

PAPER • OPEN ACCESS

A Multiple LRU List Buffer Management Algorithm

To cite this article: Xiang Wu *et al* 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* **569** 052002

View the [article online](#) for updates and enhancements.

A Multiple LRU List Buffer Management Algorithm

Xiang Wu, Delin Cai, Shujie Guan

School of Electronic Information Engineering, Anhui University, Hefei 230601, China
wuxiang_study@163.com

Abstract. Paper[1] proposes LLRU algorithm. The buffer is divided into four categories in LLRU: cold clean LRU list, cold dirty LRU list, hot clean LRU list and hot dirty LRU list. But the cold clean LRU list may be empty when the algorithm runs for a while. Then a new page just reading in buffer will be victim when the LLRU selects the victim page, leading to a clean page hard to turn into a hot page. The AM-LRU proposed in this paper ensures that the cold clean page has the chance to turn into a hot page by setting the minimum length of the cold clean LRU list and adopting secondary opportunity replacement strategy for the victim page from the hot area. As a result, compared with LLRU, the hit rate of AM-LRU can be improved and the flash reads counts, writes counts and erases counts can be reduced.

1. Introduction

With the arrival of larger data, the demand for processing speed of the system is higher, which requires the storage system to have faster response speed and greater throughput. Therefore, solid-state storage systems use flash rather than mechanical hard drives. Compared with HDD(Hard Disk Drive), SSD (Solid-State Driver) has the advantages of better performance: lower power consumption, anti-seismic, noise-free, smaller volume and so on[2]. With the emergency of large flash memory, the prices of SSD are getting lower and lower, so now SSD is being used on a large scale.

But there are differences between how SSD and HDD work. Because flash cannot be rewritten, the flash space generates garbage data (invalid data) when user data continuously write in flash and FTL(Flash Translation Layer) in SSD needs to do garbage collection (data erasure), making flash memory space available to user data. But there is a limitation to the number of times that can be erased of each block in flash[3]. When the times exceeds some value, it will become a bad block. Therefore, how to reduce the erasure and write operation to flash is a key problem. Generally, this problem can be solved in two aspects: buffer management and FTL. If the buffer management algorithm can be optimized ahead of time, it will reduce the randomness of accessed LBA(Logical Block Address). Therefore, the performance of buffer management algorithm directly affects the performance and efficiency of solid-state storage system[4].

2. Solid-State Storage System and Research Status of Buffer Management Algorithm

2.1. The structure of solid-state storage system

The FTL can divided into host based and device based[2]. Host based, which implement FTL in host, use CPU and RAM of host. Device based, which implement FTL in SSD, use CPU and RAM of SSD. Most storage systems are device based structure, as shown in Fig.1. The FTL is the core of the SSD firmware. The storage features of flash are shielded to the upper file system, and the mapping of logical address to physical address is realized. Wear balance, etc.



2.2. Research status of buffer management algorithm

As shown in Fig.1, buffer management is located above the FTL, as an important part of the storage system. Buffer management is one of the important way to improve the efficiency accessing to flash. There are many buffer management algorithms, such as LRU, CCF-LRU[5], LLRU[1], etc. The following content is a brief introduction to this algorithms:

(1) The LRU is the most classic algorithm. It organizes all the pages in the buffer into a LRU list, in which the most recently accessed pages are placed at the MRU(Most Recently Used) location and the LRU(least Recently Used) location are the least referenced pages. When the buffer is full, the page of the LRU position is directly selected to be victim. But LRU algorithms does not take into account the asymmetry of the cost of reading and writing.

(2) CCF-LRU places the pages in the buffer into two LRU lists, ML list and CCL list. The ML list hold hot clean pages and dirty pages, while the CCL list only hold cold clean pages. When selecting the victim page, as long as the CCL list is not empty, the page in LRU location of CCL is selected. when the CCL list is empty, the hot clean page in ML list will be selected. If there is no hot clean page, the dirty page in ML list will be selected based on the secondary opportunity replacement strategy.

(3) LLRU employ multiple LRU list method to organize pages in buffer. It divides pages into four categories: cold clean pages, hot clean pages, cold dirty pages and hot dirty pages. Most algorithms give priority to expel clean pages before expel dirty pages. However, LLRU proposed a mathematical model to measure the cost of expulsion when selecting pages for expulsion. The LLRU not only considers the asymmetry of read costs and write costs for flash, but also takes the different expulsions caused by different page reference frequency into account.

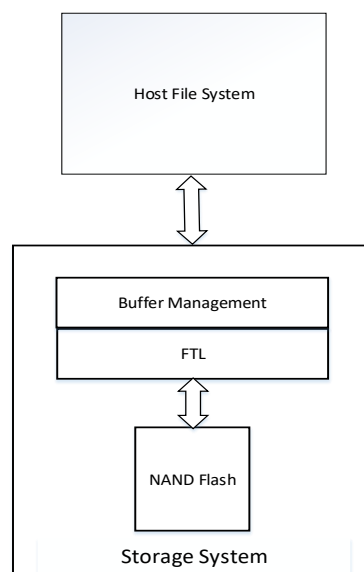


Fig.1 solid-state storage system structure

3. Adaptive Multiple LRU List Buffer Management Algorithm

3.1. AM-LRU overview

The AM-LRU proposed in this paper has the following two improvements for LLRU:

(1) LLRU based on cost function chooses victim page according to the minimum page cost. As a result, the cold clean LRU list in LLRU will be empty, leading to a deterioration in system performance. Therefore, this paper sets a minimum length `min_len` for cold clean LRU list. When the length of the cold clean LRU list is greater than or equal to `min_len`, the LRU position of the cold clean LRU list takes part in the selection of the victim page. Only select victim page in the cold dirty LRU list, hot clean LRU list and hot dirty LRU list when the cold clean LRU list is smaller than `min_len`.

(2) Using the secondary opportunity replacement strategy[6] based on cost function. The cost function proposed by the LLRU takes the page access frequency and the asymmetry between flash read and write into account. However, the pages in the hot area are still more likely to be accessed than pages in cold LRU list. The performance of the system can be further improved by employing the secondary opportunity replacement strategy for victim page from hot clean LRU list and hot dirty LRU list.

3.2. The Mathematical Model of AM-LRU

Generally, the traditional buffer management algorithm selects the victim page from two aspects :

(1) By dividing the pages into clean pages and dirty pages, firstly evicting clean pages

(2) By dividing the pages into hot pages and cold pages, firstly evicting cold pages.

But if it is cold dirty or hot clean pages, it is difficult to make the best choice. Therefore, in paper[1], a mathematical model is proposed to solve such a problem, which is also used by AM-LRU to select the minimum page cost.

$$Cost = Ec * Cnt \quad (1)$$

In equation (1), $Cost$ denote the page cost. Ec denote the cost of evict a page, including Ecc and Ecd . They respectively correspond to the evicting cost of clean page and dirty pages. For a appointed flash, Ec_c and Ec_d are constant. Cnt is times that a page has been accessed. if a page is accessed more times, its Cnt is larger.

The mathematical model uses the product of page accessed times and cost of evict a page to select the pages with the minimum page cost. Equation (1) takes account of the asymmetric characteristics of page access frequency and flash read write cost. The page of smallest $Cost$ should be selected as evicting page

3.3. AM-LRU data structure and page conversion Strategy

The pages in buffer are divided into four LRU lists: cold clean LRU list, hot clean LRU list, cold dirty LRU list and hot dirty LRU list. As shown in Fig.2, the pages in the cold clean LRU list are clean pages which are visited only once. the pages in the cold dirty LRU list are dirty pages which are only visited once. The pages in the hot clean LRU list are clean pages which are visited at least twice. Hot dirty pages, which are visited at least twice, are hold to hot dirty LRU list

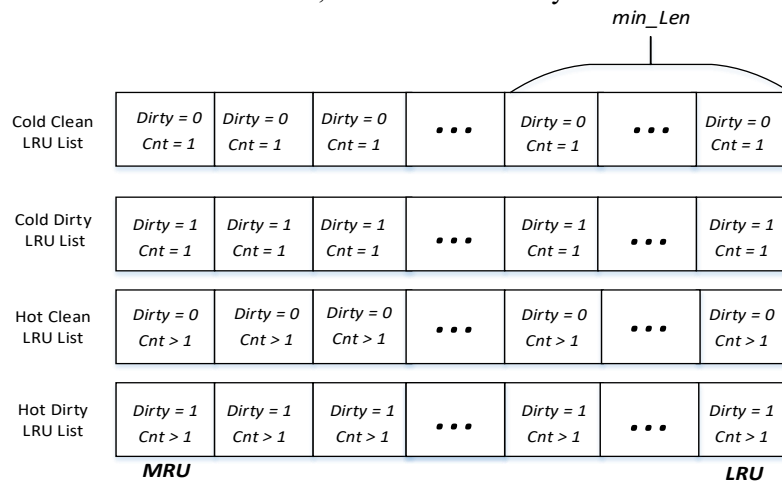


Fig.2 AM-LRU data structure

It is necessary to maintain the page status(page are dirty or clean) in the LRU list and the times of page accessed(Cnt). Where $Dirty=1$ denotes that the page is a dirty page. On the contrary, it is clean page. Cnt denotes the times of the page is accessed. When the page is accessed once, Cnt adds 1. if the new data is written to a page in buffer, the page will be set to $Dirty=1$. min_len is the minimum length of the cold clean LRU list.

The page conversion strategy of the AM-LRU follows the law shown in Table.1.

Table.1 Page conversion strategy.

Page location	Operation	
	Read	Write
Cold-Clean LRU list	Insert to MRU of HC LRU List	Insert to MRU of HD LRU List
Cold-Dirty LRU List	Insert to MRU of HD LRU List	Insert to MRU of HD LRU List
Hot-Clean LRU List	Adjust to MRU of HC LRU List	Insert to MRU of HD LRU List
Hot-Dirty LRU List	Adjust to MRU of HD LRU List	Adjust to MRU of HD LRU List

(1) When the page P located in cold clean LRU list is read, then inserting P into the MRU of the hot clean LRU list. When page P located in cold clean LRU list is written, then inserting P into the hot dirty LRU list MRU.

(2) When the page P located in cold dirty LRU list is read, then inserting P into the MRU of the hot dirty LRU list. When page P located in cold dirty LRU list is written, then P is also inserted into the MRU of the hot dirty LRU list.

(3) When the page P located in hot clean LRU list is read, then P is placed in the MRU of the hot clean LRU list. When page P located in hot clean LRU list is written, then P is inserted into the MRU of the hot dirty LRU list.

(4) The page P in the hot dirty LRU list is accessed once more. Whether it is read or written, P is placed in the MRU of the hot dirty LRU list.

(5) When the accessed page is not in the buffer, the page firstly accessed should be placed in the cold LRU list. If read, it should be placed in the cold clean LRU list, otherwise the page should be placed in the cold dirty LRU list.

3.4. Select Victim Page

The selection process of victim page of AM-LRU can be divided into two step: at first, selecting the pending victim page. Then selecting the final evicting page. Because of the limitation that minimum length of the cold clean LRU list, there are two cases that the cold clean LRU list is greater than or equal to min_len and the cold clean LRU list is less than min_len .

(1) The selection of the pending victim page

a. When the length of the cold clean LRU list is greater than or equal to min_len , the page cost of the LRU position of the four LRU lists is calculated based on the mathematical model proposed by the equation (1). Selecting the page of smallest *Cost* as the pending evicting page.

b. When the length of the cold clean LRU list is less than min_len , the cold clean LRU list does not participate in the selection process of the pending evicting page. AM-LRU Only compares the page cost of the other three LRU list. The page of minimum *Cost* is selected as the pending evicting page.

(2) The selection of the final victim page

The pending evicting page may from the hot or cold area, but the probability that the data in the hot area are accessed again is greater than that of the cold area. Therefore, if the pending evicting page are from the hot clean LRU list, it will be moved to the MRU of the cold clean LRU list and its *Cnt* will be set to 1. The pending victim page coming from the hot dirty area will be moved to the MRU of the cold dirty LRU list and the *Cnt* will also be set to 1. Then repeating entire selection process of victim page until the final evicting page is determined. If the pending victim page is located in cold area, the page will be directly chosen as final victim page.

The algorithm of selecting victim page describes as follow:

Algorithm: SelectVictim

Defintion:

CClength: the length of Cold-Clean LRU List

min_len: the minimum length set by the AM-LRU algorithm

COST_{cc}: the page cost of Cold-Clean LRU list in LRU position

COST_{cd}: the page cost of Cold-Dirty LRU list in LRU position

COST_{hc}: the page cost of Hot-Clean LRU list in LRU position

COST_{hd}: the page cost of Hot-Dirty LRU list in LRU position

PEP: the Pending eviction page

EP: the final eviction page

L_h: the hot area include Hot-Clean LRU list and Hot-Dirty LRU list

L_c: the cold area include Cold-Clean LRU list and Cold-Dirty LRU list

L_{hc}: the Hot-Clean LRU list

L_{hd}: the Hot-Dirty LRU list

Result: return victim page

1 If $CClength \geq min_len$ then

2 $COST_{cc} = Ec_c * Cnt_{cc};$

3 $COST_{cd} = Ec_d * Cnt_{cd};$

4 $COST_{hc} = Ec_c * Cnt_{hc};$

5 $COST_{hd} = Ec_d * Cnt_{hd};$

6 Else

7 $COST_{cd} = Ec_d * Cnt_{cd};$

8 $COST_{hc} = Ec_c * Cnt_{hc};$

9 $COST_{hd} = Ec_d * Cnt_{hd};$

10 Compare $COST_{cc}$ $COST_{cd}$ $COST_{hc}$ $COST_{hd}$ or $COST_{cd}$ $COST_{hc}$ $COST_{hd};$

11 find the *smallest one*;

12 $PEP = smallest\ one;$

13 If PEP in L_h then

14 If PEP is in L_{hc} then

15 delete PEP from Hot Clean LRU List to insert Cold Clean LRU list
MRU;

16 $PEP \rightarrow CNT = 1;$

17 Return $EP = SelectVictim$ //SelectVictim recursive call

18 Else //PEP is in L_{hd}

19 delete PEP from Hot Dirty LRU List to insert Cold Dirty LRU list
MRU;

20 $PEP \rightarrow CNT = 1;$

21 Return $EP = SelectVictim;$ // SelectVictim recursive call

22 Else //PEP is in L_c

23 delete PEP from L_c ;

24 Return $EP = PEP;$

4. Performance Evaluation

4.1. Simulation Platform and Algorithm Parameter Setting

Flash-DBsim[7] simulation platform is chosen to verify the performance of AM-LRU. Flash-DBsim can simulate various experimental environments in the research of flash storage system technology. According to the needs of upper layer application, The simulator can simulate SSD with different

characteristics[8]. In order to reflect the difference between AM-LRU and LRU, CCF-LRU, LLRU. This paper respectively compares the hit ratio, read flash, write flash and flash erasure times.

In this experiment, a storage device is simulated by Flash-DBsim. The storage device size is 128 MB. The detailed parameter settings are shown in Table.2.

Table.2 The detailed parameters of NAND flash

Attribute	Value
Page Size	2048 B
Block Size	64 pages
Read Latency	25 us/page
Write Latency	200 us/page
Erase Latency	1.5 ms/block
Endurance	100000

The test data set used in the experiment are generated by simulation. Different data set are used to access the simulated storage device. In this experiment, two different locality data set are generated to simulate the four algorithms. The details of test data set are shown in Table.3 .

Table.3 Test data set

Data set	Request	Locality	Read/Write
Trace1	3000000	80%/20%	65%/35%
Trace2	3000000	70%/30%	65%/35%

The locality x%/y% denotes that the x% operation takes place on the y% area, and the read/write denotes that the x% operation in all the reference is read, and the y% operation is write. In the AM-LRU, the length of the cold clean LRU list is limited. Before the simulation, the length of the cold clean LRU list should be determined. The selection of min_len is related to the test data set and buffer size. In this experiment, the min_len is 10 times of the buffer size.

According to paper [1], The optimal ratio of evicting cost between clean page *Ecc* and dirty page *Ecd* is selected as 1:18.

4.2. Results and Analysis

The experiment compares and analyzes four aspects: hit ratio, read flash count, write flash count and erase flash count for the four algorithms. For test data set trace1 and trace2, the AM-LRU obtain the best performance in four aspects.

The Fig.3~Fig.6 are diagrams of performance comparison when trace1 is used for simulation

When the buffer is 4MB, hit ratio of AM-LRU is obviously higher than that of LLRU and hit ratio is 2.10% higher. Compared with CCF-LRU, AM-LRU can improve up to 5.67% at most. Compared with LRU, AM-LRU can improve up to 9.11% at most .

When the buffer is 4MB, the read flash count of AM-LRU obviously precedes LLRU and reduced by 63001 times. Compared with CCF-LRU, AM-LRU reduce read flash count up to 170029 times at most. Compared with LRU, AM-LRU reduce read flash count up to 273192 times at most.

When the buffer is 3MB, the change of write flash count is most obvious, which is 12138 times less than LLRU. Compared with CCF-LRU, AM-LRU reduce write flash count up to 65671 times at most. Compared with LRU, AM-LRU reduce write flash count up to 213465 times at most.

The erase flash count of AM-LRU obviously precedes LLRU when buffer is 4MB, which is decreased 119 times. Compared with CCF-LRU, AM-LRU reduce erase flash count up to 1022 times at most. Compared with LRU, AM-LRU reduce erase flash count up to 3435 times at most.

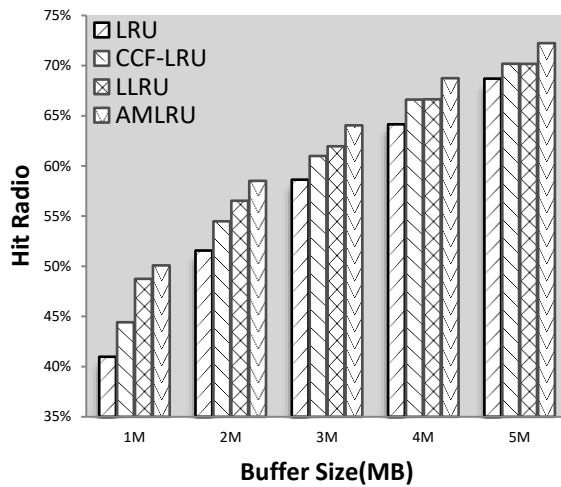


Fig.3 Hit ratio comparison of different algorithm

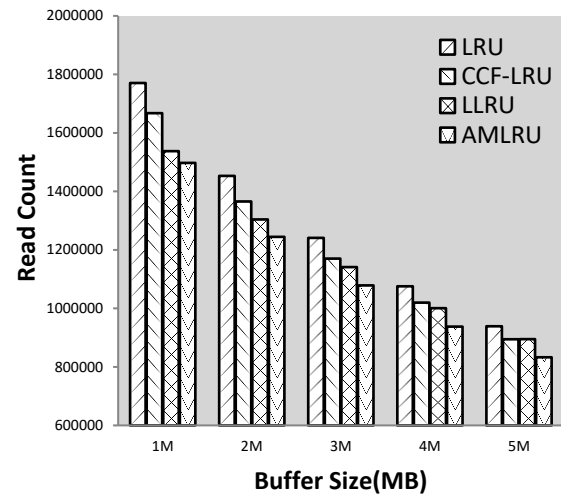


Fig.4 Read flash count comparison of different algorithm

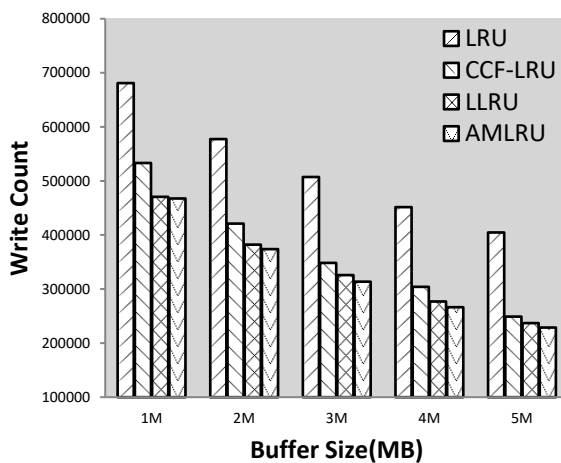


Fig.5 Write flash count comparison of different algorithm

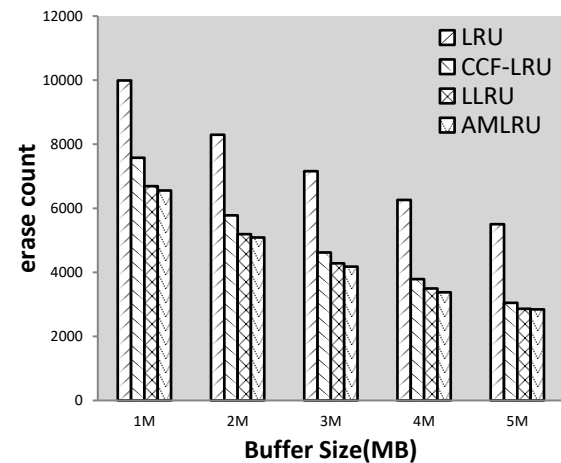


Fig.6 Erase flash count comparison of different algorithm.

The Fig.7~Fig.10 are diagrams of performance comparison when trace2 is used for simulation. When the buffer was 4MB, the hit ratio of AM-LRU is obviously higher than that of LLRU and hit ratio is 2.91% higher. The read flash count of AM-LRU obviously precedes LLRU when the buffer is 4MB, with a decrease of 87313 times. When the buffer is 1MB, the write flash count of AM-LRU obviously precedes LLRU, which is decreased by 18376 times. The change of erase flash count of AM-LRU is most obvious when the buffer is 1MB, with reduced by 286 times.

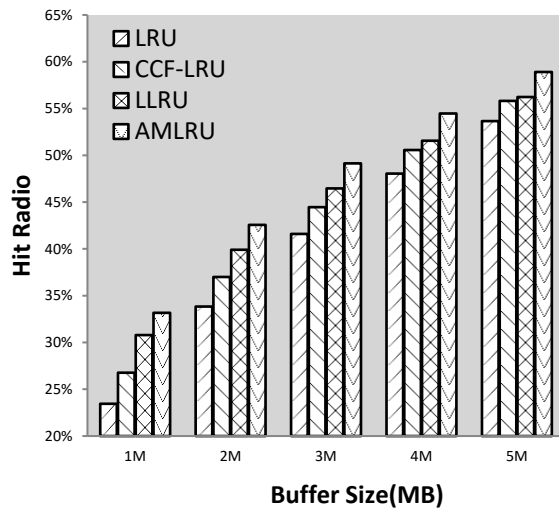


Fig.7 Hit ratio comparison of different algorithm

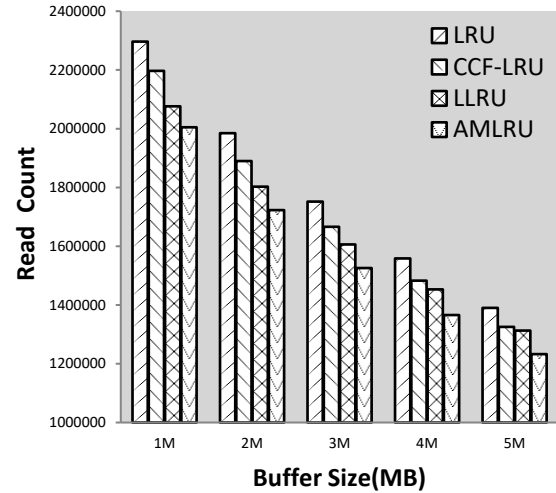


Fig.8 Read flash count comparison of different algorithm

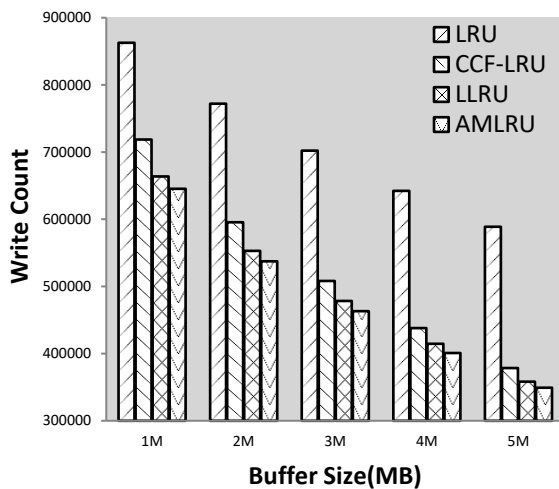


Fig.9 Write flash count comparison of different algorithm

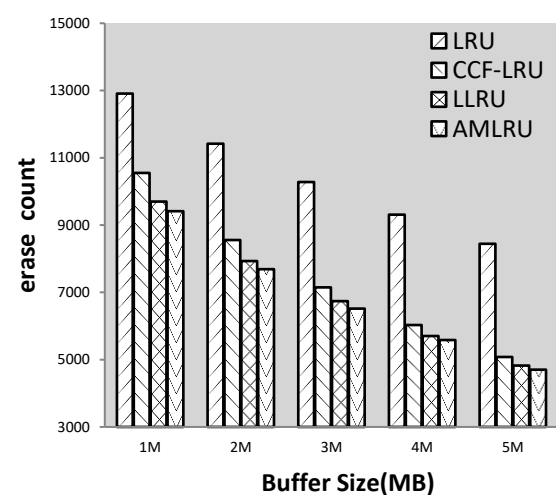


Fig.10 Erase flash count comparison of different algorithm.

5. Conclusion

On the basis of LLRU, this paper proposes AM-LRU. By setting minimum length for cold clean LRU list, it can be ensured that the pages in the cold clean LRU list can be turned into hot pages after the system runs for a period of time. Considering that the pages in the hot area are more likely to be accessed in the next reference, cost function based secondary opportunity replacement strategy is adopted for the pages in the hot area. Finally, this paper make performance comparison with CCF-LRU, LRU and LLRU in simulation platform Flash-DBsim. The environment of running Flash-DBsim is windows 7 operating system, 8GB RAM, I5-4500@2.7Ghz CPU, 1TB SSD. Through the analysis of the data, this paper can verify that the AM-LRU is superior to the other three buffer management algorithms in reading and writing times, erasing times and hit ratio of flash. Therefore, the performance of storage system also should be optimized.

References

- [1] He J, Jia G, Han G, et al. Locality-Aware Replacement Algorithm in Flash Memory to Optimize Cloud Computing for Smart Factory of Industry 4.0[J]. IEEE Access, 2017, (5):16252-16262.
- [2] SSDFans, et al. Explain Profound Theories in Simple Language[M]. Beijing: Engineering And Industry Press , 2018.(in Chinese)
- [3] K. Greene. (Feb. 16, 2008). A Memory Breakthrough. [Online]. Available: <http://www.technologyreview.com/Infotech/20148/>
- [4] Shang X, Lin Y, et al. A Buffer Management Algorithm Based on Flash Memory[J]. Computer and Modernization, 2013, (11):74-76, 81(in Chinese)
- [5] Z. Li, P. Jin, X. Su, K. Cui, and L. Yue. CCF-LRU: A new buffer replacement algorithm for flash memory[J]. IEEE Transactions on Consumer Electronics, 55(3):1351–1359.
- [6] A. Sliberschantz et al: Operating System Concepts[C]. 6th ed., John Wiley & Sons, Inc. (2004)
- [7] Jin Peiquan, Su Xuan, Li Zhi, et al. A flexible simulation environment for flash-aware algorithms[C].//The 18th ACM Conference on Information and Knowledge Management, 2009: 2093-2094.
- [8] Lin Z, et al. The concept and Technology of Flash Database.[online]. Available: <http://dblab.xmu.edu.cn>,