Proactive Data Migration for Improved Storage Availability in Large-Scale Data Centers

Suzhen Wu, *Member, IEEE*, Hong Jiang, *Senior Member, IEEE*, and Bo Mao, *Member, IEEE*

Abstract—In face of high partial and complete disk failure rates and untimely system crashes, the executions of low-priority background tasks become increasingly frequent in large-scale data centers. However, the existing algorithms are all reactive optimizations and only exploit the temporal locality of workloads to reduce the user I/O requests during the low-priority background tasks. To address the problem, this paper proposes *Intelligent Data Outsourcing* (IDO), a zone-based and proactive data migration optimization, to significantly improve the efficiency of the low-priority background tasks. The main idea of IDO is to proactively identify the hot data zones of RAID-structured storage systems in the normal operational state. By leveraging the prediction tools to identify the upcoming events, IDO proactively migrates the data blocks belonging to the hot data zones on the degraded device to a surrogate RAID set in the large-scale data centers. Upon a disk failure or crash reboot, most user I/O requests addressed to the degraded RAID set can be serviced directly by the surrogate RAID set rather than the much slower degraded RAID set. Consequently, the performance of the background tasks and user I/O performance during the background tasks are improved simultaneously. Our lightweight prototype implementation of IDO and extensive trace-driven experiments on two case studies demonstrate that, compared with the existing state-of-the-art approaches, IDO effectively improves the performance of the low-priority background tasks. Moreover, IDO is portable and can be easily incorporated into any existing algorithms for RAID-structured storage systems.

Index Terms—Low-priority background tasks, availability, proactive, temporal and spatial locality, RAID reconstruction

1 Introduction

Storage is a critical component in the data centers that continues to grow in size to accommodate the exponentially increasing volume of data. As the big data becomes the business technology of modern enterprise, the unavailability of data for even a short duration is unacceptable [8], [35]. In large-scale data centers, due to the increasing needs for system maintenance, such as replacing failed components, enhancing system performance, and expanding data capacity, data servers and storage subsystems routinely experience system upgrades.

In general, two categories of tasks exist in storage systems: high-priority foreground tasks and low-priority background tasks. Foreground tasks, initiated by the system and user applications, such as the online transaction processing (OLTP), require providing 24×4 online services. On the other hand, background tasks, invoked by repair or maintenance activities due to disk failures or crash reboot, include RAID reconstruction, RAID re-synchronization, and disk scrubbing. The purpose of these background tasks is to improve the storage availability in large-scale data centers [4].

- *S. Wu is with the Computer Science Department, Xiamen University, Xiamen, Fujian 361005, China, and with the State Key Laboratory of High-end Server & Storage Technology, Jinan, Shandong, China. E-mail: suzhen@xmu.edu.cn.*
- *H. Jiang is with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588-0150. E-mail: jiang@cse.unl.edu.*
- *B. Mao is with the Software School, Xiamen University, Xiamen, Fujian 361005, China. E-mail: maobo@xmu.edu.cn.*

Manuscript received 13 June 2013; revised 23 July 2014; accepted 1 Oct. 2014. Date of publication 3 Nov. 2014; date of current version 12 Aug. 2015. Recommended for acceptance by C.-Z. Xu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2366734

The RAID technology [25] has been widely deployed in large-scale data centers owing to its high availability. For the purpose of data integrity and reliability, RAID can recover the lost data in case of disk failures, a process also known as *RAID reconstruction* and one of the aforementioned typical background tasks in storage systems. With the growing number and capacity of disks in data centers, the slow performance improvement of the disks and the increasing disk failure rate in such environments [27], [30], execution of the RAID reconstruction task, along with other background tasks of storage systems, is poised to become the norm rather than the exception in large-scale data centers [2], [6], [9]. While periodic disk scrubbing is becoming a necessity as a measure of reliability maintenance [3], re-synchronization has also been increasingly frequently used to counter the unexpected system crashes and high sector-error rates. If another disk failure or a latent sector error [3] occurs when the background tasks are being performed, data will be lost, which is unacceptable for end users. Therefore, the performance of on-line background tasks is of great importance to the storage availability in large-scale data centers.

A number of optimizations have been proposed to optimize the low-priority background tasks, such as RAID reconstruction [12], [28], [34], [38], [41], RAID re-synchronization [7], [42], disk scrubbing [15], [24], [31] and RAID reshape [21]. However, to the best of our knowledge, all of them are failure-induced or *reactive* optimizations and thus are passive. In other words, they are triggered *after* a disk failure has been detected and focus on either improving the workflow of the background tasks [34], [38], [46] or alleviating the user I/O intensity when performing the background tasks [36], [41], [43] but *not both*. In fact, our workload

analysis reveals that the reactive optimization is far from being adequate (see Section 2.2.1 for details).

Moreover, from extensive evaluations and analysis, our previous studies and researches by others have found that user I/O intensity during background tasks has a significant impact on the performance of the on-line background tasks because there are mutually adversary impacts between background I/O requests and user I/O requests [34], [36], [41]. This is why the time spent on the on-line background tasks is much longer than that on its off-line counterpart [11]. However, existing on-line background-task optimization approaches, such as WorkOut and VDF, only exploit the temporal locality of workloads to reduce the user I/O requests, which results in very poor performance under workloads with poor temporal locality, as clearly evidenced in the results under the Microsoft Project trace that lacks temporal locality (see Section 4 for details). Therefore, we strongly believe that both the temporal locality and spatial locality of user I/O requests must be simultaneously exploited to further improve the performance of the on-line background tasks.

On the other hand, most low-priority background tasks are predictable, such as RAID reconstruction, RAID re-synchronization and RAID reshape. For example, the disk failure events in large scale data centers are shown to be predictable [10], [19], [48]. Researchers in Microsoft designed a disk failure detector that is able to predict the disk failure events by an accuracy of 88.2 percent in an evaluation on a population of 1,190 production disks [10]. The disk scrubbing processes are scheduled periodically or initiated in the night time. When the free space of the system is lower than a threshold, the RAID system will initiate a RAID reshape event. Thus, the low-priority background tasks, such as the RAID reconstruction, RAID re-synchronization, disk scrubbing and RAID shape, are all predictable prior to the actual occurrences of such events. However, the predictable characteristics of the low-priority background tasks are not well exploited by the existing optimizations.

Motivated by the above observations, in this paper we propose a novel optimization scheme, called Intelligent Proactive Data Outsourcing (IDO), to significantly improve the availability of the low-priority background tasks in large-scale data centers by proactively exploiting data access patterns to outsource data judiciously. IDO divides the entire storage space into zones and identifies the popularity of these zones in the normal operational state, in anticipation for background tasks and migration. By leveraging the prediction tools to identify the upcoming events, IDO proactively migrates the data blocks on the degraded device belonging to the hot data zones to a surrogate RAID set in the large-scale data centers. Upon a background task, most user I/O requests can be serviced directly by the surrogate RAID set instead of the much slower degraded RAID set that is performing the background tasks. By simultaneously optimizing the workflow of the background task and alleviating the user I/O intensity, the performance of the background task in the degraded RAID set is accelerated and the user I/O requests are serviced more effectively, thus significantly improving the availability of storage systems.

Although the data migration technique has been well studied for improving performance [1], [14], [16] and energy efficiency [26], [37] of storage systems, IDO adopts this technique in a unique way to significantly optimize the increasingly critical background tasks in large-scale data centers. In this paper, we implement the IDO prototype by embedding it into the Linux software RAID5/6 module on two case studies to measure IDO's performance and effectiveness. The extensive trace-driven evaluations show that IDO outperforms WorkOut [41] by a factor of up to 2.7 with an average of 2.1 in terms of the reconstruction time, and by a factor of up to 2.1 with an average of 1.5 in terms of the average user response time. IDO outperforms VDF [36] by a factor of up to 4.2 with an average of 3.1 in terms of the reconstruction time, and by a factor of up to 3.8 with an average of 2.5 in terms of the average user response time. The trace-driven evaluations also show that IDO outperforms WorkOut by a factor of up to 2.8 with an average of 2.2 in terms of the re-synchronization time, and by a factor of up to 2.0 with an average of 1.6 in terms of the average user response time.

More specifically, IDO has the following salient features:

- IDO is a *proactive* optimization that dynamically captures the data popularity and migrates the hot data blocks in a RAID system in the normal operational state.
- IDO exploits both *the temporal locality and spatial locality* of workloads on all disks to improve the performance of the background tasks.
- IDO optimizes the workflow of the background task and alleviates the user I/O intensity simultaneously to improve the performance of the background tasks.
- The results demonstrate that IDO is independent of the existing RAID tasks, thus it can also effectively improve the performance of other background tasks, such as re-synchronization, RAID reshape and disk scrubbing.

The rest of this paper is organized as follows. Background and motivation are presented in Section 2. We describe the design of IDO in Section 3. Methodology and performance results of a prototype evaluation of IDO are presented in Section 4. Related work is presented in Section 5 and the main contributions of this paper are summarized in Section 6.

2 Background and Motivation

In this section, we first describe the low-priority background tasks in storage systems. Then we show how the existing schemes are not efficient and motivate our work.

2.1 The Low-Priority Background Tasks

In large-scale data centers, the low-priority background tasks become the frequent events, such as RAID reconstruction, RAID re-synchronize, disk scrubbing, and RAID reshape. RAID reconstruction recovers the failed data to the replacement disk when a disk fails. RAID re-synchronization scans the entire RAID to find and repair the inconsistent data when an untimely system crash occurs. Disk scrubbing periodically accesses disks to detect irremediable read errors in infrequently accessed sectors.

To accommodate the explosive growth in data volume, the storage systems in large-scale data centers today

routinely are routinely constructed of hundreds of thousands of disks. Recent studies have revealed that partial or complete disk failures in large-scale data centers indicate that disk failures happen at a higher rate than expected [3], [27], [30]. Schroeder and Gibson [30] found that annual disk replacement rates in the real world exceed 1 percent, with 2-4 percent on average and up to 13 percent in some systems, much higher than 0.88 percent, the annual failure rate (AFR) specified by the manufacturer's datasheet. Bairavasundaram et al. [3] observed that the probability of latent sector errors that can lead to disk replacement is 3.45 percent in their study. These studies also show a significant amount of correlation in drive failures, indicating that, after one disk fails, another disk failure will likely occur soon [3], [27], [30]. Gibson [9] also points out that the probability of a second disk failure in a RAID system during reconstruction increases along with the reconstruction time: approximately 0.5 percent for 1 hour, 1.0 percent for 3 hours and 1.4 percent for 6 hours. A subsequent disk failure (or a series of subsequent disk failures) in the degraded RAID will cause persistent data loss for most RAID levels.

Moreover, in software RAID, unexpected system crash or software errors can make stripes in inconsistent states that can also introduce a window of vulnerability [7]. In this case, current software RAID systems always employ the re-synchronization and disk scrubbing process to find and correct the inconsistent data by scanning the whole RAID. However, these processes can take hours or even days for large-scale RAID-structured storage systems.

Due to the increasing needs for system maintenance, such as replacing failed components, enhancing system performance, and expanding data capacity, the low-priority background tasks become the common events in large-scale data centers [6], [9], [42]. Frequent or long-term downtime and data loss are obviously intolerable to the end users. Wenk showed that 50 percent of companies losing critical systems for more than 10 days never recover, 43 percent of companies experiencing a disaster never reopen, and 29 percent of the remaining companies close within two years [39]. Thus, building available storage systems in large-scale data centers is critically important.

2.2 Why Are the Existing Schemes Inefficient?

Due to the importance of low-priority background tasks on the availability of large-scale storage systems, a number of optimizations have been proposed to optimize the low-priority background tasks. However, with the predictable characteristics of low-priority background tasks and the deployment of the flash-based SSDs in large-scale data centers, the existing schemes are becoming less efficient, as elaborated next.

2.2.1 Reactive versus Proactive

Recent studies have reveal that the disk failure events could be detected by more than 88 percent [10], [19], [48] which means that the RAID reconstruction process could be predictable. Moreover, the other low-priority background tasks, such as the RAID re-synchronization, disk scrubbing and RAID shape, are all predictable prior to the actual occurrences of such events. The predictable characteristics of the low-priority background tasks could be leveraged in the optimizations.

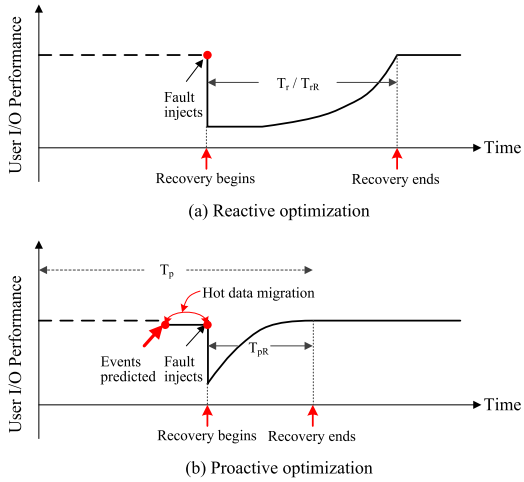


Fig. 1. Comparisons of user I/O performance and reconstruction time between (a) reactive optimization and (b) proactive optimization on RAID reconstruction. Note that T_r and T_p indicate the period of hot data identification, T_{rR} and T_{pR} denote the reconstruction time, and $T_r = T_{rR}$. In general, $T_r \ll T_p$ and $T_{rR} > T_{pR}$.

However, the existing optimizations only initiate the low-background priority tasks after a disk fails or crash reboot, which we refer to as *reactive* optimizations, and thus are passive. For example, PRO [34] and WorkOut [41] identify the popular data during on-line RAID reconstruction, which may result in insufficient identification and exploitation of the popular data. Compared with the normal mode, the period of the low-background priority tasks is too short for the optimizations to identify sufficient amount of popular data, which will be clearly evidenced by our experimental results in Section 2.3.

By monitoring the user I/O requests in the normal mode, the popular data zones can be proactively identified and migrated before and in anticipation of the low-priority background tasks. Once the low-priority background tasks are detected to happen soon, the optimization works immediately and efficiently by leveraging the data popularity information that is already identified. We call this process a *proactive* optimization, to contrast to its *reactive* counterpart. Fig. 1 shows a case study on RAID reconstruction, where to compare the user I/O performance between a reactive optimization and a proactive optimization. The performance is degraded by the RAID reconstruction and returns to its normal level after the recovery process completes. We can see that the proactive approach takes effect much faster than the reactive approach, shortening the reconstruction time. The reason is that the proactive approach with its popular data already accurately identified and migrated prior to the onset of the disk failure, can service the user requests effectively without the substantially extra amount of time required by the reactive approach to identify and migrate the popular data blocks.

In large-scale data centers consisting of hundreds of thousands of disks, the proactive optimization is of great importance because the low-priority background tasks are becoming the norm rather than the exception, for which the low-priority background tasks are thus becoming a normal operation [6], [9], [42].

2.2.2 Temporal Locality versus Spatial Locality

Access locality in storage systems is reflected by the phenomenon of the same storage locations or closely nearby storage locations being frequently and repeatedly accessed. There are two dimensions of access locality. *Temporal locality*, on the time dimension, refers to the repeated accesses to specific data blocks within relatively small time durations. *Spatial locality*, on the space dimension, refers to the clustered accesses to data objects within small regions of storage locations within a short timeframe. These two access localities are the basic design motivations for storage-system optimizations.

The existing optimizations on the low-priority background tasks, such as VDF [36] and WorkOut [42], use *request-based* optimization that only exploits the temporal locality of workloads, but not the spatial locality to reduce user I/O requests. For example, PRO [34] and VDF [36] only focus on optimizing (i.e., tracking or reducing) the user I/O requests that the failed disk receives, thus they ignore the spatial locality and the impact of the user I/O requests on the surviving disks that also have notable performance impact on RAID reconstruction. Moreover, given the wide deployment of large-capacity DRAMs and flash-based SSDs as cache/buffer devices above HDDs to exploit the temporal locality, the visible temporal locality at the HDD-based storage level is arguably very low because of the filtering of the upper-level caches, while the visible spatial locality remains relatively high. The high cost of DRAMs and SSDs relative to that of HDDs makes good design sense for new system optimizations to put the large and sequential data blocks on HDDs for their high sequential performance, but cache the random and hot small data blocks in DRAMs and SSDs for their high random performance [5], [29]. As a result, these new system optimizations likely render the existing temporal-locality-only optimizations ineffective.

To reduce the user I/O requests during the low-priority background tasks by capturing both the temporal locality and the spatial locality, we argue that *zone-based*, rather than *request-based*, data popularity identification scheme and data migration scheme should be used. For example, by migrating the "hot" and popular data zones to a surrogate RAID set immediately after a disk fails, the subsequent user I/O requests can be serviced by the surrogate set during reconstruction as many as possible. It improves the system performance by fully exploiting the spatial locality of the workload. In the meantime, the reconstruction process should firstly rebuild the hot zones, rather than sequentially from the beginning to the end of the failed disk, to take the user I/O requests into consideration. By reconstructing the hot zones first, the data-migration overhead is reduced and most of the subsequent user I/O requests can be serviced by the surrogate set during reconstruction. Consequently, the reconstruction workflow and the user I/O requests are simultaneously optimized by taking the full advantages of both the temporal locality and spatial locality of workloads.

2.3 IDO Motivation

User I/O intensity directly affects the performance of the low-priority background tasks [41]. Fig. 2 shows a case study on RAID reconstruction where we plot the amount of user I/O traffic removed by a reactive request-based optimization (Reactive-request), a reactive zone-based optimization (Reactive-zone), a proactive request-based optimization (Proactive-request) and a proactive zone-based optimization (Proactive-zone), under the three representative traces, WebSearch2.spc, Financial2.spc and Microsoft Project. Each trace is divided into two disjoint parts, one runs in the normal mode and the other runs in the reconstruction mode. The reactive optimization, either request-based or zone-based, exploits the locality of user I/O requests in the reconstruction mode and migrates the popular requests or hot data zones to a surrogate set to allow the subsequent repeated read requests to be serviced by the surrogate set. The proactive scheme, either request-based or zone-based, exploits the locality of user I/O requests by identifying the popular requests or hot zones in the normal mode and migrates the hot requests or hot data zones to a surrogate set immediately before a disk fails, allowing any subsequent read requests that hit the migrated hot requests or data zones to be serviced by the surrogate set during the period of low-priority background tasks.

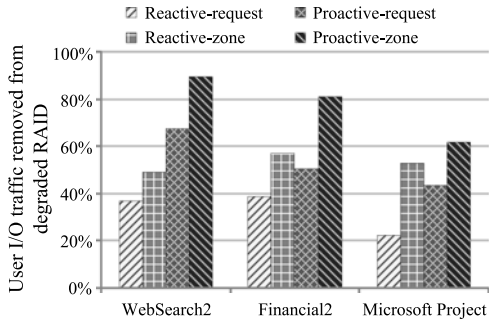


Fig. 2. A comparison of the user I/O traffic removed from the degraded RAID set by a reactive request-based optimization (Reactive-request), a reactive zone-based optimization (Reactive-zone), a proactive request-based optimization (Proactive-request) and a proactive zone-based optimization (Proactive-zone), driven by three representative traces.

We can see that the reactive-request scheme only exploits the data's temporal locality in the reconstruction mode, thus failing to remove a significant amount of user I/O requests from the degraded RAID set, as shown in Fig. 2. For traces with high spatial locality, such as the Microsoft Project trace, the reactive-zone scheme works better than the reactive-request scheme by removing an additional 30.5 percent of user I/O requests from the degraded RAID set. For traces with high locality, be it temporal locality or spatial locality, the proactive approach removes much more user I/O requests than the reactive approach. For example, the proactive-request scheme removes 41.4 and 21.2 percent more user I/O requests than the reactive-request scheme for the WebSearch2.spc and Microsoft Project traces, respectively. By combining the advantages of the proactive and zone approaches, the proactive-zone scheme removes the highest amount of the user I/O requests from the degraded RAID set, with up to 89.8, 81.2, and 61.9 percent of the user I/O requests being removed during reconstruction for the three traces, respectively.

From the case study on RAID reconstruction, it is clear that the proactive optimization is much more efficient and effective than its reactive counterparts. Moreover, exploiting

TABLE 1
Comparison of IDO with the Related Schemes

Characteristics	PRO [34]	WorkOut [41]	VDF [36]	IDO
Proactive				✓
Temporal Locality	✓	✓	✓	✓
Spatial Locality	✓			✓
User I/O		✓	✓	✓
background I/O	✓			✓

both the temporal locality and spatial locality (i.e., zone-based) is noticeably more effective than only exploiting the temporal locality (i.e., request-based), especially for the HDD-based RAIDs in the new HDD/SSD hybrid storage systems. If the hot data zones have been identified in the normal mode (i.e., when no disk fails) and the data in these hot zones is migrated to a surrogate set before the reconstruction event, both the on-line RAID-reconstruction performance and user I/O performance can be significantly improved.

We compare IDO with the state-of-the-art optimizations PRO, WorkOut and VDF based on several important RAID properties, as shown in Table 1. WorkOut [41] and VDF [36] only exploit the temporal locality of workloads to reduce the user I/O requests during reconstruction but ignore the spatial locality. PRO [34] and VDF [36] only focus on optimizing (i.e., tracking or reducing) the user I/O requests to the failed disk but ignore the impact of the user I/O requests to the surviving disks that also have notable performance impact on RAID reconstruction. In contrast, IDO tracks all the user I/O requests addressed to the degraded RAID set in the normal mode to obtain the data popularity information. Moreover, IDO exploits both the temporal locality and spatial locality of user I/O requests to simultaneously optimize the workflow of background tasks and alleviate the user I/O intensity to the degraded RAID set. Thus, the performance of both the low-priority background tasks and foreground tasks are simultaneously improved.

3 Intelligent Data Outsourcing

In this section, we first outline the main design objectives of IDO. Then we present its architecture overview and key data structures, followed by descriptions of the hot data identification and the proactive data migration processes. The data consistency issue in IDO is discussed at the end of this section.

3.1 The Design of IDO

The design of IDO aims to achieve the following three objectives.

- *Improving the storage availability*. By removing most of user I/O requests from the degraded RAID set, the low-priority background tasks can be significantly accelerated, thus shortening the window of vulnerability and reducing the probability of data loss.
- *Improving the user I/O performance*. By proactively migrating the data belonging to the hot zones to a surrogate RAID set, most subsequent user I/O requests can be serviced by the surrogate RAID set without being affected by the low-priority background tasks.
- *Providing high portability*. Since the low-priority background tasks are typically driven by many different algorithms or scheduling methods, it is desirable for IDO to be easily incorporated into the latter for their optimizations.

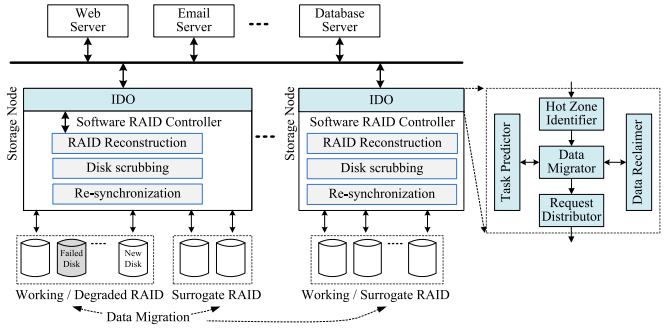


Fig. 3. An architecture overview of IDO in a data center.

3.2 Overview of the IDO Architecture

Fig. 3 shows a system architecture overview of our proposed IDO in the context of the data centers. IDO operates beneath the applications and above the RAID systems in the storage node of a large-scale data center consisting of hundreds or thousands of RAID sets. There are two types of RAID sets, *working RAID sets* and *surrogate RAID sets*. A working RAID set, upon a disk failure or crash reboot, becomes a *degraded RAID set* and is paired with a surrogate RAID set during background task. A surrogate RAID set can be a dedicated RAID set that is shared by multiple RAID sets, or a RAID set that is capacity-shared with a lightly-loaded working RAID set. The dedicated surrogate RAID set improves the system performance but introduces extra device overhead, while the capacity-shared surrogate RAID set does not introduce extra device overhead but affects the performance of its own user applications. However, both are feasible and available for system administrators to choose from based on their characteristics and the system requirements. Moreover, a surrogate RAID set can be in the local storage node or a remote storage node connected by a network.

As shown in Fig. 3, IDO consists of five key functional modules: Hot Zone Identifier, Task Predictor, Request Distributor, Data Migrator and Data Reclaimer. *Hot Zone Identifier* is responsible for identifying the hot data zones of the RAID system based on the user I/O requests. *Task Predictor* is responsible for predicting the upcoming low-priority background tasks. *Request Distributor* is responsible for directing the user I/O requests to the appropriate RAID set during background tasks, i.e., the degraded RAID set or the surrogate RAID set. *Data Migrator* is responsible for migrating the data belonging to the hot zones from the degraded RAID set to the surrogate RAID set, while *Data Reclaimer* is responsible for reclaiming all the redirected write data to the newly recovered RAID set, i.e., the previously degraded RAID set that has completed the background task. The detailed descriptions of these functional modules are presented in the following subsections.

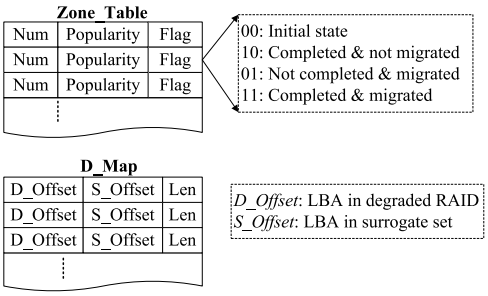


Fig. 4. Main data structures of IDO.

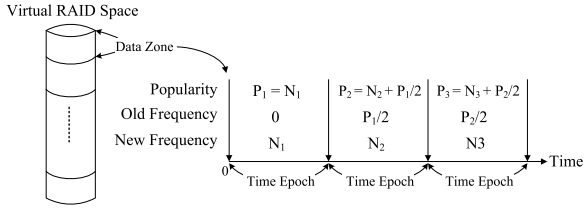


Fig. 5. Hot data identification scheme in IDO. Note that the frequency of N_* is defined as the number of user I/O requests issued to the data zone in a time epoch.

In order to support portability, IDO is designed as an independent module in the existing RAID system and interacts with the modules of background tasks, such as RAID reconstruction and re-synchronization. In the normal mode, only the Hot Zone Identifier module and the Task Predictor module are active to track the popularity of each data zone and predict the upcoming tasks. The other three modules remain inactive until the Task Predictor detects the task will happen and they are deactivated when the reclaim process completes. The reclaim thread is triggered by the background task module when the background task completes.

3.3 Key Data Structures

To identify the hot data zones and record the redirected write data, IDO relies on two key data structures, namely, *Zone_Table* and *D_Map*, as shown in Fig. 4. The Zone_Table contains the popularity information of all data zones, represented by three variables: *Num*, *Popularity* and *Flag*. *Num* indicates the sequence number of the data zone. Based on the Num value and the size of a data zone, IDO can calculate the start offset and end offset of the data zone to determine the target data zone for the incoming read request. *Popularity* indicates the popularity of the data zones. Its value is incremented when a read request hits the corresponding data zone. *Flag* indicates whether the corresponding data zone has been reconstructed and migrated. It is initialized to "00" and the different values represent different states of the corresponding data zones, as shown in Fig. 4.

Based on the Zone_Table, the user read requests addressed to the data zones that are already migrated are redirected to the surrogate RAID set to reduce the user I/O traffic of the degraded RAID set. Thus, more disk resources of the degraded RAID set can be allocated for the background task. However, if the requested data blocks have not been migrated, the read requests are issued to the degraded RAID set as usual to ensure access integrity.

IDO redirects all user write requests to the surrogate RAID set to further alleviate the I/O intensity of the degraded RAID set. The D_Map records the information of all the redirected write data, including the following three variables. *D_Offset* and *S_Offset* indicate the offsets of the redirected write data on the degraded RAID set and the surrogate RAID set, respectively. *Len* indicates the length of the redirected write data. Similar to WorkOut [41], the redirected write data is sequentially stored on the surrogate RAID set to guarantee the write performance. Moreover, the redirected write data is only temporarily stored on the surrogate RAID set and thus should be reclaimed to the newly recovered RAID set after the background task completes.

3.4 Hot Data Identification

Identifying the data popularity is very important for IDO to exploit both the temporal locality and the spatial locality of user I/O requests. We designed a dynamic hot data identification scheme by implementing the Hot Zone Identifier module, as shown in Fig. 5. Three design issues are considered in the hot data identification scheme, elaborated as follows.

First, the entire RAID device space is split into multiple equal-size data zones of multiple-stripe size each to make the accesses aligned. For example, in a four-disk RAID5 set with a 64KB chunk size (i.e., a stripe size of $3\times$ 64 KB = 192 KB), the size of a data zone should be multiple times of 192 KB. Therefore, a data zone is stripe aligned and can be fetched together to reconstruct the lost data blocks. Moreover, since the tracking and migration unit of data is a data zone, IDO can capture the *spatial locality of workloads* by migrating hot data blocks prior to the arrival of user I/O requests for those blocks. A detailed evaluation based on the selection of the different sizes of a data zone is presented in Section 4.2.2.

Second, the hot data zones in IDO are aged by decreasing their popularity values with time to effectively and accurately exploit the *temporal locality of requests*. More specifically, the popularity of a data zone in the current time epoch is calculated by first halving its popularity value from the previous time epoch before adding any values to it as more requests hit the zone in the current epoch. For example, as shown in Fig. 5, P_2 is equal to N_2 plus half of P_1 that is the popularity of the same data zone in its former time epoch. When a time epoch ends, the popularities of all data zones are halved in value.

Third, IDO updates the Zone_Table in the main memory without any disk I/O operations to avoid the impact of the hot data identification scheme on the system performance in the normal mode. When a read request arrives, IDO first checks its offset to locate which data zone it belongs to. Then, IDO increases the corresponding *Popularity* value in the Zone_Table in the main memory. Since the delay of the memory processing is much smaller than that of the disk I/O operation, the identification process in IDO has little performance impact on the overall system performance. Moreover, with the increasing processing power embedded in the storage controller, some storage systems already implement intelligent modules to identify the data popularity. Consequently, IDO can also simply utilize these functionalities to identify the hot data zones.

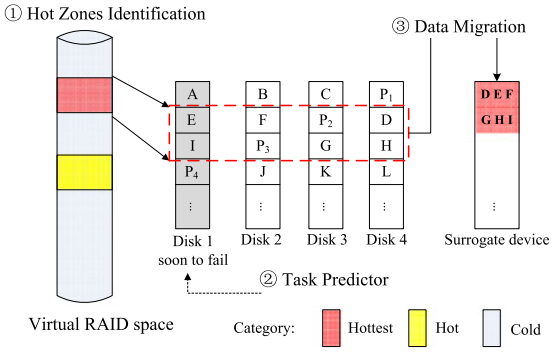


Fig. 6. The data migration process in IDO case study for RAID reconstruction.

3.5 Proactive Data Migration

Once the Task Predictor module detects that a background task will happen soon, the data migration process will be initiated to migrate the hot data zones to the surrogate RAID set. How to predict the background tasks is out of the scope of the paper. Fig. 6 shows a case study on RAID reconstruction. IDO first identifies the soon-to-fail disk, such as Disk 1 in the Fig. 6, by the Task Predictor module. Then the Data Migrator module begins to migrate all the data in these proactively identified hot zones to the surrogate RAID set. Because the data is sequentially written on the surrogate RAID set, the overhead of the data migration, i.e., writing the hot data to the surrogate RAID set, is minimal. When the data in a hot data zone has been migrated to the surrogate RAID set, the corresponding *Flag* in the Zone_Table is set to "01". In order to reduce the space overhead of the surrogate RAID set, IDO does not migrate the data in the cold data zones to the surrogate RAID set. Upon a disk fails, the RAID reconstruction process is triggered with a replacement disk in the degraded RAID set. IDO first reconstructs data on the failed disk in the hot data zones by reading the data from the surrogate RAID set. After all the hot data zones have been reconstructed, IDO continues to reconstruct the remaining data zones. After a cold data zone has been reconstructed, the corresponding *Flag* in the Zone_Table is set to "10".

The reclaim process for the redirected write data is initiated when the RAID reconstruction process completes. In the IDO design, the redirected write data on the surrogate RAID set is protected by a redundancy scheme, such as RAID1 or RAID5/6. The priority of the reclaim process is set to be lower than that of user I/O requests, which will not affect the reliability of the RAID system [41]. Therefore, the reclaim process for the redirected write data can also be scheduled in the system idle period. When a redirected write data block is reclaimed, its corresponding item in the D_Map is deleted. After all items in the D_Map are deleted, the reclaim process completes.

All incoming user I/O requests are carefully checked during the on-line RAID reconstruction period. Upon the arrival of a read request, IDO first determines its target data zone according to the Zone_Table and checks the second bit of the corresponding *Flag* to determine whether the data zone has been migrated to the surrogate set or not ("1" indicates that the data zone has been migrated, while "0" indicates the opposite). If the data zone has not been migrated, the read request is issued to the degraded RAID set and its *Popularity* is updated. Otherwise, the read request is issued to the surrogate RAID set. In order to obtain the accurate location of the read request on the surrogate RAID set, IDO checks the D_Map to determine whether the read request hits the previously redirected write data. If so, the read request is issued to the surrogate RAID set according to the *S_Offset* in the D_Map. Otherwise, the read request is issued to the surrogate RAID set according to the Zone_Table.

IDO checks whether the write request hits the D_Map when receiving a write request. If the write request hits the D_Map, the corresponding item in the D_Map is updated. Otherwise, a new item for the write request is added to the D_Map. To improve the write performance, the redirected write data is sequentially written on the surrogate RAID set.

3.6 Data Consistency

Data consistency is critical in the design of new storage systems. Two aspects are carefully considered in IDO: 1) the key data structures must be safely stored, and 2) the redirected write data must be reliably stored on the surrogate set until the data reclaim process completes.

First, to prevent the loss of the key data structures in the event of a power supply failure or a system crash, IDO stores them in a non-volatile RAM (NVRAM) or a non-volatile persistent storage device such as PCM or Flash-based SSD with a certain write-performance penalty. Since the size of Zone_Table and D_Map is generally very small, it will not incur significant extra hardware cost. Moreover, in order to improve the write performance by using the write-back technique, the NVRAM is commonly deployed in the storage controllers. Thus, it is easy and reasonable to use the NVRAM to store the key data structures.

Second, the redirected write data must be safely stored on the surrogate set. To prevent data loss caused by a disk failure on the surrogate set, the surrogate set must be configured by a redundancy scheme, such as mirroring-based (RAID1) or parity-based (RAID5/6) disk arrays, a basic requirement for the surrogate RAID set. Our previous study [41] provides a detailed analysis on how to choose a surrogate set based on the requirements and characteristics of the applications. Moreover, since the up-to-date data for a read request can be stored on either the degraded RAID set or the surrogate set, each read request is first checked in the D_Map to determine whether it should be serviced by the degraded RAID set, the surrogate set or both (when the data is partially modified) to keep the fetched data always up-to-date, until all the redirected write data on the surrogate set has been reclaimed.

4 Performance Evaluation

In this section, we present the performance evaluation of the IDO prototype through extensive trace-driven experiments.

4.1 Experimental Setup and Methodology

An IDO prototype has been implemented and embedded into the Linux software RAID (MD) as a built-in module. IDO tracks the user I/O requests in the *make_request* function to identify the data popularity in the normal mode. When low-priority background tasks are detected to be

TABLE 2
The Key Evaluation Workload Parameters

Trace	Trace Characteristic		
	Read Ratio	IOPS	Ave. Req. Size(KB)
Fin1	32.8 percent	69	6.2
Fin2	82.4 percent	125	2.2
Web2	100 percent	113	15.1
Proj	97.6 percent	29	57.8

happen, the hot data zones are first migrated to the surrogate set. When a disk fails or crash reboot, the background tasks is initiated by the *md_do_sync* function. During the background task, the incoming read requests are checked in the *make_request* function to determine by which device the requests should be serviced, so as to avoid the degraded RAID set whenever possible. All write requests are issued to the surrogate set and marked as dirty for reclaim after the background tasks complete.

The performance evaluation of IDO was conducted on a server-class hardware platform with an Intel Xeon X3440 processor and 8 GB DDR memory. The HDDs are WDC WD1600AAJS SATA disks that were used to configure both the active RAID set and the surrogate RAID set. While the active set was configured as a RAID5/6 organization, the surrogate set was configured as a RAID1 organization with two HDDs. Further, the surrogate set can be located either in the same storage node as the degraded RAID set or in a remote storage node in the data center. The rotational speed of these disks is 7,200 RPM, with a sustained transfer rate of 60 MB/s that is specified in the manufacture's datasheet. We used 10 GB of the capacity of each disk for the experiments. A separate disk was used to house the operating system (Linux kernel version 2.6.35) and other software (MD and mdadm). In our prototype implementation, the main memory was used to substitute a battery-backed RAM for simplicity.

The traces used in our experiments were obtained from the UMass Trace Repository [23] and Microsoft [22]. The two financial traces (short for Fin1 and Fin2) were collected from the OLTP applications running at a large financial institution and the WebSearch2 trace (short for Web2) was collected from a machine running a web search engine. The Microsoft Project trace was collected in a volume storing the project directories (short for Proj). The four traces represent different access patterns in terms of read/write ratio, IOPS and average request size, with the main workload parameters summarized in Table 2.

4.2 Case Study: RAID Reconstruction

In this section, we conduct experiments to evaluate how IDO boosts the performance of the RAID reconstruction. We incorporated IDO into MD's default reconstruction algorithm PR and compared IDO with two state-of-the-art RAID reconstruction optimizations, WorkOut [41] and VDF [36], in terms of reconstruction performance and user I/O performance. WorkOut tracks the user access popularity and issues all write requests and popular read requests to the surrogate set during reconstruction. VDF exploits the fact that the user I/O requests addressed to the failed disk are expensive by keeping the requested data previously stored on the failed disk longer in the storage cache and choosing data blocks belonging to the surviving disks to evict first. Because VDF is a cache replacement algorithm, we applied it to the management of the surrogate set to make the comparison fair.

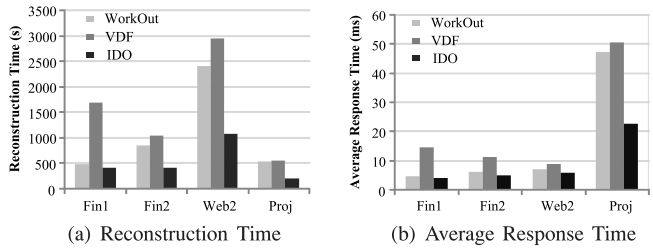


Fig. 7. The performance results of WorkOut, VDF and IDO in a four-disk RAID5 set with a stripe unit size of 64 KB, 1 MB/s minimum reconstruction bandwidth, and a local two-disk dedicated RAID1 set as the surrogate set, driven by the four traces.

4.2.1 Trace-Driven Results and Analysis

The experiments are conducted on a four-disk RAID5 set with a stripe unit size of 64 KB to compare the performance of WorkOut, VDF and IDO. Fig. 7 shows the reconstruction time and average user response time under the minimum reconstruction bandwidth of 1 MB/s, driven by the four traces. We configured a local two-disk dedicated RAID1 set as the surrogate set to boost the reconstruction performance of the four-disk degraded RAID5 set. For IDO, the data zone size was set to 12 MB and 10 percent data is proactively migrated to the surrogate set.

From Fig. 7a, we can see that IDO speeds up WorkOut by a factor of 1.2, 2.1, 2.2 and 2.7, and speeds up VDF by a factor of 4.2, 2.6, 2.7 and 2.8 in terms of the reconstruction time for the Fin1, Fin2, Web2 and Proj traces, respectively. The advantage of IDO stems from its ability to remove much more user I/O requests from the degraded RAID set than WorkOut and VDF, as indicated in Fig. 8, which enables it to accelerate the RAID reconstruction process. However, since the Fin1 trace has much more write requests than read requests (as indicated in Table 2), IDO and WorkOut have similar abilities to remove the write requests from the degraded RAID set, reducing the performance advantage of IDO over WorkOut. For the read-intensive traces, i.e., Fin2, Web2 and Proj, IDO removes much more read requests from the degraded RAID set than WorkOut. This is because IDO proactively identifies both the temporal locality and the spatial locality in the normal mode and migrates these

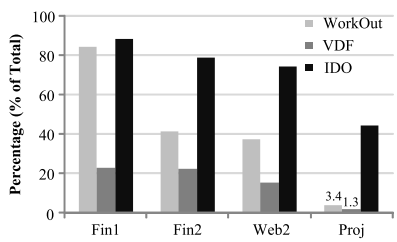
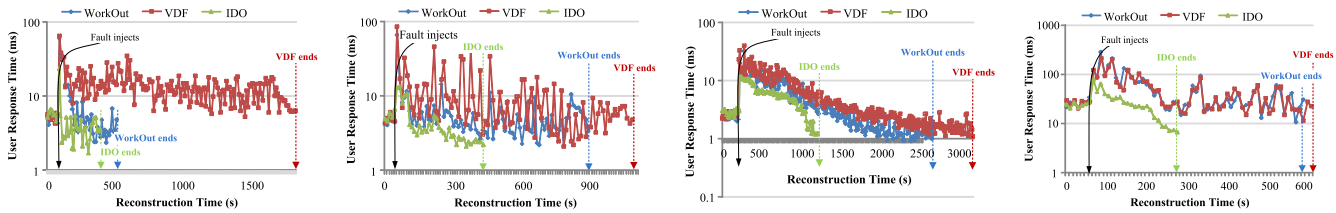


Fig. 8. Percentage of redirected user I/O requests for WorkOut, VDF and IDO under the minimum reconstruction bandwidth of 1 MB/s.



(a) Financial1.spc (b) Financial2.spc (c) WebSearch2.spc (d) MSR Project

Fig. 9. A comparison of user response times during reconstruction of WorkOut, VDF and IDO in a four-disk RAID5 set with a stripe unit size of 64 KB, 1 MB/s minimum reconstruction bandwidth, and a local two-disk dedicated RAID1 set as the surrogate RAID set, driven by the four traces.

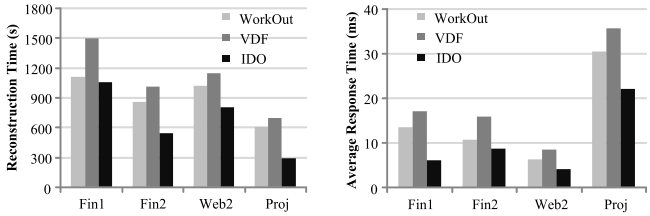
hot data zones before disk fails, while WorkOut only reactively identifies the temporal locality and migrates the popular read data after a disk fails. In this case, most subsequent read requests addressed to these hot data zones in IDO can be serviced directly by the surrogate RAID set instead of the much slower degraded RAID set. As a result, the advantage margin of IDO over WorkOut is much wider under read-intensive traces than under write-intensive traces. For example, IDO reduces the reconstruction time much more significantly than WorkOut under the Proj trace because the Proj trace has poor temporal locality, which leaves WorkOut much less room to improve than the spatial-locality-exploiting IDO.

On the other hand, from Fig. 7a, we can see that the performance of both WorkOut and IDO are better than that of VDF. The reason is that both WorkOut and IDO reduce not only the user I/O requests to the failed disk, but also the popular read requests and all write requests to the surviving disks. VDF keeps the data blocks belonging to the failed disk longer in the storage cache because servicing these data blocks is much more expensive than servicing the data blocks belonging to the surviving disks. However, if the data blocks belonging to the failed disk are already reconstructed, they will behave exactly the same as the data blocks belonging to the surviving disks because the read redirection technique will fetch these data blocks directly from the replacement disk rather than reconstructing them from the surviving disks again. In the VDF evaluation reported in [36], the experimental kernel is Linux 2.6.32 without the read redirection function that has been implemented in the Linux MD software since Linux 2.6.35 [20]. The results also confirm the conclusion made in our previous WorkOut study that the user I/O intensity has a significant impact on the on-line RAID reconstruction performance.

From Fig. 7b, we can also see that IDO speeds up WorkOut by a factor of 1.2, 1.3, 1.3 and 2.1, and speeds up VDF by a factor of 3.8, 2.4, 1.6 and 2.2 in terms of the average user response time (i.e., user I/O performance) during reconstruction for the Fin1, Fin2, Web2 and Proj traces, respectively. The reason why IDO only improves WorkOut slightly is that the minimum reconstruction bandwidth is set to be the default 1MB/s, which gives the user I/O requests a higher priority than the reconstruction requests. However, the performances of both IDO and WorkOut are better than that of VDF because IDO and WorkOut remove much more user I/O requests from the degraded RAID set than VDF, as revealed in Fig. 8. Removing the user I/O requests from the degraded RAID set directly reduces the user response time for the following two reasons. First, the response time of the redirected requests is no longer affected by the reconstruction process that competes for the available disk bandwidth with the user I/O requests on the degraded RAID set. Moreover, the redirected write data is sequentially laid out on the surrogate RAID set, thus further reducing the user response time. Second, many user I/O requests are removed from the degraded RAID set and the I/O queue on the degraded RAID set is accordingly shortened, thus reducing the response time of the remaining user I/O requests that are still serviced by the degraded RAID set.

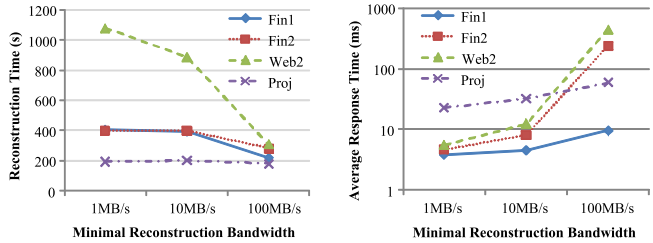
Fig. 9 compares in more details the reconstruction performances and user I/O performances of IDO, WorkOut and VDF during reconstruction, and highlights the significant advantage of IDO over WorkOut and VDF. Two aspects of the significant improvement in the user response time are demonstrated in these figures. First, the onset of the performance improvement of IDO is much earlier than that of WorkOut and VDF during reconstruction. The reason is that IDO has already captured the data popularity and migrated these hot data to a surrogate device before a disk fails, thus enabling it to optimize the reconstruction process earlier and consequently outperform both WorkOut and VDF in terms of user response time during reconstruction. Upon a disk fails, the user requests could be serviced by the surrogate device. Later on the background migration job has no effect on the user's IO performance. Second, IDO completes the reconstruction process much more quickly than WorkOut and VDF, which translates into improved reliability. The RAID reconstruction period is also called a "window of vulnerability" during which a subsequent disk failure (or a series of subsequent disk failures) will result in data loss [34]. Thus, shorter reconstruction time indicates higher reliability. Furthermore, user I/O requests in the IDO reconstruction experience a much shorter period of time in which they see increased response time than in the WorkOut and VDF reconstruction.

To gain a better understanding of the reasons behind the significant improvement achieved by IDO, we plotted the percentage of redirected requests for the three schemes under the minimum reconstruction bandwidth of 1MB/s. From Fig. 8, we can see that IDO moves 88.5, 78.7, 74.6 and 44.1 percent of user I/O requests from the degraded RAID set to the surrogate RAID set for the four traces respectively, significantly more than either WorkOut or VDF does. The reason is that both WorkOut and VDF only exploit the temporal locality of user I/O requests and VDF only gives higher priority to the user I/O requests belonging to the failed disk. In contrast, IDO exploits both the temporal locality and spatial locality of user I/O requests on all disks and migrates the data in the hot zones to the surrogate RAID set. And, most importantly, IDO uses a proactive



(a) Reconstruction Time (b) Average Response Time

Fig. 10. Reconstruction times and average user response times of RAID6 reconstruction.



(a) Reconstruction Time (b) Average Response Time

Fig. 11. Reconstruction times and average user response times of IDO as a function of different minimum reconstruction bandwidths (1, 10 and 100 MB/s) under the four traces.

optimization that is superior to the reactive optimizations, WorkOut and VDF. On the other hand, removing user I/O requests from the degraded RAID set directly reduces both the reconstruction time and user response time [41], so that IDO does much better than either WorkOut or VDF.

We also conducted experiments on a six-disk RAID6 set (4 data + 2 parity) with a stripe unit size of 64 KB under the minimum reconstruction bandwidth of 1 MB/s. In the RAID6 experiments, we configured a two-disk dedicated RAID1 set in the same storage node as the local surrogate set. In the experiments, we measured the reconstruction times and the average user response times when one disk fails.

From Fig. 10, we can see that IDO improves both the reconstruction performance and user I/O performance over the WorkOut and VDF schemes. In particular, IDO speeds up WorkOut by a factor of up to 2.1 with an average of 1.5 in terms of the reconstruction time, and by a factor of up to 2.2 with an average of 1.6 in terms of the average user response time. IDO speeds up VDF by a factor of up to 2.4 with an average of 1.8 in terms of the reconstruction time, and by a factor of up to 2.8 with an average of 2.1 in terms of the average user response time.

The reason behind the improvement on RAID6 is similar to that on RAID5. Upon a disk failure, all the disks in RAID6, as in RAID5, will be involved in servicing the reconstruction I/O requests. In the meantime, all the disks will service the user I/O requests. Thus, removing the user I/O requests can directly speed up the reconstruction process by allowing much more disk bandwidth to service the reconstruction I/O requests. Moreover, the average user response times also decrease because most of the user I/O requests are serviced by the surrogate set without interfering with the reconstruction I/O requests. IDO works in a proactive way and exploits both the temporal locality and spatial locality of the user I/O requests, thus removing much more user I/O requests from the degraded RAID set to the surrogate set (as similarly indicated in Fig. 8) and performing better than Workout and VDF.

4.2.2 Sensitivity Study

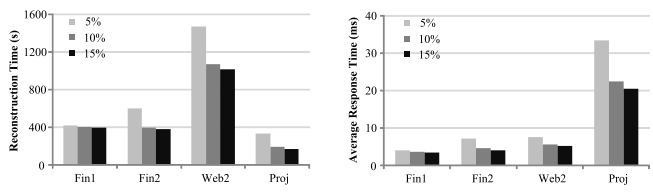
The IDO performance is likely influenced by several important factors, including the available reconstruction bandwidth, proactively migrated data percentage, the data zone size, the stripe unit size, the number of disks, and the location of the surrogate set.

Reconstruction bandwidth. To evaluate how the minimum reconstruction bandwidth affects the reconstruction performance, we conducted experiments to measure the reconstruction time and average user response time as a function of different minimum reconstruction bandwidths, 1, 10 and 100 MB/s, respectively. Fig. 11 shows the experimental results on a four-disk RAID5 set with a stripe unit size of 64 KB, a data zone size of 12 MB and 10 percent data is proactively migrated to the surrogate set.

As shown in Fig. 11a, the reconstruction time generally decreases with the increasing minimum reconstruction bandwidth. However, for the Fin1, Fin2 and Proj traces, the reconstruction times remain almost unchanged when the minimum reconstruction bandwidth changes from 1 to 10 MB/s. The reason is that when the minimum reconstruction bandwidth is 1 MB/s, the actual reconstruction speed is around 10 MB/s due to the low user I/O intensity on the degraded RAID set. In contrast, From Fig. 11b, we can see that the user response time increases rapidly with the increasing minimum reconstruction bandwidth. When the minimum reconstruction bandwidth increases, much more reconstruction I/O requests are issued at a given time, thus lengthening the disk I/O queue and increasing the user I/O response time.

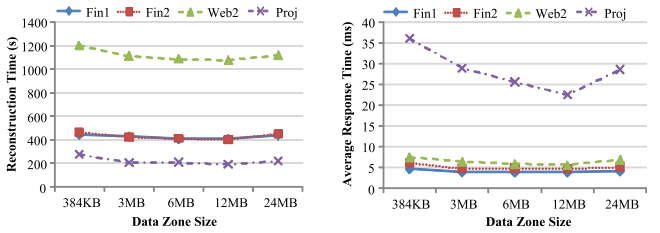
Migrated data percentage. The proactively migrated data percentage in IDO is a key factor affecting the system performance. In order to evaluate the effect of the proactively migrated data percentage on the reconstruction performance and user I/O performance during reconstruction, we conducted experiments on a four-disk RAID5 set with a stripe unit size of 64 KB, 12 MB data zone size and different data percentage, 5, 10 and 15 percent, under the minimal reconstruction bandwidth of 1 MB/s.

As shown in Fig. 12, the reconstruction time generally decreases with the increasing migrated data percentage. However, for the Fin1 trace, the reconstruction times remain almost unchanged with different migrated data percentage. The reason is that Fin1 trace has much more write requests which can be directly issued to the surrogate set. The higher write ratio makes Fin1 trace be not sensitive to the migrated data percentage. Moreover, when the migrated data



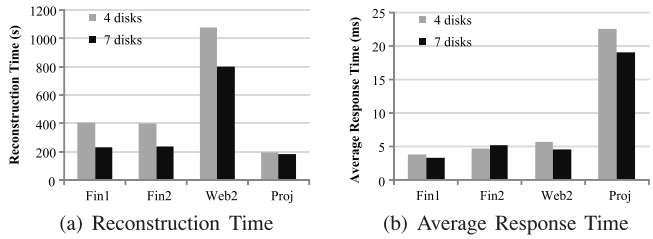
(a) Reconstruction Time (b) Average Response Time

Fig. 12. Reconstruction times and average user response times of IDO as a function of different migrated data percentage (5, 10 and 15 percent) under the four traces.



(a) Reconstruction Time (b) Average Response Time

Fig. 13. Reconstruction times and average user response times of IDO as a function of different data zone size (384 KB, 3 MB to 24 MB) under the four traces.



(a) Reconstruction Time (b) Average Response Time

Fig. 14. Reconstruction times and average user response times of IDO as a function of different numbers of disks (4 and 7) under the four traces.

percentage increases from 10 to 15 percent, both the reconstruction time and the user response time are not reduced significantly. The reason is that top popular data absorbs the most user accesses. Thus the increased migrated data contributes little to the system performance improvement. Our experiments, driven by the four traces, suggest that the migrated data percentage of 10 percent is the nearly best for system performance.

Data zone size. The data zone size in IDO is a key factor in identifying the data popularity. In order to evaluate the effect of the data zone size on the reconstruction performance and user I/O performance during reconstruction, we conducted experiments on a four-disk RAID5 set with a stripe unit size of 64 KB and different data zone sizes of 384 KB, 3 MB, 6 MB, 12 MB and 24 MB, under the minimal reconstruction bandwidth of 1 MB/s and 10 percent data is proactively migrated to the surrogate set.

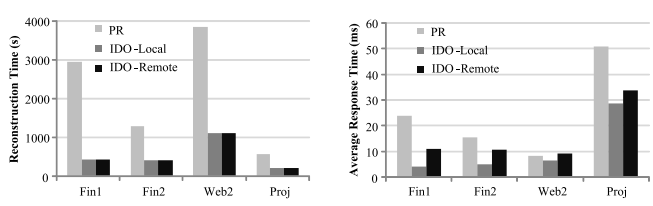
As shown in Fig. 13, the results indicate that, with a very small data zone, both the reconstruction time and the user response time are large. The reason is that the reconstruction sequentiality is destroyed when the data zone is small. Moreover, the spatial locality is somewhat weakened with a very small zone, although highly concentrated temporal locality is still captured. On the other hand, with a very large data zone, both the reconstruction time and the user response time are also large. The reason is that with a very large data zone, IDO will likely migrate much more rarely-accessed cold data blocks to the surrogate set, thus wasting the disk bandwidth resources. Our experiments, driven by the four traces, suggest that the data zone sizes between 3 to 12 MB are consistently the best.

Stripe unit size. To examine the impact of the stripe unit size on the RAID reconstruction, we conducted experiments on a four-disk RAID5 set with stripe unit sizes of 4, 16 and 64 KB, respectively. The experimental results show that the reconstruction times and user response times are almost unchanged, suggesting that IDO is not sensitive to the stripe unit size. The reason is that most user I/O requests are serviced by the surrogate RAID set, thus the stripe unit size of the degraded RAID set has no impact on these redirected user I/O requests and very little impact overall. Accordantly, the RAID reconstruction speed is not affected. Due to space limits, these results are not quantitatively shown here.

Number of disks. To examine the sensitivity of IDO to the number of disks of the degraded RAID set, we conducted experiments on RAID5 sets consisting of different numbers of disks (4 and 7) with a stripe unit size of 64 KB under the minimum reconstruction bandwidth of 1 MB/s and 10 percent data is proactively migrated to the surrogate set. From Fig. 14, we can see that the reconstruction time decreases with the increasing number of disks. The reason is that the user I/O intensity on individual disks will decrease when the RAID set consists of more disks, allowing for a shorter reconstruction time. However, the user I/O requests are not significantly affected since they are mostly serviced by the surrogate RAID set. The remaining user I/O requests still serviced by the degraded RAID set only slightly affect the total user response time.

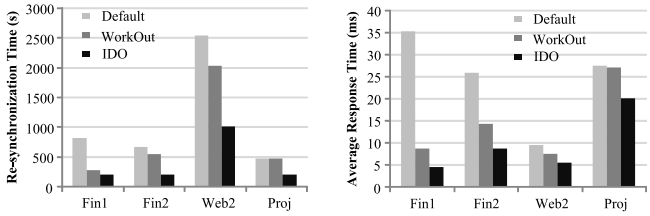
Location of the surrogate device. In a large-scale data center, the location of the surrogate set can also affect the RAID reconstruction performance. To examine the impact of the location of the surrogate set on the reconstruction performance, we conducted experiments to migrate the hot data to a different storage node connected with a gigabit Ethernet interface in a local area network.

From Fig. 15a, we can see that the reconstruction time is almost unchanged, which indicates that the reconstruction time is not sensitive to the location of the surrogate set. The reason is that no matter where the surrogate set is, the total numbers of redirected user I/O requests are almost the same, thus the reconstruction speed of the degraded RAID set is similar when configuring different surrogate sets. On the other hand, the average user response time increases significantly when configuring a remote surrogate set, as shown in Fig. 15b. The reason is that the response time of the redirected user I/O requests must now include the extra network delay in that case. However, compared with PR (the default reconstruction algorithm of Linux MD), IDO still significantly reduces the user response time and the reconstruction time with a remote surrogate set. The results suggest that in a large-scale data center, both the local and remote surrogate devices are helpful in improving the reconstruction performance, which further validates the effectiveness and adaptivity of IDO in large-scale data centers.



(a) Reconstruction Time (b) Average Response Time

Fig. 15. Reconstruction times and average user response times with respect to different locations of the surrogate set under the four traces.



(a) Re-synchronization Time (b) Average Response Time

Fig. 16. The re-synchronization performance of the default re-synchronization function in the Linux MD software without any optimization, WorkOut and IDO.

4.3 Case Study: RAID Re-Synchronization

To demonstrate how IDO can be extended to optimize other background RAID tasks, we incorporated IDO into the RAID re-synchronization module. We conducted experiments on a four-disk RAID5 set with a stripe unit size of 64 KB under the minimum re-synchronization bandwidth of 1 MB/s and 10 percent data is proactively migrated to the surrogate set, driven by the four traces. We configured a dedicated two-disk local RAID1 set as the surrogate set. The experimental results of the re-synchronization times and average user response times during re-synchronization are shown in Fig. 16.

Although the RAID re-synchronization process operates somewhat differently with the RAID reconstruction process, the re-synchronization requests also compete for the disk resources with the user I/O requests during the on-line re-synchronization period, which is similar to the latter. By redirecting a significant amount of user I/O requests away from the RAID set undergoing re-synchronization, IDO reduces both the re-synchronization time and user response time. The trace-driven evaluations show that IDO outperforms WorkOut by a factor of up to 2.8 with an average of 2.2 in terms of the re-synchronization time, and by a factor of up to 2.0 with an average of 1.6 in terms of the average user response time. The results are very similar to those RAID reconstruction experiments, as are the reasons behind them.

4.4 Overhead Analysis

Besides the device overhead for the surrogate set in the case of dedicated surrogate sets [41], we analyzes the following two overhead metrics in this paper: the performance overhead in the normal mode and the memory overhead.

4.4.1 Performance Overhead

IDO is a proactive optimization for improving the performance of low-priority background tasks. It requires a hot data identification module that may affect the system performance in the normal mode. In order to quantify how much the impact is, we conducted experiments to evaluate the user response times in the normal mode with and without the hot data identification module.

From the experimental results, we find that the user response times in the two cases remain roughly unchanged under the four traces. In the worst case, the performance with the hot data identification module activated degrades by less than 3 percent under the WebSearcch2.spc trace. The reason is that the hot data identification module only adds one extra operation (i.e., incrementing the in-memory popularity value of the corresponding data zone by one) for each request. Thus, the performance overhead is negligible in face of the high latency of the disk accesses.

4.4.2 Memory Overhead

To prevent data loss caused by power failure, IDO uses non-volatile memory to store the Zone_Table and D_Map, thus incurring extra memory overhead. However, IDO uses less non-volatile memory capacity than WorkOut. The reason is that WorkOut migrates the user requested data to the surrogate set while IDO migrates the hot data zones. In WorkOut, each migrated write request and read request requires a corresponding entry in the mapping table. However, in IDO, only the migrated write requests and hot data zones require corresponding entries in the mapping table. Since the number of migrated hot data zones in IDO is generally much smaller than the number of hot read requests in WorkOut, the memory overhead of IDO is lower than that of WorkOut.

In the above experiments on the RAID5 set with individual disk capacity of 10 GB, the maximum memory overheads are 0.12, 0.24, 0.11 and 0.07 MB for the Fin1, Fin2, Web2 and Proj traces, respectively. Accordingly, if the individual disk capacity is 1TB, the maximum memory overheads are approximately 12, 24, 11 and 7 MB, respectively. With the rapid increase in the size of memory and decrease in the cost of non-volatile memories, this memory overhead of IDO is arguably reasonable and acceptable to the end users.

5 Related Work

The scheduling of the low-priority background tasks is important for storage availability and received much more attention from the storage community. Among these studies, the RAID reconstruction is an important function in large-scale data centers. Since the RAID technology [25] was proposed in 1988, a rich body of research on the the low-priority background task optimizations has been published in the literature [12], [17], [18], [28], [32], [33], [34], [36], [38], [41], [43], [44], [45], [46], [47], [49]. Generally speaking, these approaches can be categorized into two types: optimizing the task workflow and optimizing the user I/O requests during the low-priority background tasks.

The first type of optimization approaches, such as SOR [13], DOR [12], PR [18], Live-block recovery [32], PRO [34], JOR [40], improve the reconstruction performance by optimizing the reconstruction workflow. DOR [12] assigns one reconstruction thread for each disk, unlike SOR [13] that assigns one reconstruction thread for each stripe, allowing DOR to efficiently exploit the disk bandwidth to improve the RAID reconstruction performance. Live-block recovery [32] and JOR [40] exploit the data liveness semantics to reduce the RAID reconstruction time by ignoring the "dead" data blocks on the failed disk. PRO [34] exploits the user access locality by first reconstructing the hot data blocks on the failed disk. When the hot data blocks have been recovered, the reconstruction process for read requests to the failed disk can be significantly reduced, thus reducing both the reconstruction time and user I/O response time.

Although the above workflow-optimized schemes can also improve the user I/O performance, the improvement is limited because the user I/O requests must still be serviced

by the degraded RAID set. Therefore, the disk resource contention between user I/O requests and background I/O requests still persists.

The second type of optimization approaches, such as MICRO [46], WorkOut [41], Shaper [43] and VDF [36], improves the low-priority background task performance by optimizing the user I/O requests. These optimization approaches directly improve the user I/O performance during the low-priority background task while simultaneously improving the low-priority background task performance by allocating much more disk resources to the low-priority background task. MICRO [46] collaboratively utilizes the storage cache and the RAID controller cache to reduce the number of physical disk accesses caused by RAID reconstruction. VDF [36] improves the reconstruction performance by keeping the user requests belonging to the failed disk longer in the cache. However, if the requested data belonging to the failed disk have already been reconstructed to the replacement disk, the access delays of these user requests will not be further improved because they behave exactly the same as those data blocks belonging to the surviving disks. Therefore, the advantages of VDF scheme are weakened with the new functions in software RAID module [20]. Both Shaper [43] and VDF [36] use the reconstruction-aware storage cache to selectively filter user I/O requests, thus improving both the low-priority background task performance and user I/O performance. Different from the above schemes, WorkOut [41] aims to alleviate the user I/O intensity on the degraded RAID set, not just the failed disk, by redirecting many user I/O requests to a surrogate RAID set.

While optimizing the user I/O requests can also reduce the on-line RAID reconstruction time, the performance improvement of the above user-I/O-request-optimized approaches is limited. For example, both WorkOut and VDF only exploit the temporal locality of read requests, ignoring the highly beneficial spatial locality that abundantly exists among read requests. Moreover, VDF gives higher priority to the user I/O requests addressed at the failed disk. However, the RAID reconstruction process involves all disks and the user I/O requests addressed at the surviving disks also affect the reconstruction performance. And most importantly, the access-locality tracking functions in these schemes are all initiated after a disk fails, which is passive and ineffective. Compared with the existing RAID reconstruction optimizations, IDO can effectively improve the reconstruction performance and user I/O performance simultaneously by proactively identifying and migrating the hot data zones to a surrogate RAID set to optimize both the reconstruction workflow and user I/O requests. Though the data migration technique has been well studied for performance improvement [1], [14], [16] and energy efficiency [26], [37] of storage systems, IDO adopts this technique to significantly optimize the increasingly critical the low-priority background tasks in large-scale data centers.

6 Conclusion

Due to the increasing needs for the system maintenance, such as replacing the failed components, enhancing the system performance and expanding the data capacity, the performance of the low-priority background tasks has become increasingly important for the storage availability in large-scale data centers. Our proposed IDO can substantially improve the performance of the low-priority background tasks with low cost by utilizing the free space available in these environments. IDO exploits both the temporal locality and spatial locality of the user I/O requests to identify the hot data zones in the normal mode. By leveraging the prediction tools to identify the upcoming events, IDO proactively migrates the data blocks on the degraded device belonging to the hot data zones to a surrogate RAID set in the large-scale data centers. Upon a disk failure or crash reboot, IDO enables most subsequent user I/O requests to be serviced by the surrogate RAID set, thus improving the performance of both the low-priority background tasks and the foreground tasks.

In summary, this paper has made the following contributions:

- Motivated by our extensive experimental studies and analysis, we argue that the reactive optimization is not sufficient in large-scale data centers. Thus, we propose a proactive optimization to accelerate the low-priority background tasks.
- The workload studies and analysis have shown that the spatial locality is much more important than the temporal locality at the disk level. Thus, we propose a zone-based approach to exploiting both the spatial locality and temporal locality to boost the performance of the low-priority background tasks.
- We have designed and implemented a proactive zone-based optimization by proactively exploiting the data access patterns to judiciously outsource data. Moreover, we have conducted extensive experiments on two case studies to demonstrate that IDO outperforms the existing state-of-the-art approaches.

Acknowledgments

This work was supported by the China National Natural Science Foundation No. 61100033, No. 61472336 and No. 61402385, the US NSF under Grant No. NSF-CNS-1116606 and NSF-CNS-1016609, the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, Fundamental Research Funds for the Central Universities (No. 20720140515). This work was also supported by Huawei Innovation Research Program and the State Key Laboratory of High-End Server & Storage Technology. This is an extended version of our manuscript published in the Proceedings of the 26th USENIX Conference on Large Installation System Administration (LISA'12), San Diego, CA, Dec. 2012. B. Mao is the Corresponding Author.

References

[1] R. Arnan, E. Bachmat, T. Lam, and R. Michel, "Dynamic data reallocation in disk arrays," *ACM Trans. Storage*, vol. 3, no. 1, p. 1–16, 2007.

[2] (2011, May). Storage at Exascale: Some Thoughts from Panasas CTO Garth Gibson. Interview. [Online]. Available: http://www.hpcwire.com/hpcwire/2011-05-25/storage_at_exascale_some_thoughts_from_panasas_cto_garth_gibson.html

[3] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst.*, Jun. 2007, pp. 289–300.

[4] A. Brown and D. A. Patterson, "Towards availability benchmarks: A case study of software raid systems," presented at the USENIX Annu. Tech. Conf., San Diego, CA, USA, Jun. 2000.

[5] F. Chen, D. A. Koufaty, and X. Zhang, "Hystor: Making the best use of solid state drives in high performance storage systems," in *Int. Conf. Supercomput.*, Jun. 2011, pp. 22–32.

[6] V. Deenadhayalan, "GPFS Native RAID for 100,000-Disk petascale systems," in *Proc. Large Installation Syst. Admin. Conf.*, Dec. 2011.

[7] T. E. Denehy, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Journal-guided resynchronization for software RAID," in *Proc. USENIX Conf. File Storage Technol.*, Dec. 2005, pp. 87–100.

[8] (2013, May). Era of Agile and Always-Available Data Storage. [Online]. Available: http://www.computer.org/portal/web/computingnow/archive/march2013

[9] G. Gibson, "Reflections on failure in post-terascale parallel computing. keynote," in *Proc. Int. Conf. Parallel Process.*, Sep. 2007.

[10] M. Goldszmidt, "Finding soon-to-fail disks in a haystack," in *Proc. 4th USENIX Conf. Hot Topics Storage File Syst.*, Jun. 2012, p. 8.

[11] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 4th ed. San Mateo, CA, USA: Morgan Kaufmann, 2006.

[12] M. Holland, "On-line data reconstruction in redundant disk arrays," Ph.D. dissertation, Dept. Elec. Comput. Eng., Carnegie Mellon Univ., Pittsburgh, PA, USA, Apr. 1994.

[13] M. Holland, G. Gibson, and D. P. Siewiorek, "Architectures and algorithms for on-line failure recovery in redundant disk arrays," *J. Distrib. Parallel Databases*, vol. 2, no. 3, pp. 295–335, Jul. 1994.

[14] IBM Easy Tier. [Online]. Available: http://www.almaden.ibm.com/storagesystems/projects/easytier, 2011.

[15] I. Iliadis, R. Haas, X.-Y. Hu, and E. Eleftheriou, "Disk scrubbing versus intra-disk redundancy for high-reliability RAID storage system," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst.*, Jun. 2008, pp. 241–252.

[16] S. Kang and A. Reddy, "User-centric data migration in networked storage systems," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Apr. 2008, pp. 1–12.

[17] O. Khan, R. Burns, J. S. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2012, pp. 251–264.

[18] J. Lee and J. Lui, "Automatic recovery from disk failure in continuous-media servers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 5, pp. 499–515, May. 2002.

[19] J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, and X. Liu, "Hard drive failure prediction using classification and regression trees," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 383–394.

[20] [MD PATCH 09/16] md/raid5: Preferentially read from replacement device if possible. [Online]. Available: http://www.spinics.net/lists/raid/msg36361.html, 2009.

[21] A. Miranda and T. Cortes, "CRAID: Online RAID upgrades using dynamic hot data reorganization," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2014, pp. 133–146.

[22] D. Narayanan, A. Donnelly, and A. Rowstron, "Write Off-Loading: Practical power management for enterprise storage," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2008, pp. 253–267.

[23] OLTP Application I/O and Search Engine I/O. [Online]. Available: http://traces.cs.umass.edu/index.php/Storage/Storage, 2007.

[24] A. Oprea and A. Juels, "A clean-slate look at disk scrubbing," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2010, pp. 57–70.

[25] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, Jun. 1988, pp. 109–116.

[26] E. Pinheiro and R. Bianchini, "Energy conservation techniques for disk array-based servers," in *Proc. 18th Annu. Int. Conf. Supercomput.*, Jun. 2004, pp. 68–78.

[27] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2007, pp. 17–29.

[28] J. S. Plank, M. Blaum, and J. L. Hafner, "SD Codes: Erasure codes designed for how storage systems really fail," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2013, pp. 95–104.

[29] M. Saxena, and M. M. Swift, "FlashVM: Virtual memory management on flash," in *Proc. USENIX Annu. Tech. Conf.*, Jun. 2010, pp. 187–200.

[30] B. Schroeder, and G. Gibson, "Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?" in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2007, pp. 1–16.

[31] T. J. E. Schwarz, Q. Xin, E. L. Miller, D. D. E. Long, A. Hospodor, and S. Ng, "Disk scrubbing in large archival storage systems," in *Proc. IEEE Comput. Soc. 12th Annu. Int. Symp. Modeling, Anal. Simul. Comput. Telecommun. Syst.*, Oct. 2004, pp. 409–418.

[32] M. Sivathanu, V. Prabhakaran, F. I. Popovici, T. E. Denehy, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Improving storage system availability with D-GRAID," in *Proc. USENIX Conf. File Storage Technol.*, Mar. 2004, pp. 15–30.

[33] L. Tian, Q. Cao, H. Jiang, D. Feng, C. Xie, and Q. Xin, "SPA: On-line availability upgrades for parity-based RAIDs through supplementary parity augmentations," *ACM Trans. Storage*, vol. 6, no. 4, pp. 1–23, 2011.

[34] L. Tian, D. Feng, H. Jiang, K. Zhou, L. Zeng, J. Chen, Z. Wang, and Z. Song, "PRO: A popularity-based multi-threaded reconstruction optimization for RAID-structured storage systems," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2007, pp. 277–290.

[35] J. Tucci, "Cloud + Big Data = Massive change, massive opportunity," Keynote Address at UW CSE 2011-12 Annual Industrial Affiliates Meeting, Oct. 2011.

[36] S. Wan, Q. Cao, J. Huang, S. Li, X. Li, S. Zhan, L. Yu, C. Xie, and X. He, "Victim disk first: An asymmetric cache to boost the performance of disk arrays under faulty conditions," in *Proc. USENIX Annu. Tech. Conf.*, Jun. 2011, pp. 173–186.

[37] C. Weddle, M. Oldham, J. Qian, A. Wang, P. Reiher, and G. Kuenning, "PARAID: A gear-shifting power-aware RAID," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2007, pp. 245–260.

[38] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou, "Scalable performance of the panasas parallel file system," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2008, pp. 17–33.

[39] D. Wenk, "Is 'Good Enough' storage good enough for compliance?" *Disaster Recov. J.*, vol. 17, no. 1, pp. 1–3, 2004.

[40] S. Wu, D. Feng, H. Jiang, B. Mao, L. Zeng, and J. Chen, "JOR: A journal-guided reconstruction optimization for raid-structured storage systems," in *Proc. 15th Int. Conf. Parallel Distrib. Syst.*, Dec. 2009, pp. 609–616.

[41] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao, "WorkOut: I/O workload outsourcing for boosting RAID reconstruction performance," in *Proc. USENIX Conf. File Storage Technol.*, Feb. 2009, pp. 239–252.

[42] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao, "Improving availability of RAID-structured storage systems by workload outsoucing," *IEEE Trans. Comput.*, vol. 60, no. 1, pp. 64–79, Jan. 2011.

[43] S. Wu, B. Mao, D. Feng, and J. Chen, "Availability-aware cache management with improved RAID reconstruction performance," in *Proc. IEEE 13th Int. Conf. Comput. Sci. Eng.*, Dec. 2010, pp. 229–236.

[44] L. Xiang, Y. Xu, J. C. S. Lui, and Q. Chang, "Optimal recovery of single disk failure in RDP code storage systems," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst.*, Jun. 2010, pp. 119–130.

[45] T. Xie and A. Sharma, "Collaboration-oriented data recovery for mobile disk arrays," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2009, pp. 631–638.

[46] T. Xie and H. Wang, "MICRO: A multilevel caching-based reconstruction optimization for mobile storage systems," *IEEE Trans. Comput.*, vol. 57, no. 10, pp. 1386–1398, Oct. 2008.

[47] Q. Xin, E. L. Miller, and T. J. E. Schwarz, "Evaluation of distributed recovery in large-scale storage systems," in *Proc. 13th Int. Symp. High Performance Distrib. Comput.*, Jun. 2004, pp. 172–181.

[48] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, "Proactive drive failure prediction for large scale storage systems," in *Proc. IEEE 29th Symp. Mass Storage Syst. Technol.*, May 2013, pp. 1–5.

[49] Y. Zhu, P. P. C. Lee, L. Xiang, Y. Xu, and L. Gao, "A cost-based heterogeneous recovery scheme for distributed storage systems with RAID-6 codes," in *Proc. 42nd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2012.

Suzhen Wu received the BE and PhD degrees in computer science and technology and computer architecture from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2010, respectively. She is an assistant professor at the Computer Science Department, Xiamen University. Her research interests include computer architecture and storage system. She has more than 20 publications in journal and international conferences including IEEE-TC, ACM-ToS, FAST, LISA, IPDPS, MASCOTS, and ICPADS. She is a member of the IEEE and the ACM.

Hong Jiang received the BSc degree in computer engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1982, the MASc degree in computer engineering from the University of Toronto, Toronto, Canada, in 1987, and the PhD degree in computer science from the Texas A&M University, College Station, Texas, in 1991. Since August 1991, he has been at the University of Nebraska-Lincoln (UNL), Lincoln, Nebraska, where he is a Willa Cather Professor of computer science and engineering. He is currently on leave as a program director at the National Science Foundation. At UNL, he has graduated 13 PhD students who upon their graduations either landed academic tenure-track positions in PhD-granting US institutions or were employed by major US IT corporations. His current research interests include computer architecture, computer storage systems and parallel I/O, high-performance computing, big data computing, cloud computing, performance evaluation. He serves as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. He has more than 200 publications in major journals and international conferences in these areas, including IEEE-TPDS, IEEE-TC, ACM-TACO, JPDC, ISCA, MICRO, USENIX ATC, FAST, LISA, ICDCS, IPDPS, MIDDLEWARE, OOPLAS, ECOOP, SC, ICS, HPDC, INFOCOM, ICPP, etc., and his research has been supported by NSF, DOD and the State of Nebraska. He is a senior member of the IEEE and a member of the ACM.

Bo Mao received the BE degree in computer science and technology from Northeast University, Shenyang, China, in 2005, and the PhD degree in computer architecture from the Huazhong University of Science and Technology, Wuhan, China, in 2010. His research interests include storage system, cloud computing and big data. He is an assistant professor of the Software School of Xiamen University. He has more than 20 publications in international journals and conferences including IEEE-TC, ACM-ToS, FAST, IPDPS, Cluster, LISA, MASCOTS, and ICPADS. He is a member of the IEEE.