Co-Active: A Workload-Aware Collaborative Cache Management Scheme for NVMe SSDs

Hui Sun, Shangshang Dai, Jianzhong Huang, *Member, IEEE*, and Xiao Qin, *Senior Member, IEEE*

Abstract—When it comes to NAND Flash-based solid-state disks (SSDs), cache can narrow the performance gap between user-level I/Os and flash memory. Cache management schemes impose relentless impacts on the endurance and performance of flash memory. A vast majority of existing cache management techniques adopt a passive data-update style (e.g., GCaR, LCR), thereby undermining response times in burst I/O requests-based applications[1]. To address this issue, we propose a collaborative active write-back cache management scheme, called *Co-Active*, customized for I/O access patterns and the usage status of a flash chip. We design a *hot/cold separation* module to determine whether data is cold or hot in workload. When a flash chip is idle, cold and dirty data in the cache is flushed into the idle flash chip to produce clean data. To curtail cache replacement cost, clean data are preferentially evicted amid the procedure of cache replacement. A maximum write-back threshold is configured according to the level of burst I/O requests in workload. This threshold is intended to avert redundant write I/Os flushing into flash memory, thereby boosting the endurance of flash memory. The experiments are conducted to validate the advantages of Co-Active in terms of average response time, write amplification, and erase count. The findings unveil that compared with the six popular cache management schemes (LRU, CFLRU, GCaR_CFLRU, LCR, and MQSim), Co-Active (1) slashes the average response time by up to 83.89 percent with an average of 32.7 percent; (2) drives up the performance cliff degree by up to 76.4 percent with an average of 42.3 percent; and (3) improves write amplification rate by up to 60.5 percent with an average of 5.4 percent.

Index Terms—Cache management, NAND flash, solid state disks, proactive write back

1 Introduction

NAND Flash-based solid-state disks (SSDs) are rapidly gaining popularity in storage systems thanks to high random performance and low power consumption, compared with hard drives (HDDs). SSDs are an appealing alternative to HDDs in large data-intensive computing, high-performance computing, and enterprise applications [1], [2].

A portion of DRAM, residing as a middle man between user-level I/O requests and flash memory, is performing as cache space to boost I/O performance. An I/O request page with the same logic page address (i.e., LPA) as the previous request page can be serviced by the cache to significantly shorten the average response time of I/O requests [3]. In addition, the cache reduces the frequency of I/Os accessing flash memory; thereby, enhancing the utilization of program/erase cycles (P/Es) and improving the endurance of flash memory. It is evident that a cache management scheme largely impacts the SSD performance duo to cache space limit [4]. Flash memory-aware caching techniques were proposed to incorporate temporal and spatial locality in workloads [5], [6] to minimize replacement costs induced by cache misses [7], [8]. The others promising cache management schemes cope with the real-time status of the underlying flash memory (e.g., garbage collection status and chip load) [9], [10], [11]. All these existing strategies make use of the write-back method to handle data updates, meaning that only when cache space is full, cached data can be evicted into the flash memory. Such policies leads to an excessive number of misses of new I/O requests. This cache update policy strives to place data in the cache while giving rise to uneven I/O accesses to flash memory.

It is worth noting that data distributions in flash memory is reliant on I/Os access patterns in case of a fully-loaded cache. The data-intensive degree determines the frequency of I/Os access to flash memory. Burst I/Os account for a large proportion of real-world workloads (see Section 6). Facing burst I/O accesses [12], write-back update methods lead to sharply deteriorate SSD performance [13], thereby impairing response time. This problem was addressed by a handful of novel update methods in cache management systems. For instance, Huang *et al.* [14] proposed an efficient caching scheme that periodically writes dirty data in a cache into flash memory during an idle period. MQSim [15] writes cached cold data into flash memory whenever the cache stores new I/O data. Once data is written into the flash memory, the data in the

1. Burst I/O requests must be served in a real-time manner. This type of I/O access pattern is prevalent in data-intensive workloads.

- *Hui Sun is with the School of Science and Technology, Anhui University, Hefei, Anhui 230601, China, and also with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100864, China. E-mail: sunhui@ahu.edu.cn.*
- *Shangshang Dai is with the School of Science and Technology, Anhui University, Hefei, Anhui 230601, China. E-mail: shangshangdai@stu.ahu.edu.cn.*
- *Jianzhong Huang is with the Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: hjzh@hust.edu.cn.*
- *Xiao Qin is with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849-5347 USA. E-mail: xqin@auburn.edu.*

Manuscript received 22 July 2020; revised 3 Dec. 2020; accepted 5 Jan. 2021. Date of publication 15 Jan. 2021; date of current version 28 Jan. 2021. (Corresponding author: Jianzhong Huang) Recommended for acceptance by S. Pakin. Digital Object Identifier no. 10.1109/TPDS.2021.3052028

1045-9219 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

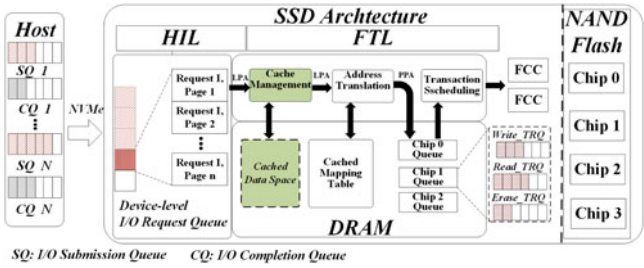


Fig. 1. A typical architecture of a modern NVMe SSD.

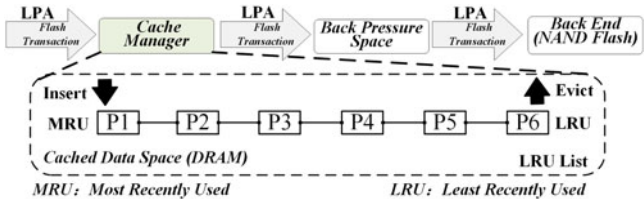


Fig. 2. A schematic diagram of LRU cache management scheme.

cache are purged. This method prevents the cache from being fully loaded and; therefore, free space in the cache is released to service I/O requests and to avert cache-replacement overhead. Although these caching schemes avoids the performance degradation caused by burst I/O requests, a huge amount of data written into flash memory adversely impact the endurance of flash memory. Because each block has a limited number of P/Es, a cache management scheme ought to optimize the endurance of flash memory as well as the overall SSD performance.

In this study, we propose a Co-Active cache management scheme, which is collaboratively aware of I/Os access patterns and the status in flash chips. Co-Active mitigates the performance degradation and performance cliff triggered by burst requests under I/O write-intensive workloads. When a chip monitor unit (see Fig. 4 in Section 4.1) in the Co-Active detects an idle flash chip, the proactive write-back module actively flushes dirty and cold data from cache to flash memory in order to originate clean data. Clean data is preferentially evicted to minimize the overhead of cache misses, because replacing clean data eschews extra write operations issued to flash memory. Co-Active is adept at alleviating performance degradation during burst I/Os under write-intensive workloads. The write-back threshold helps in suppressing the excessive number of write I/Os flushed into flash memory and; as a result, high endurance of flash memory is achieved.

The contributions in our study are summarized as follows.

▷ In Co-Active, we develop a novel data organization orchestrating cache space. Two LRU linked lists are crafted to store clean data and dirty data, respectively. When the active write-back mode is triggered, data in the dirty data LRU linked list (i.e., dirty LRU) moves to the clean data LRU linked list (clean LRU for short). The metadata of the two linked lists, aiming to enhance the performance, are maintained by means of a hash table-based cache index layer. We configure a threshold for the size of clean LRU according to the amount of data accessed by burst I/O requests. This approach is capable of slashing the excessive written data caused by active write back while improving endurance in flash memory.

▷ We design Co-Active in a way that I/Os access patterns and the status in flash chips are incorporated. More specifically, we devise a module to distinguish cold and hot data in workloads. This module verifies a given logical page address (LPA) to ensure the LPA contains frequently accessed data referred to as hot data. We employ a bloom filter to keep track of relationships between hot and cold data. The bloom filter improves retrieval performance of hot and cold data while conserving cache space. We construct a chip detection unit to detect the status (idle or busy) of a flash chip by gauging operations being executed in the flash chip and pending operations to be executed in a chip queue.

▷ We implement Co-Active in an open-source SSD simulator - MQsim. The extensive experiments are conducted to validate the advantages of Co-Active under real-world traces. Compared with the latest cache management schemes, Co-Active judiciously shortens the average response time by up to 83.9 percent with an average of 32.7 percent. And the performance cliff is reduced by up to 82.1 percent accompanied by an average of 42.3 percent. Co-Active also decreases write amplification by up to 60.5 percent with average of 5.4 percent.

2 Background

2.1 NVMe SSD Architecture

With the emerging development of SSDs, the internal bandwidth in SSDs exceeds the bandwidth of external interface connecting to a host. SSD performance is limited by the throughput of an interface such as *SATA* and *SAS* [16]. Accordingly, the PCIe-based NVMe protocol [17] becomes the standard protocol for high-performance SSDs. The NVMe protocol promotes the performance of an SSD in terms of hardware interface, data path, and software stacks [18]. Fig. 1 depicts a new modern NVMe SSD [19].

In an NVMe SSD, the host-side memory consists of I/O queues, including a submission queue (SQ) and a completion queue (CQ). The device-side system is a mixed bag of three components, namely, a host interface logic (HIL), a flash translation layer (FTL), and NAND flashes. FTL orchestrates a resource management module, I/O requests processing, garbage collection (GC) [20], [21], and wear leaving (WL) [22]. Flash transactions pass the cache management module; thereby enhancing the SSD performance. The address translation unit maintains a mapping table between logical page address (LPA) and physical page address (PPA). An out-place update is employed in page updates. Amid an excessive number of invalid pages in a block, a GC operation is prompted to erase the block. The SSD that is fueled by NAND flash [23], [24] as a storage medium encompasses multiple channels transmitting flash transactions in parallel [25]. Each channel embraces one or more chips, one of which has an array of dies containing planes. A plane in turn is composed of multiple blocks, an erase operation of which performs on a fixed number of pages. A page is the unit of a program/read operation. Four-level parallelism featured in SSDs [26] includes channel-, chip-, die-, and plane-level parallelisms, among which die- and plane-level parallelisms are reliant on advanced commands.

2.2 Cache Management Scheme

The LRU policy is a classical cache management scheme adopted in SSDs. Data managed by LRU is organized in a linked list (see Fig. 2). When a new flash transaction moves

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

to the cache manager unit, a request is determined whether it is in the cache according to the LPA. If there is a cache hit, the I/O request will be serviced by the cache in terms of updating written data or extracting read data. During the procedure of handling a cache miss, a write I/O request is inserted into the head of the LRU list (MRU); a read request ought to access required data from NAND flash. When the cache is full, data at the end of the LRU list (LRU) must be evicted to make free space that can be written. It is noteworthy that the amount of data written from the cache to NAND flash in parallel is limited by the back pressure buffer space [15], and its size is related with the physical structure in NAND flash.

From the perspective of cached-data sources, cache management schemes are categorized into three types, namely, the read cache mode, the write cache mode, and the read-write cache mode. In what follows, we elaborate these cache modes determining which type of data should be written to a cache. (1) The read cache mode writes data that are read from flash memory into the cache as the read-I/O requests miss in the cache. (2) The write cache mode writes write-I/O requests that are issued to an SSD from user-level workload into the flash memory. (3) The read-write cache mode both writes the data that are read from the flash memory and the write-I/O requests from the user-level workload into the flash memory.

In terms of updated methods of cached data, caching schemes fall into two types - write-back and write-through schemes [27], [28]. A vast majority of cache management mechanisms favor write-back schemes. When cache replacement occurs, data are written into flash memory. LRU and the latest cache management (e.g., GCaR, LCR) adopt this scheme to boost performance by curbing the number of write I/Os. A write-through scheme simultaneously writes data to cache and flash memory. This scheme maintains a remarkable data consistency between the cache and the underlying flash memory. However, frequently writing data into the flash memory inevitably slows down I/O performance due to aggressive accesses to the flash memory. Regardless of NAND flash categories like *SLC* or *QLC*, there are a limited number of program/erase cycles in flash memory medium. Compared to the write-back policy, the write-through counterpart is more likely to shorten SSD lifetime by impairing NAND flash endurance.

Wu [9] proposed an average-memory-access-time (i.e., AMAT) model to gauge the efficiency of a caching strategy. The efficiency of a caching scheme largely depends on three parameters - Hit_Time, Miss_Rate, and Miss_Penalty [9]. The access time of DRAM is constant; thus, Hit_Time remains unchanged in SSDs. Prior research is focused on improving the efficiency of caching schemes by reducing Miss_Rate and Miss_Penalty. For example, LRU [29], EELRU [30], 2Q [31], LRFU [32], ARC [5] were devised to strengthen hit ratios. Very recently, a handful of caching schemes were designed to suppress Miss_Penalty. For instance, CFLRU [7] minimizes cache-replacement cost by evicting clean data pages in cache and flash memory. GCaR [25] defers data evictions during the GC process to avert high replacement costs. Liu *et al.* discovered that the replacement cost in LCR [10] is dependent of unbalanced load in flash chips.

To quantitatively evaluate the caching performance benefits of Co-Active, we select LRU, CFLRU, GCaR-CFLRU, LCR, and MQsim as the comparison caching schemes. Please refer to Section 5.2 for the details on the qualitative comparisons among these caching schemes, cache mode (the read, write, and mixed read-write), and update methods (write through and write back).

3 Motivation

A conventional wisdom in cache management is to advocate for passive write-back update strategies. When a cache is full upon arrivals of new I/O requests, cached data must be written back into flash memory as cache replacement kicks in. The frequency of the cache replacement occurrence largely depends on I/O access patterns in workloads. Light I/O load leads to very few cache-replacement operations, thereby keeping the bandwidth utilization of flash chips at a fairly low level. Consequently, the internal parallelism of the flash memory cannot be fully exploited. On the flip side, I/O-intensive workload conditions trigger frequent cache replacement, which impairs the response time as well as SSD performance. The deteriorating performance is attributed by the fact that the bandwidth of flash chips fails in handling burst I/O accesses in a swift manner.

To combat the aforementioned issue, we employ realistic enterprise-scale traces to explore I/O access patterns.[2] The details of traces are listed in Table 1.

The first four columns represent the basic features of traces. The *Avg. accessed sectors* signifies the average number of write accessed sectors per second. The *Max. accessed sectors* means the maximum value of write accessed sectors per second in the process of running. For example, the *Avg. accessed sectors* and *Max. accessed sectors* are 1663.35 and 64126 in hm_0, respectively. The maximum value is nearly $4 * 10^4$ times larger than that of the average value, implying that the I/O access pattern exhibits imbalance to a certain extent in this trace. We employ the standard of deviation of write accessed data per second to quantify the access-imbalance degree (*Standard Deviation*). In some cases, the amount of write accessed data is orders of magnitude higher than that of the normal stage in each trace (i.e., burst access stage). When the number of write accessed sectors per second exceeds 10^3, this stage accounts for about 1 percent of the total running time; meanwhile, the amount of write accessed data is about ten times larger than that of the normal stage. In this work, we define this period as the *burst access stage*. Afterward, We count the percentage of the burst access stage and the request data ratio in the burst access stage (see the eighth and ninth columns in Table 1). For instance, 37.4 percent of accessed data is issued during a very short period (0.98 percent) under hm_0. The metric - request data ratio in burst access being divided by burst stage ratio - reveals the write access imbalance in the traces. It must be noted that the larger the ratio, the more severe the imbalance problem becomes; the ratio being close to one indicates a well-balanced workload. Importantly,

2. Because Co-Active is customized for write-intensive application scenarios, we select nine block-level traces with a high write ratio and two ones with a read-write ratio. These traces were collected from the Umass Trace Repository [33] and Microsoft Research Cambridge Repository [34], [35].

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Details of Realistic Traces

Trace	Total I/Os	Write Ratio	Avg.Size (KB)	Avg. accessed sectors[1]	Max. accessed sectors[2]	Standard Deviation[3]	Request Data Ratio in Burst Stage[4]	Burst Stage Ratio[5]	Request Data Ratio in Peak Burst Stage[6]	Peak Burst Stage Ratio[7]
hm_0	3993316	0.644	7.99	166.35	64126	1222.8	37.4%	0.98%	20.5%	0.09%
proj_0	4224524	0.87	38.03	1300.62	64480	4943.7	94%	11.2%	43%	1.7%
prxy_0	12518968	0.96	4.761	244.15	60766	1680.5	53.6%	1.2%	29.1%	0.25%
stg_0	2030915	0.84	11.57	143.35	52712	409.6	7.2%	0.24%	2.7%	0.015%
ts_0	1801734	0.82	9.003	114.49	64480	384.8	11.1%	0.37%	2.8%	0.012%
usr_0	2237889	0.595	22.66	119.98	33256	470.7	19.6%	0.67%	2.67%	0.013%
web_0	2029945	0.701	14.992	138.78	39162	679.7	22.2%	0.56%	8.4%	0.043%
Fin1	5334987	0.768	3.384	728.42	34058	517.3	40.9%	20.7%	0.11%	0.002%
src2_0	1557814	0.886	7.206	106.22	51160	515.7	15.6%	0.35%	5.6%	0.021%
hm_1	609311	0.046	15.16	105.23	36992	975.67	73.01%	1.25%	13.29%	0.0005%
WebSearch1	1055448	0.0002	15.145	1.162	320	8.106	0%	0%	0%	0%

[1]*Column 5 is the average number of write accessed sectors per second in a trace.*
[2]*Col. 6 is the max value of write accessed sectors per second in a trace.*
[3]*Col. 7 is The standard of deviation of write accessed data per second, representing the the degree of the access imbalance.*
[4]*Col. 8 means that the proportion of amount of write data accessed in burst access stage to the total amount of accessed data during a trace.*
[5]*Col. 9 is the proportion of the burst write access stage in the whole period of a trace.*
[6]*The amount of write data accessed in the peak burst access stage accounts for the proportion of the total amount of write data accessed during a trace.*
[7]*The proportion of the peak-burst access stage in the whole period of a trace.*

the SSD performance is mainly affected by the peak amount of write accessed data per unit time. Thus, we stipulate the peak burst stage as the number of write accessed sectors per second that is more than $2 * 10^4$ (last two columns in Table 1). The peak-burst-stage degree in hm_0, proj_0 and prxy_0 is stronger than that of the other traces and; therefore, Co-Active has the competitive edge over the other techniques under these traces.

We carry out experiments to investigate parallel situations of an SSD (please refer to Section 5 for the detailed experiment settings). When an I/O request arrives at the SSD, we count the number of chips that are working in a total of 24 chips in the SSD. If no chips are running, the counter is recorded as 0. If all the chips are busy, then the counter equals to 24. To avoid the contingency of a single load result, we select multiple real-world traces to separately perform statistics. Next, we plot the statistical results by means of a cumulative distribution function (CDF) in Fig. 3. We discover that the number of chips running in parallel is fewer than 5 (i.e., 80 percent chips are sitting idle) in more than 75 percent cases under the tested workloads. The time duration in which more than 20 chips work in parallel is less than 5 percent. The parallelism ratio and chip utilization are very low across all the workloads.

Most existing caching schemes, which favor an approach to preferentially evicting clean data rather than dirty one, can curb the cache-replacement cost.[3] The clean data that are evicted by cache replacement are merely dropped from the cache [7]. Thus, evicting clean data is the most cost-effective way for cache replacement. In addition, when physical page address cannot be located in the mapping table, the global translation table is accessed in the flash memory. Then, internal I/O accesses enlarge the cache-replacement cost in the case of dirty data [36]. Clean data are only passively generated by read I/Os in existing caching schemes. Under the write-intensive workloads, the cost of cache replacement is quite high due to low chance of replacing clean data.

To optimize SSD performance, we propose a novel cache management scheme to actively update cached data into flash memory by the virtue of utilizing the flash-chip bandwidth amid light I/O accesses. The overarching goal of our design is to push up the volume of clean data in cache; thereby, offering ample opportunities to evict clean data during the cache replacement. An advantage of our scheme lies in boosting SSD performance by cutting back cache-replacement cost during burst I/O accesses.

4 Co-Active Cache Management Scheme

We start this section by presenting an overview of the Co-Active cache management mechanism, followed by the implementation details of Co-Active that encompasses a manager and a write-back module powered by a collaborative-aware approach.

4.1 The Overview System of Co-Active

Fig. 4 depicts an overall system design of our proposed *Co-Active* - an active cache management scheme fueled by the collaborative write-back approach for SSDs. Co-Active is deployed in the flash transaction layer (i.e., FTL) of SSDs functioning as follows. First, an SSD receives read/write I/O requests from a host governed by the NVMe protocol.

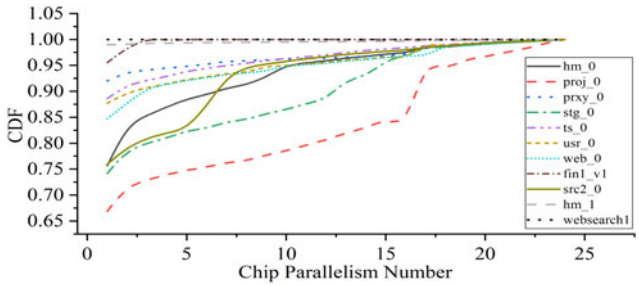


Fig. 3. The parallelism of an SSD in the term of a cumulative distribution function.

3. Clean data are referred to as cached data that are duplicated copies of those in flash memory; dirty data are newly updated data of those in flash memory.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

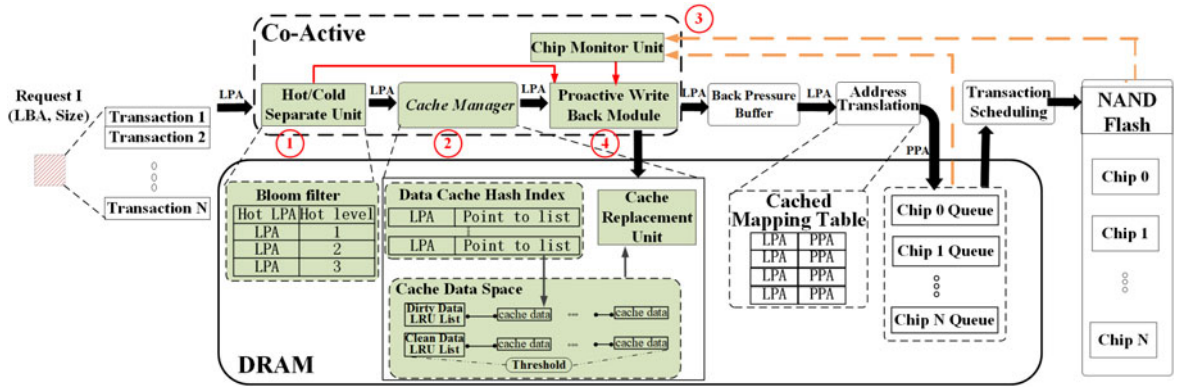


Fig. 4. The overall system design of Co-Active for NAND flash-based SSDs.

Then, an I/O request is split into flash transactions by the size of a page. Each flash transaction has a LPA and a bitmap. When flash transactions reaching the FTL are processed by Co-Active, the Co-Active system is comprised of four components. The cache management module (② in Fig. 4) achieves a two-fold goal, namely, (1) handling cache space storing data in DRAM; (2) controlling data transfers of I/O requests inside SSDs. The proactive write-back module (④ in Fig. 4) determines whether to trigger the active write-back function according to collaborative messages. The active write-back operation flushes cold and dirty data residing in the cache into the flash memory, thereby making dirty data clean. The collaborative messages mainly includes two parts, which signify cold data and idle chips. Such information is gauged by the hot/cold separation unit (① in Fig. 4) and the chip monitor unit (③ in Fig. 4). The hot-and-cold separation unit distinguishes hot from cold data; the chip monitor unit detects the load status of each chip.

The cache management module eventually writes flash transactions into the flash memory through the cache replacement or the active write-back function. Co-Active generates clean data by the virtue of the active write-back function. The proactive write-back function may push up the amount of data written into flash memory. The volume of data written into the flash memory by active write-back function is set to be a threshold, which curbs the amount of invalid data written into flash memory.

Amid the process of writing data into the flash memory, the LPA is translated into the PPA through the address translation module determining destination addresses in flash memory. FTL funnels flash transactions into the chip-level queue in DRAM according to the corresponding chip of its PPA. Flash transactions in the queue are scheduled by the transaction scheduling unit (TSU) governed by the scheduling policy. The flash transactions are delivered to the flash memory by the flash channel controllers (FCCs).

We introduce the details of the cache manager, the proactive write-back module, the hot-cold separate unit, and the chip monitor unit in the subsequent subsections.

4.2 Cache Manager

The cache manager in the Co-Active is in charge of data space and I/O-requests flow inside an SSD.

To store data in the cache, Co-Active makes use of two LRU linked lists. One list is dedicated for clean LRU and another list is reserved for dirty data. Two lists are orchestrated, because Co-Active preferentially evicts clean data when the replacement is kicked in. The two linked lists make a conspicuous difference between clean and dirty data. Compared to a single linked list design, employing two lists can effectively boost the retrieval performance of clean data. In addition, we advocate for a hash structure-based data-cache index layer on the LRU linked list. Data-cache index registers metadata related to the LRU linked list; sample metadata includes LPA and its corresponding position pointer in the LRU linked list. The data cache index is organized in the format of a hash table. Co-Active judiciously determines whether data exists in the cache; thereby updating the cache in a swift manner.

After an I/O request is split into flash transactions, the hot-and-cold-separation unit records hot and cold data (see Section 4.4.1 for the details). Then, these transactions move to the cache management module, in which the process is reliant on whether a flash transaction is read or write. In what follows, we articulate the procedures of the two flash-transaction cases.

Case 1: Flash transactions are read-I/O requests. When there is a hit in the cache, data is moved into the head of the corresponding LRU-linked list. The position pointer of the data is updated in the cache index. In case of clean data, the linked list will be clean LRU.

If there is a data miss, the flash transaction is placed into the read queue of the corresponding chip and await TSU scheduling. Next, the required data reads from the NAND flash. Please refer to Algorithm 1 for the details of the read process in Co-Active.

Algorithm 1. Read Flow for Co-Active

Require: $ReadFlashTransaction$ $RT_1, RT_2, RT_3, \ldots, RT_i, \ldots$
1: **procedure** Co-Active Cache Manager(RT_i)
2: **if** RT_i hit Data_Cache_index **then**
3: Get_slot(RTi, corresponding_LRU_list)
4: Move to corresponding_LRU_list head(RT_i)
5: Return from Cache(RT_i)
6: **else**
7: /*Hit miss*/
8: Push RT_i to Chip_Queue for UserRead
9: Return from Flash(RT_i)
10: **end if**
11: **end procedure**

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

Case 2: Flash transactions are write-I/O requests. Amid a data hit, the data is taken out from the linked list; afterward, the data is updated and inserted at the head of the corresponding LRU linked list. The position pointer of data is updated in the cache index.

Let us consider the data-miss scenario. When the free space in the cache is empty upon the arrival of a new write request, the cache replacement is provoked. Data at the end of the clean LRU is first evicted. Only when all the data in the clean LRU has been evicted, data at the end of the dirty LRU is evicted. When the cache replacement is completed, free buffer space is utilized by the transaction write. If the cache has free space, a new slot will be allocated for the newly written data; meanwhile, the data is inserted into the dirty LRU. The details of the write processes in Co-Active can be found in Algorithm 2.

Algorithm 2. Write Flow for Co-Active

Require: $WriteFlashTransaction \;\; WT_1, WT_2, \ldots, WT_i, \ldots$
procedure CO-ACTIVE CACHE MANAGER(Ti)
  if WT_i hit Data_Cache_index **then**
    Get slot from cache
    Write to Dirty_LRU_list head(WT_i)
  else
    /*Hit miss*/
    if Cache is full **then**
      /*Invoke Cache Replacement*/
      if Clean_LRU_list > 0 **then**
        Remove bottom item in Clean_LRU_list: L_x
        Write to Dirty_LRU_list head(WT_i)
      else
        /*Invoke Cache Replacement in Dirty_LRU */
        Get bottom item in Dirty_LRU_list: L_x
        Push L_x to Chip_Queue for UserWrite
        Write to Dirty_LRU_list head(WT_i)
      end if
    else
      /*Cache is not full*/
      Write to Dirty_LRU_list head(WT_i)
    end if
  end if
end procedure

4.3 The Proactive Write-Back Module

Recall that the cache manager module handles two LRU-linked lists for dirty data and clean data; thereby managing a huge amount of clean data generated by the active write-back function. The write-back module actively writes cold data in the dirty LRU back into flash memory when the flash is idle; meanwhile, dirty data are converted to clean ones.

The process of actively writing cache data into the flash memory inevitably occupies the bandwidth of flash chips, which increase the amount of write data to the flash memory. When the active write-back I/Os, user I/Os, mapping data I/Os, and I/Os in GC conflict, the SSD performance can be impaired. We devise a chip monitoring unit to determine when to invoke active write-back operations. The chip monitoring unit gauges the load of each chip. If the load reading of a chip equals to zero, the chip is in the idle mode. When an active write-back flash transaction is issued amid the idle status, this transaction can be immediately handled. Therefore, when the target chip of cached data is sitting idle, the active write-back function will be triggered. Another issue is how to obtain a chip address that is corresponding to the LPA. Co-Active leverages the default address allocation method in MQsim (CWDP [26], [37]). The active write-back module calculates the number of the channel, chip, die, and plane to be written by the LPA using (1)

$$\begin{aligned} ChannelID &= lpa\%Channel_N \\ ChipID &= (lpa/Channel_N)\%Chip_N \\ DieID &= (lpa/(Chip_N * Channel_N))\%Die_N \\ PlaneID &= (lpa/(Die_N * Chip_N * Channel_N))\%Plane_N, \end{aligned} \tag{1}$$

where $Channel_N$, $Chip_N$, Die_N, and $Plane_N$ represent the number of flash memory channels, chips per channel, dies in each chip, and the number of planes in a die. Co-Active merely keeps track of $ChannelID$ and $ChipID$ to locate the chip queue.

In the FTL module, a data allocation method (e.g., CWDP) translates LPA to a chip address. Because the proposed Co-Active is adroit at modifying the load formulas in the chip monitor to adapt to each data allocation, Co-Active is a high-performance caching strategy being compatible with data allocation in any address translation.

Now we elaborate the procedure of writing back cold dirty data. When all dirty data are actively written back, an enormous number of writes and erase operations are issued to flash memory. As a result, the flash memory's endurance deteriorates. Dirty data should be selectively written back into the flash memory. When the same LPA is written multiple times in a period of time, the last data is valid. The PPA in the mapped table records the page address that was last written. All the pages that have been written are marked as invalid; meanwhile, these invalid pages ought to be erased by the GC. Frequently written data are referred to as hot data, which should be prevented from being written into flash memory. Co-Active employs the hot-and-cold separation unit to oversee data of LPAs. When active write back occurs, the cold dirty data are written into flash memory to avert unnecessary write operations.

We introduce the concept of threshold for write-back clean data; the rational is manifested as follows. The active write-back function may result in an excessive number of writes, meaning that a tremendous amount of data in dirty LRU are written into the flash memory. In the clean LRU, there should be an amount of clean data that can meet the data volume of burst I/Os requests in workload. Thus, we set a term '*write-back threshold*' for the clean-LRU. The *write-back threshold* represents the ratio of clean-LRU size to the total cache size. A large threshold can easily spur active write back. When the amount of data in the clean LRU reaches this threshold, dirty data are no longer actively written. The threshold depends on I/O access patterns in workload. We shed more light on the threshold in the scalability experiment.

4.4 A Collaborative-Aware Approach

The collaborative-aware approach provides dependent messages for making decisions in the process of the collaborative

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
An Exampled Table of Hot-Cold Levels for Accessed Data

LPA	Hot Level
100	1
101	2

active writing-back. This approach is comprised of two parts, namely, a *hot-cold separation* unit and a *chip-monitoring* unit.

4.4.1 Hot-Cold Separate Unit

We incorporate the hot-cold separation unit in Co-Active. A table (see Table 2) is maintained to keep track of the hot-and-cold status of data.

The table is organized in the form of key-value pairs, where a key stores the LPA and a value represents the number of accesses the corresponding LPA during a period of time. All flash transactions of a workload in an SSD pass through the cold-hot-separation unit, which determines whether the LPA of the flash transaction exists in the hot-cold table. After the LPA is placed in the table, the value is set to 1 when the LPA does not exist in the table. If the LPA already resides in the table, the value increases by one. For example, suppose LPA equals to 100 and the corresponding hot level is 1. This key-value pair signifies that the logical address numbered 100 is accessed once during the current cycle.

Table 2 illustrates that the table of hot-cold levels for accessed data usually stored by means of a red-black tree or a hash table as a data structure. Hot and cold cached data are stored in cache; therefore, the low latency and space utilization are mandatory in cache management. The time complexity of the red-black tree is O(logn), so it fails in meeting the low-latency requirement. The time complexity of hash tables can reach O(1), but a hash table stores a large amount of data in the case of hash conflicts, causing the high occurrence ratio for hash conflict to consume a large amount of memory space. Thus, we advocate for the bloom filter data structure in Co-Active. The traditional bloom filter only determines whether an LPA may exist or not rather than obtaining the hash-value size of the LPA, because each element in the vector has only one bit. Thus, the counting bloom filter [38], [39] is deployed to distinguish hot and cold data as well as their access frequency, thereby conserving memory space and optimizing access performance.

In view of hot and cold distribution changes over time, the hot-cold-separation table performs a reset at the beginning of each period. Then, cold-hot status can be gauged from ground up. This iterative process staves off an infinite increment in the table. Fig. 5 illustrates the hot-cold statistics in the workload.

We classify sectors in a workload according to its access times. Then, we account the proportion of each access sector range in the entire sector range, which is named as hot-cold levels in this work. The data that is accessed merely once is cold data. In case that the number of access exceeds one, the counter is treated as thermal data referred to as heat levels. We observe that the ratio of absolute cold data is less than 50 percent, and all real-world workloads show a conspicuous mixture of hot and cold data. Thus, the hot-cold-separation unit is intended to boot overall performance of SSDs.

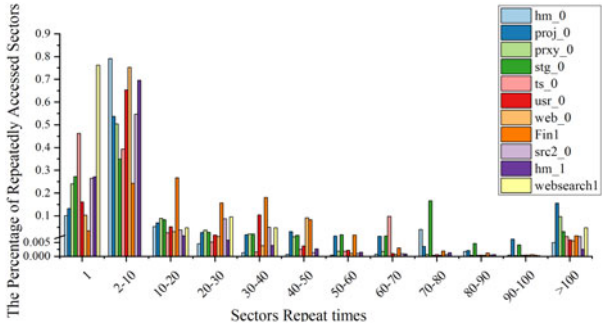


Fig. 5. Hot- and cold-data statistics in real-world traces.

When the Bloom Filter discovers that the LPA of data exists, the data is treated as hot data regardless of the data's hot level. This method is a widely adopted technique to pinpoint hot and cold data [5], [40], [41].

4.4.2 Chip Monitor Unit

The chip-monitoring unit measures load of flash chips. Chip load readings indicate how long flash transactions will be serviced by the chip upon the arrival of the transactions; load values imply a chip's busy level within a period. The chip load is calculated as follows:

$$L_{chip} = L_{extern} + L_{execute}, \tag{2}$$

where the symbol L_{chip} represents the chip load. L_{chip} is the summation of the current chip execution load $L_{execute}$ and the chip external load L_{extern}. The value of L_{chip} being zero signifies that a chip that has zero load is idle. Current load $L_{execute}$, which indicates how long the command of this chip currently being finished, is expressed as

$$L_{execute} = L_r * B_r + L_w * B_w + L_e * B_e - T_{consumed}, \tag{3}$$

where terms B_r, B_w, and B_e are the marked flag bits for the read, write, and erase operations in the chip, zero is no-operation is performed while 1 is being performed this operation. Only one flag bit can be true at the same time; meanwhile, other flag bits are set to be zero. L_r, L_w, and L_e indicate the delay of read, write, and erase operations, respectively. $T_{consumed}$ indicates how long the current chip has been executed.

The external load L_{extern} in Eq. (2) measures the estimated rest completion time for the execution of the command in the chip queue. L_{extern} can be written as as

$$L_{extern} = N_r * L_r + N_w * L_w + N_e * L_e, \tag{4}$$

where notations N_r, N_w, and N_e represent the rest numbers of read, write, and erase operations that are pending in the chip queue, respectively. It is worth noting that reads and writes are categorized into three types, namely, user reads/writes, mapped data reads/writes, and GC reads/writes; moreover, erase operations are only spawned by the GC module.

The values of N_r, N_w, and N_e can be computed as follows:

$$N_r = N_{uers_{read}} + N_{Mapping_{read}} + N_{GC_{read}}. \tag{5}$$

$$N_w = N_{user_{write}} + N_{Mapping_{write}} + N_{GC_{write}}. \tag{6}$$

$$N_e = N_{GC_{erase}}. \tag{7}$$

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3. Flash Transaction Flow for Co-Active

```
Require: FlashTransactionT1, T2, T3, ..., Ti, ...
procedure Co-Active Cache Procedure(Ti)
  function Hot/Cold SeparateTi
    if LPA of Ti is in the Hot/Cold Table then
      Hot level(Ti)++
    else
      insert Hot/Cold Table(Ti)
    end if
    if Time > bloom_filter_reset_milestone then
      Reset Hot/Cold Table
    end if
  end function

  function Co-Active Cache Manager(Ti)
    /*Detail is listed in the Read/Write flow in Co-Active */
  end function

  function Proactive Write back
    if Clean_LRU_list.size < threshold and Dirty_LRU_list.size > 0 then
      repeat Get bottom item in Dirty_LRU_list: L_x
        if L_x is Cold_Data then
          if L_x belongs to the chip with L_chip = 0 then
            Active_Write_back(L_x)
            Move_to_Clean_LRU_list(L_x)
          end if
        end if
      until x==0
    end if
  end function
end procedure
```

Being scheduled by TSU, the three types of read and write operations are featured with different priorities. These mapping- and GC-type reads and writes - assigned the highest priority - are scheduled in the first place.

The chip load calculated by the chip-monitoring unit is not the chip execution time in a sense. The time consumed by a flash memory basic operation is written as

$$T_{operation} = T_{send_command} + T_{execute} + T_{transfer}, \quad (8)$$

where time $T_{operation}$ largely depends on time $T_{execute}$. $T_{send_commend}$ represents the transmission time of the command from the flash channel controller to flash memory. $T_{execute}$ denotes the delay in performing read, write, and erase operations. $T_{Transfer}$ indicates the time spent in transferring read data from the flash memory to the flash channel controller (or to write data to the flash chip from the flash channel controller). $T_{send_commend}$ and $T_{transfer}$ vary with the amount of data transferred and different commands. In a real-world scenario, the value of $T_{execute}$ can be fickle. Moreover, when the TSU schedules commands in the chip queues, advanced commands (e.g., interleave, multi-plane, and copy-back) are executed in parallel, which impairs the accuracy of the collection and calculation for the chip load unit.

4.5 Data Flow in Co-Active

Now we articulate the data flow of Co-Active in *Algorithm 3*, which encompasses three key functions - Hot-Cold separation, Co-Active cache manager, and proactive write-back. The hot-cold-separation function registers hot-cold messages of I/O requests in the hot-cold table. The cache manager in Co-Active handles the read/write flow, the details of which are elaborated in *Algorithms 1* and *2*. The proactive write-back function adjudicates whether it is appropriate to perform the active write-back function.

TABLE 3
Simulation Parameters

SSD Organization	Host interface: PCIe 3.0 (NVMe 1.2) Total Capacity: 24 GB 6 channels 4 chips-per-channel
Flash Communication Interface	ONFI 3.1 (NV-DDR2)
Flash Microarchitecture	4 KB/page, 448 B metadata/page, 64 pages/block, 1024 blocks/plane, 2 planes/die, 2 dies/chip
Flash Latencies [42]	Read latency: 75 μs, Program latency: 750 μs, Erase latency: 3.8 ms
Flash translation layer (FTL)	GC Policy: Greedy GC Threshold: 0.05 Address Mapping: DFTL TSU Policy: Sprinkler [43] Over provisioning Ratio: 0.07

5 Experimental Setup

After describing the experimental platform in Section 5.1, we briefly elaborate five alternative cache management schemes in Section 5.2. The performance metrics are introduced in Section 5.3.

5.1 Platform

We utilize an open-source MQ-SSD simulator, MQsim [19], to validate the efficacy of Co-Active. MQsim released by the CMU-SAFARI Research Group accurately simulates front-end and back-end components in modern MQ-SSDs. In this work, we configure an SSD with 24 chips. The default cache size is 32 MB. The details of the simulation parameters are summarized in Table 3.

We apply the realistic enterprise-scale workloads (see Table 1) to study the performance of various caching schemes. Co-Active aims improving performance of applications with unbalanced write-I/O requests and the burst-access mode write-I/O requests. Co-Active increases the number of write-I/O requests triggering active write-back operations, thereby shortening the response time of burst-access requests and improving the system performance. A workload with a high write ratio is readily to evaluate the active write-back mode in Co-Active, which has an edge in the caching management. In addition, these workloads are popularly used in the prior studies focusing on caching strategies in NAND flash-based storage systems. Applying these workload conditions, we can gauge the benefits of Co-Active compared with the alternative caching strategies reported in the literature. The volume of write-I/O requests in a portion of traces is mostly small, GC procedure cannot be sparked after running the trace once. We repeatedly run these traces multiple times to simulate realistic use cases,

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

thereby gauging the impacts of different caching schemes on performance and lifetime of SSDs.

5.2 Alternative Cache Management Schemes

We compare Co-Active with the following five alternative solutions, including the LRU, CFLRU [7], GCaR-CFLRU [9], LCR [10], and MQsim [15] schemes. There are two types of cache mode, namely, the write cache mode and read-write cache mode. Besides the write-through update method, the write back is a popular update method managing cached data.

LRU. LRU_W_Mode denotes a LRU policy with the write cache mode, whereas *LRU_WR_Mode* represents LRU powered by the read-write cache mode. LRU utilizes the write-back methods.

CFLRU. CFLRU, designed for flash memory, preferentially replaces clean data to minimize replacement cost. Clean data in CFLRU are generated by writing data from the flash memory into the cache. The performance of CFLRU is reliant on the size of a write-back window. In the experiments, we set the write-back window to $C/1.5$, where C is the cache capacity. CFLRU leverages the read-write cache mode and write-back methods.

GCaR-CFLRU. The garbage-collection-aware replacement scheme aims to avert the access conflict between user I/Os and GC I/Os. *GCaR-CFLRU* pays close heed to the impact of chip GC on cache replacement, ensuring that data in a no-GC chip are preferentially evicted. GCaR-CFLRU employs the read-write cache mode and write-back methods.

LCR. LCR takes the flash chips load into account in cache replacement. When a cache replacement occurs, LCR attempts to evict clean data. In case of no clean data, LCR preferentially evicts the data corresponding to the low load of each flash chip. LCR uses read-write cache mode and write-back methods.

MQSim. MQSim represents a default caching scheme implemented in the MQSim simulator. This policy synchronously flushes cold data written to the cache into the back-end flash memory. Meanwhile, MQSim removes the cache data that have been written back into the flash memory. MQSim applies the write cache mode and write-through methods.

5.3 Tested Metrics

Average response time indicates the performance of an SSD from the perspective of individual I/O requests. This metric measures the time interval between an I/O request entering the SQ queue and the host receiving feedback from the SSD.

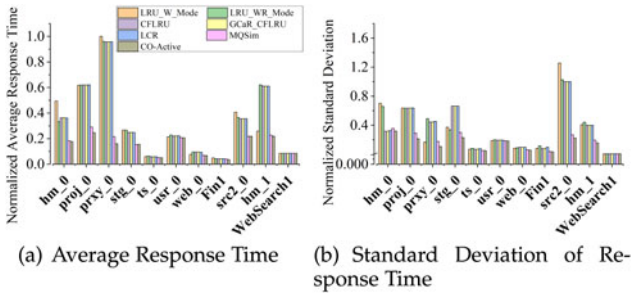
The standard deviation of response time quantifies the degree of an SSD performance cliff.

Write amplification is a ratio between the amount of data written to the flash memory and the amount of data issued by a host. A low value indicates that the amount of data written to the flash memory is small. Write amplification reflects impacts of the tested cache management schemes on SSD lifetime.

Erase count in flash memory is the most conspicuous manifestation of SSD lifetimes. Each block in flash memory has a maximum number of erase operations.

Write (Read) hit ratio is the probability of a I/O request hitting the cache.

Clean ratio suggests a proportion of clean data that are evicted when cache replacements occur.



(a) Average Response Time (b) Standard Deviation of Response Time

Fig. 6. Average Response Time (6a) and Standard Deviation of Response Time (6b) of Co-Active and the alternative caching schemes under 11 traces. The low response time signifies good performance.

6 Performance Evaluation

In this section, we shed bright light on the experimental results of Co-Active. After delving into the evaluation of the response-time performance in Section 6.1, we gauge the impacts of Co-Active on SSD endurance in terms of write amplification (Section 6.2) and erase count (Section 6.3). Sections 6.4 and 6.5 offer analysis of hit ratio and clean ratio, respectively. We summarize the empirical findings in Section 6.6.

6.1 Response Time

Fig. 6a unveils the average response time of the caching schemes in the simulated SDD driven by the 11 traces (i.e., hm_0, proj_0, prxy_0, stg_0, ts_0, usr_0, web_0, Fin1, and src2_0, hm_1, and Websearch1).

Under hm_0, proj_0, and prxy_0, Co-Active has an edge over the other caching schemes. Compared with the *LRU_W_Mode*, Co-Active slashes the response time by 64, 62, and 84 percent in the cases of hm_0, proj_0, and prxy_0, respectively. A head-to-head comparison shows that Co-Active reduces the average response time of MQSim by 7, 16, and 25 percent, respectively. The advantages of Co-Active are expected, because the three traces embrace a larger scale of burst peak I/Os than other traces (see Table 1). Co-Active writes back an excessive amount of cold and dirty data in advance during an idle period, thereby maintaining clean data to cope with burst I/O accesses. Thus, the Co-Active method avoids dirty data cache frequently being evicted as burst I/O access occurs. Under the ts_0, usr_0, and Fin1 traces, Co-Active rarely presents a significant performance improvement over the other caching schemes. Compared to the *LRU_W_Mode* policy, Co-Active cuts back the response time by 15, 3.7, and 27 percent, respectively. Co-Active reduces the average response time of MQsim by up to 0, 1.2, and 13 percent under the ts_0, usr_0, and Fin1 workload conditions. CFLRU, GCaR_CFLRU, and LCR customized for SSDs hold no performance edge over LRU. In MQSim, there is a back pressure buffer between cache and flash memory. This layer limits the amount of data transferred from the cache to flash memory. When a burst peak I/O access occurs, these write-back mode caching schemes (i.e., LRU, CFLRU, GCaR_CFLRU) force a large amount of dirty data to be evicted, thereby suppressing the performance of cache replacement.

MQsim incorporates the cold-and-hot separation to update data to flash memory in a real-time manner. MQsim improves the SSD performance by deleting data written back in the cache to speed up burst peak I/O accesses. The disadvantage

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

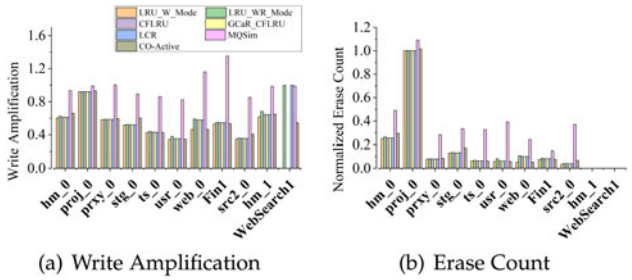


Fig. 7. The impact of co-active and the alternative caching schemes on endurance of flash memory under the 11 traces. Low write amplification and the erase count values signify good performance.

of MQsim, however, lies in extra write operations issued in flash memory. Such a setback enlarges write amplification and erase operations. The details are elaborated in Section 6.2.

Under the hm_1 and Websearch1 traces, Co-Active presents the different improvement with the other caching schemes, although these are high read I/O ratio traces (read I/O ratio exceeds than 99 percent, details see Table 1). Under the hm_1 workload, Co-Active reduces the average response time by 15.7, 64.86, 64.29, 64.29, 64.29, 3.98 percent compared with *LRU_W_Mode*, *LRU_WR_Mode*, CFLRU, GCaR-CFLRU, LCR, and MQSim. The response time of Co-Active is on par with those of the other cache schemes under Websearch1. The performance improvements achieved by Co-Active become more pronounced under a high percentage of burst write accesses. The request data ratio in the peak-burst stage of hm_1 is the highest (13.29 percent) among all the workloads; the request data ratio in the peak-burst stage of Websearch1 is the lowest (0 percent) among its peer. Therefore, Co-Active holds a notable edge over the other cache schemes under hm_1. Co-Active maintains similar performance as those of the others under Websearch1, because the read I/O ratio of Websearch1 is more than 99.9998 percent with a read hit ratio close to 0 percent (see Fig. 8b). Consequently, the cache performance is hardly optimized by anyt caching scheme in this case.

The standard deviation of response time can be gauged to assess the optimization of SSD performance cliff with the various caching schemes. Fig. 6b reveals that Co-Active is adroit at lowering the response-time standard deviations of the LRU_W_Mode, LRU_WR_Mode, CFLRU, GCaR-CFLRU, LCR, and MQSim schemes by up to 82, 78, 76, 76, 78, and 40 percent, respectively. Unlike the other cache techniques, Co-Active judiciously addresses the performance-cliff issue by striking a performance balance to properly deal with burst I/O requests.

6.2 Write Amplification

The cache residing in an SSD temporally stores data that are repeatedly updated multiple times in memory, followed by being written into the flash memory. Thus, the write amplification is improved by reducing the amount of data written into the flash memory.

Fig. 7a reveals that the write amplification of MQSim is much higher than those of the other solutions. Testing the 11 traces, we observe that the write amplification of MQSim is 29, 6, 40, 33, 50, 58, 60, 61, and 53 percent higher than those of Co-Active. Co-Active has an edge over MQsim in terms of write amplification, because MQsim employs only the hot-and-cold separation to distinguish all data followed by flushing data into the flash memory in a swift manner. MQsim inevitably causes excessive data written into flash memory, thereby leading to a high write amplification. Nevertheless, MQSim does optimize the write amplification of SSDs where there is the lack of cache. The write-back mode caching scheme presents write-amplification improvement, because data are written back to flash memory only when a cache replacement emerges. Once data in the LRU linked list is updated, the data moves to the head of the LRU linked list and; then, the writing-back procedure is delayed in flash memory.

After testing the ts_0, usr_0, web_0, Fin1 traces, we demonstrate that Co-Active delivers the lowest write amplification among its peers. Under hm_0, stg_0, src2_0 and hm_1, Co-Active's write amplification is 9, 16, 16.1 and 5.3 percent higher than those of the optimal LRU. On the flip side, Co-Active curtails LRU's response time by 64, 42, 46 and 15.7 percent, respectively, because Co-Active advocates for an active write-back method to boost SSD performance. Co-Active seamlessly integrates the cold-and-hot separation module, clean-and-dirty-data separation module, idle chip load, and a clean LRU threshold to subdue write-back conditions. The Co-Active method is adept at boosting SSD performance and lifetime. By changing the threshold of a clean LRU, Co-Active well maintains the balance between performance and SSD lifetime, the details of which are articulated in the scalability experiments.

6.3 Erase Count

When the number of free blocks is lower than the set threshold, GC operations are kicked in to collect invalid data pages and to erase blocks. The types of flash memory changes from SLC to 3D NAND flash. The maximum number of erasures plummets by a few orders of magnitude [44] even though the capacity increases.

SSD lifetime-aware caching schemes play a vital role in NAND flash-based SSDs. The number of available erase operations intuitively reflects SSD lifetime. Fig. 7b shows the numbers of erasures of the evaluated caching schemes managing the replayed real-world traces. The numbers of erasures are *zero* under hm_1 and Websearch1, because the read I/O ratios of both workloads exceed 99 percent, causing no GC operations. In MQSim, the SSD has more erase operations than those of the other strategies. In the context of the eleven traces (i.e., hm_0, proj_0, prxy_0, stg_0, ts_0, usr_0, web_0, Fin1, src2_0, hm_1, and Websearch1), Co-Active prunes the erase operations of MQSim by 40, 7, 70, 48, 81, 85, 78, 48, and 82 percent, respectively. It is evident that Co-Active optimizes SSD lifetime while exhibiting remarkable I/O performance.

6.4 Hit Ratio

The hit ratio is a mixed bag of both read hit ratio and write hit ratio in the experiments.

Write Hit Ratio. A write-hit ratio, which indicates a proportion of write requests that hit cache, largely depends on caching schemes in traditional storage systems. In NAND Flash-based SSDs, a vast majority of the caching schemes are an expansion of LRU or the like. Among these cache policies, there is no conspicuous discrepancy with respect to

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

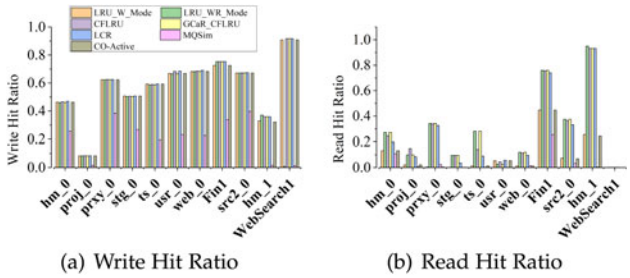


Fig. 8. The hit ratio of the co-active and the alternative caching schemes under the 11 traces.

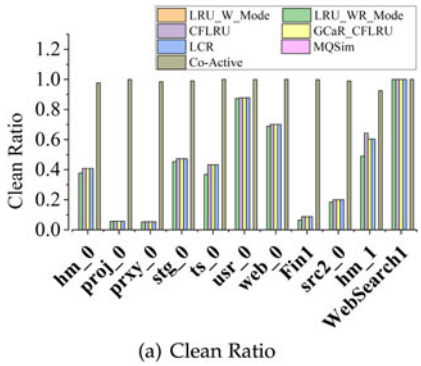


Fig. 9. The clean ratios of co-active and the alternative caching schemes under the 11 traces. A high clean ratio indicates good performance.

write hit ratio. The write hit ratio of MQSim lies below those of the other caching schemes, because MQSim deletes data in the cache when the data are written to flash memory. Thus, the MQSim's cache is not working in a full capacity in most cases compared with the other policies, thereby giving rise to a low hit ratio.

Figs. 8a and 7a depict the comparisons of the cache techniques in terms of write-hit ratio and write amplification. The results plotted in Fig. 7a suggest that a high write-hit ratio leads to a dropped write amplification. This trend is expected, because data in the cache ought to be updated when a write request hits the cache to bypass flash memory. For example, the overall write-hit ratio is the lowest while the corresponding write amplification surges to the highest level under the *proj* workload.

In the cases of hm_1 and Websearch1, the difference of the actual number of write hit counts is marginal among the cache schemes, due to the small write I/O ratios (less than 1.0 percent) under these traces.

Read Hit Ratio. Fig. 8b unravels that the read-hit-ratio distribution is inconsistent with that of the write-hit ratio among all the tested caching schemes. The read-hit ratio of the *LRU_WR_Mode*, CFLRU, GCaR-CFLRU, and LCR edge out the other three cache managements. Owing to temporal and spatial locality, loaded data are likely to be repeatedly accessed in the following period. Thus, the four read-write cache strategies offer higher read-hit ratios than those of the other three policies. Among the four caching schemes, *LRU_WR_Mode* is slightly superior to the other three, because the other strategies preferentially replace clean data in the write-back window. The read-hit ratio of MQSim is the lowest among its peers; the rationale is similar to that of the write-hit ratio. Co-Active's read-hit ratio is almost identical to that of *LRU_W_Mode*, because both cache managements deploy the write-cache mode, in which data accessed from flash memory are not written to the cache.

The average response time is nearly the same across all the cache schemes under Websearch1, because Websearch1's read I/O ratio is above 99.9998 percent a accompanied by a read hit ratio of almost 0 percent (see Fig. 8b). As such, all the caching schemes can hardly optimize performance under this workload.

From the experiments focused on the read/write cache hit ratio, our discover that it is nontrivial to quantify the advantages of caching schemes using the cache-hit ratio for NAND flash-based SSDs. Read/write hit ratio of MQSim is the lowest across the board, but MQSim presents good performance compared with the other caching schemes. Our findings suggest that developers should comprehensively analyze the characteristics of the flash memory medium and the overhead of cache replacement during a design phase of a caching scheme.

6.5 Clean Ratio

A clean ratio quantifies a proportion of replaced clean data when the cache replacement occurs. Recall that the replacement cost of clean data is lower than that of dirty data. The CFLRU scheme preferentially replaces clean data. Although GCaR opts to replace clean data, the clean data is generated by the hit-miss of reads, which makes GCaR a passive policy.

Fig. 9 unveils that clean ratio of MQSim and *LRU_W_Mode*-based caching schemes is zero under the workload, because both schemes belong to the write-cache mode where all evicted data are dirty data. Compared to the *LRU_WR_Mode* techniques, CFLRU improves the clean ratio by 8, 2, 3, 4, 15, 0.5, 1.7, 25, 8, 31 and 0.01 percent in the experiments driven by the 11 traces. If the write-back window of CFLRU grows, the clean ratio can be further pushed up at a marginal rate. Co-Active also enhances the clean ratio of the *LRU_WR_Mode* scheme under the 11 workloads. The clean ratio of Co-Active is almost close to 100 percent under all the tested traces, because Co-Active actively flushes dirty data from cache into flash memory to originate clean data. Meanwhile, clean data is a priority to be evicted to curtail cache replacement cost. Therefore, Co-Active significantly optimizes the overall performance of SSDs.

6.6 Summary

In a nutshell, the percentage of evicting clean data in Co-Active is almost close to 100 percent for most cases; as a result, Co-Active outshines the other cache strategies in terms of cache-replacement cost. Co-Active employs the active write-back function to yield clean data during idle time periods and to preferentially evict clean data in peak accesses. Therefore, Co-Active embraces performance advantages amid I/O burst accesses (e.g., the hm_0, proj_0 and prxy_0 traces). Under the hm_0, proj_0 and prxy_0 workload conditions, Co-Active cuts back the average response time of GCaR-CFLRU by 51.3, 60.2, 83.18 percent, and the performance cliff by 1, 66, 76.05 percent. Compared with MQSim, Co-Active slashes the number of erase operations by 40.81, 7, 70 percent and the average response time by 5, 16.7, 25 percent. The findings strongly reveal that compared with the existing cache solutions, Co-Active is conducive to boosting both the performance and endurance of NAND Flash-based SSDs.

7 Performance Optimization

In this section, we shed bright light on a comparison between Co-Active and MQSim+. It is worth noting that the

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

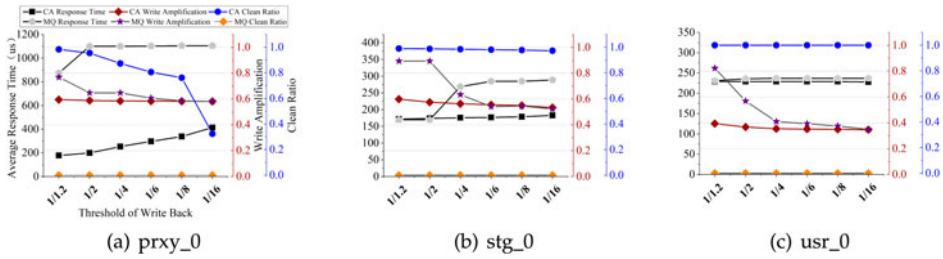


Fig. 10. The impact of write-back thresholds on co-active performance. Good performance is manifested by low average response time, low write amplification, and high clean ratios.

term 'MQSim+' means that a write-back threshold is forged in the caching strategy of MQSim. We articulate the performance optimization of Co-Active that are impacted by various thresholds of clean LRU list (Section 7.1), cache size (Section 7.2), and the functions of separation unit and clean-LRU thresholds (Section 7.3). According to the degree of peak-burst I/O access in workloads, we select three traces to study the performance optimization of Co-Active. The request data ratio in the peak-burst stage of prxy_0 is the highest (29.1 percent) in workloads. The value of stg_0 is the middle (8.4 percent). And the value of usr_0 is lower (2.67 percent). The traces details are listed in Table 1.

7.1 Write-Back Thresholds in Co-Active and MQSim

We devise a write-back threshold in MQSim's cache strategy; the new version of MQSim is referred to as MQSim+. Various write-back threshold values are preset in MQSim+.[4] The extensive experiments are performed to validate the benefits of active write-back mechanism in Co-Active by the comparison of Co-Active and MQSim+. The x-axis in Fig. 10 represents the values of write-back threshold. The symbols CA and MQ denote Co-Active and MQSim+ caching schemes.

Figs. 10a, 10b, and 10c unveil the comparison results under prxy_0, stg_0, and usr_0. Under prxy_0, the response time of Co-Active is superior to that of MQSim at the threshold of $1/1.2 \sim 1/16$, representing an average optimization of 73.7 percent. The response time of MQSim+ with the threshold of 1/1.2 (the threshold with the least restriction on write-back) is 72.8 percent worse than that of MQSim without the threshold. The result indicates a notable impact of the write-back threshold in MQSim+ on the response time. As the write-back threshold decreases (e.g., from 1/1.2 to 1/16), both Co-Active and MQSim share a decreasing trend in write amplification. Nevertheless, Co-Active outperforms MQSim+ by an average of 8.3 percent. This improvement is rooted in Co-Active's larger limitation on write-back operation compared with MQSim+. Under stg_0, Co-Active enhances response time and write amplification by averages of 27.7 and 11.49 percent over MQSim+. Under usr_0, Co-Active improves MQSim+'s response time and write amplification by averages of 2.96 and 25.91 percent. The results confirm that the collaborative active write-back mechanism in Co-Active significantly optimizes cache performance.

As the value of write-back thresholds decreases, the frequency of active write back reduces; as such, write amplification drops. In this case, MQSim and Co-Active share the same tendency. Without loss of generality, we delve into the write-back threshold's impacts on average response time, write amplification, and clean ratio in Co-Active.

Clean ratio is reversely correlated with response time. Thus, a high clean rate leads to a low response time, which translates to superb performance. The clean ratio is largely reliant on the write-back threshold value under the prxy_0 load, because the ratio and the peak value of burst I/O accesses are large in these traces (see also Table 1). When the volume of burst peaks services-based request data is larger than that of clean data in the cache, clean data become insufficient for effective cache replacement. Therefore, the clean ratio drops along with the deteriorated overall performance. With a small write-back threshold value, the clean ratio tends to reach 100 percent when the peak burst I/Os are low. In this case (e.g., the usr_0 trace), increasing write-back thresholds fails in boosting the clean-ratio performance.

Workloads have various degrees of burst-access patterns (see Table 1), it is unlikely for Co-Active to achieve the best performance when a constant threshold is configured across all the workloads. Nevertheless, in the experiments, a static value of write-back threshold is taken in Co-Active for each workload in this work. The write-back thresholds value can be adjusted with cache size in accordance with the burst- I/O degree. Write amplification is used as a reference; the write-back threshold is adjusted to minimize the value of write amplification. The write-back threshold is determined within an acceptable range of write amplification, thereby optimizing write amplification and response time in SSDs. Given a fixed-sized cache, we raise the write-back thresholds to enhance clean ratios with the available write amplification in applications where the clean ratios are evidently affected by the write-back thresholds. Otherwise, we push down the threshold value to strengthen the write amplification.

As one of our future research directions, we will implement a dynamic write-back threshold mechanism in Co-Active to feed the needs of changing peak-burst-stage degrees.

7.2 Cache Size

Now we are in a position to elaborate the impact of cache size on Co-Active under the prxy_0, stg_0 and usr_0 workload. It is noteworthy that the cache size varies from 2 to 128 MB. Fig. 11 shows that the average response time decreases with an increasing cache size. Nevertheless, the response-time

4. In MQSim+, the write-back threshold is represented by a fraction. For example, the ratio 1/16 indicates that the write-back threshold is 1/16 of the cache size. When the cached data exceeds the threshold, no write-back operation will occur. If the threshold is set to a smaller value, the much restrictive the write-back operation is, which less likely triggers a write-back operation. When the data volume in the clean data linked list reaches a preset threshold, the active write back operation is not triggered.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

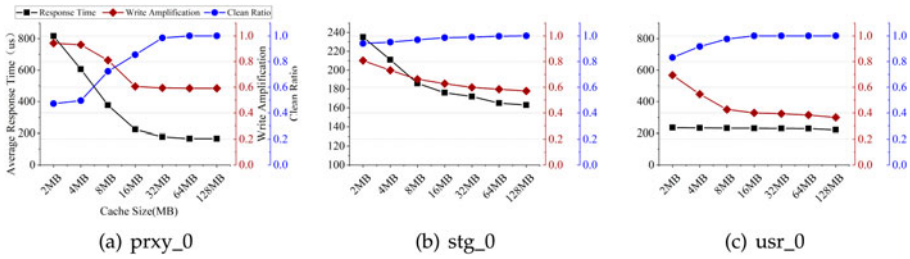


Fig. 11. The impact of cache size on co-active performance. Superb performance is manifested by low average response time, low write amplification, and high clean ratio.

trends differs among the evaluated traces. In the prxy_0 case, for example, the response time plummets by approximately 78 percent as the buffer size is expanded from 2 to 32 MB. When the cache size increases from 32 to 128 MB, the changes in response become stable with merely 6 percent reduction. In the usr_0 trace, the average response time remains relatively flat with the increment of the cache size. Write amplification presents a steady status after decrements.

In summary, an increasing clean ratio impacts the average response time. When the clean ratio is 100 percent, the impact of cache size on response time is diminishing. This trend illustrates that the prioritized replacement for clean data in Co-Active is adroit at enhancing cache performance.

7.3 Benefits of Separation Unit and Clean-LRU Thresholds in Co-Active

Co-Active initially writes back dirty data to generate clean data. Meanwhile, Co-Active prioritizes the replacement of clean data to lift the clean ratio. Co-Active imposes two restrictions on triggering active write back: (1) writing cold data into flash memory and (2) configuring the threshold of clean data in the LRU list (or write-back thresholds, see Section 4.3).

We conduct a group of experiments to quantify the impact of the hot-and-cold separation unit and clean-LRU thresholds on Co-Active driven by the prxy_0, stg_0, and usr_0 workload conditions (see Fig. 12). We configure four customized Co-Active versions. It is worth noting that Co-Active is a system fully integrating the two components; Co-Active-1 means the Co-Active edition without the cold-and-hot separation unit; Co-Active-2 is another version that lacks the clean-LRU threshold; Co-Active- - represents an original system in which both the cold-and-hot separation unit and the threshold are disabled.

Under the two traces, there are no significant differences in response time and clean ratio in the four Co-Active editions. However, disabling the two components makes a surged write amplification. For example, in the prxy_0 trace, write amplification of Co-Active-1, Co-Active-2, and Co-Active- - increases by 19.3, 64.7, and 153.4 percent, respectively. In the case of stg_0, write amplification is elevated by 10.1, 49.1, and 66.2 percent, respectively. Under the usr_0 workload, this metric climbs by 42.1, 103, and 165.2 percent, respectively. Thus, the findings evidently demonstrate that the hot-and-cold separation unit coupled with the threshold of clean data in the LRU list are capable of curtailing write amplification.

8 Related Work

Caching is one of the most effective approaches to optimizing SSD performance. A handful of prominent studies were conducted to shape cache management schemes. Data residing in a cache are evicted or updated according to a replacement strategy. Prevalent cache replacement strategies include, but not limit to, FIFO, LRU, and LFU [29], [45], among which LRU is the most ubiquitously deployed policy.

With the advancement in flash memory, caching schemes that take into account the features of NAND flash memory have gained much attention in the research community. CFLRU [7] divides a LRU linked list into a work areas and a write-back area. When a cache replacement occurs, clean data in the write-back area is preferentially evicted. In follow-up studies, CFLRU were extended to address a few setbacks. For instance, CFDC [46] organizes the corresponding linked lists for dirty data and clean data, aiming to enhance retrieval performance amid the replacement of cache pages. AD-LRU [47] embraces two LRU lists to store cold data and hot data. The cold-data list only manages data that are accessed once, whereas the hot-data list registers data that are loaded two times or more. These CFLRU-based cache management schemes, however, are facing the same obstacle. When a cache replacement kicks in, there exists almost no clean data in the cache for replacement under the write-intensive workload with burst I/Os.

Caching schemes designed for flash storage systems have a block-level caching module supporting hybrid address mappings of log block. These schemes, taking a block as a

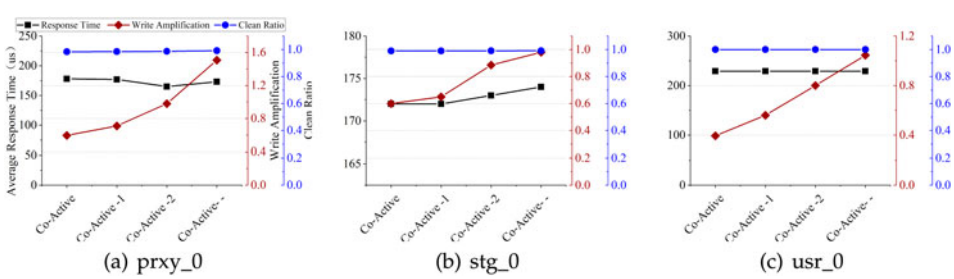


Fig. 12. The impact of the hot-and-cold separate unit and the threshold of *Clean LRU* on co-active performance. Superb performance is manifested by low average response time, low write amplification, and high clean ratio.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

basic storage unit of LRU, include FAB [48], BPLRU [8], and PUD-LRU [49]. The caching strategies cutback the number of merging operations of data blocks and log blocks in flash memory. Unfortunately, the performance of the above techniques is dismal under frequent-random access workloads.

Intriguing and novel updating methods were proposed in the realm of caching schemes. For example, Huang *et al.* [14] devised a caching management scheme that periodically writes data back to flash in a proactive mode. An SSD has three states (idle, run, suspend) and the specific period is configured in advance. The MQSim simulator [15] adopts a write-through strategy combined with a cold-and-hot separation unit. MQsim relentlessly maintains the state of free space, suppressing the frequency of cache replacement to enhance SSD performance. However, the aforementioned caching schemes cause enormous writes into flash memory, which raises write amplification and shortens SSD lifespan.

Recently, GCaR [9] incorporated the GC state of the backend flash memory into the cache management scheme. The GC process originates a large number of internal reads, writes, and erase operations, which adversely affect SSD performance by conflicting with external I/Os. GCaR makes the best effort to keep data handled by the garbage-collection module in the cache. LCR [10], an extended version of GCaR, quantifies a chip's load that depends on internal requests in the chip queue and external requests. During the data replacement in cache, clean data are evicted followed by data with low chip load. The ECR [11] caching scheme optimizes LCR to avoid the negative impact of adjusting the size of a write-back window. Nevertheless, these caching scheme adopts a passive updating method. Facing burst peak I/O services under write-intensive workloads, frequent replacements of dirty data tend to result in a sharp performance cliff.v

Co-Active, adopting the active write-back module, creates sufficient amount of clean data during idle time periods. Thanks to such a write-back strategy, Co-Active edges out the existing caching techniques when it comes to write-intensive workload conditions.

9 Conclusion

We overcome the setbacks of traditional NAND flash-based cache management schemes, where clean data are passively generated by read I/Os keeping the proportion of evicting clean data very small. Amid the presence of burst peak I/Os, the traditional caching schemes induce long response time in SSDs. We proposed Co-Active, a cache management scheme for NVMe SSDs, to incorporate the active write-back function to write dirty data from the cache into flash memory during idle time periods.

Co-Active pushes the percentage of evicting clean data to almost 100 percent under write-intensive workloads, thereby significantly curtailing cache replacement overhead. Co-Active utilizes the chip-level parallelism inside SSDs to concurrently write data from the cache to the flash memory without imposing extra bandwidth overload. Co-Active's cold-and-hot separation unit combined with the clean data threshold is conducive to cutting back the number of write operations caused by the existing write-back cache strategies that impair SSD lifetime. With a cache-data index module in place, Co-Active judiciously enhances the cache-replacement performance by the virtue of the clean-data and dirty-data LRU lists. The experimental results unveil that Co-Active shortens the average response time by up to 83.9 percent with an average of 32.7 percent; Co-Active alleviates the performance-cliff degree of LRU by up to 82.1 percent with an average of 42.3 percent.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants 61702004, 62072001, and 61821003, in part by the State Key Laboratory of Computer Architecture (ICT, CAS) under Grant CARCH201915, and in part by the Natural Science Research Projects at Higher Institutions in Anhui Province under Grant KJ2017A015. The work of Xiao Qin was supported by the U.S. National Science Foundation under Grants IIS-1618669, OAC-1642133, and CCF-0845257.

References

[1] Z. Yang *et al.* "AutoTiering: Automatic data placement manager in multi-tier all-flash datacenter," in *Proc. IEEE 36th Int. Perform. Comput. Commun. Conf.*, 2017, pp. 1–8.

[2] Y. Jin, H.-W. Tseng, Y. Papakonstantinou, and S. Swanson, "KAML: A flexible, high-performance key-value SSD," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2017, pp. 373–384.

[3] J. Wan, W. Wu, L. Zhan, Q. Yang, X. Qu, and C. Xie, "DEFT-cache: A cost-effective and highly reliable SSD cache for RAID storage," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2017, pp. 102–111.

[4] A. Jain and C. Lin, "Back to the future: Leveraging Belady's algorithm for improved cache replacement," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit.*, 2016, pp. 78–89.

[5] N. Megiddo and D. S. Modha, "ARC: A self-tuning, low overhead replacement cache," in *Proc. 2nd USENIX Conf. File Storage Technol.*, 2003, pp. 115–130.

[6] S. Jiang and X. Zhang, "LIRS: An efficient low inter-reference recency set replacement policy to improve buffer cache performance," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 1, pp. 31–42, 2002.

[7] S.-Y. Park, D. Jung, J.-U. Kang, J.-S. Kim, and J. Lee, "CFLRU: A replacement algorithm for flash memory," in *Proc. Int. Conf. Compilers Archit. Synthesis Embedded Syst.*, 2006, pp. 234–241.

[8] H. Kim and S. Ahn, "BPLRU: A buffer management scheme for improving random writes in flash storage," in *Proc. 6th USENIX Conf. File Storage Technol.*, 2008, pp. 1–14.

[9] S. Wu, Y. Lin, B. Mao, and H. Jiang, "GCaR: Garbage collection aware cache management with improved performance for flash-based SSDs," in *Proc. Int. Conf. Supercomput.*, 2016, pp. 1–12.

[10] C. Liu *et al.*, "LCR: Load-aware cache replacement algorithm for flash-based SSDs," in *Proc. IEEE Int. Conf. Netw. Archit. Storage*, 2018, pp. 1–10.

[11] H. Chen, Y. Pan, C. Li, and Y. Xu, "ECR: Eviction-cost-aware cache management policy for page-level flash-based SSDs," *Concurrency Comput., Practice Experience*, 2019, Art. no. e5395.

[12] C. Gao *et al.*, "Exploiting parallelism for access conflict minimization in flash-based solid state drives," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 168–181, Jan. 2018.

[13] M. Jung, "Exploring design challenges in getting solid state drives closer to CPU," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1103–1115, Apr. 2016.

[14] M. Huang, O. Serres, V. K. Narayana, T. El-Ghazawi, and G. Newby, "Efficient cache design for solid-state drives," in *Proc. 7th ACM Int. Conf. Comput. Front.*, 2010, pp. 41–50.

[15] MQSim Github repository. 2020. [Online]. Available: https://github. com/CMU-SAFARI/MQSim

[16] S. Kim and J.-S. Yang, "Optimized I/O determinism for emerging NVM-based NVMe SSD in an enterprise system," in *Proc. 55th Annu. Des. Autom. Conf.*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3195970.3196085

[17] Kimber Smith-Fidler. 2020. [Online]. Available: https://www.nvmedeveloperdays.com/English/Collaterals/Press_Releases/2018/20181205_G2M_NVMeResearch.pdf

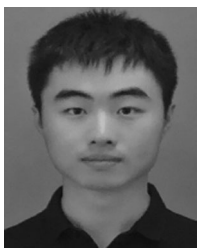
[18] A. Tavakkol *et al.* "FLIN: Enabling fairness and enhancing performance in modern NVMe solid state drives," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit.*, 2018, pp. 397–410.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.

[19] A. Tavakkol, J. Gómez-Luna, M. Sadrosadati, S. Ghose, and O. Mutlu, "MQSim: A framework for enabling realistic studies of modern multi-queue SSD devices," in *Proc. 16th USENIX Conf. File Storage Technol.*, 2018, pp. 49–66.
[20] S. Yan *et al.*, "Tiny-tail flash: Near-perfect elimination of garbage collection tail latencies in NAND SSDs," *ACM Trans. Storage*, vol. 13, no. 3, Oct. 2017. [Online]. Available: https://doi.org/10.1145/3121133
[21] J. Guo, Y. Hu, B. Mao, and S. Wu, "Parallelism and garbage collection aware I/O scheduler with improved SSD performance," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2017, pp. 1184–1193.
[22] Micron Technology, Inc. 2020. [Online]. Available: https://www.micron.com/-/media/client/global/documents/products/technical-note/nand-flash/tn2961_wear_leveling_in_nand.pdf
[23] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," *Proc. IEEE*, vol. 105, no. 9, pp. 1666–1704, Sep. 2017.
[24] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives," in *Proc. IEEE*, vol. 105, no. 9, pp. 1666–1704, 2017, doi: 10.1109/JPROC.2017.2713127.
[25] J. Guo, Y. Hu, B. Mao, and S. Wu, "Parallelism and garbage collection aware I/O scheduler with improved SSD performance," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2017, pp. 1184–1193.
[26] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, "Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity," in *Proc. Int. Conf. Supercomput.*, 2011, pp. 96–107.
[27] T. Blaß, S. Hahn, and J. Reineke, "Write-back caches in WCET analysis," in *Proc. 29th Euromicro Conf. Real-Time Syst.*, 2017, pp. 26:1–26:22. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/7158
[28] R. I. Davis, S. Altmeyer, and J. Reineke, "Analysis of write-back caches under fixed-priority preemptive and non-preemptive scheduling," in *Proc. 24th Int. Conf. Real-Time Netw. Syst.*, 2016, pp. 309–318. [Online]. Available: https://doi.org/10.1145/2997465.2997476
[29] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proc. ACM SIGMETRICS Conf. Meas. Model. Comput. Syst.*, 1990, pp. 143–152.
[30] Y. Smaragdakis, S. Kaplan, and P. Wilson, "EELRU: Simple and effective adaptive page replacement," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 27, no. 1, pp. 122–133, 1999.
[31] T. Johnson *et al.*, "2Q: A low overhead high performance buffer management replacement algorithm," in *Proc. 20th Int. Conf. Very Large Data Bases*, 1994, pp. 439–450.
[32] D. Lee *et al.*, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001.
[33] UMass trace repository. 2020. [Online]. Available: http://traces.cs.umass.edu/index.php/Storage/Storage
[34] Snia traces. 2020. [Online]. Available: http://iotta.snia.org/traces/388
[35] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: Practical power management for enterprise storage," in *Proc. 6th USENIX Conf. File Storage Technol.*, 2008, pp. 253–267.
[36] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings," *ACM SIGPLAN Notices*, vol. 44, no. 3, pp. 229–240, 2009.
[37] A. Tavakkol, P. Mehrvarzy, M. Arjomand, and H. Sarbazi-Azad, "Performance evaluation of dynamic page allocation strategies in SSDs," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 2, pp. 1–33, 2016.
[38] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 254–265, 1998.
[39] J.-W. Hsieh, T.-W. Kuo, and L.-P. Chang, "Efficient identification of hot data for flash memory storage systems," *ACM Trans. Storage*, vol. 2, no. 1, pp. 22–40, 2006.
[40] Z. Fan, D. H. C. Du, and D. Voigt, "H-ARC: A non-volatile memory based cache policy for solid state drives," in *Proc. 30th Symp. Mass Storage Syst. Technol.*, 2014, pp. 1–11.
[41] S. Wan, Q. Cao, X. He, C. Xie, and C. Wu, "An adaptive cache management using dual LRU stacks to improve buffer cache performance," in *Proc. IEEE Int. Perform. Comput. Commun. Conf.*, 2008, pp. 43–50.
[42] Micron Technology, Inc. 2020. [Online]. Available: https://datasheetspdf.com/pdf/843436/Micron/MT29F256G08CKCAB/1
[43] M. Jung and M. T. Kandemir, "Sprinkler: Maximizing resource utilization in many-chip solid state disks," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit.*, 2014, pp. 524–535.
[44] S. Liu and X. Zou, "QLC NAND study and enhanced gray coding methods for sixteen-level-based program algorithms," *Microelectron. J.*, vol. 66, pp. 58–66, 2017.
[45] D. Lee *et al.* "On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 1999, pp. 134–143.
[46] Y. Ou, T. Härder, and P. Jin, "CFDC: A flash-aware replacement policy for database buffer management," in *Proc. 5th Int. Workshop Data Manage. New Hardware*, 2009, pp. 15–20.
[47] P. Jin, Y. Ou, T. Härder, and Z. Li, "AD-LRU: An efficient buffer replacement algorithm for flash-based databases," *Data Knowl. Eng.*, vol. 72, pp. 83–102, 2012.
[48] H. Jo, J.-U. Kang, S.-Y. Park, J.-S. Kim, and J. Lee, "FAB: Flash-aware buffer management policy for portable media players," *IEEE Trans. Consum. Electron.*, vol. 52, no. 2, pp. 485–493, May 2006.
[49] J. Hu, H. Jiang, L. Tian, and L. Xu, "PUD-LRU: An erase-efficient write buffer management algorithm for flash memory SSD," in *Proc. IEEE Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, 2010, pp. 69–78.



Hui Sun received the PhD degree from the Huazhong University Science and Technology, Wuhan, China, in 2014. He is an associate professor of computer science with Anhui University. His research interests include computer systems, edge computing, performance evaluation, non-volatile memory-based storage systems, file systems, and I/O architectures.



Shangshang Dai is currently working toward the MS degree at Anhui University, Hefei, China. His main research interests include computer systems, edge computing, and key-value storage systems.



Jianzhong Huang (Member, IEEE) received the PhD degree in computer architecture from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2005, and completed the post-doctoral research in information engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2007. He is currently an associate professor with the Wuhan National Laboratory for Optoelectronics, HUST. His research interests include computer architecture and dependable storage systems.



Xiao Qin (Senior Member, IEEE) received the BS and MS degrees in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 1992 and 1999, respectively, and the PhD degree in computer science from the University of Nebraska-Lincoln, Lincoln, Nebraska, in 2004. He is currently an alumni professor and the director of Graduate Programs wih the Department of Computer Science and Software Engineering, Auburn University. He won an NSF CAREER Award, in 2009. His research interests include parallel and distributed systems, storage systems, fault tolerance, real-time systems, and performance evaluation.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:38:00 UTC from IEEE Xplore. Restrictions apply.