A-Cache: Asymmetric Buffer Cache for RAID-10 Systems Under a Single-Disk Failure to Significantly Boost Availability

Bin Zhou®, Hong Jiang®, Fellow, IEEE, Qiang Cao®, Senior Member, IEEE, Shenggang Wan®, and Changsheng Xie®, Member, IEEE

Abstract—The RAID-10 architecture has been widely deployed in commercial and industrial storage environments over the past two decades due to its high reliability, availability, and performance. However, during the recovery process of a single disk failure, which accounts for more than 99.75% of the disk failure scenarios, it is still at a high risk of data loss and suffers from a degradation of user I/O performance, which results from the severe interference between the user and recovery I/Os. Based on our observations and analyses, we find highly asymmetric disk bandwidth utilization and disk I/O interference during the recovery process of the RAID-10 systems under the single faulty-disk condition. Motivated by the fact that this asymmetry can be leveraged to significantly and simultaneously speed up the recovery and user I/O performances. As a result, we propose a novel asymmetric buffer cache management scheme, called A-Cache, to mitigate this asymmetry by allocating cache space asymmetrically between the disks that do not participate in the recovery process and those that do. To verify the effectiveness of the A-cache, we have integrated A-Cache into the popular cache algorithms LRU, named A-LRU. The evaluation of our prototype system demonstrates that A-LRU is able to significantly speed up the recovery speed and average user I/O latency under various typical configurations, compared to the original LRU, without any additional hardware cost.

Index Terms—Buffer cache replacement algorithm, performance, reconstruction, reliability.

I. INTRODUCTION

RAID-10 systems are widely deployed in commercial and industrial production environments to provide highly reliable and available online storage services [1]–[5]. However, when a disk failure occurs and a recovery process is initiated, the RAID-10 system can still face a high risk of data loss and a heavy degradation of I/O performance compared to its failure-free state. This is because under recovery, not only are the disks in a RAID system (including the RAID-10 system) required to provide continuous service to the disk I/Os incurred by user requests, referred to as user I/Os, but they must also perform a recovery process to reconstruct the lost data on the faulty disk to a spare disk, inducing additional disk I/Os that are referred to as recovery I/Os. Due to the unique way by which hard disk drives process I/O operations, the recovery and user I/Os can seriously interfere with each other, thus leading to longer user I/O latency and recovery time simultaneously, leading to a higher risk of data loss as well [6]–[16]. In this article, we focus on the recovery period of the RAID-10 system with a single faulty disk for the following reasons. First, over 99.75% of disk failures [17], [18] are single-disk failures. Second and importantly, when one disk in a RAID-10 system fails, the data on its mirror disk is in a critical state in that any subsequent sector errors on the mirror disk, which can be caused by high-fly writes, scratches, and smeared soft particles [19], can lead to unrecoverable data loss. Therefore, shortening the length of the recovery period, which is often referred to as the window of vulnerability, is critically important in improving the reliability and availability of the RAID-10 system.

By analyzing the distribution of the recovery and user I/Os among all surviving disks, and the interference between the recovery and user I/Os during the recovery process of the RAID-10 system, we obtain the following key observations on two types of asymmetry that provide important insights and opportunities to simultaneously and significantly speed up the recovery and user I/O performances during RAID-10’s recovery process.

1) Asymmetric Utilization of Disk Bandwidth: The asymmetric spatial distribution of recovery I/Os and the asymmetric temporal distribution of the user I/Os combine to cause the asymmetric utilization of disk bandwidth among the surviving disks of a RAID-10 with a single
faulty disk. More specifically, while almost all disk I/O bandwidth on the disks participating in recovery (DP) is spent on performing both the user and recovery I/Os, the bandwidth on the disks not participating in recovery (DNP) is not efficiently explored and exploited because the DNP disks serve the user I/Os exclusively often with surplus/unused bandwidth.

2) **Interference Between Recovery and User I/Os:** Due to the spatial locality usually existing exclusively among the user I/Os or exclusively among the recovery I/Os, disk head seek distances in such exclusive scenarios, where a disk is accessed only by the user I/Os or the recovery I/Os, are usually much shorter than that in a case where the user I/Os and recovery I/Os are mixed and interleaved on a disk because of the frequent long-distance seek operations. These long-distance seek operations exacerbate the bandwidth contention on the DP disks by significantly reducing the efficiency in which this bandwidth is used, leading to both poor user I/O performance and poor recovery performance.

However, the buffer cache commonly deployed on top of the disk array typically overlooks the above two types of asymmetry and allocates buffer symmetrically between the DP and DNP disk groups. We argue that the above asymmetry can be mitigated for improving recovery and user I/O performances by an asymmetric buffer allocation. This motivates us to propose a novel mechanism, called asymmetric cache (A-Cache), taking full-associative cache as an example, to improve the data reliability and user I/O performance of RAID-10 systems during the recovery process of their single faulty-disk.

More specifically, A-Cache dynamically reallocates a certain amount of cache space originally assigned to the DNP disks to the DP disks during recovery. As a result, more user requests targeted at the DP disks can be absorbed by the increased cache space, and more user requests targeted at DNP disks are transformed to user I/Os on the DNP disks with surplus disk bandwidth due to the decreased cache space, mitigating the asymmetric utilization of disk bandwidth between DP and DNP disks. Furthermore, since the saved bandwidth on the DP disks is used to perform recovery I/Os, not only are the recovery I/Os given more disk bandwidth but they are also relieved of much interference from the user I/Os to allow for better exploitation of the strong spatial locality among recovery I/Os, leading to significantly improved recovery performance; in the meanwhile, due to the increasing number of recovery I/Os with a short seek latency, the queuing time of user I/Os on the DP disks will also be reduced, which leads to improved user I/O performance on the DP disks. On the other hand, since the DNP disks only need to handle user I/Os exclusively and given the surplus bandwidth on these disks, the increased user I/Os there can be very efficiently serviced, thus mitigating the impact of the increasing user I/Os on the user I/O performance on the DNP disks.

The main contributions of the A-Cache study include the following.

1) We provide a comprehensive analyses on the asymmetry property of RAID-10 systems under a single disk failure during recovery in distribution and interference of disk I/Os, which provides insights into and opportunities for simultaneously speeding up the recovery and user I/O performances.

2) We develop a novel buffer-cache mechanism, called A-Cache, in which the buffer cache space originally allocated to the DNP disks is redistributed to the DP disks during recovery in an asymmetric manner to effectively mine the observed two types of asymmetry for significantly improved recovery and user I/O performances simultaneously.

3) We demonstrate the feasibility and applicability of A-Cache through an example of integrating it into the simple but classic cache replacement algorithm of least-recently used (LRU).

4) A prototype of A-Cache based on LRU is implemented to evaluate the effectiveness and efficiency of A-Cache. The extensive trace-driven evaluation of the A-Cache prototype shows that A-Cache is able to simultaneously improve the user performance and the recovery speed over the state-of-the-art schemes. For example, in an 8-disk RAID-10 system with a single faulty disk, under a read-most workload DA, it can speed up the recovery process by up to 9.25X as well as the average response time of user disk I/Os by 1.03X, or speed up the response time of user disk I/Os by 1.37X as well as the recovery speed by 1.95X.

The remainder of this article is organized as follows. Section II reviews the background of existing techniques developed to speed up RAID-10’s recovery and the buffer cache replacement algorithms. In Section III, we describe our observations on the recovery process of the RAID-10 system, which motivated us to propose A-Cache. Section IV represents the general design of the architecture of A-Cache, and the detailed design of A-Cache through an LRU-based example. Section V describes the prototype of A-Cache. Section VI describes the experimental settings and extensive trace-driven evaluation results of A-Cache. Section VII discusses the application space of A-Cache beyond the RAID-10 system. We conclude this article in Section VIII.

II. BACKGROUND AND RELATED WORK

In this section, we will provide the necessary background knowledge about the RAID-10 system, the existing optimizations to speed up the recovery process of RAID-10 systems, and cache replacement algorithms, the main topics of the focus for our A-Cache research.

A. RAID-10

RAID-10 system is a combination of a RAID-0 system and a RAID-1 system. In most cases, RAID-10 provides better throughput and latency than all other RAID levels except RAID-0 (which wins in throughput). Thus, it is the preferable RAID level for I/O-intensive applications, such as database, e-mail, and Web servers, as well as for any other use requiring high disk performance.

To smooth the gap between the upper-level computational components and the lower-level storage devices, a buffer cache is commonly deployed in RAID-10 systems on top of
disks. A buffer cache management module (CM) [20], [21] is employed to manage the buffer cache algorithms which selects replacement pages residing in the buffer cache.

On the other hand, a disk array management module (DM) [22]–[24] is commonly deployed in RAID-10 systems to be in charge of managing the underlying disk arrays and to provide the detailed configuration information of the underlying disk array to the buffer CM module.

B. Speeding Up the Recovery Process

As we mentioned in Section I, the poor performance in recovery speed and user I/O response time are induced by the interference between recovery and user I/Os. As a result, many solutions have been proposed to speed up the recovery process in RAID-10 systems under faulty conditions [6]–[16], [25], which are either based on the recovery I/Os [6]–[11], [14], or focused on the user I/Os [12], [13], [15], [16], [25].

The recovery-I/O-oriented optimizations mainly rely on the static data layout to balance or reduce the recovery I/Os, and/or on judiciously scheduling recovery I/Os at the runtime (e.g., adjusting I/O granularity, sequence, or address).

Menon and Mattson [7] presented a method to distribute spare capacity to all disks, which not only reduces the amount of data loss per disk but also parallelizes the reconstruction process. However, the amount of data accessed per year [26] has become an important measure of the reliability for hard disk drives in that a disk containing spare space may be less reliable than a dedicated disk. The track-based recovery (TBR) [9] algorithm provides a tradeoff between block-based recovery and cylinder-based recovery, and balances the user I/O latency and the recovery window. However, TBR requires much more buffer space than the block-based recovery algorithms. The pipelined recovery (PR) scheme [10] improves on TBR by significantly reducing the buffer requirements to be close to that of the block-based recovery algorithms. The disk-oriented recovery (DOR) algorithm [8], [27] rebuilds the array at the disk level instead of the stripe level. With this approach, DOR can absorb the bandwidth of the array as much as possible. The popularity-based recovery (PRO) algorithm [11], [28]–[30] can mitigate the severe interference between the user and recovery I/Os by reconstructing the user requested hot zone first. However, when that zone has already been reconstructed, the user-recovery interference will become worse.

Other approaches aim to reduce the arrival rate of user I/Os to the surviving disks during recovery by exploiting the I/O processing ability of the spare disk, using additional hardware and/or buffer scheduling.

Two techniques called redirection of reads and piggybacking of writes [25], [31] are proposed to reduce the user I/Os on the surviving disks that participate in recovery by employing the reconstructed spare disk to absorb parts of the I/Os to the faulty disk. In fact, given the recovery process of RAID-10 systems with a single faulty disk, these two techniques will balance the user I/Os between the only two DP.

MICRO [12] achieves improved recovery performance by writing the in-memory surviving data of the faulty disks into a spare disk first and using a file popularity table to find the hotspot. MICRO treats all the blocks in the cache equally and is not aware of the aforementioned underlying asymmetry between DP and ones not participating (see Section III for more details).

WorkOut [13] and IDO [16] are both array-caching-array methods that offload the write I/Os and read I/Os to another disk array called the surrogate array. As a result, they simultaneously speed up the recovery process and improve the user I/O latency. However, both WorkOut and IDO require an extra disk array to help with the recovery, which, apart from the additional hardware cost, may incur extra problems. For example, the data temporarily stored on the surrogate array can face security problems, such as information leakage, or even tampered information. In addition, during the long recovery process, the I/Os to the outsourced data may compete with the original user I/Os with the same target data for the I/O bandwidth, leading to unexpected performance problems (e.g., the disks in the surrogate array also need to service mixed user I/Os with poor spatial locality).

Similar to the latter type of technique, A-Cache also tries to solve the problem by manipulating user I/Os. In addition, A-Cache is not in conflict with the redirection of reads, piggybacking of writes, and MICRO, and it is cost effective compared to WorkOut and IDO.

C. Buffer Cache Replacement Algorithm

The buffer cache is an integral part and works closely with RAID-based storage systems. To improve the efficiency of the buffer cache, researchers have proposed many stack-based cache-replacement algorithms, such as LRU [32], 2Q [33], MQ [34], [35], LIRS [36], [37], ARC [38], DULO [39], VDF [15], and FBF [20] to name just a few most representative ones. Each cache-replacement algorithm weighs the cached blocks with a different method, such as access interval, number of I/Os induced by a miss, and so on, then decides which blocks to evict.

The LRU algorithm, a stack-based algorithm, is one of the most popular and effective policies for buffer CM. When a block needs to be inserted into the cache, the candidate block to be evicted is the block that is the least recently used. Many variants of the LRU algorithm, such as LRU-k, 2Q, MQ, LIRS, and ARC, which are also stack-based algorithms, have been designed and demonstrated good performance under various scenarios. Different from the above algorithms that only consider temporal locality, FBF [20], DULO [39], and DISKSEEK [40] consider both temporal and spatial locality when a block needs to be replaced.

VDF [15] is a cache replacement scheme that is activated only during recovery and treats the blocks belonging to the faulty disks more favorably, which is different from all the above algorithms that treat blocks from different disks symmetrically. However, VDF aims at reducing the total number of user I/Os by reducing the number of degraded reads, which only occur in parity-based RAID systems, e.g., RAID-5 or RAID-6 systems, but not the mirror-based RAID-10 systems. And more importantly, it still in effect balances the user I/Os among all the underlying surviving disks.
In this section, we motivate the A-Cache research by analyzing our key observations on the significant asymmetry in disk bandwidth utilization between the DP and DNP disks in RAID-10 systems.

A. Asymmetric Disk Bandwidth Utilization

Fig. 1 illustrates an example of the recovery process of a RAID-10 system with 8 disks. As shown in this figure, the distribution of recovery I/Os among the underlying disks is asymmetric. The lost data on the faulty disk (disk 0) is fetched from its mirror disk (disk 1) and then written to the spare disk, which results in recovery I/Os on these two disks, referred to as DP disks; whereas, there is no recovery I/O on the remaining disks (disks 2∼7), referred to as DNP disks.

In the meanwhile, the temporal distribution of user I/Os is also asymmetric, due to the known fact that the user I/O rate rarely reaches its peak value in practical scenarios, as concluded in many workload studies, such as those on Facebook’s image storage system Haystack [41] and Microsoft’s production servers [42].

These two types of I/O workload asymmetry together result in the asymmetric utilization of disk bandwidth between the DP and DNP disks during recovery, as follows. During recovery, the unused bandwidth on the DP disks is used to schedule recovery I/Os. Clearly, while the recovery speed will be limited and the user performance degraded on these DP disks because of the sharing and interference between the user I/Os and recovery I/Os on the DP disks, the same unused bandwidth on the DNP disks is usually wasted.

B. Interference Between User and Recovery I/Os

In fact, due to the unique way by which hard disk drives process I/O operations, the interference between user I/Os and recovery I/Os is usually much more severe than that exclusively among the user I/Os or exclusively among the recovery I/Os, as illustrated in Fig. 2.

This is because the spatial locality usually existing among the user I/Os implies that the seek distance between two consecutive requested addresses of user I/Os is usually very short, resulting in a short seek latency, e.g., the user I/Os U1∼4 in Fig. 2(a). Similarly, the spatial locality among the recovery I/Os is even stronger, since recovery I/Os are typically sequential, giving rise to even shorter seek latency, e.g., the last recovery I/O R3 in Fig. 2(b). In contrast, the mixed user and recovery I/Os rarely share any spatial locality, leading to very long seek latency, due to the following two reasons. First, the recovery process typically needs to scan an entire disk to recover the lost data of a faulty disk in a RAID-10 system. Second, the working set of a user workload is typically a very small portion of all data stored in a RAID-10 system. As a result, the interleaved user and recovery I/Os will very likely lead to frequent back-and-forth long-distance seek operations, resulting in increased I/O latency for both the user, e.g., the user I/O U4 in Fig. 2(b), and recovery I/Os, e.g., the recovery I/Os R1∼2 in Fig. 2(b).

Due to the above asymmetric interference between the user and recovery I/Os, the free disk bandwidth on the DP disks is poorly utilized in that a significant portion of that bandwidth is wasted in scheduling long-distance seek operations, leading to both poor recovery and user performances on the DP disks. In fact, the user I/O performance on the DP disks trends to be much poorer than that on the DNP disks where there is no such I/O interference, since the user I/Os on the DP disks suffer from not only a long service latency but also a long queuing latency induced by the long-distance seek operations, e.g., the user I/O U4 in Fig. 2(b).

C. Our Approach: Asymmetric Buffer Cache Management

Buffer cache is commonly deployed in RAID-10 systems on top of disks. Cache replacement algorithms are employed to manage pages residing in the buffer cache. In fact, the user I/Os arriving at the disks are actually misses in the buffer cache.

Contrary to the conventional wisdom behind all existing replacement algorithms, however, we argue that it is the asymmetry that should be explored and exploited in the buffer cache, a belief that is rooted in the aforementioned asymmetric disk bandwidth utilization and disk I/O interference on the
recovery process of a single faulty disk after an asymmetric buffer allocation between the DP and DNP disk groups. (a) Disk I/Os on a DNP Disk with asymmetric buffer allocation, three new users I/Os, denoted by NU1 ~ 3, are sent to the DNP disk, in addition to the four old users I/Os. However, the DNP disk can easily handle these extra user I/Os due to its surplus disk bandwidth and the low interference among the user I/Os. (b) Disk I/Os on a DP disk due to the asymmetric buffer allocation, the old user I/O U3 is absorbed by the upper-level buffer. As a result, the user I/O U4 need not wait for a long time to be scheduled, since R2 has a very short seek latency owing to the spatial locality among exclusively recovery I/Os. Furthermore, more recovery I/Os (R4 ~ 7) can be executed, thus speeding up the recovery process.

Based on our observations and analyses, we believe that the buffer cache can be asymmetrically and judiciously partitioned between the DNP and DP disks to divert user I/Os from the heavily contended DP disks to the uncontended DNP disks. Fig. 3 illustrates the operational states and timings of I/Os on the DP and DNP disks after the asymmetric buffer allocation.

By giving more cache space to the DP disks, the number of cache misses and thus user I/Os targeted at the DP disks may be reduced accordingly, e.g., the absorbed user I/O U3 in Fig. 3(b). As a result, the recovery I/Os on the DP disks have more available disk bandwidth to speed up the recovery process, e.g., the extra recovery I/Os R4~7 in Fig. 3(b) compared to Fig. 2(b). In addition, with the further increasing number of recovery I/Os with short seek distances, the user I/O on the DP disks may benefit from a shorter queuing time, e.g., the user I/O U4 in Fig. 3(b), since the user I/Os are typically preferentially scheduled to the recovery I/Os to maintain a high user I/O performance.

On the other hand, the decreased cache space allocated for the DNP disks will result in increased misses, and thus user I/Os targeted at these DNP disks. The DNP disks should be able to readily accommodate these extra user I/Os because of their available surplus disk I/O bandwidth and the relatively low interference exclusively among the user I/Os, compared to the DP disks, e.g., the status of I/O scheduling on the DNP disk in Fig. 3(a). All the above analyses are proved by our experiment results as shown in Fig. 10.

Therefore, we propose an asymmetry-exploiting approach, called A-Cache, to speed up both the recovery performance and the user I/O performance of the RAID-10 systems with a single-faulty disk by focusing on exploiting the asymmetry in cache allocation. With its asymmetric buffer CM, A-Cache is able to effectively mine the asymmetry in disk bandwidth utilization between the DNP and DP disks and in disk I/O interference between the user and recovery I/Os to simultaneously boost the recovery and user I/O performances.

### IV. DESIGN OF A-CACHE

In this section, we first present the general design principle of A-Cache by way of an example that demonstrates how A-Cache is integrated with the LRU algorithm. Before our discussion, we summarize the abbreviations in this section in Table I.

#### A. General Design Principles

The architecture of A-Cache, designed on the general description given in Section III-C, is illustrated in Fig. 4. To allocate the cache space asymmetrically during the recovery process, two existing functional modules are enhanced in the RAID-10 systems. One is the enhanced CM module, which is in charge of enforcing both the symmetric (regular) and asymmetric cache space management policies under the failure-free and single faulty-disk conditions, respectively. The other is the enhanced DM module, which is in charge of sending control messages to CM to start or stop the asymmetric cache-space management, providing the detailed configuration information of the underlying disk array to CM, and observing important parameters to support dynamic and adaptive asymmetric cache replacement.

Under the failure-free condition, the buffer cache for the disk array will be managed by a regular algorithm and CM treats all the underlying disks symmetrically. To be able to asymmetrically allocate the cache space under the single faulty-disk condition, CM must obtain the detailed layout information of the underlying disk array, including the total number and device IDs of the disks, the strip size, and the interleaving algorithm that balances the user I/Os among all disks. With such information, CM will be able to determine which cache blocks belong to which disks.

Once a disk in the RAID-10 system fails and the recovery process starts, CM will send a message to inform CM about the failure and the device IDs of the DP disks. When CM receives the message about the failure, the asymmetric DP
and DNP disk-aware cache-space management is activated. CM will try to dynamically and adaptively evict blocks in the cache space to balance all disk I/Os among all underlying disks, including the DP and DNP disks.

When the recovery process is finished or a second disk fails, DM will inform CM with another message, which forces CM to stop/adjust the asymmetrical cache-space management.

Clearly, to make A-Cache practical, it needs a smooth transition between asymmetric disk-aware and symmetric cache replacement, which will be demonstrated with an example of integrating A-Cache with LRU, a simple but classic stack-based cache replacement algorithm in the next section.

### B. Integrating A-Cache With LRU

LRU is a typical and widely used stack-based cache replacement algorithm. The main rationale behind LRU is to replace the least recently used block when the stack is full and a miss occurs. A linked list is usually used to organize all blocks in the cache to maintain an $O(1)$ temporal complexity for looking up the LRU block. To integrate A-Cache with LRU, we keep the original organization of blocks in the cache through a global LRU-style linked list (denoted by GL) under the failure-free condition. Meanwhile, to keep a low temporal complexity for disk-aware block lookup under the faulty condition, the blocks from one disk and its mirror disk are organized into a dedicated LRU-style linked list (denoted by DL), since these two disks maintain the same data.

In summary, the cache blocks are linked into two lists: 1) a DL and 2) the GL. Fig. 5 demonstrates the block organization of an LRU-based A-Cache built upon an 8-disk RAID-10 system.

Similar to the original LRU algorithm, when a block is accessed (either a hit or a miss occurs), it will be placed into the most recently used (MRU) positions of both the GL and the corresponding DL, whether the replacement is symmetric or asymmetric. When the symmetric cache replacement is adopted, the lookup for the block to evict is conducted through the GL. In contrast, when the asymmetric cache replacement is activated, the eviction of cache blocks is conducted via the DLs. In doing so, A-Cache can work in both asymmetric
disk-aware and symmetric styles, and have a smooth transition between these two styles.

Now the only remaining key problem is how to dynamically and adaptively choose the block to evict via the DLs under the faulty condition, which is discussed in the next section.

### C. Making A-Cache Dynamic and Adaptive

It is important to conduct dynamic and adaptive cache replacement when allocating buffer cache space asymmetrically. For example, too large an extra cache space allocated to the DP disks will have a diminishing return on reducing the misses to the DP disks to speed up the recovery, due to the descending marginal effect of the cache space (e.g., in usual, hits per MB allocated cache space will rapidly decrease); meanwhile, the misses to the DNP disks will dramatically increase with the diminishing cache space. On the other hand, if the cache space allocated to the DP disks is too small, the free bandwidth on the DNP disks may not be effectively exploited.

By analyzing the I/O scheduling conditions on the DNP and DP disks, respectively, as described in Section III-C, we have the following interesting observations on the average response time (including queuing time and service time) of the user I/Os.

1) The average response time on the DP disks is longer than that on the DNP disks when the recovery I/Os begin to be scheduled, due to the heavy interference between the user and recovery I/Os, as illustrated in Fig. 2.
2) The average response time of user I/Os on the DNP disks is likely to increase with the increasing cache misses on the DNP disks due to the increasing probability of queuing, as illustrated in Fig. 3(a).
3) The average response time of user I/Os on the DP disks is likely to decrease due to the shorter queuing time, as illustrated in Fig. 3(b), and then becomes stable with the reduction of cache misses on the DP disks during the recovery process. Consider the extreme case in which there is only one user I/O per second. The user I/O is likely to wait for the finishing of a recovery I/O with a short seek latency due to almost all recovery I/Os are sequential in such scenarios, and a stable servicing time which is dominated by the seek latency between the user requested zone and recovery requested zone.

Based on these observations and insights, we choose the average response time of recent user I/Os in each DL as the
metric to conduct the eviction of blocks. More specifically, we record the response time of recent $k$ user I/Os in each DL, and accumulate these $k$ response time whenever a user I/O is finished. When a block needs to be evicted under the single faulty-disk condition, we choose the LRU block in the DL having the minimum accumulation of response time as the candidate.

This scheme is also adjustable, by multiplying the average response time of the user I/Os on the DP disks with a constant coefficient. More specifically, when we turn up that coefficient, it implies that the blocks from the DP disks have a higher priority to stay in the cache, which leads to an increased recovery speed. Otherwise, it implies that these blocks have a lower priority, lowering the recovery speed.

Detailed descriptions of A-LRU for RAID-10 of $2^n$-disk are given in Algorithm 1.

Algorithm 1: A-LRU for RAID-10 of $2^n$ disks

Input: The request stream $x_1, x_2, x_3, \ldots, x_i, \ldots$

A_LRU_Replace($x_i$)

/*For every $i \geq 1$ and any $x_i$, one and only one of the following cases must occur. And we assume $x_i$ should be in $DL_j$*/

if There are no DNP disks in the RAID-10 then
  /*The RAID-10 works under failure-free state or every disk pair (one disk and its mirror disks) has one and only one faulty disk. The cache should be managed in original LRU style*/
  if $x_i$ is in GL then
    /*A cache hit has occurred.*/
    Move $x_i$ to the heads of GL and $DL_j$.
  else
    /*A cache miss has occurred.*/
    if Cache is full then
      /*Replace the LRU block of GL*/
      Delete the LRU block of GL to get a free block, and also delete this block from the corresponding DL.
    else
      /*The RAID-10 works under failure state and there is at least one DNP disk. The cache should be managed in asymmetrical style.*/
      Get a free block.
      Load $x_i$ from the disk and update the response times of recent $k$ user I/Os of $DL_j$.
      Add $x_i$ to the heads of GL and $DL_j$.
  end
else
  /*The RAID-10 works under failure state and there is at least one DNP disk. The cache should be managed in asymmetrical style.*/
  if $x_i$ is in GL then
    /*A cache hit has occurred.*/
    Move $x_i$ to the heads of GL and $DL_j$.
  else
    /*A cache miss has occurred.*/
    if Cache is full then
      /*Replace the LRU block of the $DL_m$ which has the smallest average response time of recent $k$ user I/Os*/
      Delete the LRU block of $DL_m$ to get a free block, and also delete this block from GL.
    else
      /*The RAID-10 works under failure state and there is at least one DNP disk. The cache should be managed in asymmetrical style.*/
      Get a free block.
      Load $x_i$ from the disk and update the response times of recent $k$ user I/Os of $DL_k$.
      Add $x_i$ to the heads of GL and $DL_k$.
end

V. PROTOTYPING OF A-CACHE

In this section, we present our prototype, which is implemented to evaluate the effectiveness and efficiency of A-Cache with representative real-world traces. After a general description of the prototype, we detail its key functional modules.

A. Overview of the A-Cache Prototype

The prototype of A-Cache is a multithread application, which consists of three key functional modules, namely, trace handling, buffer CM, and DM, as illustrated in Fig. 6.

The trace handling module, the front-end of the prototype, translates the trace records into standard I/O requests (rw, offset, and length) and sends these requests to the CM module. The buffer CM module will divide the request into block-aligned pieces. When a read miss or a write-back operation occurs, I/O request(s) is/are generated in the unit of a block and sent to the underlying DM module. The DM module performs real disk I/Os according to these I/O requests; in the meanwhile, it also performs recovery I/Os on the DP disks when there is no user I/O being executed or queued and the recovery process is activated. In what follows we elaborate on these three functional modules.

B. Trace Handling

In our evaluation, we choose four typical workloads, including DA, DT, W2, and F1, from two different environments. The DA and DT traces were collected in production server environments by the Microsoft Corporation in 2008 [42]. The W2 and F1 traces from the SPC-1 benchmark suites were collected in a popular search engine and OLTP applications...
running at two large financial institutions [43], respectively. They are all block-level traces. Table II summarizes the key features of these traces, including the working set size, denoted by WSS, of read and write requests, the total amount of data to be read and written, and the size of the dataset. We assume a block size of 64 kB in our evaluation and analyzes.

The W2 trace is an almost read-only workload. The DA trace is a read-dominated workload with 93.1% block requests in it being reads. The DT trace is a read-write-balanced workload, considering the high penalty of write in RAID-10 systems, with about 34.7% block requests in it being writes. The F1 trace is a write-dominated trace with about 77.2% block requests in it being writes. Although these traces cover a wide range of workloads in terms of their read-write ratio, the cache size, number of disks, and arrival rates of user disk I/Os have significant impacts on the efficiency as verified in Section VI.

To facilitate the evaluation, we merge all requests in each trace within a consecutive address space, since all traces record requests to multiple devices except for the DA trace. By scanning all requests in each trace file offline, we can find out the maximum requested offset, which is used to generate the device capacity that is typically larger than that maximum offset. Then, we assign a new offset for each request (denoted by Offs), by its device number (denoted by Dev), the device capacity (denoted by Cap), and its old offset (denoted by Offo), as illustrated in (1). Since there are many devices (16) in the DT trace and the maximum offset is nearly 300 GiB, we simply ignore the device IDs in our evaluation to avoid generating out-of-bound I/Os.

\[
\text{Offs} = \text{Dev} \times \text{Cap} + \text{Offo}. \quad (1)
\]

To emulate the real-world environments as we analyzed in Section III, we take an open-loop measurement on A-Cache in our prototyping, where all requests in one trace are replayed according to their timestamps as recorded in the original trace file. In addition, we evaluate the efficiency of A-Cache under different disk I/O arrival rates by scaling the timestamps with the same ratio.

### C. Buffer Cache Management

The buffer CM of our prototype is implemented according to the design principles presented in Section IV. In this implementation, the clean blocks and dirty blocks are separately maintained in dedicated caches and managed by the original LRU algorithm and the LRU-based A-Cache, respectively, for performance comparison purposes. The ratio of the clean-block cache size to the dirty-block cache size is fixed and equal to the ratio of the working set size of read requests to that of write requests. A hash table, with sufficient entries to accommodate all blocks in both caches, is deployed to speed up the block lookup process.

When a block read miss occurs, the whole block is requested from the underlying disk array module. Cache destage operations (to write back dirty data) are performed when the number of dirty blocks is above a watermark set at 90% of the capacity of the dirty-block cache. A dirty block is directly written back, i.e., writing the whole block to the underlying disk array, if the block is fully dirty, meaning that all bits in this block are dirty, or some of the bits are dirty and the remaining bits are consistent with the corresponding bits on disks. Otherwise, a read-modify-write operation is performed to write back the block. The destage operations are asynchronous, and the maximum number of concurrent destage operations is limited to the number of disks in the underlying RAID-10 system.

To sufficiently warm up the caches, we run each trace for the first one-third of requests before measuring the performance. After the caches are warmed up, we immediately activate the asymmetric cache space management policy and the recovery process.

### D. Disk Array Management

Except for the enhanced functions mentioned in Section IV-A, such as sending control messages and providing the detailed configuration information to CM, and observing important parameters to support dynamic and adaptive asymmetric cache replacement, the DM module has the following two major functions.

1) **Performing User I/Os**: When the DM module receives a block I/O request from the CM module, it first translates the I/O request to the disk array into I/O request(s) to individual disk(s). Each disk in the disk array has a dedicated thread that performs the user I/Os from the upper-level buffer cache. The user I/Os are performed via libaio [44], a widely used tool for conducting asynchronous I/Os. We choose to perform the user I/Os in the direct I/O model to avoid the requested block being reached by the buffer cache in the operating system. The response time of \( k \) most recent user I/Os is recorded. The average value of these response time is used by the asymmetric eviction scheme in the CM module to determine which cache blocks to evict. For the DNP disks, we employ a simple approach balancing the user read I/Os between a pair of mirrored disks. That is, if the user read I/O can be serviced by either disk A or disk B, a user-I/O requested block with an even block number is sent to disk A; otherwise, it will be sent to disk B. For the DP disks, we simply reduce the portion of the user read I/Os that the spare disk can service, e.g., to one-third of the total number of user read I/Os, and the mirror disk of the faulty disk services the remaining user read I/Os.

2) **Performing Recovery I/Os**: The recovery I/Os on each DP disk are also performed by a dedicated thread. To maintain a low average response time of user I/Os

<table>
<thead>
<tr>
<th>Name</th>
<th>WSS-R (Block)</th>
<th>WSS-W (Block)</th>
<th>#Reads</th>
<th>#Writes</th>
<th>Dataset (GiB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2</td>
<td>189953</td>
<td>6</td>
<td>505535</td>
<td>922</td>
<td>120</td>
</tr>
<tr>
<td>F1</td>
<td>25925</td>
<td>60017</td>
<td>1260851</td>
<td>4293832</td>
<td>24</td>
</tr>
<tr>
<td>DA</td>
<td>69844</td>
<td>3711</td>
<td>2073164</td>
<td>154668</td>
<td>120</td>
</tr>
<tr>
<td>DT</td>
<td>1506153</td>
<td>2596582</td>
<td>15763679</td>
<td>8390818</td>
<td>300</td>
</tr>
</tbody>
</table>
and absorb the available bandwidth for recovery, the recovery I/Os will be scheduled immediately when all previous user I/Os are finished and no user I/O is queuing, which is similar to DOR. A dedicated recovery buffer is deployed to balance the recovery progress between the two DP disks. The recovery thread of the mirror disk fetches blocks (of size of 64 KiB each) one by one from that disk to the recovery buffer. Meanwhile, the recovery thread of the spare disk writes the data in the recovery buffer back to that spare disk at a size of 4 MiB. The recovery read I/Os will not be scheduled when the recovery buffer is full. In our prototype, the size of the recovery buffer is fixed at 10 MiB.

VI. EXPERIMENTS

In this section, we first introduce our prototype-based trace-driven experiments for evaluating A-Cache’s effect on speeding up the recovery and user I/O performances. We then present and analyze the results of the experiments.

A. Evaluation Environment

In our experiments, we employ a Supermicro storage server with two Intel Xeon CPU X5650 @2.67-GHz processors (6 cores per processor), and 16-GB DDR3 main memory. All disks are Western Digital WD10EALS 3.5” 1TB SATA HDDs, which are connected by an Adaptec 31605 RAID controller with a 256-MB cache. We disable the RAID function of the hardware controller and only use the direct I/O mode to connect each disk. The cache and prefetch functions on the controller are also disabled. The operating system is Linux Ubuntu 16.04 LTS X86 with the kernel version 4.15.0.

The underlying disks are configured based on a predefined RAID-10 system specifying the strip size at a fixed 512 kB and the data layout of the near copy model. We consider disks 0 and 1 in the RAID-10 system as the DP disks in our experiments. The starting address of the recovery process is 0.5 TB in a disk, which is half of the capacity of a disk. The number of recent user I/Os to be observed is set to 10. And the remaining parameters are configured as that described in our prototype implementation.

We conduct our measurement on the efficiency of A-Cache along three important dimensions, namely, cache size (number of blocks in cache), number of disks in the RAID-10 system (DP and DNP disks), and arrival rates of user disk I/Os on each disk (by scaling the timestamp proportionally as mentioned in Section V), because they have significant impacts on the efficiency of A-Cache. The coefficient of the accumulated value of the response time of the most recent k (set to 10) user I/Os of the DP disks is set as 1.0, so that the eviction in LRU-based A-Cache takes place when the dedicated linked list (DL, Section 4.2) has the smallest accumulated value.

We report the speedup of recovery speed and average response time of user disk I/Os of the LRU-based A-Cache (denoted by A-LRU) over the original LRU without A-Cache (denoted by LRU). For fairness in comparison, we replay the same number of block requests for LRU and A-LRU during the recovery process at the same request arrival rate.

Before presenting our experimental results, we show in Fig. 7 the ratio of the number of disk I/Os to the number of block requests to cache, referred to as IOR, as a function of cache size, from 1/1000 to 1024/1000 of the working set size of a given workload. Note that IOR is different from miss rate because of the write penalty in RAID-10 (i.e., each block write translates to two disk I/Os). The purpose of this figure is to show how workloads of different characteristics, shown in Table II of Section V-B, behave differently in cache misses. These differing cache-miss behaviors will be used to explain the reasons behind some of the experimental results presented and discussed next.

B. Performance as Function of Cache Size

Fig. 8 shows the experimental results under different cache sizes. We choose only four typical cache sizes for each workload, and the basic unit of the cache sizes is 1/1000 of the working set size of each workload summarized in Table II of Section V-B, e.g., the points 1, 32, 128, and 512 of the W2 workload in Fig. 8 denotes that the cache sizes are 1/1000, 32/1000, 128/1000, and 512/1000 of the working set size of the W2 workload, respectively. The reasons for using those cache sizes are explained as follows.

1) A lower-bound of cache size, i.e., 1/1000 of the working set size, is considered in our evaluation.

2) A upper-bound of cache size, the IOR of which is close to that of caching all blocks in the working set of each workload, i.e., 512/1000, 8/1000, 512/1000, and 512/1000 of the working set sizes of the W2, F1, DA, and DT workloads, respectively, are considered in our evaluation. We do not choose 1024/1000 or 1000/1000 of the working set sizes as the upper-bound because
A-Cache would not induce any benefit in those cases, as shown in Fig. 7.

3) The remaining cache sizes in our evaluation are chosen between the lower-bound and upper-bound. In addition, the IOR should be sensitive to the cache size, e.g., 128/1000 and 256/1000 under the DA workload. For a fair comparison, we fix the other parameters. More specifically, the number of disks in the disk array is fixed at eight. And the arrival rate of user disk I/Os is fixed at around 50 IOPS per disk on average (under LRU), which is about half to one-third of the peak IOPS that a disk can service, noting that different workloads may have different peak IOPS due to their differences in dataset size and read-write ratio in Table II of Section V-B.

The experimental results demonstrate that A-Cache can significantly improve the recovery speed. For example, it can speed up the recovery process by up to 2.87X, 4.91X, 4.63X, and 1.59X under the W2, F1, DA, and DT workloads respectively. Furthermore, except for the F1 workload, A-Cache can simultaneously improve the latency of user disk I/Os to various degrees, e.g., it can speed up the average response time of user I/Os by up to 1.17X under the DA workload as well as the recovery speed by 3.99X.

To further understand the evaluation results, we present the miss rates in cache of DP and NDP disks under the LRU and A-LRU algorithms corresponding to the cases in our evaluation in Fig. 9. Generally speaking, the effectiveness of A-Cache in speeding up the recovery performance is determined by the sensitivity of the miss rate in cache of the DP disks (denoted by $S_{\text{miss}}^{(\text{DP})}$), as derived in (2), to the extra cache received by those DP disks. More specifically, the more sensitive the miss rate is, the higher efficiency can be achieved by A-LRU, and vice versa. For example, for configurations of 1/1000 of working set size for the W2 and DA workloads, A-Cache is almost not able to speed up the recovery, because: 1) the miss rates in cache under LRU of the DP disks are very high, i.e., about 0.64X and 0.92X, respectively and 2) the improvements of the miss rate in cache of the DP disks by using A-LRU are low, i.e., about 0.62X and 0.91X, respectively

$$S_{\text{miss}}^{(\text{DP})} = \frac{R_{\text{LRU}}^{(\text{DP})} - R_{\text{ALRU}}^{(\text{DP})}}{R_{\text{LRU}}^{(\text{DP})}}$$

$R_{\text{LRU}}^{(\text{DP})}$ denotes the miss rate in cache under LRU of the DP disks, and $R_{\text{ALRU}}^{(\text{DP})}$ denotes the miss rate in cache under A-LRU of the DP disks.

In addition, the effectiveness of A-Cache in speeding up the user I/O performance is determined by the sensitivity of the miss rate in cache of the DNP disks (denoted by $S_{\text{miss}}^{(\text{DNP})}$), as derived in (3), to the cache sacrificed by those DNP disks. More specifically, the more sensitive the miss rate is, the lower efficiency can be achieved by A-LRU, and vice versa. For example, under all the configurations of the F1 workload, A-Cache even increases the user IO latency, because: 1) the miss rates in cache under LRU of the DNP disks are relatively low, i.e., less than 0.2 and 2) the increase of miss rates in cache of the DNP disks by using A-LRU is very high, i.e., higher than 0.3X

$$S_{\text{miss}}^{(\text{DNP})} = \frac{R_{\text{ALRU}}^{(\text{DNP})} - R_{\text{LRU}}^{(\text{DNP})}}{R_{\text{LRU}}^{(\text{DNP})}}$$

$R_{\text{LRU}}^{(\text{DNP})}$ denotes the miss rate in cache under LRU of the DNP disks, and $R_{\text{ALRU}}^{(\text{DNP})}$ denotes the miss rate in cache under LRU of the DNP disks.
Fig. 10 shows the normalized average response time on the DP and DNP disk pairs by using LRU and A-LRU. The baseline is the average response time by using the original LRU. And all these measurement results are collected in a RAID-10 with eight disks under the W2 workload with the cache size of 512/1000 of the working set size of the W2 workload. The average response time of user I/Os on the DP disk pairs significantly decreases, but those on the DNP disk pairs increases, by using A-LRU, as we analyzed in Section III-C.

C. Performance as Function of Number of Disks

Fig. 11 shows the experimental results under various numbers of disks from 4 to 12. In addition, and without loss of generality, by referring to Fig. 8, the cache size is fixed at 128/1000, 2/1000, 256/1000, and 128/1000 of the working set size of the W2, F1, DA, and DT workloads, respectively. The arrival rate of user disk I/Os is fixed at around 50 IOPS per disk on average.

The effectiveness of A-Cache in speeding up the recovery and user I/O performances typically increases with the number of disks in the RAID-10 system. This is because the larger the number of disks in the array is, the larger number of the DNP disks there will be. With the same number of the DP disks in our experiments, more DNP disks imply that larger extra buffer space can be reallocated to the DP disk and that fewer user I/Os will be serviced by each DNP disk on average. The detailed miss rates for different cases are presented in Fig. 12. It is worth noting that the miss rate in cache of the DP disk under the A-LRU algorithm is not improved when the number of disks increases from 8 to 12 except the DA workload. As a result, the recovery speedup under the DA workload decreases and demonstrates different trends from the other workloads.

D. Performance as Function of Arrival Rate of User Disks I/Os

Fig. 13 shows the experimental results under different levels of the arrival rate of user disk I/Os (around 25, 50, and 75 IOPS per disk, denoted by L, M, and H, respectively). In addition, the number of disks is fixed at eight, and without loss of generality, by referring to Fig. 8, the cache size is also fixed at 128/1000, 2/1000, 256/1000, and 128/1000 of the working set size of the W2, F1, DA, and DT workloads, respectively. It is worth noticing that the speedup on the recovery speed under a high user I/O arrival rate of the DA workload is up to 9.25X.

We find that the effectiveness of A-Cache in speeding up the recovery performance usually increases with the arrival rate of user I/Os. On one hand, with the similar reduction of miss rates on the DP disks (by using A-Cache), e.g., Fig. 14(a) and (d), A-Cache can save more disk bandwidth consuming under a higher user I/O arrival rate. Therefore, more disk bandwidth can be used for recovery in such scenarios. On the other hand, assume that the bandwidth of a disk is fixed. A higher user I/O arrival rate usually means a lower remaining disk bandwidth (in terms of IOPS) for recovery. Due to the lower baseline (original disk bandwidth for recovery), even the same increase of disk bandwidth for recovery would result in a relatively higher speed up in recovery performance.

As shown in Fig. 14, the miss rates are different for the same workload under different user I/O arrival rates, because different numbers of user requests are conducted. To ensure the fairness of the comparisons, we fix the recovery time for all the configurations in our experiments. As a result, a higher user I/O arrival rate implies that more user requests are conducted, which leads to the change of the reaccess pattern, even under the same workload as shown in Fig. 13.

As we mentioned in Section IV-B, at the beginning of recovery, the buffer cache space for each disk pair is approximately the same. After the activation of A-Cache, the reallocation of buffer space from the DNP disks to the DP disks is actually caused by cache misses that will cause the eviction of old cache blocks. In other words, there is a warm-up period. Typically, a higher arrival rate of user I/Os implies a higher
will quickly lead A-Cache to its best effect. Due to the relatively short measurement time in our experiments, a higher warm-up speed will quickly lead A-Cache to its best effect.

VII. FURTHER DISCUSSION

In this section, we will make some further discussion on the application range of A-Cache. A-Cache can be applied in SSDs. On the other hand, A-Cache can be applied to a wide range of recovery processes of the various RAID systems with both DP and DNP disks, which is discussed as follows.

1) A-Cache Can Be Applied in SSDs: The aforementioned buffer cache replacement algorithm can also be applied in SSDs. Wu et al. [21], [45] proposed buffer cache replacement algorithm, when some data blocks in the cache are selected for replacement, the data blocks belonging to the chips that are ongoing collection garbage (GC) cannot be evicted, only selecting the data blocks in cache belonging to the chips that are not ongoing GC can be evicted. This buffer cache replacement algorithm is asymmetric, and similar to our A-Cache.

2) RAID-10 Systems Under the Recovery Process of Multiple Faulty Disks: The RAID-10 system can tolerate multiple failures as long as no two failures happen to the same pair of mirrored disks. In a RAID-10 system, if there are any DNP disks in the RAID-10 system, the asymmetry discussed in Section III-A and III-B will be present in that RAID-10 system, making A-Cache applicable. In addition, due to the increasing number of the DP disks and decreasing number of the DNP disks, the effects of A-Cache on speeding up the recovery process might be compared to the case of using A-Cache during the single faulty disk condition.

3) RAID Systems That Can Fully or Partially Tolerate Concurrent Double Disk Failures Under the Recovery Process of a Single Faulty Disk: We take the RAID-6 system as an example, which can tolerate concurrent double disk failures. Since any k surviving strips in a RAID-6 stripe with k+2 strips can be used to recover all data in that stripe, one DNP disk can be separated from all surviving disks in a RAID-6 system, if the data for recovery are all fetched from the remaining k surviving disks that are then treated as DP disks. This constitutes the necessary condition for applying A-Cache. The difference between using A-Cache in a RAID-6 system and using it in a RAID-10 system mainly lies in the number of the DP disks relative to that of the DNP disks. For example, there is only one DNP disk in a RAID-6 system, which can make A-Cache less effective than when it is applied to a RAID-10 system where there are multiple DNP disks. Besides the RAID-6 systems, A-Cache can also be used to other RAID systems that can partially tolerate concurrent double disk failures, such as RAID-01, RAID-1E, and de-clustering RAID systems, under the recovery process of a single faulty disk.

VIII. CONCLUSION

To simultaneously speed up the recovery and user I/O performances of RAID-10 systems under the recovery process of a single faulty disk, we propose an asymmetric buffer CM scheme, called A-Cache. The rationale behind A-Cache is to dynamically and adaptively allocate the buffer cache space belonging to the DNP disks to those disks that do (DP disks), to exploit the surplus free bandwidth and the relatively low interference among the exclusively user I/Os on the DNP disks, which in turn mitigates the high interference between the user I/Os and recovery I/Os and mines the low interference exclusively among the recovery I/Os on the DP disks. Results of our prototype-based extensive trace-driven experiments demonstrate that A-Cache can dramatically improve the user I/O performance and recovery speed simultaneously.

ACKNOWLEDGMENT

The authors are grateful to anonymous reviewers for their helpful comments and suggestions. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

REFERENCES


[17] B. Schroeder and G. A. Gibson, “Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?” in Proc. 5th USENIX Conf. File Storage Technol. (FAST), San Jose, CA, USA, Feb. 2007, pp. 1–16.


Hong Jiang (Fellow, IEEE) received the B.Sc. degree in computer engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1982, the M.A.Sc. degree in computer engineering from the University of Toronto, Toronto, ON, Canada, in 1987, and the Ph.D. degree in computer science from the Texas A&M University, College Station, TX, USA, in 1991.

He is currently the Chair and the Wendell H. Nedderman Endowed Professor of Computer Science and Engineering Department, University of Texas at Arlington, Arlington, TX, USA. Prior to joining UTA, he has served as a Program Director with National Science Foundation from 2013 to 2015, and he has been with the University of Nebraska–Lincoln, Lincoln, NE, USA, since 1991, where he was a Willa Cather Professor of Computer Science and Engineering. He has graduated 16 Ph.D. students, who upon their graduations either landed academic tenure-track positions in Ph.D.-granting U.S. institutions or were employed by major U.S. IT corporations. He has also supervised 15 postdoctoral fellows and visiting scholars. He has more than 250 publications in major journals and international conferences in these areas, including the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, PROCEEDINGS OF IEEE, ACM Transactions on Architecture and Code Optimization, ACM Transactions on Storage, Journal of Parallel and Distributed Computing, ISCA, MICRO, USENIX ATC, FAST, EUROSYS, LISA, SIGMETRICS, ICDCS, IPDPS, MIDDLEWARE, OOPSLA, ECOOP, SC, ICS, HPDC, INFOCOM, and ICPP. His research has been supported by NSF, DOD, and industry. His present research interests include computer architecture, computer storage systems and parallel I/O, high-performance computing, big data computing, cloud computing, and performance evaluation.

Dr. Jiang recently served as an Associate Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He is a member of the ACM.

Qiang Cao (Senior Member, IEEE) received the B.S. degree in applied physics from Nanjing University, Nanjing, China, in 1997, the M.S. degree in computer technology and the Ph.D. degree in computer architecture from the Huazhong University of Science and Technology, Wuhan, China, in 2000 and 2003, respectively.

He is currently a Full Professor with the Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology. His research interests include computer architecture, large scale storage systems, and performance evaluation. Prof. Cao is a Senior Member of China Computer Federation and a member of the ACM.

Shenggang Wan received the B.S. degree in computer science and technology, the M.S. degree in software engineering, and the Ph.D. degree in computer science and technology from the Huazhong University of Science and Technology, Wuhan, China, in 2003, 2005, and 2010, respectively.

His research interests include dependable storage systems and coding theory.

Changsheng Xie (Member, IEEE) received the B.S. and M.S. degrees in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1982 and 1988, respectively.

He is currently a Professor with the Department of Computer Engineering, HUST. He is the Deputy Director of the Wuhan National Laboratory for Optoelectronics, Wuhan. His research interests include computer architecture, disk I/O system, networked data storage system, and digital media technology.

Prof. Xie is the Vice Chair of the Expert Committee of Storage Networking Industry Association, China. He is a member of the IEEE Computer Society.