Considering Block Popularity in Disk Cache Replacement with Solid State Drive

Mandar Tawde

Computer Engineering, Universal College of Engineering, Kaman
Survey Number 146, Chinchoti Anjur Phata road,
Near Bhajansons and Punyadham,
Vasai, Thane, Maharashtra 401212
mandar258@gmail.com

Abstract — **The Solid State Drive (SSD) is now becoming a main stream in storage systems. It is widely deployed as cache for hard disk drive (HDD) to speed up the execution of data intensive applications. In this paper we propose a novel block replacement algorithm for flash-based disk cache, named Block Replacement based on Popularity (BRP). Using the frequency and recency of block access, it calculates the block popularity to select the block which will be evicted from SSD. This avoids cache pollution and keeps popular blocks in SSD cache, leading to high hit ratio. Meanwhile, the proposed scheme reduces block replacements, and thus incurs less write operations to SSD. As a result, BRP enhances the performance of storage and the lifetime of SSD. Computer simulation demonstrates that the proposed scheme consistently outperforms five existing cache replacement algorithms with two different kinds of traces.**

Keywords — **Disk Cache, Solid State Disk, Cache Replacement, Block Popularity, Ghost Queue.**

I. INTRODUCTION

NAND flash-based Solid State Drive (SSD) is now widely used in various computing environment such as laptop, desktop, and server storage system due to numerous merits including non-volatility, fast random access, shock resistance, small size, and low power consumption. SSD has attracted a lot of attentions recently as effective replacement of Hard Disk Drives (HDDs). However, the price of SSD is still higher than HDD although it is decreasing rapidly. Therefore, SSD is desirable to serve as cache storage of HDD for cost efficiency.

Meanwhile, SSD has a few undesirable features including limited erase/write cycles and drastically shortened lifetime with excessive operations. These are serious issues, and need to be taken care of by software approach. Efficient disk cache management is required at the HDD level to avoid unnecessary writes to SSD. For example, sequential access pattern is often observed in the scan operation handled with typical data analysis applications. If the LRU policy is applied to manage the HDD, all sequentially-accessed data blocks will be moved from HDD to SSD. Caching these data blocks in SSD inevitably requires a large number of write operations to SSD, but without any benefit of caching as these blocks are never reused again. On the other hand, cache pollution occurs as the spaces are wasted to cache useless data blocks. The traditional cache management is not suitable for SSD cache.

In this paper, thus, we propose a novel block replacement algorithm for SSD-based disk cache, named Block Popularity Replacement based on Popularity (BRP). BRP is different from other algorithms in considering the frequency and recency of block, trying to keep hot blocks in SSD cache and evict seldom accessed blocks out of SSD cache. BRP uses block popularity to identify potentially popular blocks. As a result, blocks in SSD cache tend to be more popular and kept longer, leading improved the hit ratio. Meanwhile, BRP reduces the number of block replacements and hence incurs less write operation to SSD.

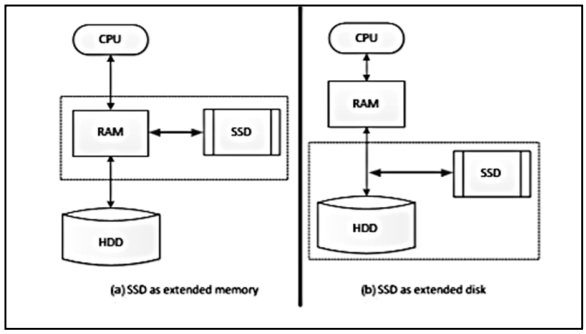
The rest of the paper is organized as follows. In Section 2 the basics of SSD and the issues with the existing cache management algorithms are discussed. Section 3 introduces the proposed scheme and the conclusion is given in Section 4.

II. BACKGROUND AND RELATED WORKS

This section introduces the hardware characteristics and the operational mechanism of flash memory along with the software layer providing the block device interface.

A. SSD-BASED CACHE

There are basically two usage models for SSD-based disk cache (Fig. 1). In the first model, SSD is used as an extension of system memory. In this model RAM and SSD are managed as a single unified cache tier for HDD. In the other model (Fig.2), it is used as extended disk lying beneath the standard block interface to serve as a second level cache. It is transparent to other components in the system.



Due to the inclusive property of multi-level cache, the extended disk model is often inferior to the other one with respect to hit ratio. However, implementing the extended memory model usually involves the modification of the operating system or the application itself. This is expensive or even impossible often. Consequently, the extended disk model is more common in the field, and it is adopted in this research.

B. Cache Replacement

The LRU replacement policy is widely used for the replacement of cache because of the simplicity, which replaces the least recently used page in the cache. It performs well in some applications where a burst of references to infrequently used blocks may cause replacement of commonly referenced blocks in the cache [3, 13]. The LRU-K [2] replacement policy considers the time of the K^{th}-to-last reference to the page. It replaces the page of the least reference frequency and the least recency of the reference.

The existing replacement algorithms for traditional hard disks were designed to minimize the page miss ratio assuming the cost of page read and write are equal. In the case when flash memory is used as storage medium, it is necessary to reflect the characteristics of flash memory in cache management. This paper distinguishes itself from the existing algorithms in considering the recency and frequency for block or page replacement. The residence time in SSD Queue and hit count of a block are used to decide the popularity.

III. THE PROPOSED SCHEME

In this section the proposed BRP scheme is presented, which exploits the popularity of blocks for block replacement with SSD cache. According to LARC, the accessed blocks are classified into two types, one time accessed blocks and blocks accessed more than one time. The second type block is most likely to be accessed again later, and thus need to be kept in the SSD cache. The BRP scheme calculates the popularity of blocks to keep the seldom accessed blocks out of SSD Cache. It will prevent cache pollution and lead to high hit ratio and less write operations.

A. The Basic Approach

The basic idea of BRP is to properly identify seldom accessed blocks and evict them out of the SSD cache. This is achieved by the notion of block popularity, which is managed by S_Queue. It also adopts ghost queue as a filter used to reduce the SSD write operations. S_Queue is a queue whose entry indicates the value of popularity of the corresponding SSD block, with the block of highest (lowest) popularity at the head (tail). Ghost queue contains the metadata of the block referenced once, with the most (least) recently used one at the head (tail). Algorithm 1 shows the operation of the proposed scheme using ghost queue for block replacement.

The size of Ghost Queue (SizeGQ) indicates how long a block needs to be placed in Ghost Queue. Obviously, large size Ghost Queue leads to high hit ratio, but also increases write operations. Therefore, SizeGQ is dynamically changed according to the hit ratio of SSD cache. The self-tuning policy will be effective for coping with varying block access patterns.

Algorithm 1. Ghost Queue-based Replacement

```
Initialize : SizeGQ = 0.1 * SizeSQ
if requested Block Bn is in S_Queue then
    SizeGQ = max(0.1 * SizeSQ , SizeGQ - SizeSQ / SizeSQ - SizeGQ)
    Redirect the request to the corresponding SSD block
    return
end if
SizeGQ = min(0.9 * SizeSQ, SizeGQ + SizeSQ / SizeGQ)
if Block Bn is in GhostQ then
    Remove Block Bn from GhostQ
    Allocate the Block Bn to the S_Queue
    if SSDQ > SizeSQ then
        Remove the lowest popularity of Block in S_Queue
        Allocate the Block Bn to S_Queue
    else
        Allocate the Block Bn to free S_Queue Block
    end if
else
    Allocate Block Bn to the MRU block in GhostQ
    if GhostQ > SizeGQ then
        Remove the LRU Block of GhostQ
    end if
end if

*SizeSQ : Size of SSD Queue, SizeGQ : Size of Ghost Queue,
```

Fig. shows the three cases of operations of block move,

- Case (1): When a request for a block arrives, SSD Queue is first checked if the requested block is in the SSD cache or not. If true, the access information on the block is updated.
- Case (2): If the requested block is not found, Ghost Queue is checked for its presence. Then the requested block is moved to the SSD cache by referring to the metadata of the block in Ghost Queue.
- Case (3): When the requested block is not in both SSD Queue and Ghost Queue, the metadata of the request block is put in the head of Ghost Queue.

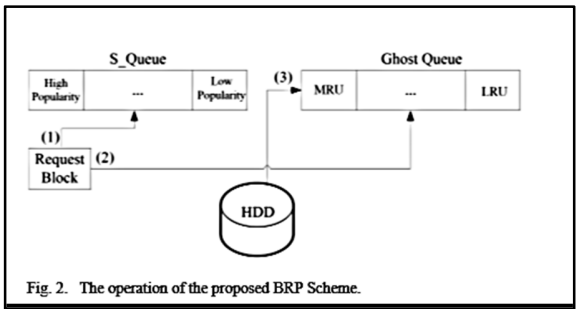


Fig. 2. The operation of the proposed BRP Scheme.

The operation of the proposed BRP Scheme.

B. Block Popularity

To avoid cache pollution, a new metric called Block Popularity is adopted, which is decided by residence time and, hit count of the block. It is calculated using the variables summarized in Table I.

TABLE I.
VARIABLES USED TO CALCULATE BLOCK POPULARITY

Variable	Description
P(n)	Popularity of Block_n
TA(n)	The last access time of Block_n
TR(n)	Residence time of Block_n
H(n)	Hit count of Block_n

The residence time of a block, TR, is defined as the time interval between the current time t, and the last access time of the block. It is thus

$$T_R(n) = t - T_A(n) \quad \text{----------(1)}$$

TR of a block indicates how long the block has been staying in SSD cache since the last reference. The second information on a block is the hit count, defined as the number of read/write hits on the block while it is in the SSD cache. When a block is put in SSD cache, the block hit count is initialized to 1, and it is increased whenever the block in SSD Cache is requested.

Using the block hit count and residence time, the popularity of block n, P(n), is obtained as follows.

$$P(n) = \frac{H(n)}{T_R(n)} = \frac{H(n)}{t - T_A(n)}$$

-------- (2)

If TR(n) is 0, Block_n has just been put in SSD Queue. P(n) is initialized to 1. After the block is requested again, the P(n) value is changed. If the residence time of a block is not taken into account in deciding its popularity, the temporal locality is missed. Even with the same hit count, the blocks perhaps have different residence times, and consequently different probability of access in the future.

Compared to the existing schemes, the proposed scheme can effectively allocate the blocks to SSD cache based on the block popularity. Fig. 3 shows an example of cache replacement with the popular LRU cache replacement policy.

Here the SSD cache is composed of eight blocks. B43 is the LRU block when SSD Queue is full, and thus it has the highest priority of eviction from the cache. On the other hand, B53 is MRU block, and thus it will stay long in the cache (also SSD Queue). B43 has only one hit after located in SSD Queue, it has not been requested again until t = 1000. However, when requested at t = 1000, it is moved to the head of the queue.

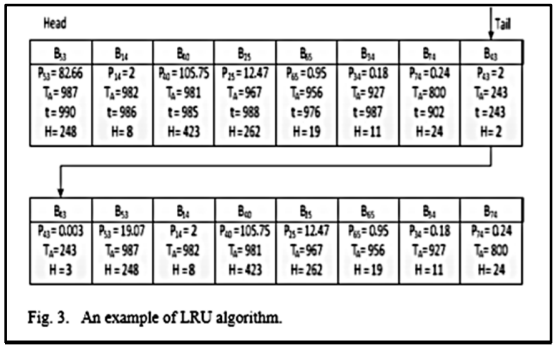


Fig. 3. An example of LRU algorithm.

Fig 4 shows the operation of the proposed BRP scheme for the same situation with Fig. 3. The first row of Fig. 4 is sorted by block popularity. Observe from the figure that B43 was first located in the middle of the SSD Queue. Recall that the P value is initialized to 1(P43 = 1) when the block is inserted in SSD Queue for the first time. At t = 1000, B43 is requested again as in Fig. 3, where it was located in the MRU position. Nevertheless, it is moved to the tail of the SSD Queue with the proposed scheme since B43 has the lowest popularity. In other words, B43 is "cold" block. If B43 is not requested again, it wastes the capacity of SSD cache until finally evicted from SSD Queue. Observe that the residence time of B43 in Fig. 3 is much longer than that of Fig. 4. In the long run, B43 will degrade the hit ratio. The policy employed in the proposed scheme can effectively prevent cache pollution and keep "hot" blocks in SSD Queue for higher hit ratio.

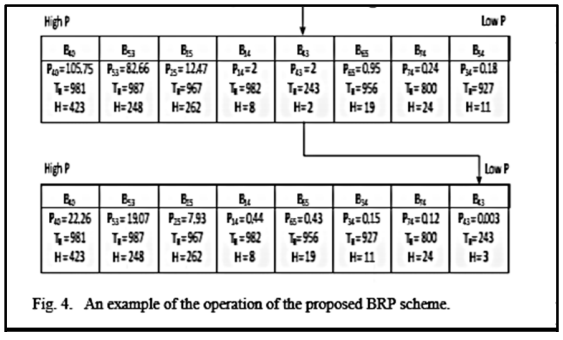


Fig. 4. An example of the operation of the proposed BRP scheme.

Algorithm 2. Cache Replacement with Block Popularity

```
1: if requested Block Bn is in GhostQ
2:     Block Bn -> Hit Count = 1;
3:     Block Bn -> Block Popularity = 1;
4:     Block Bn -> Last accessed time = Current accessed time;
5:     Allocate the Block Bn  in  S_Queue
6: end if
7: if requested Block Bn  is in  S_Queue
8:     Block Bn  -> Hit Count ++;
9:     Update the Block Bn -> Last accessed time;
10:    Block Bn BP = Hit Count /residence time;
11:    Update the Popularity of Block Bn
12: end if
13: else
14:    if  S_Queue is Full then
15:        for each Block Bn -> BP in  S_Queue do
16:            Calculate Bn -> BP = Hit Count / t – Last accessed time;
17:            Evict the lowest Popularity of Block Bn in  S_Queue
18:            Allocate the requested Block Bn of  GhostQ to S_Queue
19:        end for
20:    end if
```

Algorithm 2 presents the cache replacement with block popularity. The scanning process (line 14-20) computes popularity of all the blocks in the SSD Queue. Lines 1-6 are for a block inserted in the SSD Queue first time. Lines 7-12 are for updating the hit count when the requested block is hit. Also, the residence time in SSD Queue is calculated based on the requested time. Using the block information, BRP updates the popularity of the blocks. According to the calculated popularity of the block, the block of lowest popularity will be evicted from SSD Queue, and then a new block which is hit from the Ghost Queue is allocated.

IV. CONCLUSION

In this paper we have proposed an enhanced cache replacement scheme for flash-based disk cache. It considers both the frequency and recency of the access to the blocks to decide whether the blocks are kept in SSD or not. This avoids cache pollution and keeps popular blocks in SSD cache, leading to high hit ratio. Also, the proposed scheme reduces the frequency of block replacement, and thus incurs less write operations to SSD. Consequently, it enhances the performance of the storage and lifetime of the SSD.

REFERENCES

S. Huang et al., "Improving Flash-based Dish Cache with Lazy Adaptive Replacement," in Proc. Mass Storage Systems and Technologies(MSST),2013 IEEE 29th Symposium on, 2013 May 6- 10,pp. 1-10

SanghyunYoo.," Considering Block Popularity in Disk Cache Replacement for Enhancing Hit Ratio with Solid State Drive", SNPD 2015, June 1-3 2015