An Adaptive Double Area Page Replacement Algorithm for NAND Flash

Jiatong Li*, Huaixiang Hu
Fifteenth Research Institute, China Electronic Technology Group Corporation, Beijing 100083, China
*995380196@qq.com

Abstract. In order to improve the hit rate of memory pages and prolong the service life of flash memory, an adaptive double area page replacement algorithm is proposed. The buffer is divided into page area and page cluster area. The initial size of each area is set. The size of the area is dynamically changed according to the number of page hits in the two areas. The replacement unit of the page area is a single page. The CCF-LRU algorithm is used in page area replacement policy. When the page in the page area needs to be replaced, the replaced page is put into the page cluster area. The replacement unit of page cluster area is the whole page cluster. When the page area cluster needs to be replaced, the dirty memory pages in the page cluster are divided into small memory pages with the same size as the flash memory pages, and only the dirty small memory pages are written back. According to the analysis of experimental results, compared with CF-LRU, CCF-LRU and FAB, the hit rate is increased by 9%, 8% and 11% respectively.

Keywords: Flash, Page Replacement , LRU

1. Introduction
Flash memory is widely used as secondary storage in PCs, servers, mobile terminals, and computing centers [1]. Flash memory [2-4] is divided into NOR and NAND. The transmission efficiency of NOR is very high, but the erasing speed and writing speed are slow. The overall performance of NOR is not as good as NAND. So most Flash storage systems use NAND. The physical space of NAND is composed of several physical blocks. Each block is composed of several physical pages. NAND read and write operations are based on pages, and erase operations are based on blocks. Flash memory must be erased before data can be written, and there is a limit on the number of erasing [5-6]. Exceeding the limit number of erase operations will reduce the performance of the system. At the same time, the read overhead of flash memory is much smaller than the write overhead.

One way to design a flash-based page replacement algorithm is to increase the hit rate of data in the memory and reduce frequent data swapping in and out. Many flash-based page replacement algorithms have been proposed one after another: such as CFLRU (Clean First LRU) [7], FAB [8] (Flash-aware buffer management policy), CCF-LRU [9] (Clod Clean First LRU).

CFLRU is the first page replacement algorithm designed for Flash storage. The algorithm is based on the LRU algorithm. The LRU linked list is divided into Work Area and Clean First Area. Work Area consists of recently used pages, and most cache hits are in this area. Prioritize replacement of clean pages in the clean priority area. If there are no clean pages in this area, replace the dirty pages at...
the end of the LRU list. But the algorithm does not consider the problem of repeated hits of clean and dirty pages, and it is easy to replace clean and dirty pages that are frequently accessed.

The CCF-LRU algorithm adds a cold and hot detection mechanism on the basis of CFLRU. The algorithm maintains two LRU linked lists: a cold clean page linked list and a mixed linked list. CCF-LRU solves the problem that hot clean pages are easily replaced repeatedly in the CFLRU algorithm. But when the cold clean page list is empty, the clean page which was just read is likely to be replaced without becoming a hot page.

The FAB algorithm is mainly an algorithm designed for media data. A large number of sequential reads and writes will result in low algorithm performance. Therefore, FAB does not use pages as replacement units, but uses read-write blocks as units for replacement. Put the pages of the same read-write block into one group, and each group performs block replacement according to the LRU algorithm. The FAB algorithm has a small application range. Block replacement can easily increase write overhead.

According to the analysis, it can be seen that the existing flash memory-based page replacement algorithms are designed for the asymmetry of flash memory read and write, reducing the number of erasing and writing overhead. But these algorithms still have some problems. First of all, the characteristics of the data are not considered. When both sequential read and write operations and random read and write operations exist, there is no algorithm that can perform well in both scenarios. Second, the algorithm does not impose a limit on the number of cold clean pages. Clean pages that have just entered memory are easily swapped out. Finally, when the dirty page is written back, it will cause additional write overhead.

In view of the above three problems, in order to increase the hit rate of sequential read and write operations while ensuring the hit rate of random read and write data, and minimize the number of flash memory erasing and writing and write overhead, an adaptive Dual-area flash page swap algorithm. The following is a detailed description of the algorithm.

2. Algorithm principle

2.1. The total flow

The algorithm divides the cache area into page cache area (page-area, P area) and page cluster cache area (page cluster-area, C area). The P area size and the C area size add up to the entire buffer size. For each area, we set a size value, \(P_{size}\) and \(C_{size}\). When the page of the area is hit, the setting value of the area is increased by one. If the actual size of the area (the actual size of the P area is denoted by \(P_{real\_size}\) later, and the actual size of the C area is denoted by \(C_{real\_size}\) later) exceeds the initial size, when a page replacement is needed, the pages of the area are replaced. The unit of page replacement in the P area is one page, and the replaced pages in the P area will not be directly cleared out of the cache, but placed in the C area. The page replacement unit of the C area is the entire page cluster, so that multiple dirty pages can be directly written back to the flash memory at one time, thereby reducing the number of erasing of the flash memory and increasing the service life of the flash memory. The algorithm flow is shown in Fig.1. The specific replacement strategy for P and C areas is not shown in the figure.

2.2. P area page replacement strategy

P area maintains two LRU linked lists, one is a cold clean page linked list (CL), and the other is a mixed linked list (ML, contains hot clean pages, hot dirty pages, and cold dirty pages). When the page hits or is stored in the P area, the specific algorithm steps are as follows:

1) If the page hits the P area, \(P_{size}+1\), \(C_{size}-1\), and P area do not need to replace the page. If the page hits ML, move the page to the MRU end of the linked list. If it is a cold dirty page, change the mark as a hot dirty page. If the page hits the CL, put it into the MRU end of the ML and set the mark as hot.
2) If the page is a new page, determine the size of $P\_real\_size$ and $P\_size$. If $P\_real\_size<P\_size$, put it into the MRU end of CL; otherwise put the LRU end page into the C area.

![Algorithm flowchart](image)

Fig.1 Algorithm flowchart.

3) If the page is in the C area, the page is moved into the P area, $P\_size-1$, $C\_size+1$. If the page is a clean page, the operation is the same as 2. If the page is a dirty page, change the page mark to hot and put it into the MRU end of ML. At this time, it is necessary to determine the size of $P\_real\_size$ and $P\_size$.

4) $P\_real\_size<P\_size$, put into the MRU end of ML

5) $P\_real\_size>P\_size$, P area expels the LRU end page of ML. Determine the type of the LRU end page, if it is a cold dirty page, put it in the C area; if it is a hot clean page, change the mark to cold put it in the MRU end of the CL, and put the LRU end page of the CL in the C area.

1.1 C area page replacement strategy

According to the comparison between $C\_real\_size$ and $C\_size$, it is judged whether the C area needs to be replaced. If $C\_real\_size>C\_size$, the entire page cluster in the C area needs to be replaced. In this way, the flash memory can be erased and can be written back to multiple ones at a time, which is beneficial to prolong the service life of the flash memory.

The memory page size and the flash page size are generally not equal. The typical memory page size is generally 4KB, and the flash memory page size is 512B. Therefore, when a dirty memory page is written back, 8 write operations are required, and the partially written flash memory page is actually useless. In the experiment, the memory page division size needs to be adjusted according to the actual flash page size. When the page cluster replacement is to be replaced, find the dirty pages inside, and divide the dirty memory page into 8 small dirty memory pages equal in size to the flash memory page. Each small page adds a flag to compare small memory pages with flash memory pages, and set the flag to 1 if the content is different. When writing back, only the small dirty memory pages whose flag bit is 1 are written.

3. Experimental environment

The experiment is based on the Linux system and uses FlashSim for simulation experiments[10]. FlashSim is an open source SSD simulator with a variety of built-in FTL strategies. The result ob-
tained by using FlashSim is very similar to the actual running result of the flash memory system. And FlashSim can also provide various statistical information such as response time and energy consumption. Collect the read and write requests of three applications as Trace. This article selects three common application services: Database-Server, Mail-Server, and Web-Server. The specific parameters of the data are shown in Tab.1.

| Trace          | Write Request | Read Request | Unique Pages | Locality |
|----------------|---------------|--------------|--------------|----------|
| Database-Server| 2176548       | 12633458     | 564132       | high     |
| Mail -Server   | 4361260       | 507521       | 493124       | medium   |
| Web-Server     | 1138649       | 12654773     | 2968703      | low      |

Tab.1 Trace Details

4. Results and analysis

4.1. Cache hit ratio

The hit rate of the algorithm is shown in Fig.2. Under the three types of Trace, the hit rate of the DA-LRU (Double Area LRU) algorithm is higher than the other three algorithms. Under Mail-Server, the hit rate increases the most, which is 9% higher than CF-LRU, 8% higher than CCF-LRU, and 11% higher than FAB.

4.2. Write count

The fewer write operations the algorithm has in memory, the better the performance of the algorithm. Figure 3 shows the number of write operations of different algorithms in the three types of Trace. Under the three types of Trace, the number of write operations of the DA-LRU algorithm is basically the lowest. When the memory is less than 2k pages, because the memory is too small and frequent page replacement is required, the number of write operations of the four algorithms is basically the same. With the increase of the cache, the number of writes of each algorithm is gradually reduced, reflecting the advantages of the algorithm inferior. The reduction of write times in Web-Server is not as great as Database-Server and Mail-Server. This is because Web services are more sequentially accessed, and many pages are accessed only once.
The results of Mail-Server are listed in Table 2, which shows the write count and hit ratio of the four buffer replacement algorithms.

| Cache Size | 2K | 8K | 16K | 64K | 2K | 8K | 16K | 64K |
|------------|----|----|-----|-----|----|----|-----|-----|
| CFLRU      | 23.67 | 34.39 | 37.52 | 53.47 | 231 | 172 | 126 | 45  |
| CCF-LRU    | 23.38 | 35.81 | 37.97 | 54.69 | 237 | 170 | 125 | 45  |
| FAB        | 22.06 | 35.16 | 37.35 | 52.57 | 239 | 186 | 143 | 47  |
| DA-LRU     | 22.94 | 36.24 | 40.18 | 62.53 | 205 | 150 | 109 | 43  |

5. Conclusion

In order to improve the hit rate of memory pages and prolong the service life of flash memory, an adaptive dual-region flash memory page replacement algorithm is proposed. The improvement of the algorithm is as follows: ① The flash memory area is divided into page replacement area and page cluster replacement area. When writing back to flash memory, multiple pages are written back at once. Therefore, the number of flash erasing can be reduced and the life of the flash memory can be extended. ② When the page area is replaced, the cold clean pages are put into the page cluster area instead of directly discarding, so as to avoid the clean pages that have just entered the memory from being swapped out directly, which improves the memory hit rate.

Experimental results show that the algorithm has a higher hit rate than CF-LRU, CCF-LRU, and FAB algorithms in different application environments, and the number of write operations is less than that of CF-LRU, CCF-LRU, and FAB algorithms. But for Database-Server, the hit rate of this algorithm is almost equal to that of CFLRU. In Mail-Server, the number of write operation is more than CFLRU in some cases with different cache size. These problems need to be further studied.

References
[1] Fukami A, Ghose S, Luo Y, Cai Y and Mutlu O 2017 Improving the reliability of chip-off forensic analysis of NAND flash memory devices Digital Investigation 20(5) pp S1-S11
[2] Yan H and Yao Q 2014 An efficient file-aware garbage collection algorithm for NAND Flash-based consumer IEEE Transactions on Consumer Electronics 60(4) pp623-627
[3] Cai Y, Ghose S, Luo Y, Mai K, Mutlu O and Haratsch E. F 2017 Vulnerabilities in MLC NAND flash memory programming: experimental analysis, exploits, and mitigation techniques IEEE International Symposium on High Performance Computer Architecture
[4] Wenjing Z, Mingwu L and Jiwu S 2010 Flash memory technology Journal of Computer Research and Development 047(004) pp716-726
[5] Jing T and Lei T 2018 MAND flash wear leveling based on multi-threshold Microelectronics &...
Peng C, Chen W, Luo Y, Zhang F and Guo X 2020 Low-energy proton-induced single event effect in nand flash memories Nuclear Instruments and Methods in Physics Research Section A Accelerators Spectrometers Detectors and Associated Equipment 969 164064

Park S.Y, Jung D, Kang J. U, Kim J.S and Lee J 2006 CFLRU: a re-placement algorithm for flash memory CASES pp234-241

Jo H, Kang J. U, Park S.Y, Kim J.S and Lee J 2006 FAB: Flash-aware buffer management policy for portable media players IEEE Transactions on Consumer Electronics 52(2) pp485–493

Li Z, Jin P, Su X, Cui K and Yue L 2009 CCF-LRU: a new buffer replacement algorithm for flash memory IEEE Transactions on Consumer Electronics 55(3) pp1351-1359

Kim Y, Tauras B and Gupta A 2009 Flashsim: A simulator for nand flash-based solid-state drives First International Conference on Advance in System Simulation pp125-131