value.

If the buffer list is filled with pages as shown in Fig. 9, we check whether there is a Cold Clean Page according to the priority proposed in this paper. Currently, P1, P4, and P5 are Cold Clean Pages. The HC_Value is checked here. In the above situation, since the HC_Value of P5 is the smallest at 0.33, P5 is selected as the replacement target page.

Fig. 10 shows the EraseConsider algorithm for selecting replacement page candidates considering the deletion frequency.

As shown in Fig. 10, the deletion algorithm works as follows. First, the number of page deletions that exist in the buffer list is obtained, and an average value is assigned to Avg. Next, the number of deletion operations for pages in the buffer is compared with the Avg value, and pages smaller than the average are extracted as Pages and used to perform the SelectVictim algorithm. Fig. 11 shows the SelectVictim algorithm for selecting the pages to be replaced in the buffer.



Page#	1	2	3	4	5
Hot/Cold	C	H	H	C	C
HC_Value	0.5	0.79	0.75	0.375	0.33
Partial	0	0	0	0	0
EraseCount	2	9	3	5	0

Clean Page List

Page#	6	7	8	9	10
Hot/Cold	C	C	H	H	H
HC_Value	0	0.375	0.75	0.5625	0.8125
Partial	1/8	1/8	1	3/8	1
EraseCount	4	7	6	8	2

Dirty Page List

Fig. 9. Example of selecting the page to be exchanged for the IBRA algorithm.

function	EraseConsider
Explaination	Selection of candidates for replacement page considering deletion frequency
Input	Blist: Buffer List Avg: Average vlaue of deletions Erase_Value: Number of deletions Pages: Intermediate results page

```
1:  BEGIN
2:      FOR EACH Page is Blist
3:          Avg ← AVG(Page.Erase_Value)
4:      END FOR
5:      FOR EACH Page is Blist
6:          IF(Page.Erase_Value<Avg)
7:              Pages ← Page
8:          END IF
9:      END FOR
10:     SelctVitim(Pages);
11: END
```

Fig. 10. Algorithm to consider deletion operation.

As shown in Fig. 11, the process of selecting the replacement object selection algorithm is as follows. First, if a Cold page exists in the buffer list, and if a Clean page exists in the CLlist, assign the page with the lowest Hot_Cold_Value among the Clean pages to Victim. If there is no page in the CLlist, assign the page that has the smallest value among the products of Partial_Value and Hot_Cold_Value among the pages whose Partial_Value for the dirty page is <1 in the DRlist to Victim. If the Dirty page of the DRlist does not have a page with a Partial value less than 1, assign the page with the lowest Hot_Cold_Value among the pages whose Partial_Value for the dirty page is not <1 to Victim.

However, if there is no Cold page in the buffer list, the page with the lowest Hot_Cold_Value of the Clean pages among the Hot pages, rather than the Cold page existing in the CLlist is assigned to Victim and the Victim is returned. If there is no page in the CLlist, assign the page that has the smallest value among the products of Partial_Value and Hot_Cold_Value among the pages whose Partial_Value for the dirty page is <1 in the DRlist to Victim. If the Dirty page of the DRlist does not have a page with a Partial <1, the page with the lowest Hot_Cold_Value among the pages whose Partial_Value of the dirty page is not <1 is assigned to Victim and the Victim is returned.

function	SelectVictim
Explanation	Select the page to replace in the buffer
Input	Blist: Buffer(Page) List DRlist: Dirty Page List Hot_Cold_Value: Hot/Cold Value CLlist: Clean page List Victim: Replacement target page Partial_Value: Partial value of the Dirty page Pages: Intermediate results page
output	Replacement target page

```
BEGIN
    IF(ColdPage is Blist)
        FOR EACH CleanPage in CLlist
            Victim ← MIN(CleanPage.Hot_Cold_Value)
        END FOR
        FOR EACH DirtyPage in DRlist
            IF(DirtyPage.Partial_Value < 1)
                Victim ← MIN(DirtyPage.Partial_Value * DirtyPage.
                Hot_Cold_Value)
            ELSE
                Victim ← MIN(DirtyPage.Hot_Cold_Value)
            END IF
        END FOR
    ELSE
        FOR EACH CleanPage in CLlist
            Victim ← MIN(CleanPage.Hot_Cold_Value)
        END FOR
        FOR EACH DirtyPage in DRlist
            IF(DirtyPage.Partial_Value < 1)
                Victim ← MIN(DirtyPage.Partial_Value * DirtyPage.
                Hot_Cold_Value)
            ELSE
                Victim ← MIN(DirtyPage.Hot_Cold_Value)
            END IF
        END FOR
    END IF
    RETURN Victim
END
```

Fig. 11. Algorithm for page selection to be exchanged.

4. Performance Evaluation

This paper evaluates the performance of the IBRA algorithm using Flash-DBSim [13], a simulation platform capable of evaluating flash memory-based algorithms. Flash-DBSim is often used to evaluate flash memory-based buffer replacement algorithms because users can specify flash memory specifications. Table 2 shows the specifications of the detailed flash memory that was set up to evaluate the performance of the IBRA algorithm this paper proposes.

We verified the efficiency of the IBRA algorithm presented in this paper by comparing the buffer replacement algorithms based on existing flash memory with AD-LRU, CLRU, and HDC. The traces are used to evaluate the performance of the IBRA algorithm and the characteristics of each trace are shown in Table 3.

Table 2. Flash memory specifications

Item	Value
Page size (byte)	4,096
Block size (page)	64
Page reading speed (μs/page)	25 (max)
Page writing speed (μs/page)	220
Block delete speed (ms/block)	1.5
Deletion threshold (durability)	100,000

Table 3. Trace characteristics

Trace name	Number of request	Read/Write ratio (%)	Reference pattern
Random Access	1,000,000	50/50	Uniform
Read-Most	1,000,000	90/10	Uniform
Write-Most	1,000,000	10/90	Uniform

The performance is compared by applying the 8 MB buffer size to the algorithm, and the buffer hit rate, number of write operations, and execution time are each compared in order.

4.1 Buffer Hit Rate

As shown in Fig. 12, the IBRA ratio is the highest in all traces T1, T2, and T3, and the IBRA hit ratio at T1 is 84%. The Hit Ratio at T2 is 61% and the Hit Ratio at T3 is 90. T3 has the highest hit rate because T3 slows the replacement of dirty pages by 10%/90% of the read/write ratio.

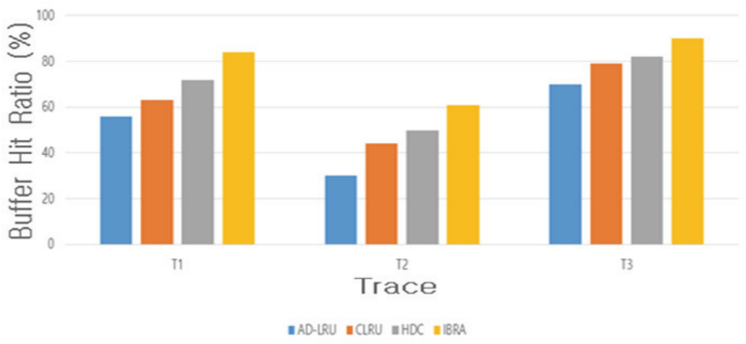


Fig. 12. Trace stars existing algorithm and IBRA buffer hit ratio.

We verified the efficiency of the wavelet histogram generation system developed in this paper using the Page Traffic statistical information data (180 million) that was accumulated during the week of the

Wikipedia Hits Log [12]. The Page Traffic statistical information data used the value of the language attribute from the visitor page among the four attributes (visitor page language, page name, number of page requests, and response contents size) as the input data for the performance evaluation. The size of the generated histogram (the number of coefficients included in the histogram) and the generation time of the wavelet histogram were measured.

4.2 Number of Write Operations

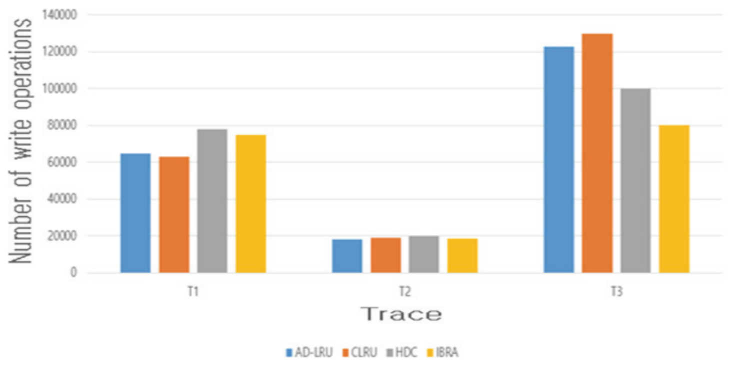


Fig. 13. Trace the existing algorithms and IBRA write operations.

As shown in Fig. 13, the number of write operations in the trace T1 is 75,000 times, the number of write operations in T2 is 19,500 times, and the number of write operations in T3 is 80,000 times. T3 has a higher write operation rate than other traces because the read/write ratio is 10%/90%. However, since the algorithm proposed in this paper selects a candidate group with a small number of deletion operations and selects a replacement target page, the lowest number of write operations is shown in T3.

4.2 Run Time

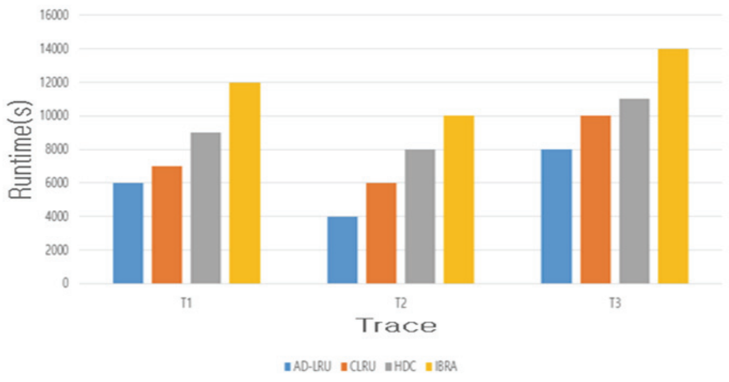


Fig. 14. Trace the existing algorithms and IBRA run time.

As shown in Fig. 14, the run time of IBRA in this paper is 1.5–2 times longer than other algorithms in all traces. Since the candidates are selected before the replacement page is selected in this paper, the run time is slightly increased. However, the write operation is about 10 times slower than the read operation in flash memory, and the erase operation is about 10 times slower than the write operation and about 100 times slower than the read operation. Therefore, in terms of overall flash memory management, it is more important to reduce the write and erase operations while increasing the buffer hit rate rather than the run time.

5. Conclusion

As we are entering the Big Data age in which the amount of data will increase exponentially, devices that can store data are also constantly evolving. Flash memory, which is a type of nonvolatile memory, has advantages of being faster and lighter than a hard disk, so they are being increasingly adopted as storage devices in various fields in recent years. The buffer is intended to store pages with a large number of references to reduce the speed difference between the CPU and the storage device. A buffer replacement algorithm has been proposed to improve the buffer's performance and an existing buffer replacement algorithm has been proposed based on a hard disk with the same operation speed. Therefore, since conventional algorithms are unsuitable for flash memories that have different operation speeds, many studies on buffer replacement algorithms that consider the characteristics of flash memories have been conducted recently.

Among the proposed buffer replacement algorithms that consider the characteristics of flash memory, AD-LRU considers the reference count, but since it is a binary type indicating re-referencing, it is difficult to indicate the accurate reference count of a page if it is referenced more than twice. Since the CLRU considers the elapsed time when distinguishing hot/cold, indicating the page's update pattern, it takes a long time to execute, and it has a disadvantage because HDC does not consider the reference count.

Therefore, this paper first divides the pages into six groups, classifies them in more detail than previous studies, and presents a Hot and Cold Classification Algorithm to consider reference times and reference times together. Furthermore, considering the limited lifetime of the flash memory, the candidates for replacement pages were selected based on the number of deletions.

Finally, we compared the proposed IBRA algorithm with AD-LRU, CLRU, and HDC. The IBRA algorithm showed the highest buffer hit rate, IBRA had the lowest number of write operations, and IBRA showed the third fastest execution time.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2017R1A2B4011243).

References

[1] S. Ahn. S. Hyun. T. Kim, and H. Bahn, "A compressed file system manager for flash memory based consumer electronics devices," *Journal of IEEE Transactions on Consumer Electronics*, vol. 59, no. 3, pp. 544-549, 2013.

[2] H. Li, C. Yang, and H. Tseng, "Energy-aware flash memory management in virtual memory system," *Journal of IEEE Transactions on Very Large Scale Integration Systems*, vol. 16, no. 8, pp. 952-964, 2008.

[3] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Boulder, CO, 1990, pp. 143-152.

[4] S. Jiang, F. Chen, and X. Zhang, "CLOCK-Pro: an effective improvement of the CLOCK replacement," in *Proceeding of the USENIX Annual Technical Conference*, Anaheim, CA, 2005, pp. 323-336.

[5] S. Jiang and X. Zhang, "LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance," in *Proceeding of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Marina Del Rey, CA, 2002, pp. 31-42.

[6] T. Johnson and D. Shasha, "2Q: a low overhead high performance buffer management replacement algorithm," in *Proceeding of the 20th International Conference on Very Large Data Bases*, Santiago de Chile, Chile, 1994, pp. 439-450.

[7] E. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," in *Proceeding of ACM SIGMOD International Conference on Management of Data*, Washington, DC, 1993, pp. 297-306.

[8] H. Jung, H. Shim, S. Park, S. Kang, and J. Cha, "LRU-WSR: integration of LRU and writes sequence recording for flash memory," *Journal of IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1215-1223, 2012.

[9] Z. Li, P. Jin, X. Su, K. Cui, and L. Yue, "CCF-LRU: a new buffer replacement algorithm for flash memory," *Journal of IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1351-1359, 2009.

[10] S. Y. Park, D. Jung, J. U. Kang, J. S. Kim, and J. Lee, "CFLRU: a replacement algorithm for flash memory," in *Proceeding of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, Seoul, Korea, 2006, pp. 234-241.

[11] G. Xu, F. Lin, and Y. Xiao, "CLRU: a new page replacement algorithm for NAND flash-based consumer electronics," *Journal of IEEE Transactions on Consumer Electronics*, vol. 60, no. 1, pp. 38-44, 2014.

[12] M. Lin, S. Chen, G. Wang, and T. Wu, "HDC: an adaptive buffer replacement algorithm for NAND flash memory-based databases," *Optik-International Journal for Light and Electron Optics*, vol. 125, no. 3, pp. 1167-1173, 2014.

[13] X. Su, P. Jin, X. Xiang, K. Cui, and L. Yue, "Flash-DBSim: a simulation tool for evaluating flash-based database algorithms," in *Proceedings of International Conference on Computer Science and Information Technology*, Beijing, China, 2009, pp. 185-189.

Jeong-Joon Kim https://orcid.org/0000-0002-0125-1907

He received his B.S. and M.S. in Computer Science at Konkuk University in 2003 and 2005, respectively. In 2010, he received his Ph.D. in at Konkuk University. He is currently a professor at the department of Computer Science at Korea Polytechnic University. His research interests include database systems, big data, semantic web, geographic information systems (GIS), and ubiquitous sensor network (USN), etc.