Cache Miss Frequency based Caching Mechanism to Prevent Performance Degradation of SSD Cache

Sang-Bok Heo[1], Dong Hoon Choi[2], Jae-Woo Chang[1], Heeseung Jo[1]

[1]Dept. of Computer Engineering, Chonbuk National University, Jeonju, South Korea
{sangbokheo, jwchang, heeseung}@jbnu.ac.kr
[2]Korea Institute of Science and Technology Information(KISTI), Daejeon, South Korea
choid@kisti.re.kr

Abstract

Compared to HDD, SSD performs better than HDD and makes less noise with low power. The higher price of SSD than HDD, however, is the only obstacle for the wide adoption of SSD. Recently, in order to overcome this limitation, the SSD caching mechanism is increasingly being adopted. As an example for SSD caching mechanism, Flashcache system software in Linux makes SSD as a cache of HDD to provide the performance of SSD with the capacity of HDD. In general, Flashcache shows the best performance in write-back mode but if the size of workloads is larger than the capacity of SSD, it often shows worse performance than HDD. In this paper, we propose a cache miss frequency based replacement mechanism to prevent the performance degradation case, named the CMF mechanism. If the cache miss ratio exceed a certain threshold, the CMF mechanism makes Flashcache not to cache the content of HDD. In our evaluation, the CMF mechanism shows better performance than the conventional Flashcache by 43~65% in case of the worst case scenario.

Keywords-component; SSD cache; replacement algorithm, Flashcache

I. Introduction

SSD shows better performance with less noise and power consumption, compared to HDD. Despite of these advantages, HDD is more widely used, since SSD is still more expensive than HDD. For example, while a HDD is $156 per 1TB, a SSD is $510 per 1TB [1]. Therefore, many companies reluctant to use SSD except for high performance critical systems. One way to resolve this problem is SSD caching mechanism which uses fast SSD as a cache over HDD having large capacity.

Flashcache, one of the typical SSD caching mechanism, is developed by Facebook in 2010 and released as open source [2]. Flashcache provides three cache modes which are write-around, write-through, and write-back. Write-around mode is for the read only caching. The request data block is written directly to HDD for write operation and loaded to SSD only for read operation. In write-through mode, all data block requests of read and write are committed into both of SSD and HDD. Therefore, the performance of write-through mode is slow according to that of HDD. In the case of write-back mode, since updating is performed only on SSD, it can achieve the best performance in general among three modes. However, write-back mode includes consistency problem, since the contents of SSD and HDD can be different. Therefore, Flashcache in write-back mode flushes dirty blocks of SSD into HDD when SSD cache is full. The issue is that this consistency mechanism may occur poor performance temporarily when the size of workload is larger than the capacity of SDD, as shown Figure 1. This pre-evaluation shows the execution time of sequential 8GB write operation under 5GB SSD cache. In the case of HDD, it takes 59 seconds while Flashcache shows 100 seconds. This is caused by the replacement mechanism of Flashcache in write-back mode.

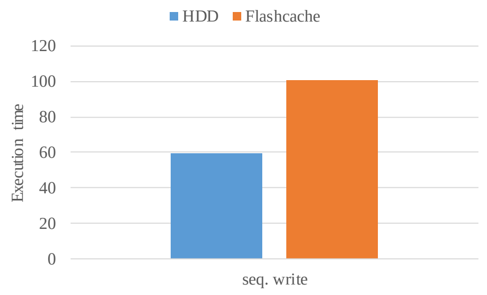

Fig. 1 The execution time of sequential write operation on each HDD and Flashcache.

In this paper, we propose a mechanism that can avoid the performance degradation of SSD even in the worst case scenario, named the CMF mechanism. The proposed mechanism monitors the cache miss ratio and temporarily stops caching when the cache miss ratio is below a certain threshold to prevent continuous block replacement between SSD and HDD.

II. Cache Miss Frequency (CMF) Mechanism

To monitor the cache miss ratio, the CMF mechanism requires to count the number of all operations and the number of hit case. Using the two values, we can calculate the cache miss ratio. If the miss ratio becomes over a preconfigured threshold value, SSD caching is turned off to prevent continuous replacement between SSD and HDD. The threshold value is empirically set to 80% in our prototype implementation.

Also, when turning off cache, the CMF mechanism stores the time which will be used for reactivating caching mechanism.

To re-enable turned-off cache, the CMF mechanism uses the number of operations and time period together. Considering that the best time to reactivate cache is after the current exacerbating workload is end, the CMF mechanism checks the number of block operations every minute. If the number of increased operations is lower than a certain threshold, the CMF mechanism guesses the workload is passed by and reactivates SSD cache again.

In this algorithm, we need to define additional values that counts the number of all operations and the number of hit case during the time period. The values are different from the overall counters and used for calculating the cache miss ratio between checking time terms. By updating the cache miss ratio to a new value, the CMF mechanism can decide caching on-off even after the cache is reactivated.

III. Evaluation

Table I shows the system specifications used for the performance evaluation of the CMF mechanism.

TABLE I
THE SYSTEM SPECIFICATIONS

Type	Description
CPU	Intel core i5-2500 3.30GHz
Mem	2GB
OS	Ubuntu 12.04
Kernel	Linux 3.10.0
SSD	840 EVO 120GB
HDD	WDC WD10EZEX-00RKKA0 1TB

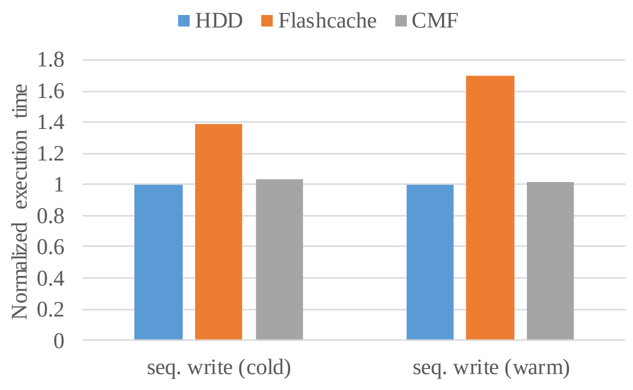

Fig. 2 The normalized execution time of sequential write operation with HDD, Flashcache, and the CMF mechanism.

Figure 2 shows the evaluation result comparing HDD, Flashcache, and the CMF mechanism. For this evaluation, we intentionally make a situation where the performance degradation occurs by using sysbench micro benchmark [3]. The size of SSD cache is configured to 5GB, and that of workload is 8GB. The cold case and the warm case denotes the status of SSD cache is cold and warm for each. In the case of Flashcache, it takes much longer time than other schemes by 82 seconds in cold and 100 seconds in warm. On the other hand, the CMF mechanism shows 57 seconds for both cold and warm, and the result is 1.75 times better compared to that of Flashcache. Although the performance of the CMF mechanism seems to be same with HDD, it is only in this worst case scenario. In other operations such as random read and write, Flashcache and the CMF outperforms HDD incomparably.

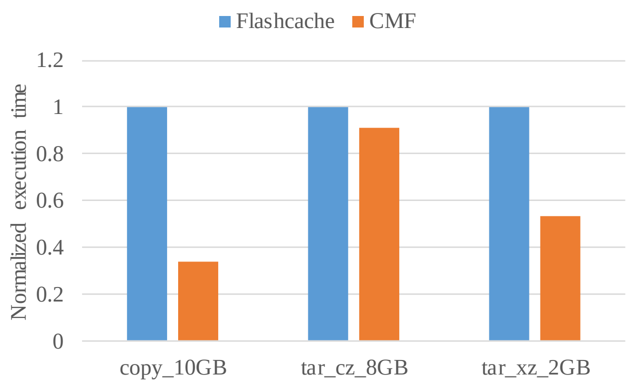

Fig. 3 The normalized execution time of real workloads.

Figure 3 shows the evaluation result using real workloads. The copy_10GB denotes copying ten 1GB mp4 video files, tar_cz_8GB denotes compress 8GB of files, and tar_xz_2GB denotes decompress a 2GB file into 8GB files. In the case of copy_10GB, while Flashcache takes 374 seconds, it takes 130 seconds in the CMF mechanism. The performance gap is 2.87 times. When compressing files, it takes 447 and 411 seconds for each. In the case of decompression, the execution times are 227 seconds and 122 seconds. Although the performance gain is different due to the characteristics of workloads, we can confirm that the CMF mechanism is highly effective from the evaluation results.

IV. Conclusion

In spite of the high performance capabilities of SSD, it is still too expensive in aspect of GB per dollar. Therefore, using SSD as a cache of HDD is an efficient approach for commercial companies. In this work, we propose and implement an enhanced mechanism named CMF for Flashcache. The key feature of the CMF mechanism is that monitoring the cache miss ratio of Flashcache and blocking temporarily the caching ability when the cache miss ratio is lower than a certain threshold. The basis of the CMF mechanism is that the write-back mode of Flashcache exacerbates the overall performance, even lower than HDD, when the size of an application workload is larger than that of SSD cache. Through the evaluation of the CMF mechanism, we confirm that Flashcache can achieve much better performance even in the worst case scenario.

For future work, we plan to optimize the CMF mechanism more and develop a dynamic cache miss threshold mechanism for various workload patterns. We expect the CMF mechanism can be applied to other SSD caching mechanisms such as bcache and dm-cache.

Acknowledgment

This Research has been performed as a collaborative research project of project Building the System for Sharing and Convergence of Scientific Big Data and supported by the KOREA INSTITUTE of SCIENCE and TECHNOLOGY INFORMATION (KISTI). It was also supported by the research project on High Performance and Scalable Manycore Operating System through the Ministry of Science, ICT, and Future Planning, Korea (grant 14-824-09-011).

References

[1] http://www.storagereview.com/ssd_vs_hdd
[2] http://en.wikipedia.org/wiki/Flashcache
[3] http://wiki.gentoo.org/wiki/Sysbench