Improve Performance of Flash-based SSDs through Multi-Subpage Merge and Page-Level Temperature Recognition

Xiangwei Zeng

School of Information Science and Technology, JiNan University, Guangzhou 510632, China.

xzengwei1313@qq.com

Abstract

Flash memory-based solid-state disks(SSDs) have better performance than magnetic disks and are gradually replacing hard disks in desktop systems. However, although DRAM is embedded in SSD as a cache, SSD may also happen unstable write performance with continuous writing, because non-overwrite write and garbage collection(GC) operations are frequently triggered when physical pages are written. In this paper, we propose a new cache management strategy called MSAC, which manages the cache through multi-subpage merging algorithm and page-level temperature identification algorithm. We implement MSAC in SSDsim and test it with real workloads. Our experimental results show that MSAC can effectively improve SSD performance and prolong SSD lifetime by reducing the write average response time by up to 48.8% and the number of GC by up to 33.4%.

Keywords

Flash; Non-overwrite; Garbage collection; Page reconstruction; Data temperature identification.

1. Introduction

Flash memory has been gradually applied to large-scale storage systems because of its fast read-write speed, low energy consumption and light weight. Compared with hard disk drives(HDD), SSD has high random-access performance because it has no mechanical seeking time[1, 2, 3].

In addition to ordinary read and write operations, erasure operations are also one of the frequent operations in SSD. A SSD consists of several flash chips, each flash chip contains a large number of flash blocks, and each flash block is composed of a large number of flash pages[4,5,6]. Flash page is the smallest unit to read and write flash memory, and flash block is the smallest unit to erase flash memory. Flash memory has the physical characteristics that must be erased before writing, so the update operation of solid-state disk usually writes to other idle physical pages first, marks the former physical pages as invalid pages, and finally modifies the mapping relationship. This update operation is called remote update operation. Flash memory system uses remote update operation to replace the local update operation of disk. Because of this update feature, SSD will generate invalid data pages in the process of using. In order to recover these invalid pages, the software system inside SSD will issue a garbage collection (GC) command[7,8]. First, the valid pages in the recovered flash block (recovery block) will be moved to other free blocks, and then the recovery block will be erased, which completes the recovery of the recovery block. However, garbage collection will take up the bus of the flash chip. If a request is sent at this time, it will lead to a delay in waiting, which will reduce the overall performance of SSD. Therefore, it is also important to reduce the amount of garbage collected during SSD use.

2. MSAC

2.1 System Overview

A flash system of flash-based SSD hardware and software on the system including flash chip and FTL consisting of caching strategies, page mapping, wear leveling, and garbage collection. Our proposed

MSAC has three important components, including Multi-Subpage Merge(MSM), Page Level Temperature Identification(PLTI), and Merge-Page Mapping(MMAP).

We apply MSAC to the SSD internal onboard cache instead of the traditional LRU algorithm for write cache. The read and write requests sent from Applications will be parsed via File system and Block device driver and sent to the SSD through the host interface. SSD processing of read and write requests generally includes: cache management and read and write operations to the flash memory. In MSAC, the data in the cache queue will be written back to flash memory according to the MSM and PLTI algorithms. When a write back operation is triggered in buffer, MSM merges the data of multiple pages in the write cache that are less than one page size to meet the requirement of one page size for the write back operation, which will avoid non-overwrite writes. After each trigger of the MSM algorithm, a merge page map is generated for the relevant logical page. The mapping table composed of the merge page map is called MMAP.The original logical page and physical page are one-to-one mapping, which will become one-to-one and many-to-one mapping coexist after the MSM is triggered. Therefore, multiple mappings will be required when read request arrived, this can cause MSM write performance to be superior but performance to be mediocre. Therefore, we propose PLTI, using the locality of I/O workload, divide the buffer area logic into temperature search area and non-search area, and set the temperature value for each logical page in the cache. To select the appropriate logical page for the write back operation by judging the temperature value priority. Improve overall performance by mitigating read performance issues with MSM by increasing cache hit ratios.

2.2 Multi-Subpage Merge

There are many cache management algorithms in FTL. MSAC is improved on the basis of LRU, because it needs to take advantage of the locality of I/O workload. Fig. 1 shows the operating principle of the MSM model in MSAC. In the buffer node, the nodes satisfying the condition are sequentially searched from the tail (such as D10), and such nodes (such as D2) are found to merge the node with the tail node, and the two nodes will be delete from the buffer. The MSM model detects whether the valid data size of D10 is full of a page size before the D10 be write to flash. If so, the MSM operation is skipped, otherwise, enter the MSM processing function and check whether the valid data of the page full one page size starts at the end of the LRU queue. If not, it is check whether the page can be merged into one page size with the tail of queue page. If so, algorithm 1 will merge the page with the tail of queue page into a new page and write back to flash memory. At the same time, it will modify the mapping table of related page and delete the page and the tail of queue page from the buffer. If the page does not meet the MSM condition or there are no eligible pages in the LRU queue, then enter the MSTI's PLTI model for processing. The MSM model eliminates the non-overwrite writes that may occur during page write back as much as possible in the buffer and reduces the number of write operations, thus it is greatly improving the write performance of the SSD.

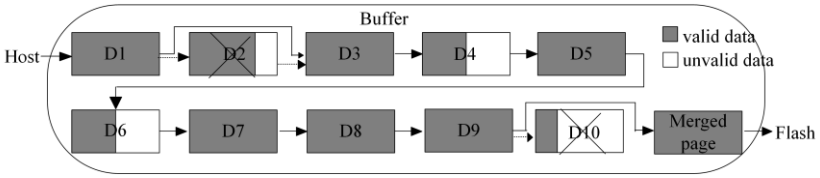


Fig. 1 An example of subpage merge operation

2.3 Page Level Temperature Identification

PLTI splitting the buffer logic into temperature search area and non-search area. When the tail page in the queue does not satisfy the MSM condition, the FTL will execute the PLTI function. If the page is cold data, write back directly to the flash memory and modify the mapping table. If the page is warm data, search for cold data from the tail of queue in the temperature search area and exchange with the page of the queue to write back to the flash memory. If no cold page in the temperature search area, the tail of the team is directly written back to the flash memory and update the mapping table. If the tail page of the queue is hot page, it will search for the cold and warm page in the temperature search area according to the write back priority(cold page > warm page > hot page). If

found the cold page and replace with the tail page of queue then write back to flash. If the cold page is not found and then the warm page will be searched, the replacement method is the same as the cold page. When there is no cold and warm page in the temperature search area, the tail hot page of queue will be write back.

3. Evaluation

3.1 Experiment environment

To evaluate the efficiency of the MSAC, we implemented the MSAC on SSDsim[2]. The SSDsim is an open-source SSD simulator that can realistically and effectively simulate the performance of FTL algorithms in hardware environments. Table 1 shows our hardware configuration parameters and performance calculation parameters. Our experiments used a 3.4GHz, 8-core Intel processor with 8GB memory.

Table 1 The SSD model parameters

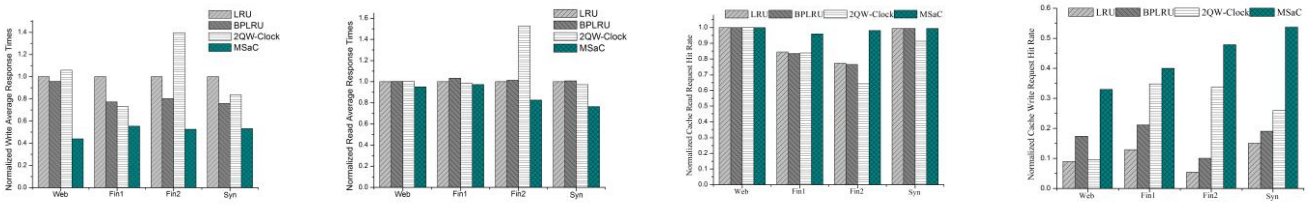
Parameter	Value
SSD Capacity	8GB
Aged Ratio	70%
Channel Number	2
Chip Numbers	4
Dies Per Chip	2
Planes Per Die	2
Blocks Per Plane	2048
Page Per Blocks	64
Page Size	4KB
4KB-Page Read	20us
4KB-Page Write	200us
Block Erase	1.5ms

3.2 Workload

We selected four I/O workloads from the real device layer of Microsoft Cambridge Research[9-13]. The web trace comes from the web search server, most of which are read request, the Fin1 and Fin2 trace are from the financial server, mainly based on write requests; and Syn trace is from the mail server, the number of read and write requests are not much different in proportion.

3.3 Performance results

Average response time: Fig.2 (a) and Fig.2 (b) shows the average response time for read and write requests normalized under four real load drivers. It can be seen from (a) that compared with LRU[13], BPLRU[1], and 2QW-Clock[5], the average response time of MSAC write requests is reduced by 48.8%, 30.8%, and 46.7%, respectively, on average. Obviously, it is because the sub-page merge module merges most of the data pages that may undergo non-overwrite write operations in the cache area, so that a large number of write operations can be directly responded with the write operation time of one page, and at the same time, it is greatly reduced. This increases the number of write operations and therefore greatly improves the response time of write requests. It can be seen from (b) that compared with the LRU, BPLRU, and 2QW-Clock algorithms, the average response time of MSAC read requests is reduced by 12.5%, 12.6%, and 9%, respectively, on average. It can be seen from Fig.2 that BPLRU has a slight decrease in read request performance. The read and write performance of 2QW-Clock in Fin2 load is reduced because about 97% of the requests in Fin2 trace are small write requests, while 2QW- The Clock algorithm is mainly aimed at improving the hit rate, and it does not perform well with lowercase requests. It can be seen that MSAC can maintain good read and write performance regardless of the type of load it serves.



(a)write request agervage response time

(b) read request agervage response time

(c) write request hit ratio

(d) read request hit ratio

Fig. 2 The normalized write and read average response time, write and read request hit ratio, driven by the real workloads

Hit ratio: In order to better understand the results of the previous average read and write response time experiments, we experimentally check the MSAC read and write request hit ratio under real load. Fig.2 (c) shows the comparison results of the write request hit ratios of MSAC, LRU, BPLRU, and 2QW-Clock after being standardized under four real trace drivers. The experimental results show that MSAC can significantly improve the write cache hit rate of SSD, compared with the other three comparative experiments, it has improved by 33.0%, 26.8% and 16.3% on average. Fig.2 (d) shows the read request hit ratio after normalization of four comparative experiments under real trace load. It can be found that the read request hit rate under the load of Web and Sys trace is close to 1, so it is difficult to improve the read performance of these two traces corresponding to that shown in Fig.2 (b). Under four loads, MSAC improves the read request hit rates by up to 21.0%, 21.0%, and 34.0%, respectively, compared to the three comparative experiments. Of course, MSAC's ability to significantly increase the write request hit rate is closely related to the PTLI model's ability to take full advantage of the locality of the load.

4. Conclusion

In this paper, we propose MSAC, and before the cache triggers a write request to write back, the write request is processed as follows: (1) When the valid data of the logical page that is about to be written back is less than a full page size, the page and the cache are Perform logical page merge operations on other logical pages of the same nature and modify and add new mapping relationships; (2) When the logical page that is about to be written back does not meet the conditions in (1), the PLTI function is triggered, which is the temperature of the logical page The value adjusts the logical page position in order of priority, and finally writes back. Experimental results show that our proposed MSAC algorithm can effectively improve SSD write performance. Compared with the three existing methods, the write performance of MSAC is improved by up to 48.8%, and the number of GCs is reduced by up to 33.4%.

References

[1] Kim, Hyojun, and Seongjun Ahn. ”BPLRU: A Buffer ManagementScheme for Improving Random Writes in Flash Storage.” FAST.Vol. 8. 2008.C.

[2] Hu Yang, Jiang Hong, Feng Dan, et al. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity [C]//Proc of the international conference on Supercomputing. ACM, 2011: 96-107

[3] Lu, Youyou, Jiwu Shu, and Weimin Zheng. ”Extending the lifetime of flash-based storage through reducing write amplification from file systems.” Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13). 2013.

[4] Wu Suzhen, Mao Bo, et al. GCaR: Garbage collection aware cache management with improved performance for flash-based SSDs [C]//Proc of the 2016 International Conference on Supercomputing. ACM, 2016: 28

[5] He Dan, Wang Fang, Feng Dan, et al. 2QW-Clock: An efficient SSD buffer management algorithm [C] //2015 IEEE 22nd International Conf on High Performance Computing (HiPC). IEEE, 2015: 47-53

[6] Seol, Jinho, et al. "A buffer replacement algorithm exploiting multichip parallelism in solid state disks." Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems. ACM, 2009.

[7] SLC vs. MLC: An Analysis of Flash Memory. In white paper. Super Talent Technology ,Inc. http: //www.supertalent.com/datasheets/SLC_vs_MLC%20whitepaper.pdf

[8] Jung H, Shim H, Park S, et al. LRU-WSR: integration of LRU and writes sequence reordering for flash memory[J]. IEEE Transactions on Consumer Electronics, 2008, 54(3): 1215-1223

[9] UMass Trace Repository. http://traces.cs.umass.edu

[10] Microsoft Enterprise Traces. http: //iotta. snia.org /traces/list/BlockIO

[11] Chan J C W, Ding Q, Lee P P C, et al. Parity Logging with Reserved Space: Towards Efficient Updates and Recovery in Erasure-coded Clustered Storage [C] //Proc of the 12th USENIX Conference on File and Storage Technologies (FAST 14). 2014: 163-176

[12] Yan Shiqin, Li Huaicheng, Hao Mingzhe, et al. Tiny-tail flash: Near-perfect elimination of garbage collection tail latencies in NAND SSDs [J]. ACM Trans on Storage (TOS), 2017, 13(3): 2

[13] Megiddo N, Modha D S. Outperforming LRU with an adaptive replacement cache algorithm [J]. Computer, 2004, 37(4): 58-65