PT-LRU: A Probabilistic Page Replacement Algorithm for NAND Flash-based Consumer Electronics

Jinhua Cui, Weiguo Wu, Yinfeng Wang, and Zhangfeng Duan

Abstract —*The flash memory is being used to an ever-increasing extent in consumer electronics for the purpose of the dominant secondary storage device. Aiming at the buffer replacement policy for flash-based storage devices in efficiency, a new efficient flash-aware buffer replacement algorithm based on probability is proposed, where three LRU linked lists are used to manage the buffer. The proposed algorithm delays the eviction of hot clean pages by evicting the cold clean pages preferentially. When the cold clean linked list is empty, the pages in the cold dirty linked list will be expelled from the buffer with a higher probability, whereas the probability of evicting the hot clean pages is lower. In that way, hot clean pages cannot reside in the buffer for a long time. Moreover, an interpretation model for describing the proposed algorithm is developed from FSM. Highlights of the experiments include an evaluation with the dynamic write/read ratio traces. Experimental results and detailed comparisons with other similar best-known competitor algorithms on the static write/read ratio traces and the dynamic write/read ratio traces show that not only does the proposed algorithm reduce the write counts, but also it offers a trade-off between the hit ratio and the overall runtime in most cases. The adaptability of the proposed method is also verified by the experiments on various kinds of traces including random, read-most, write-most and Zipf.*[1]

Index Terms — Buffer Management, Probabilistic Algorithms, Flash Memory, Consumer Electronics

I. Introduction

Consumer electronics are widely equipped with the NAND flash memory as storage devices to achieve low power consumption, shock resistance, and low noise. Because of the mechanical characteristics of NAND Flash memory, it is no longer restricted to seek penalty and rotational latency. Moreover, it offers substantially higher speed performance relative to magnetic disks in terms of the random read request and the random write request, which potentially eliminates I/O bottlenecks in the bandwidth-intensive data center. Three basic operations are involved in NAND flash memory, namely read, write and erase, respectively. Among them, read/write operations are performed in units of pages, and an erase operation is performed on block granularity. Flash memory has the characteristics of asymmetric I/O latencies for read, write, and erase. Furthermore, data are performed on out-of-place update mechanism to deal with the unique hardware constraint. Additionally, when the erasure limit is reached, the corresponding block will be worn out and becomes unmanageable and unreliable.

Buffer replacement policy can optimize the I/O sequence and reduce disk accesses, thus improving the overall efficiency of the storage system. LRU (Least Recently Used) policy considers the locality of references. Therefore, most storage systems choose LRU or upgraded algorithms of LRU as the buffer replacement policy. However, the I/O performances for read and write operations are asymmetric in flash memory. So the design for buffer replacement algorithm in NAND flash-based consumer electronics should be reconsidered. These algorithms should not only consider the hit ratio, but also should take into account to reduce the counts of write operations and raise the counts of read operations moderately.

Previous research efforts on flash-based buffer replacement algorithms showed their better performance than traditional buffer replacement algorithms, however, some factors had not been fully considered. For instance, CFLRU [1] does not care about the access frequency of pages in the buffer. LRU-WSR [2] ignores the access frequency of clean pages resided in the buffer. Therefore, a novel flash-aware buffer replacement algorithm that enhances the previous APB-LRU [3] and LRU-WSR methods by reducing the number of write/erase operations and retaining a high buffer hit ratio is proposed.

In this paper, the management of the buffer is composed by three main lists: a cold clean LRU list, a cold dirty LRU list and a mixed LRU list. Preferentially, PT-LRU (Probabilistic Triplicate LRU) evicts the LRU pages from the cold clean LRU list. After that, when the cold clean list is empty, the pages in the cold dirty LRU list will be expelled from the buffer with a higher probability, whereas the probability of

[1] This work was supported in part by the National Natural Science Foundation of China under grant NO.91330117, NO.61202041 and National High-Tech Research and Development Plan of China under the grant NO.2012AA01A306.

Jinhua Cui is with the school of Electronic and Information Engineering, Xi'an Jiaotong University, Shaanxi 710049, China (e-mail: cjhnicole@gmail.com)
Weiguo Wu is with the school of Electronic and Information Engineering, Xi'an Jiaotong University, Shaanxi 710049, China (e-mail: wgwu@mail.xjtu.edu.cn)
Yinfeng Wang is with the department of Software Engineering, ShenZhen Institute of Information Technology, Guangdong 518172, China (e-mail: wangyf@mailst.xjtu.edu.cn)
Zhangfeng Duan is with the school of Electronic and Information Engineering, Xi'an Jiaotong University, Shaanxi 710049, China (e-mail: lzcgwushuang@163.com)

Contributed Paper
Manuscript received 09/03/14
Current version published 01/09/15
Electronic version published 01/09/15.

expelling the hot clean pages from residence is lower. Furthermore, PT-LRU is validated by two types of traces, including dynamic and fixed write-read ratio traces, because write-read ratio in the real world may change continuously.

The rest of this paper is organized as follows: In Section 2, the related work is sketched. In Section 3, an efficient buffer replacement algorithm called PT-LRU is described in detail. Section 4 describes the details about the experiments and the performance evaluation results. Finally, it concludes this paper and outlines its future work in Section 5.

II. Related work

A. Flash Memory

Flash memory has two different kinds: NOR flash and NAND flash. The distinguishing design characteristic of bus interface makes the major difference between these two types. Moreover, in the NOR flash the size of an erase-block ranges from 64 to 128 Kbytes, whereas the NAND flash has erased blocks range from 8 to 32 Kbytes in size. It is appropriate to choose the NOR flash when the system is in need of code storage and execute-in-place applications, whereas the NAND flash is ideal for data storage because of high density [2]. This paper pays close attention to the NAND flash memory, which is usefully exploited for storing data.

Flash memory shows asymmetric I/O latencies for read, write, and erase operations. Table I extracted from Lee *et al* [4] compares the distinguishing characteristics in terms of the access time and the energy consumption. Compared with the NOR flash, the NAND flash devotes more time to a read operation and much less time to a write operation, which leads to less erasing time. Moreover, the latency of a write operation is about 200 times more than that of a read operation in the NOR flash, and the ratio downs to about 10 times in the NAND flash, as shown in Table I.

TABLE I
Characteristics of Two Types of Flash Memories

Flash	Access Time (us/4KB)			Energy Consumption (uJ/4KB)		
	Write	Read	Erase	Write	Read	Erase
NOR	14054.4	53.8	15616	3251.2	8.6	3609.6
NAND	1833	284.2	499.2	59.6	9.4	16.5

B. Buffer Replacement Algorithm for Flash Memory

Generally, traditional buffer replacement algorithms [5]-[7] are designed for hard disk memory, aiming at improving the hit ratio and thus resulting in better I/O performance. Among them, most algorithms focus on the frequency or the recency of pages references, because of the fact that pages references show a characterization of temporal locality. Write costs and read costs are treated equally in these algorithms. However, in flash memory, the cost of evicting a dirty page is much more expensive than that of reading a page. Hence, operating systems that use flash memory as the secondary storage should reconsider asymmetric I/O costs of read and write operations in terms of energy consumption and access time when they evict pages to reclaim free buffer slot.

The first algorithm called CFLRU is designed for flash memory by Park *et al* [1]. The basic idea is to split the buffer LRU list into the working region and the clean-first region, and it adopts a policy that clean pages in the clean-first region are chosen as victim pages in preference to dirty pages until there is no clean page in the clean-first region. Therefore, it is clear that CFLRU reduces the number of flash write operations by keeping a certain amount of dirty pages in page cache.

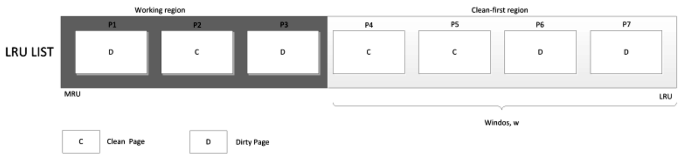


Fig. 1. Example of CFLRU Page Replacement Algorithm

In Fig. 1, this is an example to illustrate the CFLRU page replacement algorithm. Suppose data pages are accessed in the order P7, P6, P5, P4, P3, P2, and P1. Under the CFLRU algorithm, the sequence of the victim is P5, P4, P7, and P6. Hence, CFLRU preferentially evicts the least recently used clean page P5 in the clean-first region to reduce the number of flash write operations irrespective of the page P7 that has a higher priority to be referenced than page P5 in the near future, thus enhancing the overall performance of flash-based storage devices.

However, window size needed by CFLRU restricts the flexibility of the algorithm for tasks with various workloads. Furthermore, the frequency of the pages in the buffer is not exploited. Retaining cold dirty pages in the buffer and evicting hot clean page will generate more read counts than normal LRU, which results in poor I/O performance.

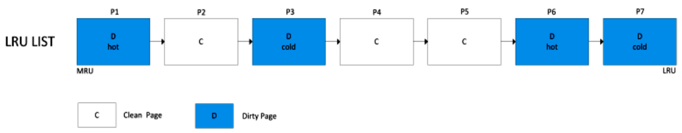


Fig. 2. Example of LRU-WSR Page Replacement Algorithm

LRU-WSR [2] enhances LRU by reordering write operations of not-cold dirty pages from the buffer cache to the flash storage. As illustrated in Fig. 2, LRU-WSR maintains a page list by the LRU order and every dirty page with an additional cold flag. When physical memory does not have a free buffer slot, pages will be checked in order until a victim page is chosen by following rules: clean or cold dirty pages will be chosen as victim pages directly and other types of pages will be labeled as cold dirty. LRU-WSR considers the access frequency of dirty pages to avoid permanently keeping cold dirty pages in the buffer, and therefore it reduces the number of write/erase as well as prevents serious degradation of a buffer hit ratio. However, the access frequency of clean pages is not taken into account, which leads to keeping cold clean pages in the buffer potentially. As a result, the problem in CFLRU is still not well addressed.

Li *et al* [8] present a new buffer replacement algorithm called CCF-LRU, which differentiates clean pages into hot and cold ones, and evicts cold clean pages firstly and delays the eviction of hot clean pages. CCF-LRU uses a cold clean LRU list to store cold clean pages, which are selected as the victims preferentially, and employs a mixed LRU list to hold dirty pages and hot clean pages, which are selected based on the policy like LRU-WSR. The policy evicts clean pages preferentially, which may lead to frequent evictions of newly referenced pages and lower the hit ratio.

AD-LRU [9] suggests controlling the length of the cold queue to solve the problem. Unlike CCF-LRU, both cold clean pages and cold dirty pages are stored in the list called the cold LRU list, while others are stored in the hot LRU list. If the length of the cold list is less than the minimal size *min_lc*, AD-LRU will choose the victim from the hot LRU list based on the strategy in CCF-LRU. Otherwise, the pages in the cold LRU list will be evicted. Actually, the problem in CCF-LRU is not substantially resolved, because clean pages in the cold list are still evicted preferentially and the evictions of newly-read pages are unavoidable, lowering the hit ratio and degrading the overall performance.

Lin *et al* [3] propose APB-LRU, a probability-based buffer replacement algorithm for flash-based databases that focuses on improving overall I/O performance of a storage system as well as preventing serious degradation of a buffer hit ratio. APB-LRU also divides the buffer into two regions similar to those in AD-LRU. However, different from AD-LRU, APB-LRU adopts a new policy based on probability, where the evicted probability of cold clean pages is greater than that of cold dirty pages. The disadvantage of the algorithm is that a dirty page may be evicted even if there are many cold clean pages, which may lead to more costs than other algorithms in some cases.

III. THE PT-LRU ALGORITHM

At the beginning of this section, the overall runtime of flash memory under the buffer layer is stated in NAND flash-based consumer electronics. Then the proposed page management with three linked lists is presented. Based on that design, the buffer management is reconsidered as the finite state machine [10]. Moreover, the detailed explanation of the proposed algorithm called PT-LRU is explained.

A. Theories of Motivation

When upper applications request the data page p in NAND flash-based consumer electronics, the overall runtime of flash memory under the buffer layer is formulized as below:

$$R(p) = (1 - P_{hit})C_{read}T_{read} + (1 - P_{hit})P_{dirty}C_{write}T_{write} + (1 - P_{hit})P_{dirty}C_{erase}T_{erase} + P_{hit}T \tag{1}$$

R(p): the overall runtime, *Phit*: the probability of hit, *Cread*: the read count, *Tread*: the time of reading page p from secondary storage device, *Pdirty*: the probability of dirty page, *Cwrite*: the write count, *Twrite*: the time of writing page back to secondary storage device, *Cerase*: the erase count, *Terase*: the time of erasing page back to secondary storage device, *T*: the time of reading page p.

In order to facilitate understanding of the equation, equation (1) is converted to (2) as shown below:

$$R(p) = (1 - P_{hit})T_{read}(C_{read} + P_{dirty}C_{write}T_{write}/T_{read} + P_{dirty}C_{erase}T_{erase}/T_{read}) + P_{hit}T \tag{2}$$

Note that *Tread*, *Twrite/Tread* and *Terase/Tread* depend on the underlying flash chips. The types of the flash memory are crucial in determining these variables. Moreover, the overall runtime of hitting the page p is much less than that of missing page p. More generally, three simplified parameters are invoked in (3). Briefly and intuitively, equation (2) is converted to (3) as follows:

$$R(p) = \alpha(1 - P_{hit})(C_{read} + \beta P_{dirty}C_{write}) + \gamma \tag{3}$$

Accordingly, to improve the overall performance efficiency should focus on reducing the number of write operations moderately, and at the same time retaining a high buffer hit ratio.

It was verified by theory and practice that the number of write operations would decrease by evicting clean pages preferentially, which would result in the better buffer performance. In PT-LRU, three LRU queues are designed to manage the buffer, namely LC (the cold clean linked list), LD (the cold dirty linked list), and LH (the mixed LRU linked list). The schematic of these linked lists is exhibited in Fig. 3.

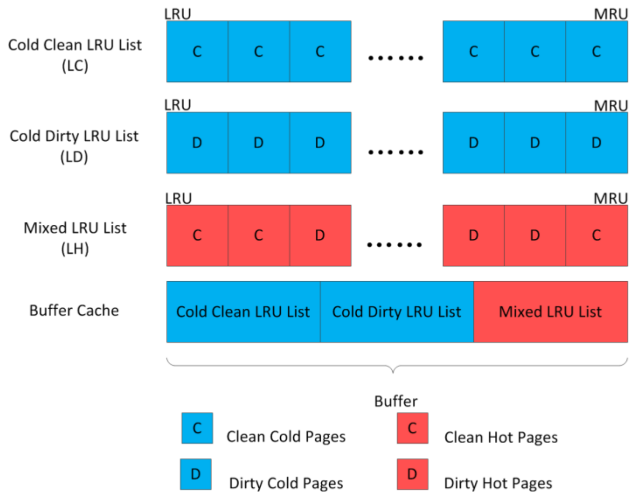


Fig. 3. The Linked Lists Schematic in the Buffer

Based on the schematic drawing, Fig. 4 illustrates an interpretation model developed from FSM. Upon initialization, all pages are deposited in the secondary storage devices, referred to herein as the flash memory which is the circle 0 in Fig. 4. The objective of PT-LRU is to improve the overall I/O performance by focusing on preventing serious degradation of the hit ratio and reducing the write counts during the replacement process.

In order to accomplish this goal, PT-LRU tries to evict cold clean pages with low access frequencies in the replacement process, which is associated with the curve from the circle 1 to the circle 0. If the length of LC linked list is too short, which means the frequent eviction of recently-referenced pages in the cold clean LRU linked list, the cold dirty pages will be dropped from the buffer by a probability that is associated with the curve from the circle 3 to the circle 0 in Fig. 4. Moreover, the probability of evicting the hot clean pages is much smaller than that of evicting the cold dirty pages. The curve from the circle 2 to the circle 0 means to drop hot clean pages from the buffer.

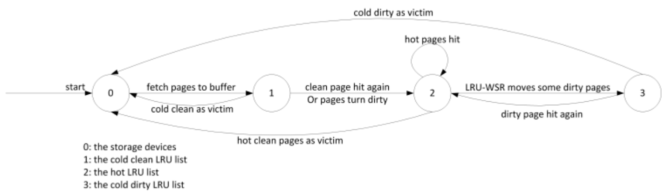


Fig. 4. FSM of PT-LRU

The main features of PT-LRU are summarized as follows:

(1) Three LRU linked lists are designed to capture both the recency and the frequency of pages references, among which LC and LD linked lists accommodate the pages referenced only once, and LH linked list maintains the rest pages.

(2)When the buffer does not have a free page slot for the new access page, PT-LRU preferentially evicts the LRU pages from the cold clean linked list. If the cold clean pages are not enough, cold dirty pages will be thrown out of memory with a probability.

(3) PT-LRU offers a trade-off between the hot and cold lists. To prevent the length of the hot linked list from being too long, PT-LRU utilizes the second-chance policy [2] to move a modest number of hot dirty pages to LD linked list when the eviction of the hot clean pages happens. Furthermore, the second-chance policy ensures that dirty pages in the LH linked list will not be stored in the buffer for a relatively extended period of time.

B. The Proposed PT-LRU Algorithm

To reduce the number of write operations to flash memory and prevent seriously degradation of the hit ratio, a new page replacement algorithm called PT-LRU is proposed for consumer electronics which have the NAND flash-based storage device. PT-LRU delays the eviction of hot clean pages by evicting the cold clean pages preferentially. Moreover, PT-LRU prevents hot clean pages residing in the buffer for a long time by evicting the cold dirty pages and hot clean pages with a probability.

The pseudo-codes of PT-LRU are illustrated in Fig. 5 and Fig. 6. When the page in the buffer is referenced again, PT-LRU adjusts the position of the page to the MRU position of the LH linked list. Once it is the first referenced page, it will be added into the buffer. Firstly, PT-LRU checks whether the buffer has a free page slot or not. If there are enough free page frames, the page will be inserted into the MRU position of the LC linked list. Alternatively, replacement occurs to eject some pages from the buffer to make space for the faulted page. In case that the chosen victim page is a dirty page, PT-LRU should write it back to consumer electronics which have the NAND flash as the secondary storage device.

```
Algorithm1 PT-LRU_manage()
Data queues：
      LC: the cold clean queue; LD: the cold dirty queue; LH: the mixed queue;
Out: The reference to the requested data page p in the buffer

if p in the buffer
    move p to the MRU position of LH;
    return the reference to p;
else if free space in The buffer
    put p in the MRU position of LC;
    return the reference to p;
else //replacement
    if LC is not NULL
        victim = the LRU page in LC;
    else
        calculate Replace-Flag by pro; //(0.5 <pro<1)
        if (Replace-Flag==1 and cold dirty pages in LD)
            victim=the LRU page in LD;
        else
            victim= PT-LRU_EvictHot();
    if victim is dirty
        write victim to flash memory;
    put p in the MRU of LC;
    return the reference to p;
```

Fig. 5. The Page Management Algorithm of PT-LRU

```
Algorithm2   PT-LRU_EvictHot()
Data queue：
    LC: the cold clean queue; LD: the cold dirty queue; LH: the mixed queue
Out: the reference to the victim data page

for (each page from LRU position of LH)
      victim=LRU page of LH;
      if(victim is hot clean)
            return victim;
      else if(cold-flag of victim is set)
            move victim  to MRU position of LD;
      else
            move victim to MRU position of LH;
            set cold-flag of victim;
end for
```

Fig. 6. The Eviction of Hot Pages in PT-LRU

During the eviction procedure, PT-LRU scans through the LC linked list preferentially. The LRU page in the LC linked list is chosen as the victim in the event of enough pages in that queue. When the LC linked list does not contain any pages, PT-LRU will calculate a replacement flag with a probability *pro*. If the replacement flag is 1 and the LD linked list has the cold dirty pages in it, PT-LRU will choose cold dirty pages from LD linked list as victims. Conversely, PT-LRU evicts the first hot clean page in LH linked list by the LRU order. Fig. 6 illustrates the pseudo-code of evicting hot pages. To maintain appropriate length in the LH linked list, the eviction of hot clean pages will use the second-chance policy to move an appropriate amount of dirty pages into the LD queue, and thus PT-LRU is self-tuning. Consequently, PT-LRU avoids the hot pages residing in the buffer for a considerable amount of time.

IV. Performance Evaluation

In this section, the experiments are performed to compare PT-LRU with other ones, namely LRU， CFLRU, CCF-LRU, APB-LRU, AD-LRU, and LRU-WSR, with respect to the buffer hit ratio, the number of write operations, and the overall runtime. Performance measurements are gathered by running the synthetic traces as well as the real workloads collected from enterprise servers, which are the static write/read ratio trace data, and dynamic write/read ratio trace data, respectively.

A. Experiment Setup

The simulation experiments are carried out under a flexible simulation environment called Flash-DBSim [11]. Flash-DBSim is used to the performance evaluation, because it is reusable, configurable and flexible for evaluating flash-based buffer replacement algorithms. In the experiments, an NAND flash chip with 64 pages per block is simulated in the platform. The erase limitation is 100,000 cycles per block. The detailed parameters settings of the selected flash chip are listed in Table II.

TABLE II
PARAMETERS CHARACTERISTICS OF THE FLASH MEMORY

Parameters	Value
Page Size	2,048B
Page Count per Block	64pages
Block Count	1,024blocks
Read Cost	25us/page
Write Cost	200us/page
Erase Cost	1.5ms/block
Erase Limitation	100,000 cycles

Here, a new challenge appears in choosing parameters of those algorithms. During the performance evaluation, the hit ratio of each buffer replacement policy is closely related to the parameters settings of the algorithm. The parameter *w* of the CFLRU algorithm is set to 0.5, which means half length of the buffer is used as the clean-first region. The parameter *min_lc* of AD-LRU is set to 0.1, which specifies the lower bound of the cold queue is 0.1. Parameter lower bound of hotspots in APBLRU is set to 0.8, and the probability of evicting the clean pages is the value of erase/read ratio. The parameter *pro* in PT-LRU is set to 0.8, which means the probability of evicting a cold dirty page is set to 0.8. The impact of parameter *pro* in PT-LRU will be discussed later in this section.

B. Performance Evaluation on the Synthesized Traces with Fixed Write/Read Ratio

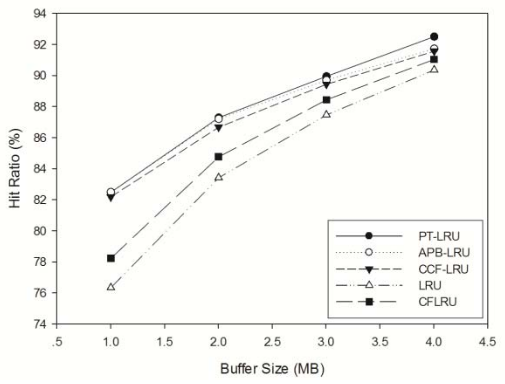
Four types of synthetic traces are picked up in the performance evaluation, which are the read-most trace (e.g., decision support systems), the write-most trace (e.g., OLTP systems), the random trace, and the Zipf trace, respectively. Total reference in table III states all the references in the trace. A read /write ratio "*x%/y%*" in Table III means that the read and write operations in the traces are of *x* and *y* percentages respectively. The locality "*m%/n%*" in Table III refers to that *m%* of total references are intensively performed in the *n%* of all pages.

TABLE III
FIVE TYPES OF SYNTHESIZED TRACES DATA

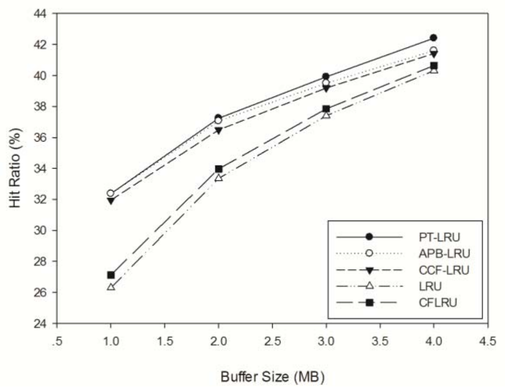
type	Total references	Pages accessed	Read/write ratio	Locality
T00	3,000,000	10,000	80%/20%	80%/20%
T01	3,000,000	10,000	50%/50%	50%/50%
T10	3,000,000	10,000	90%/10%	80%/20%
T11	3,000,000	10,000	10%/90%	80%/20%

Fig. 7 shows the hit ratios of various buffer replacement algorithms under different buffer sizes. Under various traces data, the hit ratio of PT-LRU is higher than that of other policies in most cases. PT-LRU has the better performance, because it considers the properties of pages more comprehensively and detailedly when eliminating victims. Firstly, it does not only evict the cold clean pages which decrease the hit ratio, but also choose cold dirty pages with low reference frequency as victims. Moreover, it may choose hot clean pages to be the victim pages occasionally. Although the probability of evicting hot clean pages is much lower than that of evicting cold dirty pages, it eventually achieves a higher hit ratio when many cold dirty pages in the buffer will be referenced again soon. In these typical scenarios, since other algorithms are not concerned with this problem, they will consequently result in poor performance. Furthermore, PT-LRU is also a well-considered policy on hot dirty pages. The reason is that it uses the second-chance policy to move some dirty pages with infrequently references into the MRU position of the LD queue.

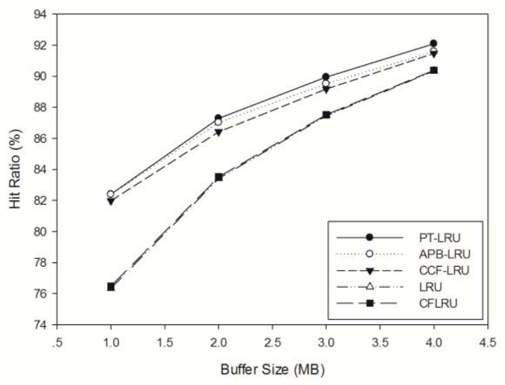
Fig. 8 gets the whole write counts of the physical pages flushed to the NAND flash memory. The write counts consist of the number of evicting dirty pages during page replacement and the total number of the dirty pages remaining in the buffer at the end of the simulation. After running all those algorithms in Flash-DBSim, these results are obtained. As LRU is not natively designed to reduce the number of writing pages back to storage devices, it gets the maximum value of the write counts among these policies. CFLRU is the first novel policy for flash memory cache. The write counts of CFLRU are considerably lower than these of LRU, on account of differentiating pages into clean and dirty ones and also delaying the eviction of dirty pages. Then CCF-LRU generates the third largest write counts, owing to considering the characteristics of data pages much thoroughly. It not only differentiates clean pages into hot and cold ones, but also considers the cold-hot differentiation among dirty pages. In the next, APB-LRU achieves the fourth largest value of write counts due to the greater probability of choosing cold clean pages than that of electing cold dirty pages as victims all the time. The write counts of PT-LRU are minimized under different traces in most cases. Because it preferentially evicts the LRU cold clean pages from LC queue, and then has a higher probability of evicting cold dirty pages.



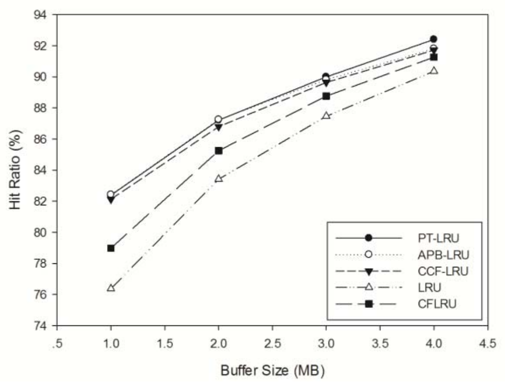
(a) T00



(b) T01

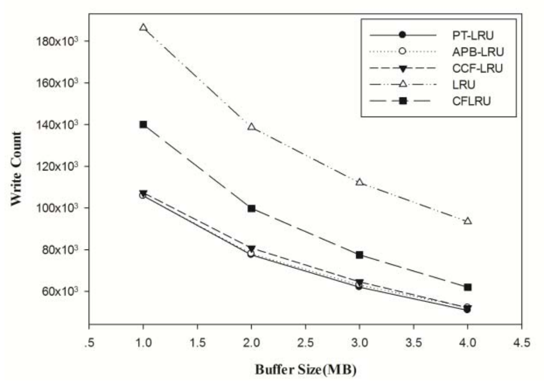


(c) T10

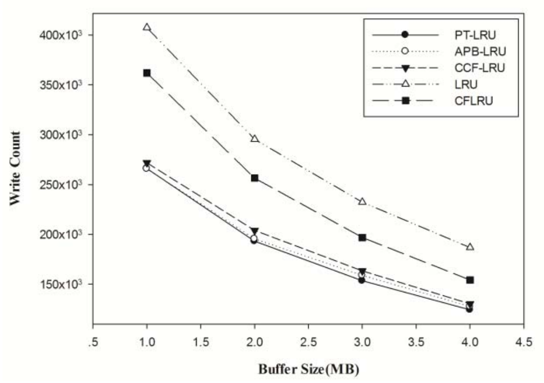


(d) T11

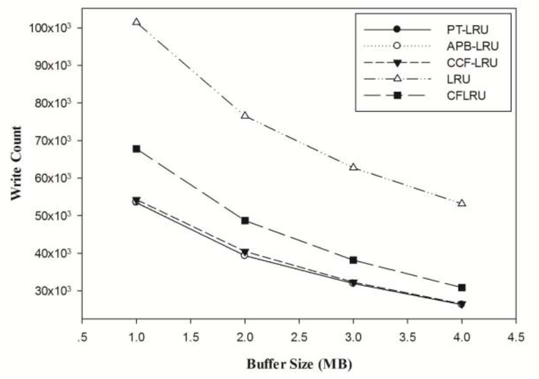
Fig. 7. Hit ratios for the traces under various buffer sizes



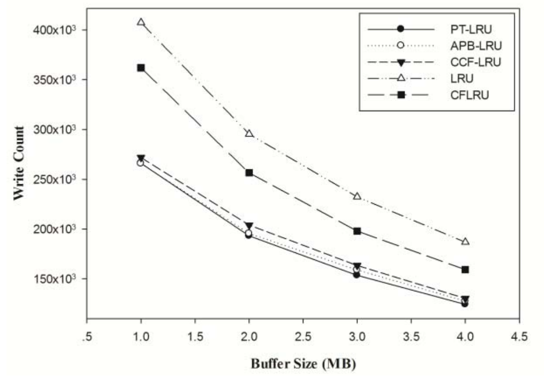
(a) T00



(b) T01

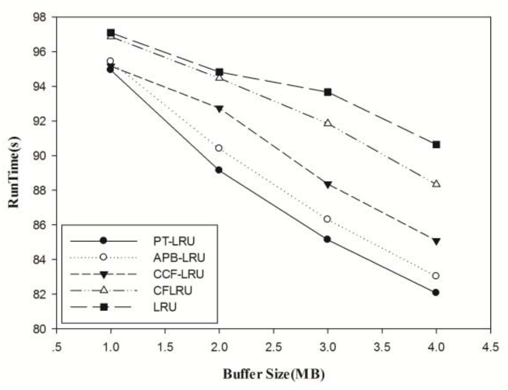


(c) T10

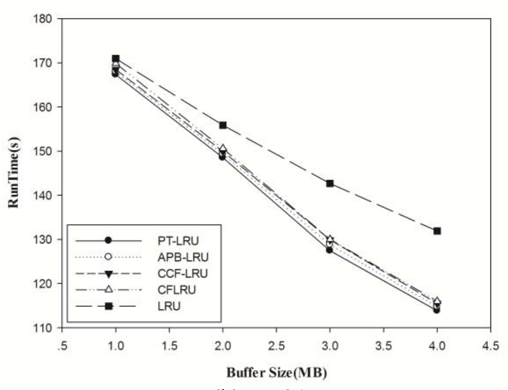


(d) T11

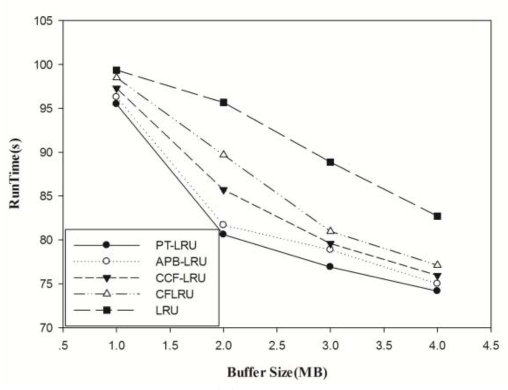
Fig. 8. Write counts for the synthesized traces



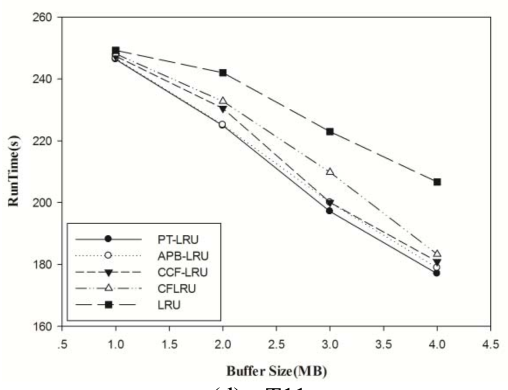
(a) T00



(b) T01

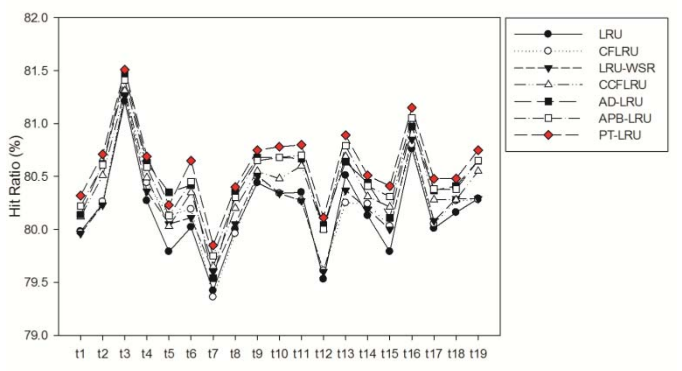


(c) T10

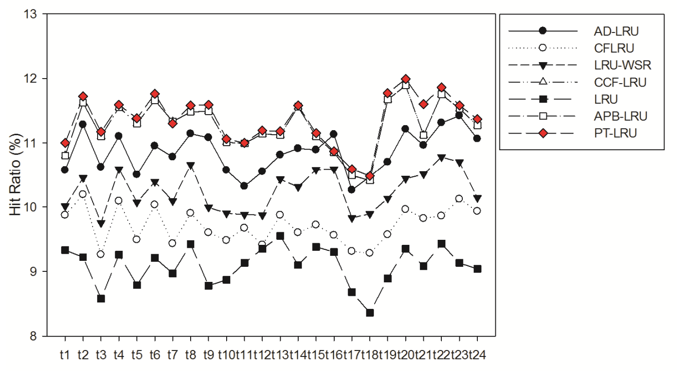


(d) T11

Fig. 9. Runtime for the traces under various buffer sizes

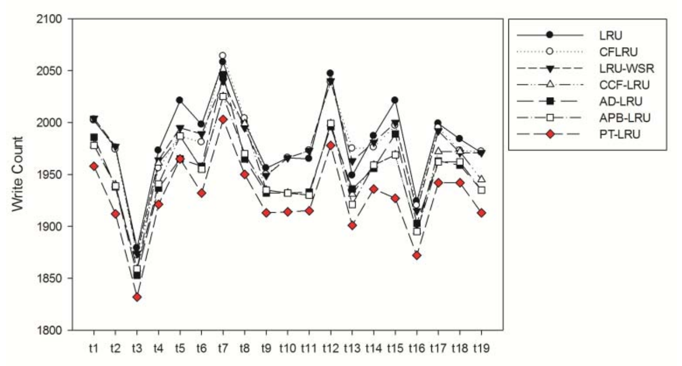


(a) Trace 1

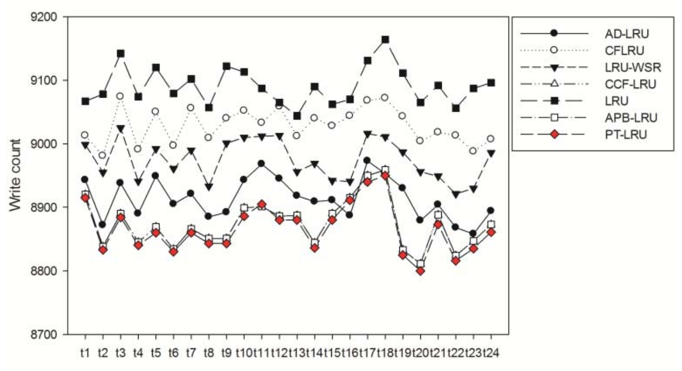


（b）Trace 2

Fig. 10. Hit ratios under dynamic write/read ratio



(a) Trace 1



（b）Trace 2

Fig. 11. Write counts under dynamic write/read ratio

Fig. 9 shows the overall runtime with different traces under various buffer sizes. Runtime mainly composes of two parts: the I/O execution time and the running time of each algorithm. The running time has a subtle impact on the overall runtime compared with the I/O execution time. The I/O execution is subject to the buffer cache hit ratio, which is mainly influenced by the read and write operations. Each operation time is determined by the physical access time (see Table I). As the hit ratio continues to increase and there is some degradation in write counts, the number of the faulted pages is expected to decrease rapidly, which leads to sharply decrease the I/O execution time. Accordingly, the overall runtime is highly influenced by the hit ratio and the number of write counts to the flash memory. Fig. 9 shows that PT-LRU has the least overall runtime among all the other replacement algorithms all the time because of its better performance in terms of the buffer hit ratio and the number of write counts.

C. Performance Evaluation on the Real OLTP Traces with Dynamic Write/Read Ratio

So far, experimental results have verified the validities of these buffer replacement algorithms with the random trace, the read-most trace, the write-most trace, and the Zipf trace, which all have the permanent write/read ratio in the mass. The two one-hour OLTP traces contain 197, 582 and 221, 638 references to 76493 unique pages, respectively. The trace files are fed to the buffer management simulator. To observe the variation trend of the overall I/O performance more intuitively and truly, the hit ratios and the number of write operations are added up every three-minute. Table IV shows the specific characteristics of the two OLTP traces.

TABLE IV
TWO TYPES OF OLTP TRACES DATA

Attribute	**Trace 1**	**Trace 2**
Total Requests	197,582	221,638
Page Size	2KB	2KB
Data Size	10GB	11GB
Duration	57min	72min
Read/Write Ratio	normal distribution	random distribution

The hit ratios of different buffer replacement algorithms are drawn in Fig. 10. LRU has the lowest hit ratio in the graph, whereas PT-LRU has the highest hit ratio in most cases. At the t6 moment in Fig. 10(a), the hit ratio of PT-LRU is obviously higher than that of others. PT-LRU overwhelms others in terms of the hit ratio in most cases. The reason is that PT-LRU evicts cold clean pages preferentially when there are enough cold clean pages in the buffer. Moreover, PT-LRU chooses cold dirty or hot clean pages as victim pages, where the probability of evicting cold dirty pages is much higher than that of evicting hot clean pages. Furthermore, with the same locality, the hit ratio of PT-LRU is much higher than that of other replacement algorithms when the read/write ratio is larger, i.e. from t9 to t11 moments in Fig. 10(a). Although those algorithms have the different hit ratios, the overall trends of those curves are practically the same as PT-LRU. Actually, some intervals are inconsistent with that viewpoint. From t9 to t11 moment in Fig. 10(b), the dynamic hit ratio curve of LRU policy is decreasing while other curves are increasing. The reason is that, in some typical scenarios, LRU can effectively evict pages while others are more complicated to choose pages as victims. The read locality of the trace is more suitable for LRU. But LRU considers fewer properties of the trace data, which leads to lower the buffer hit ratio.

Fig. 11 compares the write counts of various buffer replacement algorithms under dynamic write/read ratios. The read counts curves of those algorithms under dynamic write/read ratio are not drawn, because their overall trends are nearly the same as the write counts curves. In Fig. 11, PT-LRU outperforms the other ones in better performance in terms of write counts. The proposed PT-LRU algorithm may evict some hot clean pages, which may lower hit ratio. Nevertheless, PT-LRU responds to reduce much more write operations as well as the runtime. There is an interesting phenomenon. The read/write ratio in Fig. 11(a) exhibits a normal distribution. However, statistics suggest that the variation of the write counts is no related to the normal distribution, because the write counts are mainly subject to the buffer hit ratio.

D. Effects of Parameter pro on the Performance Evaluation of the Proposed PT-LRU Algorithm

The parameter *pro* is invoked in the proposed PT-LRU algorithm. *Pro* refers to the probability of evicting cold dirty pages. The scaling values of the parameter *pro* range from 0.5 to 1.0. *Pro* should be greater than 0.5, because the cost of evicting hot clean page is slightly higher than the cost of evicting cold dirty page, which has been explained in CCF-LRU[8]. PT-LRU evicts the cold clean pages preferentially. However, evicting the clean pages excessively will result in degrading the hit ratio seriously, and then the overall elapsed time is increased. Hence, when there is no page in cold clean list, PT-LRU should balance eviction evenly across the cold clean LRU list.

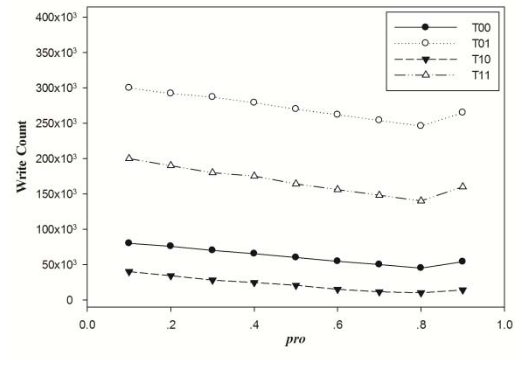


Fig. 12. Effects of *pro* under the traces

Fig. 12 illustrates the performance with parameter *pro* for the four synthesized traces used previously in this section. They are representatives of the zipf trace, the read-most trace, the write-most trace, and the random trace, respectively. According to Fig. 12, *pro* has much impact on performance. As the probability of evicting cold dirty pages is higher, the write counts are decreasing. However, when its value is greater than 0.8, it leads to serious degradation in the eviction of hot clean pages, and then the length of the hot queue will be somehow difficult to control. In that way, it deteriorates overall I/O performance.

Setting the optimal value of the parameter for various workloads is difficult. Accordingly, theoretical proofs are not developed to set the optimal value of these parameters. Nevertheless, it shows that the better set is *pro*=0.8, according to the experimental results for the four typical kinds of traces.

V. Conclusion

In consumer electronics which have the NAND flash-based storage device, the speed of a write operation is slightly slower than that of a read operation, and the speed of an erase operation is much slower than that of a write operation. To achieve high overall performance, the buffer replacement algorithms should be designed to reduce the write counts as well as the read counts. PT-LRU delays the eviction of hot clean pages by evicting the cold clean pages preferentially. When the cold clean list is empty, cold dirty pages or the hot clean pages with dissimilar probabilities of the evictions will be replaced with the newly fetched page. To avoid permanently keeping dirty pages in the buffer, the cold-page detection is used. Additionally, experimental results show the effectiveness and scalability of the proposed PT-LRU algorithm by being performed on four kinds of synthesis traces representing various kinds of workloads. Simultaneously, different from the static write/read ratio traces, the OLTP traces with dynamic write/read ratio are also invoked in the experiment. The results show that the proposed PT-LRU algorithm outperforms its competitors in terms of hit ratios, write counts and the overall runtime in most cases.

References

[1] S. H. 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.

[2] H. Jung, H. Shim, S. Park, S. Kang, and J. Cha, “LRU-WSR: Integration for LRU and writes sequence reordering for flash memory,” *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1215-1223, Aug. 2008.

[3] Z. Y. Lin, M. X. Lai, Q. Zou, Y. S. Xue, S. Y. Yang, “Probability-Based Buffer Replacement Algorithm for Flash-Based Database,” *Chinese Journal of Computers*, vol. 36, no. 8, pp. 1568-1581, Aug. 2013.

[4] H. Lee and N. Chang, “Low-Energy Heterogeneous Nonvolatile Memory Systems for Mobile Systems,” *Journal of Low Power Electronics*, vol. 1, no. 1, pp. 52-62, Apr. 2005.

[5] 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. 439-450, Sep. 1994.

[6] 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.

[7] D. Lee, J. Choi, J. Kim, S. Noh, S. Min, Y. Cho, et al., “LRFU: A Spectrum of Policies that Subsumes the LRU and LFU Policies,” *IEEE Trans. on Computers*, vol. 50, no. 12, pp. 1352-1361, Dec. 2001.

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

[9] P. Q. Jin, Y. Ou, T. Härder, and Z. Li, “AD-LRU: An efficient buffer replacement algorithm for flash-based databases,” *Data &Knowledge Engineering*, vol. 72, pp. 83-102, Feb. 2012.

[10] A. V. Aho, and J. D. Ullman, *Principles of compiler design*, Addision-Wesley Pub. Co., Boston, 1977, pp. 33-81.

[11] P. Q. Jin, X. Su, Z. Li, and L. H. Yue, “A flexible simulation environment for flash-aware algorithms,” *in Proc. 18th ACM conference on Information and knowledge management,* Hong Kong, China, pp. 2093-2094, Nov. 2009.

BIOGRAPHIES

Jinhua Cui received the B.S. degree from Southwest University, Chongqing, China, in 2012. She is currently a Ph.D. candidate of Xi’an Jiaotong University. Her major research fields are big data and flash memory based storage system.

Weiguo Wu received his B.S., M.S. and Ph.D. degrees in computer science, all from Xi’an Jiaotong University, Shaanxi, China, in 1986, 1993 and 2006, respectively. He is now a professor of Electronic and Information Engineering, Xi’an Jiaotong University. His research interests include embedded system, VHDL, cloud computing, etc.

Yinfeng Wang received his Ph.D. degrees in computer science from Xi’an Jiaotong University, Shaanxi, China. He is now with the Department of Software Engineering, Shenzhen Institute of Information Technology. His major research fields are in-memory database and cloud computing, etc.

Zhangfeng Duan received the B.S. degree from Xi’an University of Science and technology, Shaanxi, China, in 2012. He is currently a graduate student at the school of Xi’an Jiaotong University. His major research fields are index and flash memory based storage system.