Improving Availability of RAID-Structured Storage Systems by Workload Outsourcing

Suzhen Wu, *Member*, *IEEE*, Hong Jiang, *Senior Member*, *IEEE*,
Dan Feng, *Member*, *IEEE*, Lei Tian, *Student Member*, *IEEE*, and Bo Mao

Abstract—Due to the contention for the shared disk bandwidth, the user I/O intensity can significantly impact the performance of the online low-priority background tasks, thus reducing the reliability and availability of RAID-structured storage systems. In this paper, we propose a novel and practical scheme, called *WorkOut* (I/O *Work*load *Out*sourcing), to significantly boost the performance of those low-priority background tasks. WorkOut effectively outsources all write requests and popular read requests originally targeted at the degraded RAID set that is performing the low-priority background tasks to a surrogate RAID set. The lightweight prototype implementation of WorkOut and extensive trace-driven and benchmark-driven experiments on two case studies demonstrate that, compared with existing approaches, WorkOut effectively improves the performance of the low-priority background tasks, such as RAID reconstruction and RAID resynchronization. Importantly, WorkOut is portable and can be easily incorporated into any existing optimizing algorithms for RAID-structured storage systems.

Index Terms—Low-priority background tasks, RAID reconstruction, reliability, availability, performance evaluation.

1 Introduction

Generally speaking, there are two classes of tasks for storage systems: high-priority foreground tasks and low-priority background tasks [4]. Foreground tasks are generated from the system applications, such as the online transaction processing (OLTP), which require providing 24×7 online services. Background tasks are generated from maintenance or repair activities due to disk failures, such as RAID reconstruction, RAID resynchronization, and disk scrubbing. These background tasks aim at improving the performance and reliability of storage systems.

In face of high disk failure rates and untimely system crash, the low-priority background tasks become more and more frequent in large-scale storage systems. With the ever growing number of disks in data centers and the significant disk failure rate, the reconstruction might soon become the common mode of operation in large-scale systems rather than the exception [11], [38], [41] and the periodically disk scrubbing might be quite necessary [5]. The resynchronization also should be taken into account for the unexpected system crash and the high software error rate [22]. Moreover, since the capacity of drives grows at a much higher rate than other performance parameters, such as the bandwidth and position time [12], the completion time for these background tasks will increase in the future. As a result, improving the performance of the low-priority background tasks is becoming a growing and critical concern.

The performance of the low-priority background tasks depends on the following two factors: 1) The time it takes to complete the background tasks, since longer completion times for the background tasks translate to a longer "window of vulnerability," in which a disk failure or a power down may cause persistent data loss. 2) The impact of the background tasks on the foreground application, that is, to what degree are the user requests affected by the ongoing background tasks.

Similar to the RAID reconstruction, other low-priority background tasks also fall into two different categories [13]: *offline*, when the RAID devotes all of its resources to performing the background tasks without serving any user I/O requests, and *online*, when the RAID continues to serve user I/O requests and performs the background task concurrently.

Obviously, the offline background tasks are much faster than their online counterparts, but they are unpractical in environments that require high availability, as the RAID needs to be taken offline when performing the background tasks.

On the other hand, the online background tasks allow the foreground traffic to continue concurrently, but take longer time than their offline counterparts due to the disk bandwidth contention between the background and foreground processes. From our primary experiments (see Section 2.2), we find that the I/O requests generated by the foreground applications significantly affect the performance of the low-priority background tasks.

Many approaches have been proposed to improve the performance of the low-priority background tasks, such as RAID reconstruction [4], [14], [16], [27], [48], RAID resynchronization [9], and disk scrubbing [20], [35], [43].

- *S. Wu is with the Computer Science Department, Xiamen University, Room 404, Science and Research Bldg. No. 2, Haiyun Campus, Xiamen 361005, China. E-mail: suzhen66@gmail.com.*
- *H. Jiang is with the Department of Computer Science and Engineering, University of Nebraska—Lincoln, 217 Schorr Center, 1101 T Street, Lincoln, NE 68588-0150. E-mail: jiang@cse.unl.edu.*
- *D. Feng, L. Tian, and B. Mao are with the Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology, Huazhong University of Science and Technology, F307, Division of Data Storage System, Wuhan, 430074 China.*
  E-mail: {dfeng, ltian}@hust.edu.cn, maobo.hust@gmail.com.

Manuscript received 24 Jan. 2010; revised 4 June 2010; accepted 21 July 2010; published online 6 Oct. 2010.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-2010-01-0045.
Digital Object Identifier no. 10.1109/TC.2010.206.

0018-9340/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

We note that all these approaches focus on a single RAID set. In this paper, we propose a new approach that exploits the fact that most data centers contain a large number of RAID sets.

Inspired by recent work on write offloading [33], [57] and data migration [3], [29], [54], we propose *WorkOut* to significantly improve the performance of the low-priority background tasks by I/O *Work*load *Out*sourcing. The main idea behind WorkOut is to temporarily redirect all write requests and popular read requests originally targeted at the RAID set that is performing the background tasks to a *surrogate RAID set*. The surrogate RAID set can be a set of spare disks or free space on another RAID set. Since the background tasks can cause the performance degradation of RAID-structured storage systems, we refer to the RAID set that is performing background tasks as *degraded RAID set* in this paper.

WorkOut has twofold benefits. First, WorkOut reduces the impact of the low-priority background tasks on the foreground traffic because most user I/O requests can be served by the surrogate RAID set, and hence, no longer compete with the background tasks for disk bandwidth. Second, WorkOut speeds up the process of the background tasks, since more bandwidth can be devoted to it.

We have implemented a lightweight prototype of WorkOut in the Linux software RAID module (MD) and carried out WorkOut on two case studies: RAID reconstruction and RAID resynchronization. Extensive trace-driven and benchmark-driven experiments show that WorkOut significantly outperforms the reconstruction algorithms PR [27] and PRO [49] in both reconstruction time and user response time. The results also demonstrate that WorkOut can effectively improve the performance of other low-priority background tasks, such as the RAID resynchronization.

The rest of this paper is organized as follows: Background and motivation are presented in Section 2. The design of WorkOut is described in Section 3. Performance evaluations of WorkOut based on a prototype implementation and two case studies are presented in Section 4. We analyze the reliability of WorkOut in Section 5 and present the related work in Section 6. We summarize the main contributions of the paper and point out the directions for future research in Section 7.

2 Background and Motivation

In this section, we first describe the low-priority background tasks in storage systems and point out their challenges. Then we show the mutually adversary impact of the low-priority background tasks and foreground I/O requests by experiments. Based on the experimental results and the access locality characteristics, we conclude the motivation of WorkOut.

2.1 Low-Priority Background Tasks

To improve the performance and reliability of storage systems, there are many background tasks, such as RAID reconstruction, RAID resynchronize, and disk scrubbing. *RAID reconstruction* recovers the failed data to the replacement disk when a disk fails. *RAID resynchronization* 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.

Recent studies of field data in large-scale storage systems indicate that partial or complete disk failure rates are significant [5], [11], [22], [38], [41]. Schroeder and Gibson [41] found that the 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 rates (AFRs) specified by the manufacturer's datasheet. Bairavasundaram et al. [5] observed that the probability of latent sector errors, which can lead to disk replacement, is 3.45 percent in their study. Those failure rates, combined with the continuously increasing number of disks in large-scale storage systems, raise concerns that in future storage systems, the recovery mode might become the common mode of operation [11] and the periodical disk scrubbing is also quite necessary [5]. Furthermore, studies show a significant amount of correlation in drive failures, indicating that, after one disk fails, another disk failure will likely occur soon [5], [11], [22].

In software RAID, unexpected system crash or software errors can make stripes in inconsistent states that can also introduce a window of vulnerability [26]. In this case, current software RAID systems always employ the resynchronization 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 [9].

In addition to the above traditional low-priority background tasks, other background tasks are also emerging, such as garbage collection in new storage devices (e.g., flash-based drives [1], [6]) to achieve wear-leveling and data migration [29], [54] to improve the performance or energy efficiency. Although the priority of these background tasks is lower than that of the foreground tasks, they are persistent and essential for the system operation, reliability, and power.

Due to their low-priority characteristics, the background tasks are always assigned only a few disk resources. However, many researchers argued that the background tasks cannot be postponed all the time. For reasons of reliability and performance, they should be completed early and effectively [39], [32]. These trends make the low-priority background tasks, such as RAID reconstruction, RAID resynchronization, and disk scrubbing, increasingly important and urgent in building high performance and reliable storage systems.

2.2 Mutually Adversary Impact of Low-Priority Background Tasks and Foreground I/O Requests

The requests of the online low-priority background tasks and the foreground user I/O requests compete for the restricted disk resources and adversely affect each other. User I/O requests increase the completing time for the background tasks, while the background tasks increase the user response time. In this section, we will illustrate this issue through experiments on the online RAID reconstruction.

Fig. 1 shows the reconstruction times and user response times of a five-disk RAID5 set with a stripe unit size of 64 KB in three cases: 1) offline reconstruction, 2) online

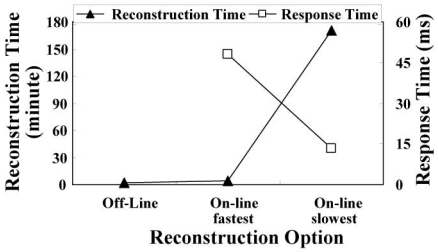

Fig. 1. RAID reconstruction and its performance impact.

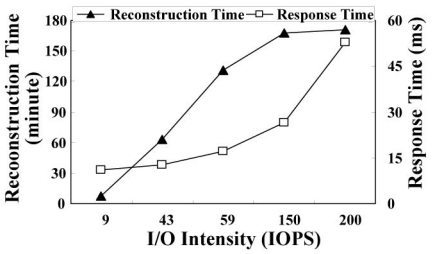

Fig. 2. I/O intensity impact on RAID reconstruction.

reconstruction at the highest speed (when RAID favors the reconstruction process), and 3) online reconstruction at the lowest speed (when RAID favors the user I/O requests). In this experiment, the capacity of each disk is limited to 10 GB. User I/O requests are generated by Iometer [21] with 20 percent sequential and 60/40 percent read/write requests of 8 KB each. As shown in Fig. 1, the user response time increases significantly along with the reconstruction speed three times ($3\times$) more than that during the normal period. The online reconstruction process at the lowest speed takes 70 times ($70\times$) longer than its offline counterpart.

The performance of reconstruction impacts both the reliability and availability of storage systems [13]. Storage system reliability is formally defined as the mean time to data loss (MTTDL) and increases with the decreasing mean time to repair (MTTR). Ironically, decreasing the MTTR (i.e., speeding up the reconstruction) by throttling the foreground user requests can lead Service-Level Agreement (SLA) violations, which are also perceived as reduced availability in many environments. Ideally, one would like to reduce both the reconstruction time and user response time in order to improve both the reliability and availability of RAID-structured storage systems.

Fig. 2 shows how user I/O intensity affects the performance of reconstruction. The experimental setup is the same as that in Fig. 1, except for the different I/O intensities. Moreover, the reconstruction process is set to yield to the user I/O requests (i.e., RAID favors the user I/O requests). From Fig. 2, we can see that both the reconstruction time and user response time increase with the *I/O Per Second* (IOPS) of user I/O requests. When increasing the IOPS from 9 to 200, it reaches its maximum of 200, the reconstruction time increases by a factor of 20.9 and the average user response time increases by a factor of 3.76.

Experiments on the other low-priority background tasks show the same trends as the RAID reconstruction. We believe that reducing the amount of user I/O requests directed to the degraded RAID set that is performing the background tasks is an effective approach to simultaneously reducing the completion time for the background tasks and alleviating the user performance degradation, thus improving both the reliability and availability. However, naively redirecting all requests to a surrogate RAID set might overload the surrogate RAID set and runs the risk that lots of work is wasted by redirecting requests that will never be accessed again. Our idea is, therefore, to exploit locality in the request stream and redirect only requests for popular data.

2.3 Workload Locality

Access locality is one of the key web workload characteristics [2], [7], [8]. Studies observe that 10 percent of files accessed on a web server approximately account for 90 percent of requests and 90 percent of bytes transferred [2]. Studies also find that 20-40 percent of files are accessed only once for web workloads [8].

Caches have been widely employed to improve the storage system performance by exploiting the access locality. While the storage cache is proven very effective in capturing the workload locality, it is so small in capacity compared with the typical storage device that it usually cannot capture all workload locality. So, the locality underneath the storage cache can still be effectively mined and utilized [28], [48]. For example, PRO [48] utilizes the workload locality at the block level, thus reducing the reconstruction time by up to 44.7 percent and the user response time by 3.6-23.9 percent simultaneously. Based on the study on C-Miner [28] that mines the block correlation below the storage cache, correlation-directed prefetching and data layout reduce the user response time by 12-25 percent.

2.4 Motivation

Low-priority background tasks become more and more important in face of such high partial and complete disk failures rates, the unexpected system crash and software errors, and the requirements of the low-power storage system. When I/O requests of the foreground tasks and background tasks are scheduled concurrently, the background requests have negative impact on the performance of the foreground tasks, and vice versa.

Based on the above experiments and observations, WorkOut exploits the access locality to reduce the foreground I/O intensity by redirecting all the write requests and popular read requests to a surrogate RAID set (idle and fast) in large-scale storage systems. Thus, the background tasks can complete much more effectively and quickly. Moreover, since the redirected requests are served by the surrogate RAID set, their performance can also be improved significantly. In such a way, WorkOut effectively boosts the availability of storage systems.

In this paper, we simply define the *popular data* in the WorkOut design as the data that have been read at least twice. Different from read requests, write requests can be served by any persistent storage device. Thus, WorkOut redirects all write requests to the surrogate RAID set.

3 WorkOut

In this section, we first outline the main principles guiding the design of WorkOut. Then, we present an architectural

overview of WorkOut, followed by a detail description of the data structures and algorithm. At the end of this section, we discuss the design choice and data consistency issues of WorkOut.

3.1 Design Principles

WorkOut focuses on outsourcing I/O workloads to boost the performance of the low-priority background tasks. It aims to achieve reliability, availability, flexibility, extendibility, and portability as follows:

Reliability. To reduce the window of vulnerability, and thus, improve the reliability, the low-priority background tasks should not be starved and their completion times must be significantly reduced. Since the foreground user I/O intensity severely affects the background tasks, WorkOut aims to reduce the the I/O intensity on the degraded RAID set by redirecting I/O requests away from it.

Availability. To avoid the user-perceived performance degradation and violation of SLAs, the user response time during the period when the background tasks are performing must be significantly reduced. WorkOut strives to achieve this goal by significantly reducing, if not eliminating, the contention between the user I/O requests and the background tasks, by outsourcing the I/O workloads to a surrogate RAID set.

Flexibility. Since modifying the organization of an existing RAID is inconvenience and expensive, such modification should be completely avoided, and instead, a separate surrogate RAID can be utilized judiciously and flexibly. In the WorkOut design, the surrogate RAID set can be a dedicated RAID1 set, a dedicated RAID5 set, or a live RAID set that uses the free space of another operational (live) RAID set. Using an RAID as the surrogate set ensures that the redirected write data are safe-guarded with redundancy, thus guaranteeing the consistency of the redirected write data. How to choose an appropriate surrogate RAID set is based on the requirements on overhead, performance, reliability, maintainability, and trade-offs between them.

Extendibility. Since I/O intensity impacts the performance of not only the reconstruction but also other low-priority background tasks, such as RAID resynchronization and disk scrubbing, the idea of WorkOut should be readily extendable to improve the performance of these background tasks.

Portability. Since the low-priority background tasks are typically driven by many different algorithms or scheduling methods, it is desirable for WorkOut to be easily incorporated into the latter for their optimizations.

3.2 WorkOut Architecture Overview

An architecture overview of WorkOut is shown in Fig. 3. In our design, WorkOut is an augmented module to the RAID software operating underneath the storage cache in a system with multiple RAID sets. WorkOut interacts with the modules of the low-priority background tasks, such as RAID reconstruction, RAID resynchronization, and disk scrubbing, but is implemented independently of them. WorkOut can be incorporated into any RAID software and also other low-priority background tasks.

As shown in Fig. 3, there are five key functional components in WorkOut. *Administration Interface* provides

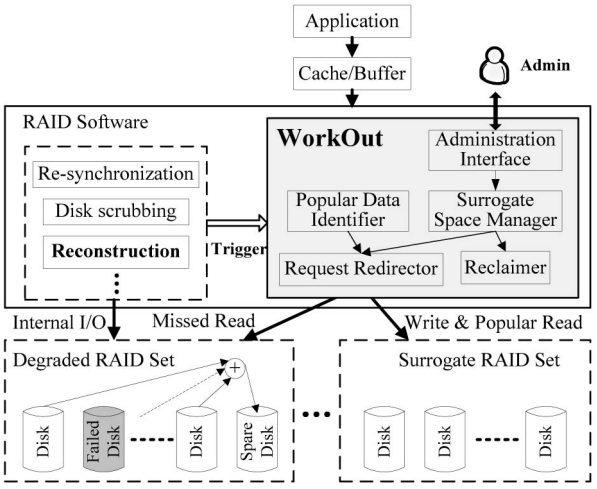

Fig. 3. An architecture overview of WorkOut.

an interface for system administrators to configure the WorkOut design options. *Popular Data Identifier* is responsible for identifying the popular read data. *Request Redirector* is responsible for redirecting all write requests and popular read requests to the surrogate RAID set, while *Reclaimer* is responsible for reclaiming all redirected write data back after the background task completes. *Surrogate Space Manager* allocates and manages a space on the surrogate RAID set for each degraded RAID set and controls the data layout of the redirected data in the allocated space.

WorkOut is automatically activated by the modules of the background tasks when their corresponding threads are initiated and deactivated when the reclaim process completes. In other words, WorkOut is active throughout the entire period when performing the background tasks and the reclaim process. Moreover, the reclaim thread is triggered by the modules of the background tasks when they complete.

In our WorkOut design, the idea of a dedicated surrogate RAID set is targeted at a typical data center where the surrogate RAID set can be shared by multiple degraded RAID sets with requirements on either a space-division or a time-division basis. The two types of RAID sets are not a one-to-one mapping. The device overhead is incurred only when the RAID set is performing the background tasks and typically amounts to a small fraction of the surrogate set capacity. For example, extensive experiments on our prototype implementation of WorkOut show that no more than four percent of the capacity of a four-disk surrogate RAID5 set is used.

Based on the preconfigured parameters, the Surrogate Space Manager allocates a disjoint space for each degraded RAID set that requests a surrogate RAID set. It prevents the redirected data from being overwritten by the redirected data from other degraded RAID sets that also request a surrogate RAID set. Noticeably, the space allocated to a certain degraded RAID set is not fixed but can be expanded. For example, the Surrogate Space Manager first allocates an estimated space required for a typical degraded RAID set and, if the allocated space is used up to a preset threshold (e.g., 90 percent), it will allocate some extra space to this RAID set. In this paper, we mainly consider the scenario where there is at most one degraded RAID set at any given

TABLE 1
Characteristic Comparisons of Three Optional Surrogate RAID Sets Used in WorkOut

Optional surrogate RAID set	*Device Overhead*	*Performance*	*Reliability*	*Maintainability*
A dedicated surrogate RAID1 set	medium	medium	high	simple
A dedicated surrogate RAID5 set	high	high	high	simple
A live surrogate RAID set	low	low	medium-high	complicated

time. Implementing and evaluating WorkOut in a large-scale storage system with multiple concurrent degraded RAID sets are work in process.

3.3 Data Structure and Algorithm

WorkOut depends on two key data structures to redirect requests and identify popular data, as follows:

- *D_Table* contains the log of all redirected data, including the following four important variables:
  - *D_Offset* indicates the offsets of the redirected data in the degraded RAID set.
  - *S_Offset* indicates the offsets of the redirected data in the surrogate RAID set.
  - *Length* indicates the length of the redirected data.
  - *D_Flag* indicates whether it is the redirected write data from the user application ($D\_Flag = true$) or the redirected read data from the degraded RAID set ($D\_Flag = false$).
- *R_LRU* is an LRU list that stores the information (*D_offset* and *Length*) of the most recent read requests. Based on R_LRU, popular read data can be identified and redirected to the surrogate RAID set.

WorkOut focuses on outsourcing user I/O requests when the degraded RAID is performing the background task concurrently, but does not modify the algorithm and scheduling of the background task. How to perform a background task itself remains the responsibility of the functional module of this task and depends on the specific algorithm and scheduling behind this task.

All write requests are redirected to the surrogate RAID set and recorded in D_Table. For each read request, D_Table is first checked to determine whether the data are in the surrogate RAID set. If the read request does not hit D_Table, it will be served by the degraded RAID set. Then, if it hits R_LRU, the read data are considered popular and redirected to the surrogate RAID set, and its information is recorded in D_Table. Otherwise, the read request will be served by the surrogate RAID set if the read data are already in it. In particular, if only a portion of the read data is in the surrogate RAID set, i.e., it partially hits D_Table, the read request will be split and served by the two RAID sets. In order to achieve a better performance, the redirected data are laid out sequentially like LFS [40] in the allocated space on the surrogate RAID set.

The redirected write data are only temporarily stored in the surrogate RAID set, and thus, should be reclaimed back after the background task completes. To ensure data consistency, the log of reclaimed data should be deleted from D_Table after the write succeeds. Since the redirected read data are already in the RAID set that has completed the background task, it need not be reclaimed as long as logs of such data are deleted from D_Table. In order not to affect the performance of the RAID system, the priority of the reclaim process is set to be the lowest, which will not affect the reliability of the redirected data as explained in Section 5.

During the reclaim period, all requests must be checked in D_Table to ensure data consistency. If a write request hits D_Table and its D_Flag is true, meaning that it will rewrite the data that are still in the surrogate RAID set, the corresponding log in D_Table must be deleted after writing the data to its correct location to prevent the new write data from being overwritten by the reclaimed data. On the other hand, if a read request hits D_Table and its D_Flag is true, meaning that the up-to-date targeted data have not been reclaimed, the read request should be served by the surrogate RAID set.

3.4 Design Choices

WorkOut can redirect data to different persistent configurations of storage devices as follows:

A dedicated surrogate RAID1 set. In this case, WorkOut stores the redirected data in *two mirroring disks*, namely, a dedicated surrogate RAID1 set. The advantage of this design option is its high reliability, simple space management, and moderate device overhead (i.e., two disks), while its disadvantage is obvious: relatively low performance gain due to the lack of I/O parallelism.

A dedicated surrogate RAID5 set. In favor of reliability and performance (access parallelism), a dedicated surrogate RAID5 set with *several disks* can be deployed to store the redirected data. The space management is simple, while the device overhead (e.g., four disks) is relatively high.

A live surrogate RAID set. WorkOut can utilize the free space of another live surrogate RAID set in a large-scale storage system consisting of multiple RAID sets and does not incur any additional device overhead that the first two design options cannot avoid. In this case, WorkOut gains high reliability owing to its redundancy, but requires complicated maintenance. Due to the contention between the redirected requests from the degraded RAID set and the native I/O requests targeted at the live surrogate RAID set, the performance in this case is lower than that in the former two design options.

The three design options are all feasible and can be made available for system administrators to choose from through the Administration Interface based on their characteristics and trade-offs, as summarized in Table 1. In this paper, the prototype implementation and performance evaluations are centered around the dedicated surrogate RAID5 set, although sample results from the other two design choices are also given to show the quantitative differences among them.

3.5 Data Consistency

Data consistency in WorkOut includes three aspects: 1) The key data structures should be safely stored until the reclaim process completes, 2) The redirected data must be reliably stored in the surrogate RAID set, and 3) The user read requests must fetch the up-to-date data.

First, since we must ensure never to lose the contents of D_Table during the entire period when WorkOut is activated, it is stored in an NVRAM to prevent data loss in the event of a power supply failure or a system crash. Fortunately, D_Table is, in general, very small (see Section 4.5), and thus, will not incur significant hardware cost. Moreover, since the performance of battery-backed RAM, a *de facto* standard form of NVRAM for storage controllers [10], [18], [19], is roughly the same as the main memory, the write penalty due to D_Table updates can be negligible.

Second, in order to avoid data loss caused by a disk failure in the surrogate RAID set, all redirected write data in the surrogate RAID set should be protected by a redundancy scheme. To simplify the design and implementation, the redirected read data are stored in the same manner as the redirected write data. If a disk failure in the surrogate RAID set occurs, data will no longer be redirected to the surrogate RAID set and the redirected write data should be 1) reclaimed back to the degraded RAID set or 2) redirected to another surrogate RAID set if possible. Our prototype implementation adopts the first option, i.e., these redirected write data should be reclaimed back to the degraded RAID set. We will analyze the reliability of WorkOut in Section 5.

Finally, since the up-to-date data for a read request can be stored either in the degraded RAID set or in the surrogate RAID set if the data are popular or modified by a previous write request, every read request is first checked in D_Table to determine whether it should be served by the degraded RAID set, the surrogate RAID set, or both (if the data are partially modified) to keep the fetched data always up-to-date, until all the redirected write data are reclaimed.

4 Performance Evaluations

In this section, we first describe the prototype implementation of WorkOut, followed by an introduction to the experimental setup and the methodology used in this performance study. Then, through extensive trace-driven and benchmark-driven experiments, we evaluate the performance of WorkOut on two case studies of low-priority background tasks: RAID reconstruction and RAID resynchronization. Finally, we analyze the overhead and complexity of WorkOut.

4.1 Experimental Setup and Methodology

We have implemented WorkOut by embedding it into the Linux Software RAID (MD) as a built-in module. In order not to impact the normal performance of RAID, WorkOut is activated only when the low-priority background task is initiated. WorkOut tracks the user I/O requests in the *make_request* function and issues them to the degraded RAID set or the surrogate RAID set based on the request type and D_Table.

We conduct our performance evaluation of WorkOut on a platform of server-class hardware with an Intel Xeon 3.0 GHz processor and 1 GB DDR memory. We use two Highpoint RocketRAID 2220 SATA cards to house 15 Seagate ST3250310AS SATA disks. The rotational speed of these disks is 7,200 RPM, with a sustained transfer rate of 78 MB/s, and the individual disk capacity is 250 GB. A separate IDE disk is used to house the operating system (Fedora Core 4 Linux, kernel version 2.6.11) and other software (MD and mdadm). For the footprint of the workloads, we limit the capacity of each disk to 10 GB in the experiments. For simplicity, we use the main memory to substitute a battery-backed RAM.

TABLE 2
The Trace Characteristics

Trace	*Characteristics*		
	Write Ratio	*IOPS*	*Average Request Size(KB)*
Fin1	67.18%	69	6.2
Fin2	17.61%	125	2.2
Web	0%	113	15.1

Generally speaking, there are two models for trace replay: open-loop and closed-loop [31], [42]. The former has the potential to overestimate the user response time measure since the I/O arrival rate is independent of the underlying system, and thus, can cause the request queue to grow rapidly when the system load is high. The opposite is true for the closed systems as the I/O arrival rate is dictated by the processing speed of the underlying system and the request queue is generally limited in length. In this paper, we use an open-loop model (trace replay with RAIDmeter [48]) and a closed-loop model (TPC-C-like benchmark [50]) to evaluate the performance of WorkOut.

The traces used in our experiments are obtained from the Storage Performance Council [51], [46]. The Fin1 and Fin2 traces were collected from the OLTP applications running at a large financial institution and the WebSearch2 trace (or Web) was collected from a machine running a web search engine. They represent different access patterns in terms of write ratio, IOPS and average request size, as shown in Table 2. The write ratio of the Fin1 trace is the highest, followed by the Fin2 trace. The read-dominated web trace exhibits the strongest locality in its access pattern. Since the request rate in the web trace is too high to be sustained by our degraded RAID set, we only use one part of it that is attributed to device zero, while the part due to devices one and two is ignored.

Since the three traces have very limited footprints, that is, the user I/O requests are congregated on a small part of the degraded RAID set (e.g., less than 10 percent of an eight-disk RAID5 set for the Fin1 trace), their replays may not realistically represent a typical scenario where user requests may be spread out over the entire disk address space. To fully and evenly cover the address space of the degraded RAID set, we scale up the address coverage of the I/O requests by multiplying the address of each request with a constant scaling factor without changing its size. While the main adverse impact of this trace scaling is likely to be on those requests that are originally sequential but can subsequently become nonsequential after scaling, the percentage of such sequential requests in the three traces is relatively small at less than four percent [53]. Thus, we

TABLE 3
The Reconstruction Time Results

Traces	Reconstruction Time (second)						
	Off-line	PR	WorkOut+PR	speedup	PRO	WorkOut+PRO	speedup
Fin1	136.4	1121.8	203.1	5.52	1109.6	188.3	5.89
Fin2		745.2	453.3	1.64	705.8	431.2	1.64
Web		9935.6	7623.2	1.30	9888.3	7851.4	1.26

TABLE 4
The Average User Response Time Results

Traces	Average User Response Time (millisecond)							
	Normal	Degraded	PR	WorkOut+PR	speedup	PRO	WorkOut+PRO	speedup
Fin1	7.9	9.5	12.7	4.4	2.87	9.8	4.6	2.15
Fin2	8.1	13.4	25.8	9.7	2.66	23.0	10.2	2.25
Web	18.5	27.0	38.6	28.3	1.36	35.6	29.1	1.22

believe that the adverse impact is rather limited in this study and far outweighed by the benefits.

We also find that the observed trends are similar for the original and scale trace, suggesting that neither is likely to generate noticeably different conclusions for the study. Nevertheless, we choose to present the results of the latter for the aforementioned reasons.

The trace replay tool is RAIDmeter [48], [49] that replays traces at the block level and evaluates the user response time of the storage device. The performance of the low-priority background tasks is evaluated in terms of the completion time and average user response time.

The TPC-C-like benchmark is implemented with TPCC-UVA [37] and the Postgres database. It generates mixed transactions based on the TPC-C specification [50]. Twenty warehouses are built on the Postgres database with the ext3 file system on the degraded RAID set. Transactions, such as PAYMENT, NEW_ORDER, and DELIVERY, generate read and write requests. To evaluate the WorkOut performance, we compare the transaction rates (*transactions per minute*) that are generated at the end of the benchmark execution.

4.2 Case Study: RAID Reconstruction

In this section, we conduct experiments to evaluate how WorkOut boosts the performance of the online RAID reconstruction. First, we describe the configuration of the RAID system. Then, we present the trace-driven experimental results of WorkOut and investigate some sensitivity factors of WorkOut, along with a few experiments that study and compare different types of surrogate RAID sets. In addition to the trace-driven experiments, we conduct and report results of the benchmark-driven experiments to more thoroughly evaluate WorkOut.

By setting the reconstruction bandwidth range, MD assigns different disk bandwidth to serve user I/O requests and reconstruction requests and ensures that the reconstruction speed is confined within the set range (i.e., between the minimum and maximum reconstruction bandwidth). For example, if the reconstruction bandwidth range is set to be the default of 1-200 MB/s, MD will favor the user I/O requests while ensuring that the reconstruction speed is at least 1 MB/s. Under heavy I/O workloads, MD will keep the reconstruction speed at approximately 1 MB/s, but allows it to be much higher than 1 MB/s when the I/O intensity is low. At one extreme when there is no user I/O, the reconstruction speed will be roughly equal to the disk transfer rate (e.g., 78 MB/s in our prototype system). Equivalently, the *minimum reconstruction bandwidth* of X MB/s (e.g., 1, 10, and 100 MB/s) refers to a reconstruction range of X-200 MB/s in MD. When the minimum reconstruction bandwidth is set to 100 MB/s, which is not achievable for most disks, MD utilizes any disk bandwidth available for the reconstruction process.

To better examine the WorkOut performance on existing RAID reconstruction algorithms, we incorporate WorkOut into MD's default reconstruction algorithm PR, PRO-powered PR (PRO for short), and JOR-powered PR (JOR for short). Pipeline Reconstruction (PR) [27] utilizes the sequential property of track retrievals to pipeline the reading and writing processes. Popularity-based multi-threaded Reconstruction Optimization (PRO) [48], [49] allows the reconstruction process to rebuild the frequently accessed areas prior to other areas. JOurnal-guided Reconstruction (JOR) [56] only recovers the failed data on the used stripes by monitoring the storage space utilization status at the block level.

4.2.1 Trace-Driven Evaluations

We first conduct experiments on an eight-disk RAID5 set with a stripe unit size of 64 KB while running PR, PRO, and WorkOut-powered PR and PRO, respectively. Tables 3 and 4, respectively, show the reconstruction times and average user response times under the minimum reconstruction bandwidth of 1 MB/s, driven by the three traces. We configure a four-disk dedicated RAID5 set with a stripe unit size of 64 KB as the surrogate RAID set.

From Table 3, one can see that WorkOut speeds up the reconstruction time by a factor of up to 5.52, 1.64, and 1.30 for the Fin1, Fin2, and web traces, respectively. The significant improvement achieved on the Fin1 trace is due to the fact that 84 percent of requests (69 percent of writes plus 15 percent of reads) are redirected away from the degraded RAID set (see Fig. 4), which enables the speed of the online reconstruction to approach that of the offline counterpart. In our experiments, the offline reconstruction time is 136.4 seconds for PR on the same platform. Moreover, WorkOut outsources 36 and 34 percent of the

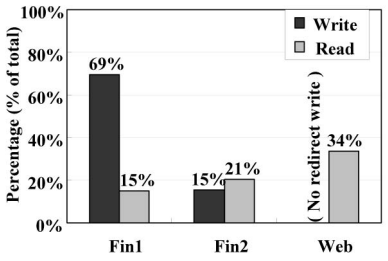

Fig. 4. Percentage of redirected requests for WorkOut.

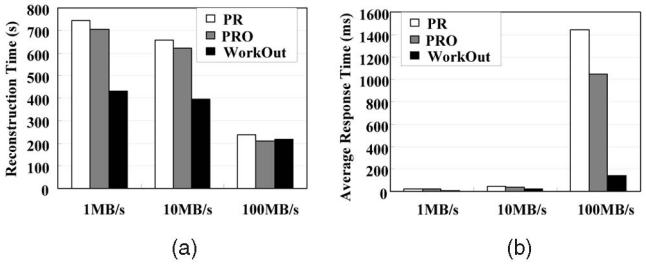

Fig. 5. Comparisons of reconstruction times and average user response times with respect to different minimum reconstruction bandwidth. (a) Reconstruction time. (b) Average user response time.

user I/O requests away from the degraded RAID set for the Fin2 and web traces, which is much fewer than that for the Fin1 trace, thus reducing the reconstruction time accordingly.

Table 4 shows that, compared with PR, WorkOut speeds up the average user response time by a factor of up to 2.87, 2.66, and 1.36 for the Fin1, Fin2, and web traces, respectively. For Fin1 and Fin2, the average user response times during reconstruction under WorkOut are even better than that in the normal or degraded period. The reasons why WorkOut achieves significant improvement on user response times are threefold. First, a significant amount of requests are redirected away from the degraded RAID set, as shown in Fig. 4. Their response times are no longer affected by the reconstruction process. Second, the redirected data are laid out sequentially in the surrogate RAID set, thus further speeding up the user response time. Third, since many requests are outsourced, the I/O queue on the degraded RAID set is shortened accordingly, thus reducing the response times of the remaining I/O requests served by the degraded RAID set. Therefore, the average user response time with WorkOut is significantly lower than that without WorkOut, especially for the Fin1 trace.

Tables 3 and 4 show that WorkOut-powered PRO performs similarly to WorkOut-powered PR. The reason is that WorkOut redirects all write requests and popular read requests to the surrogate RAID set, thus reducing the degree of popularity of I/O workloads retained on the degraded RAID set that can be exploited by PRO. Based on this observation, in the following experiments, we only compare WorkOut-powered PR (*short for WorkOut*) with PR and PRO.

4.2.2 Sensitivity Study

WorkOut's performance is likely influenced by several important factors, including the available reconstruction bandwidth, the stripe unit size, the size of the degraded RAID set, and the RAID level. Due to lack of space, we limit our study of these parameters to the Fin2 trace. Other traces show similar trends as the Fin2 trace.

Reconstruction bandwidth. To evaluate how the minimum reconstruction bandwidth affects the reconstruction performance, we conduct experiments that measure the reconstruction times and average user response times as a function of different minimum reconstruction bandwidth 1, 10, and 100 MB/s, respectively. Fig. 5 plots the experimental results on an eight-disk RAID5 set with a stripe unit size of 64 KB.

Fig. 5a shows that WorkOut speeds up the reconstruction time more significantly with a lower minimum reconstruction bandwidth than with a higher one. The reason is that the reconstruction process already exploits all available disk bandwidth when the reconstruction bandwidth is higher, thus leaving very small room for the reconstruction time to be further improved.

From Fig. 5b, in contrast, the user response time increases rapidly with the increasing minimum reconstruction bandwidth for both PR and PRO, but much more slowly for WorkOut. WorkOut speeds up the user response time significantly, by a factor of up to 10.2 and 7.38 over PR and PRO, respectively, when the minimum reconstruction bandwidth is set to 100 MB/s. From this viewpoint, the user response time with WorkOut is much less sensitive to the minimum reconstruction bandwidth than that without WorkOut. In other words, if the reconstruction bandwidth is set very high or the storage system is reliability-oriented, that is, the reconstruction process is given more bandwidth to favor the system reliability, the user response time improvement by WorkOut will be much more significant. Moreover, the user response time during reconstruction for PR and PRO is so long that it will likely violate SLA, and thus, become unacceptable to the end users.

Stripe unit size. To examine the impact of the stripe unit size, we conduct experiments on an eight-disk RAID5 set with stripe unit sizes of 16 and 64 KB, respectively. The experimental results show that WorkOut outperforms PR and PRO in the reconstruction time as well as the average user response time for both stripe unit sizes. Moreover, the reconstruction times and average user response times of WorkOut are almost unchanged, suggesting that WorkOut is not sensitive to the stripe unit size.

Number of disks. To examine the sensitivity of WorkOut to the number of disks of the degraded RAID set, we conduct experiments on RAID5 sets consisting of different numbers of disks (5, 8, and 11) with a stripe unit size of 64 KB under the minimum reconstruction bandwidth of 1 MB/s. Fig. 6 shows the experimental results for PR, PRO, and WorkOut.

Figs. 6a and 6b show that for all three approaches, the reconstruction time increases and the user response time decreases for higher number of disks in the degraded RAID set. The reason is that more disks in an RAID set imply not only a larger RAID group size, and thus, more disk read operations to reconstruct a failed drive, but also higher parallelism for the I/O process. However, WorkOut is less sensitive to the number of disks than PR and PRO.

RAID level. To evaluate WorkOut with different RAID levels, we conduct experiments on a four-disk RAID10 set and an eight-disk RAID6 set with the same stripe unit

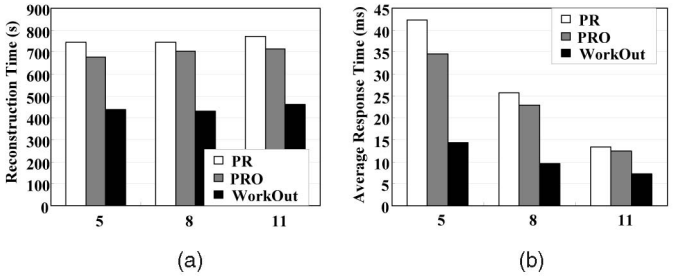

Fig. 6. Comparisons of reconstruction times and average user response times with respect to the number of disks. (a) Reconstruction time. (b) Average user response time.

size of 64 KB under the minimum reconstruction bandwidth of 1 MB/s. In the RAID6 experiments, we measure the reconstruction performance when two disks fail concurrently.

From Fig. 7, one can see that WorkOut speeds up both the reconstruction times and average user response times for the two sets. The difference in the amount of the performance improvement seen for RAID10 and RAID6 is caused by the different user I/O intensities, since the RAID10 and RAID6 sets have different numbers of disks. The user I/O intensity on individual disks in the RAID10 set is higher than that in the RAID6 set, thus leading to longer reconstruction times.

On the other hand, since each read request to the failed disks in an RAID6 set must wait for its data to be rebuilt on the fly, the user response time is severely affected for PR, while this performance degradation is significantly lower under WorkOut due to its external I/O outsourcing. For the RAID10 set, however, the situation is quite different. Since the read data can be directly returned from the surviving disks, Workout provides smaller improvements in the user response time for RAID10 than for RAID6.

4.2.3 Different Design Choices

All experiments reported up to this point in this paper adopt a dedicated surrogate RAID5 set. To examine the impact of different types of surrogate RAID set on the WorkOut performance, we also conduct experiments with a dedicated surrogate RAID1 set (two mirroring disks) and a live surrogate RAID set (replaying the Fin1 trace on a four-disk RAID5 set). Similar to the experiments conducted in the PARAID [54] and write offloading [33] studies, we reserve the 10 percent portion of storage space at the end of the live RAID5 set to the store data redirected by WorkOut. The degraded RAID set is an eight-disk RAID5 set with a stripe unit size of 64 KB and under the minimum reconstruction bandwidth of 1 MB/s.

The experimental results show that the reconstruction times achieved by WorkOut are almost the same for the three types of surrogate RAID set and outperform PR as expected, as shown in Table 3, as WorkOut outsources the same amount of requests during reconstruction. The user response times are somewhat different, as shown in Fig. 8. The dedicated surrogate RAID5 set results in the best user response times.

From Fig. 8, one can see that the dedicated surrogate RAID sets (both RAID1 and RAID5) outperform the live surrogate RAID set in the user response time. The reason is the contention between the native I/O requests and the redirected requests in the live surrogate RAID set. Serving the native I/O requests not only increases the overload on the surrogate RAID set, compared with the dedicated surrogate RAID set, but also destroys some of the sequentiality in the LFS style writes. The redirected requests also increase the overall I/O intensity on the live surrogate RAID set and affect its performance. Our experimental results show that the performance impact on the live surrogate RAID set is 43.9, 23.6, and 36.8 percent, on average, when the degraded RAID set replays the Fin1, Fin2, and web traces, respectively. The experimental results are consistent with the comparisons in Table 1.

4.2.4 Benchmark-Driven Evaluations

In addition to the trace-driven experiments, we also conduct experiments on an eight-disk RAID5 set with a stripe unit size of 64 KB under the minimum reconstruction bandwidth of 1 MB/s, driven by a TPC-C-like benchmark.

From Fig. 9a, one can see that PRO performs almost the same as PR due to the random access characteristics of the TPC-C-like benchmark. Since WorkOut outsources all write requests that are generated by the transactions, both

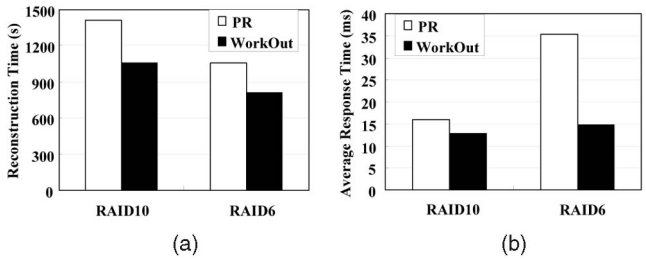

Fig. 7. Comparisons of reconstruction times and average user response times with respect to different RAID levels. (a) Reconstruction time. (b) Average user response time.

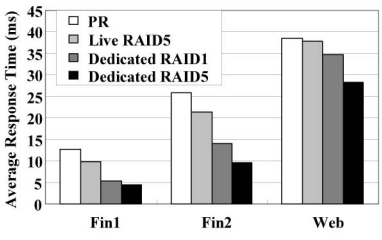

Fig. 8. A comparison of average user response times for different types of surrogate RAID set.

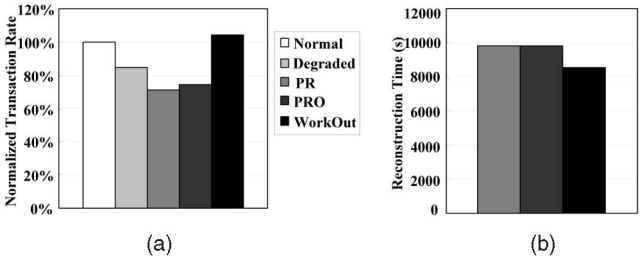

Fig. 9. Comparisons of reconstruction times and transaction rates driven by the TPC-C-like benchmark. (a) Transaction rate. (b) Reconstruction time.

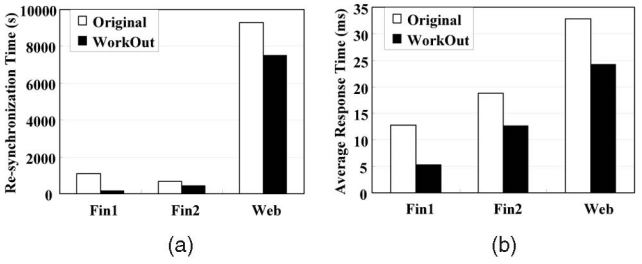

Fig. 10. Comparisons of resynchronization times and average user response times during resynchronization. (a) Resynchronization time. (b) Average user response time.

the degraded RAID set and surrogate RAID set serve the benchmark application, thus increasing the transaction rate. WorkOut outperforms PR and PRO in terms of transaction rate, with an improvement of 46.6 and 36.9 percent, respectively. It also outperforms the original system in the normal mode (the normalized baseline) and the degraded mode, with an improvement of 4.0 and 22.6 percent, respectively.

On the other hand, since the TPC-C-like benchmark is highly I/O intensive, all disks in the RAID set are driven to saturation; thus, the reconstruction speed is kept at around its minimum allowable bandwidth of 1 MB/s for PR and PRO. As shown in Fig. 9b, the reconstruction times for PR and PRO are similar at 9,835 and 9,815 seconds, respectively, while that for WorkOut is 8,526 seconds, with approximately 15 percent improvement over PR and PRO. WorkOut gains much less in the reconstruction time with the benchmark-driven experiments than with the trace-driven experiments. The main reason lies in the fact that the very high I/O intensity of the benchmark application constantly pushes the RAID set to operate at or close to its saturation point, leaving very little disk bandwidth for the reconstruction process even with some of the transaction requests being outsourced to the surrogate RAID set.

4.3 Case Study: RAID Resynchronization

To demonstrate how WorkOut optimizes other low-priority background tasks, such as RAID resynchronization, we conduct experiments on an eight-disk RAID5 set with a stripe unit size of 64 KB under the minimum resynchronization bandwidth of 1 MB/s, driven by the three traces. We configure a dedicated four-disk RAID5 set with a stripe unit size of 64 KB as the surrogate RAID set. The experimental results of the resynchronization times and average user response times during resynchronization are shown in Figs. 10a and 10b, respectively.

Although the resynchronization process performs somewhat differently from the reconstruction process, its requests also compete for the disk resources with the user I/O requests. By redirecting a significant amount of user I/O requests away from the RAID set during resynchronization, WorkOut can reduce both the resynchronization times and user response times. The results are very similar to that in the reconstruction experiments, so are the reasons behind them.

4.4 Portability Evaluation

In this paper, we also incorporate WorkOut into JOR [56] to evaluate the portability of WorkOut. One disadvantage of JOR is that it does not consider the negative impact of the user I/O requests on the reconstruction process, while WorkOut should not waste much time in recovering the unused space. Since WorkOut and JOR do not modify the data layout or the reconstruction workflow of RAID and they are orthogonal, they can optimize the reconstruction process jointly.

We conduct experiments to compare the reconstruction times of PR, JOR-powered PR, WorkOut-powered PR, and WorkOut-powered JOR. The experimental setup is the same as that described in Section 4.2.1. We set different percentage of unused storage space in the best case, where the unused space is compacted into a single portion of the disk array [56]. From Fig. 11, we can see that in spite of PR or JOR-powered PR, WorkOut can continue to reduce the reconstruction time under a certain percentage of the unused storage space.

WorkOut and JOR optimize the RAID reconstruction process from two different angles. Thus, directly comparing the performances of WorkOut and JOR makes no sense. However, the results demonstrate the good interoperability of WorkOut and JOR. When they optimize the reconstruction process jointly, the performance can be optimized. Moreover, the portability of WorkOut is also demonstrated.

4.5 Overhead and Complexity Analysis

4.5.1 *Memory Overhead.*

To prevent data loss, WorkOut uses nonvolatile memory to store D_Table, thus incurring extra memory overhead. For example, in the RAID reconstruction experiments, the amount of memory consumed is largest when the minimum reconstruction bandwidth is set to 1 MB/s, since in this case, the reconstruction time is the longest and the amount of the redirected data is the largest. In the above experiments on the RAID5 set with individual disk capacity of 10 GB, the maximum memory overheads are 0.14, 0.62, and 1.69 MB for the Fin1, Fin2, and web traces, respectively. However, the memory overhead incurred by WorkOut is only temporary and will be removed after the reclaim process completes. With the rapid increase in the memory size and decrease in the cost of nonvolatile memories, this memory overhead is arguably reasonable and acceptable to the end users.

4.5.2 *Device Overhead*

WorkOut is designed for use in a large-scale storage system consisting of many RAID sets that share one surrogate RAID composed of the spare disks. In such an environment, the device overhead introduced by WorkOut is small, given that a single surrogate RAID can be shared by many production RAID sets. Nevertheless, for a small-scale

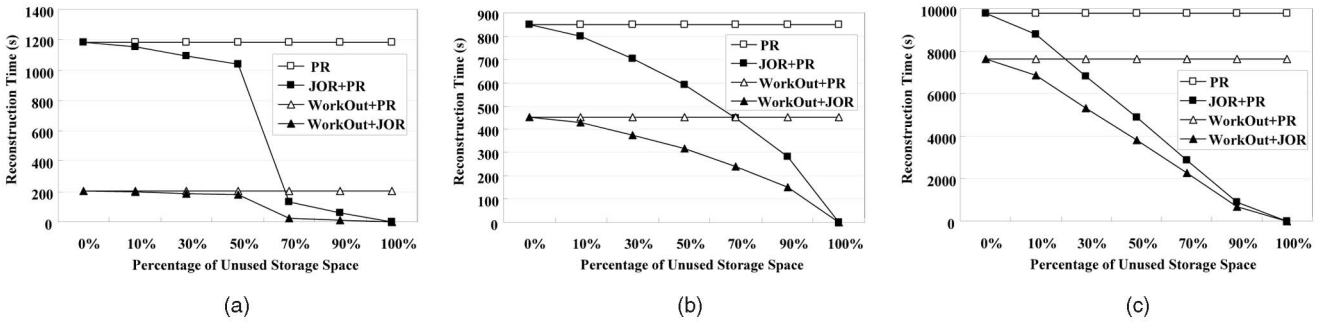

Fig. 11. Comparisons of reconstruction times of WorkOut and JOR. (a) Financial1. (b) Financial2. (c) Websearch.

storage system composed of only one or two RAID sets with few hot spare disks, the device overhead of a dedicated surrogate RAID set in WorkOut cannot be ignored. In this case, to be cost-effective, we recommend the use of a dedicated surrogate RAID1 set instead of a dedicated surrogate RAID5 set, since the device overhead of the former (i.e., two disks) is lower than that of the latter (e.g., four disks in our experiments).

To quantify the cost-effectiveness of WorkOut in this resource-restricted environment, we conduct experiments and compare the performance of WorkOut (eight-disk data RAID5 set plus four-disk surrogate RAID5 set) with that of PR (12-disk data RAID5 set), i.e., we use the same number of disks in both systems. Experiments are run under the minimum reconstruction bandwidth of 1 MB/s and driven by the Fin2 trace. The results show that WorkOut speeds up the reconstruction time of PR significantly, by a factor of 1.66. The average user response time during reconstruction achieved by WorkOut is 16.5 percent shorter than that achieved by PR, while the average user response time during the normal period in the eight-disk RAID5 set is 20.1 percent longer than that in the 12-disk RAID5 set due to the reduced access parallelism of the former. In summary, we can conclude that WorkOut is cost-effective in both the large-scale and small-scale storage systems.

4.5.3 Power Overhead

An additional surrogate set of WorkOut has also increased the power consumption of storage system. In a large-scale storage system, the power overhead of the surrogate set that is composed of several disks can be negligible relative to the huge power consumption of the whole system. In a small-scale storage system, however, this power overhead must be taken into account. However, since the surrogate set only runs in the time when WorkOut is activated and some degraded RAID sets request to use it, it operates in the energy-saving mode most of time, resulting in relatively low power consumption. Moreover, compared with the significant improvement on the reliability and availability of the storage system, we believe that the small power overhead is a reasonable trade-off for WorkOut.

4.5.4 Implementation Complexity

WorkOut contains 780 lines of added or modified code to the source code of the Linux software RAID (MD), with most lines of code added to md.c and raidx.c, while 37 lines of data structure code added to md_k.h and raidx.h. Since most of the added code is independent of the underlying RAID layout, they are easy to be shared by different RAID levels. Moreover, the added code is independent of the reconstruction module, so it is easy to adapt the code for use with other low-priority background tasks. All that needs to be done is modifying the corresponding flag that triggers WorkOut. Due to the independent implementation of the WorkOut module, it is portable to other software RAID implementations in other operating systems.

5 Reliability Analysis

In this section, we adopt the MTTDL metric to estimate the reliability of WorkOut taking into account the reconstruction scenario. We assume that thedisk failures are independent events following an exponential distribution of rate μ, and the repairs follow an exponential distribution of rate ν. For simplicity, we do not consider the latent sector error in the system model. Although some of these assumptions are not true for the real storage systems, they are necessary for using the stochastic models with finite numbers of states [36].

According to the conclusion about the reliability of RAID5 [60], MTTDL of an eight-disk RAID5 set achieved by PR and PRO is

$$MTTDL_{RAID5-8} = \frac{15\mu + \nu}{56\mu^2}. \quad (1)$$

Fig. 12 shows the state transition diagram for a WorkOut-enabled storage system configuration consisting of an eight-disk data RAID5 set and a four-disk surrogate RAID5 set. Note that by design, WorkOut always reclaims the redirected write data from the surrogate RAID set upon a surrogate disk failure. Once the reclaim process is completed, there is no more valid data of the degraded RAID set on the surrogate RAID set. Therefore, there is no need to

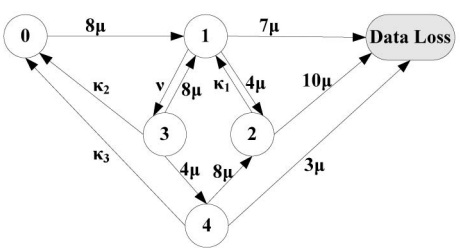

Fig. 12. State transition diagram for a WorkOut-enabled storage system configuration consisting of an eight-disk RAID5 set and a four-disk surrogate RAID5 set.

reconstruct data on the failed disk of the surrogate RAID set. This means that the degraded surrogate RAID set can be recovered by simply replacing the failed disk with a new one, resulting in a newly recovered operational four-disk surrogate RAID5 set ready to be used by the eight-disk degraded data RAID5 set. As a result, the state transition diagram only shows the reclaim process but not the reconstruction process of the four-disk surrogate RAID5 set.

State <0> represents the normal state of the system when its eight data disks are all operational. A failure of any of the eight data disks would bring the system to state <1> and a subsequent failure of any of the remaining seven data disks would result in data loss. A failure of any of the four surrogate disks in state <0> does not affect the system reliability of the eight-disk RAID5 set as long as the redirected write data on the former are reclaimed back to the latter, and thus, it is omitted from the state transition diagram. In state <1>, a failure of any of the four surrogate disks would bring the system to state <2>. A second failure in either the eight-disk data RAID5 set (one out of seven) or the four-disk surrogate RAID5 set (one out of three) in state <2> would result in data loss.

In state <2>, WorkOut reclaims the redirected write data back to the eight-disk data RAID set, which brings the system back to state <1> and follows an exponential distribution of rate κ_1. This transition implicitly assumes that while the redirected write data are being reclaimed from the surrogate set to the data set, the reconstruction process on the latter is temporarily suspended. This simplifying assumption is justifiable and will not affect the result noticeably since the reclaim time is much shorter than the reconstruction time on the eight-disk data RAID set.

Finishing the reconstruction process of the eight-disk data RAID5 set would bring the system from state <1> to state <3>, where the redirected write data have not been reclaimed. Then finishing the reclaim process would bring it back to state <0>, which follows an exponential distribution of rate κ_2. In state <3>, a failure of any of the eight data disks would bring the system to state <1>, and a failure of any of the four surrogate disks would bring the system to state <4>, where the redirected write data are not protected by redundancy. In state <4>, WorkOut also reclaims the redirected write data back to the eight-disk data RAID set, which brings the system back to state <0> and follows an exponential distribution of rate κ_3. In state <4>, a failure of any of the eight data disks would bring the system to state <2> and a second-disk failure in the four-disk surrogate RAID5 set would result in data loss.

Since κ_1, κ_2, and κ_3 all represent the rate at which the redirected write data are reclaimed, it is reasonable to assume that they are equal to a fixed reclaim rate κ, since the amount of the redirected write data should be roughly the same and the rate of transferring this data should also be the same under reasonable circumstances for all the three cases.

The Kolmogorov system of differential equations describing the behavior of WorkOut is expressed in (2):

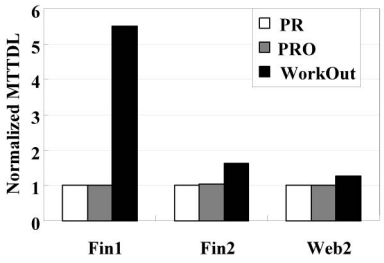

Fig. 13. Comparisons of the mean times to data loss. *Note that the normalized baselines are the MTTDLs achieved by PR driven by the three traces, respectively.*

$$\begin{cases} \frac{dp_0(t)}{dt} = -8\mu p_0(t) + \kappa_2 p_3(t) + \kappa_3 p_4(t), \\ \frac{dp_1(t)}{dt} = -(11\mu + \nu)p_1(t) + 8\mu p_0(t) + 8\mu p_3(t) + \kappa_1 p_2(t), \\ \frac{dp_2(t)}{dt} = -(10\mu + \kappa_1)p_2(t) + 4\mu p_1(t) + 8\mu p_4(t), \\ \frac{dp_3(t)}{dt} = -(12\mu + \kappa_2)p_3(t) + \nu p_1(t), \\ \frac{dp_4(t)}{dt} = -(11\mu + \kappa_3)p_4(t) + 4\mu p_3(t), \end{cases} \tag{2}$$

where $p_i(t)$ is the probability that the disk array is in state <i> with the initial condition $p_0(0) = 1$ and $p_i(0) = 0$ for $i \neq 0$.

The Laplace transformation of (2) is

$$\begin{cases} sp_0^*(s) - 1 = -8\mu p_0^*(s) + \kappa_2 p_3^*(s) + \kappa_3 p_4^*(s), \\ sp_1^*(s) = -(11\mu + \nu)p_1^*(s) + 8\mu p_0^*(s) + 8\mu p_3^*(s) + \kappa_1 p_2^*(s), \\ sp_2^*(s) = -(10\mu + \kappa_1)p_2^*(s) + 4\mu p_1^*(s) + 8\mu p_4^*(s), \\ sp_3^*(s) = -(12\mu + \kappa_2)p_3^*(s) + \nu p_1^*(s)), \\ sp_4^*(s) = -(11\mu + \kappa_3)p_4^*(s) + 4\mu p_3^*(s). \end{cases} \tag{3}$$

Observing that the MTTDL of the disk array is given by

$$MTTDL = \sum_i p_i^*(0). \tag{4}$$

Using (4), we solve the disk array of Laplace transformation for $s = 0$ and use (3) to compute MTTDL of WorkOut in Fig. 12:

$$MTTDL_{WorkOut} = \frac{\mu(89\mu + 15\kappa)(12\mu + \kappa) + \nu(69\mu^2 + 15\mu\kappa + \kappa^2)}{8\mu^2(396\mu^2 + 117\mu\kappa + 7\kappa^2 + 12\mu\nu)}. \tag{5}$$

Fig. 13 plots comparisons of the MTTDLs achieved by PR, PRO, and WorkOut, which are normalized to the MTTDLs achieved by PR driven by the three traces, respectively. The disk failure rate μ is assumed to be one failure every 100,000 hours, which is a conservative estimate to the values quoted by disk manufactures. The disk repair times, when the individual disk capacity is 250 GB, are 25 times the results listed in Table 3, where the capacity of each disk is limited to 10 GB. κ is assumed to be equal to the corresponding ν, which is actually overestimated. From Fig. 13, one can see that WorkOut increases the MTTDL and improves reliability with the decreasing the MTTR, especially for the write-intensive trace (i.e., Fin1). Moreover, if we alter κ to be several times smaller or larger than ν, the black bar remains almost unchanged, suggesting

TABLE 5
A Comparison of Systems Related to WorkOut

Scheme	*Purpose*	*Environment*	*Data Migration*		
			What	*When*	*Where*
Write off-loading [33]	Energy efficiency	Enterprise storage systems	Write data	Write-dominated period	From spin-down volumes to spin-up volumes
Everest [34]	Performance	Data center	Write data	Peak load period	From overloaded volumes to lightly loaded volumes
PARAID [54]	Energy efficiency	Single RAID set	All data	Gear shifting	From spin-down disks to active disks
Data reallocation [3]	Performance	Multidisk systems	All data	Dynamic to workloads	From busy disk/volumes to less busy ones
Cuckoo [25]	Performance	Clustering servers	Frequently-read, rarely-updated files	N/A	From busy servers onto others
User-centric migration [23]	Performance	Networked storage systems	Data that is currently read or written	Migration on access	From busy devices to the least busy ones
WorkOut	Reliability & performance	Systems with multiple RAID sets	Write data and popular read data	Redirect on demand	From a degraded RAID set to a surrogate RAID set

that the reclaim time does not affect the reliability of the RAID system, and thus, can be excluded from the reconstruction time.

6 Related Work

6.1 Low-Priority Background Task Scheduling

RAID reconstruction algorithms, such as DOR [14], PR [27], PRO [48], and others [4], [15], [16], [17], [24], [55], [58], [59], have been extensively studied. However, they mostly focus on improving the performance in the context of a single drive, e.g., by optimizing the reconstruction workflow [14], [27] or the reconstruction sequence [4], [48] *in a single RAID set*. They try to achieve a trade-off between the reconstruction bandwidth and the I/O serving bandwidth to satisfy the requirements of the end users. By utilizing the file system's semantic knowledge, the live-block recovery method in D-GRAID [45] only reconstructs the live data to the hot spare disk. By monitoring the storage space utilization status at the block level, JOR [56] only recovers the failed data on the used stripes.

Other work tries to improve the reconstruction performance by reorganizing the data layout in an RAID set. Parity declustering [15] decreases the parity group size to boost the scalable RAID rebuild rates. The client-driven rebuild approach [55] based on a per-file RAID layout allows the clients to rebuild files in parallel, thus achieving better recovery performance. In large-scale distributed storage systems, FARM [59] exploits the excess disk capacity to reduce the recovery time.

By introducing the declared mode to provide a record of all outstanding writes in case of a crash, the journal-guided resynchronization approach has improved the reliability and availability of software RAID, while the performance loss during the normal period is little [9].

To cope with the unrecoverable of latent sector errors and improve the reliability of disk systems, disk scrubbing periodically accesses the disks to detect the sector errors and correct the affected sectors [35], [43]. However, study results demonstrate that the reliability improvement due to disk scrubbing is sensitive to the workload [20].

Free block scheduling [30], [47] has been proposed to better utilize the disk bandwidth by letting the background applications make steady forward progress, with no effect on the foreground response times. Argon [52] uses automatically configured prefetch/write-back sizes to bound the inefficiency arising from the interservice disk and cache interference in a storage system which is shared by several services. An algorithmic framework [32] estimates when and for how long idle times can be utilized by the low-priority background tasks, without violating the performance goals of foreground tasks.

While all the above algorithms focus on improving the performance by optimizing the organization of work *within a disk or a single RAID set*, our work increases the performance of the low-priority background tasks by outsourcing I/O workloads. Importantly, WorkOut is orthogonal to and can further improve the above techniques.

6.2 Write Offloading and Data Migration

Our study is related in spirit to write offloading [33], [34] and data migration [3], [23], [25], [29], [54], but with distinctively different characteristics, as shown in Table 5.

Write offloading [33] redirects the writes from one volume to another, to prolong the idle period for one volume allowing the system to spin down disks for saving energy. Similarly, Everest [34] offloads the writes from the overloaded volumes to the lightly loaded ones to improve the performance during peaks.

Data migration [29] moves data from one storage device to another, e.g., for the purpose of load balancing (or load concentration), failure recovery, or system expansion. Data migration has been used in the context of improving energy efficiency in PARAID [54], improving the performance by data reallocation (e.g., in the products of EMC's Symmetrix family [3]), for the read request offloading in Cuckoo [25] and for the user-centric data migration in networked storage systems [23].

In contrast, WorkOut improves the performance of the low-priority background tasks by *temporarily* redirecting writes and popular reads and reclaiming the redirected write data back after the background tasks complete.

7 Conclusion

In this paper, for significantly boosting the performance of the low-priority background tasks, we propose *WorkOut* (I/O *Work*load *Out*sourcing) that outsources a significant amount of user I/O requests away from the degraded RAID set, which is performing the background task, to a surrogate RAID set. We have implemented a lightweight prototype of WorkOut in the Linux software RAID. In a detailed experimental evaluation, we demonstrate that, compared

with the existing reconstruction algorithms PR and PRO, WorkOut significantly speeds up the reconstruction time and average user response time simultaneously.

In summary, this paper makes the following main contributions:

- We propose WorkOut to outsource the I/O workloads away from the degraded RAID set. It tackles I/O intensity that is one of the most important factors adversely affecting the performance of the low-priority background tasks. To the best of our knowledge, it has not been adequately addressed by the previous studies [4], [48].
- WorkOut has a distinctive advantage of improving the performance of the low-priority background tasks. It is a very effective optimization scheme focusing on optimizing the write-intensive workloads, a roadblock for many of the existing approaches [48].
- We conduct comprehensive experiments on our lightweight prototype implementation to evaluate the performance of WorkOut and its sensitivity to a number of workloads and system parameters.
- We provide insights and guidance for storage system designers and administrators by exploiting three WorkOut design options based on their device overhead, performance, reliability, maintainability, and trade-offs.
- Besides the RAID reconstruction, we demonstrate how WorkOut can be easily deployed to improve the performance of other low-priority background tasks, such as RAID resynchronization. Moreover, WorkOut is portable and complementary to and can be easily incorporated into most existing optimized approaches to further improve their performance.

WorkOut is an ongoing research project and we are currently exploring several directions for future work. One is to obtain the liveness information at the block level, thus making WorkOut more transparent to the file system and more effectively utilize the free space on a live surrogate RAID set. The reason is that utilizing the free space on a live RAID set at the file system level is complicated as the file system must be engaged to discover, assign, protect, and manage the free space [44]. Another is to conduct detailed experiments to measure the impact of WorkOut on other low-priority background tasks, such as disk scrubbing and block-level backup and snapshot.

Acknowledgments

This work was supported by the National Basic Research 973 Program of China under Grant No. 2011CB302301, the National High Technology Research and Development Program ("863"Program) of China under Grant No. 2009AA01A401, 2009AA01A402, the Changjiang innovative group of Education of China No. IRT0725, the China National Science Foundation under Grant No. 61025008, and the US National Science Foundation (NSF) under Grant No. NFS-CCF-0621526, NSF-CCF-0937993, NSF-CNS-1016609, and NSF-IIS-0916859. This is an extended version of our manuscript published in the Proceedings of the Seventh USENIX Conference on File and Storage Technologies (FAST '09). Most work of Suzhen Wu was done in Huazhong University of Science and Technology when she was a PhD student. Dan Feng is the corresponding author.

References

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J.D. Davis, M. Manasse, and R. Panigrahy, "Design Tradeoffs for SSD Performance," *Proc. Ann. Technical Conf. (USENIX '08)*, June 2008.

[2] M. Arlitt and C. Williamson, "Web Server Workload Characterization: The Search for Invariants," *Proc. Int'l Conf. Measurement and Modelling of Computer Systems (SIGMETRICS '96)*, May 1996.

[3] R. Arnan, E. Bachmat, T.K. Lam, and R. Michel, "Dynamic Data Reallocation in Disk Arrays," *ACM Trans. Storage*, vol. 3, no. 1, 2007.

[4] E. Bachmat and J. Schindler, "Analysis of Methods for Scheduling Low Priority Disk Drive Tasks," *Proc. Int'l Conf. Measurement and Modelling of Computer Systems (SIGMETRICS '02)*, June 2002.

[5] L.N. Bairavasundaram, G.R. Goodson, S. Pasupathy, and J. Schindler, "An Analysis of Latent Sector Errors in Disk Drives," *Proc. SIGMETRICS '07*, June 2007.

[6] F. Chen, D.A. Koufaty, and X. Zhang, "Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives," *Proc. Int'l Joint Conf. Measurement and Modelling of Computer Systems (SIGMETRICS/Performance '09)*, June 2009.

[7] L. Cherkasova and G. Ciardo, "Characterizing Temporal Locality and Its Impact on Web Server Performance," Technical Report HPL-2000-82, Hewlett Packard Laboratories, July 2000.

[8] L. Cherkasova and M. Gupta, "Analysis of Enterprise Media Server Workloads: Access Patterns, Locality, Content Evolution, and Rates of Change," *IEEE/ACM Trans. Networking*, vol. 12, no. 5, pp. 781-794, Oct. 2004.

[9] T.E. Denehy, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau, "Journal-Guided Resynchronization for Software RAID," *Proc. Conf. File and Storage Technologies (FAST '05)*, Dec. 2005.

[10] EMC Storage Products, http://www.emc.com/products/category/storage.htm, 2010.

[11] G. Gibson, "Reflections on Failure in Post-Terascale Parallel Computing. Keynote," *Proc. Int'l Conf. Parallel Processing (ICPP '07)*, Sept. 2007.

[12] J. Gray, "Rules of Thumb in Data Engineering. Keynote Address," *Proc. Int'l Conf. Data Eng. (ICDE '00)*, Feb. 2000.

[13] J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, fourth ed. Morgan Kaufmann, 2006.

[14] M. Holland, "On-Line Data Reconstruction in Redundant Disk Arrays," PhD thesis, Carnegie Mellon Univ., Apr. 1994.

[15] M. Holland and G. Gibson, "Parity Declustering for Continuous Operation in Redundant Disk Arrays," *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '92)*, Oct. 1992.

[16] R. Hou, J. Menon, and Y. Patt, "Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array," *Proc. Hawaii Int'l Conf. System Sciences (HICSS '93)*, 1993.

[17] R. Hou and Y. Patt, "Using Non-Volatile Storage to Improve the Reliability of RAID5 Disk Arrays," *Proc. Int'l Symp. Fault-Tolerant Computing (FTCS '97)*, 1997.

[18] HP Disk Storage Systems, http://h18006.www1.hp.com/storage/disk_storage/index.html, 2010.

[19] IBM Disk Storage Systems, http://www-03.ibm.com/systems/storage/disk/, 2010.

[20] I. Iliadis, R. Haas, X.-Y. Hu, and E. Eleftheriou, "Disk Scrubbing versus Intra-Disk Redundancy for High-Reliability RAID Storage System," *Proc. Int'l Conf. Measurement and Modelling of Computer Systems (SIGMETRICS '08)*, June 2008.

[21] Iometer, http://sourceforge.net/projects/iometer, 2010.

[22] W. Jiang, C. Hu, Y. Zhou, and A. Kanevsky, "Are Disks the Dominant Contributor for Storage Failures? A Comprehensive Study of Storage Subsystem Failure Characteristics," *Proc. Conf. File and Storage Technologies (FAST '08)*, Feb. 2008.

[23] S. Kang and A.L.N. Reddy, "User-Centric Data Migration in Networked Storage Systems," *Proc. IEEE Int'l Symp. Parallel and Distributed Processing (IPDPS '08)*, Apr. 2008.

[24] H.H. Kari, H.K. Saikkonen, N. Park, and F. Lombardi, "Analysis of Repair Algorithms for Mirrored-Disk Systems," *IEEE Trans. Reliability*, vol. 46, no. 2, pp. 193-200, June 1997.

[25] A.J. Klosterman and G. Ganger, "Cukoo: Layered Clustering for NFS," Technical Report CMU-CS-02-183, Carnegie Mellon Univ., Oct. 2002.

[26] A. Krioukov, L.N. Bairavasundaram, G.R. Goodson, K. Srinivasan, R. Thelen, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau, "Parity Lost and Parity Regained," *Proc. Conf. File and Storage Technologies (FAST '08)*, Feb. 2008.

[27] J.Y.B. Lee and J.C.S. Lui, "Automatic Recovery from Disk Failure in Continuous-Media Servers," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 5, pp. 499-515, May 2002.
[28] Z. Li, Z. Chen, S.M. Srinivasan, and Y. Zhou, "C-Miner: Mining Block Correlations in Storage Systems," *Proc. Conf. File and Storage Technologies (FAST '04),* Mar. 2004.
[29] C. Lu, G.A. Alvarez, and J. Wilkes, "Aqueduct: Online Data Migration with Performance Guarantees," *Proc. Conf. File and Storage Technologies (FAST '02),* Jan. 2002.
[30] C.R. Lumb, J. Schindler, G.R. Ganger, D.F. Nagle, and E. Riedel, "Towards Higher Disk Head Utilization: Extracting Free Bandwidth from Busy Disk Drives," *Proc. Symp. Operating Systems Design and Implementation (OSDI '00),* Oct. 2000.
[31] M.P. Mesnier, M. Wachs, R.R. Sambasivan, J. Lopez, J. Hendricks, G.R. Ganger, and D. O'Hallaron, "//TRACE: Parallel Trace Replay with Approximate Causal Events," *Proc. Conf. File and Storage Technologies (FAST '07),* Feb. 2007.
[32] N. Mi, A. Riska, X. Li, E. Smirni, and E. Riedel, "Restrained Utilization of Idleness for Transparent Scheduling of Background Tasks," *Proc. Int'l Joint Conf. Measurement and Modelling of Computer Systems (SIGMETRICS/Performance '09),* June 2009.
[33] D. Narayanan, A. Donnelly, and A. Rowstron, "Write Off-Loading: Practical Power Management for Enterprise Storage," *Proc. Conf. File and Storage Technologies (FAST '08),* Feb. 2008.
[34] D. Narayanan, A. Donnelly, E. Thereska, S. Elnikety, and A. Rowstron, "Everest: Scaling Down Peak Loads Through I/O Off-Loading," *Proc. Symp. Operating Systems Design and Implementation (OSDI '08),* Dec. 2008.
[35] A. Oprea and A. Juels, "A Clean-Slate Look at Disk Scrubbing," *Proc. Conf. File and Storage Technologies (FAST '10),* Feb. 2010.
[36] J.-F. Pâris, A. Amer, and D.D.E. Long, "Using Storage Class Memories to Increase the Reliability of Two-Dimensional RAID Arrays," *Proc. IEEE Int'l Conf. Modeling, Analysis and Simulation of Computer and Telecomm. Systems (MASCOTS '09),* Sept. 2009.
[37] J. Piernas, T. Cortes, and J.M. García, "Tpcc-uva: A Free, Open-Source Implementation of the Tpc-c Benchmark," http://www.infor.uva.es/~diego/tpcc-uva.html, 2005.
[38] E. Pinheiro, W.-D. Weber, and L.A. Barroso, "Failure Trends in a Large Disk Drive Population," *Proc. Conf. File and Storage Technologies (FAST '07),* Feb. 2007.
[39] A. Riska and E. Riedel, "Idle Read After Write—IRAW," *Proc. Ann. Technical Conf. (USENIX '08),* June 2008.
[40] M. Rosenblum and J.K. Ousterhout, "The Design and Implementation of a Log-Structured File System," *Proc. ACM Symp. Operating Systems Principles (SOSP '91),* Oct. 1991.
[41] B. Schroeder and G. Gibson, "Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You?" *Proc. Conf. File and Storage Technologies (FAST '07),* Feb. 2007.
[42] B. Schroeder, A. Wierman, and M. Harchol-Balter, "Open versus Closed: A Cautionary Tale," *Proc. Conf. Networked Systems Design and Implementation (NSDI '06),* May 2006.
[43] 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," *Proc. IEEE Int'l Conf. Modeling, Analysis and Simulation of Computer and Telecomm. Systems (MASCOTS '04),* Oct. 2004.
[44] M. Sivathanu, L.N. Bairavasundaram, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau, "Life or Death at Block-Level," *Proc. Symp. Operating Systems Design and Implementation (OSDI '04),* Dec. 2004.
[45] 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," *Proc. Conf. File and Storage Technologies (FAST '04),* Mar. 2004.
[46] Storage Performance Council, http://www.storageperformance.org/home, 2010.
[47] E. Thereska, J. Schindler, J. Bucy, B. Salmon, C.R. Lumb, and G.R. Ganger, "A Framework for Building Unobtrusive Disk Maintenance Applications," *Proc. Conf. File and Storage Technologies (FAST '04),* Apr. 2004.
[48] 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," *Proc. Conf. File and Storage Technologies (FAST '07),* Feb. 2007.
[49] L. Tian, H. Jiang, D. Feng, Q. Xin, and X. Shu, "Implementation and Evaluation of a Popularity-Based Reconstruction Optimization Algorithm in Availability-Oriented Disk Arrays," *Proc. IEEE Conf. Mass Storage Systems and Technologies (MSST '07),* Sept. 2007.
[50] TPC-C Specification, http://www.tpc.org/tpcc/, 2010.
[51] UMass Trace Repository, http://traces.cs.umass.edu/index.php/Storage/Storage, 2010.
[52] M. Wachs, M. Abd-El-Malek, E. Thereska, and G.R. Ganger, "Argon: Performance Insulation for Shared Storage Servers," *Proc. Conf. File and Storage Technologies (FAST '07),* Feb. 2007.
[53] M. Wang, "Performance Modeling of Storage Devices using Machine Learning," PhD thesis, Carnegie Mellon Univ., Jan. 2006.
[54] C. Weddle, M. Oldham, J. Qian, A.A. Wang, P. Reiher, and G. Kuenning, "PARAID: The Gear-Shifting Power-Aware RAID," *Proc. Conf. File and Storage Technologies (FAST '07),* Feb. 2007.
[55] 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," *Proc. Conf. File and Storage Technologies (FAST '08),* Feb. 2008.
[56] S. Wu, D. Feng, H. Jiang, B. Mao, L. Zeng, and J. Chen, "JOR: A Journal-guided Reconstruction Optimization for RAID-Structured Storage Systems," *Proc. Int'l Conf. Parallel and Distributed Systems (ICPADS '09),* Dec. 2009.
[57] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao, "WorkOut: I/O Workload Outsourcing for Boosting RAID Reconstruction Performance," *Proc. Conf. File and Storage Technologies (FAST '09),* Feb. 2009.
[58] T. Xie and H. Wang, "MICRO: A Multilevel Caching-Based Reconstruction Optimization for Mobile Storage Systems," *IEEE Trans. Computers,* vol. 57, no. 10, pp. 1386-1398, Oct. 2008.
[59] Q. Xin, E.L. Miller, and T.J.E. Schwarz, "Evaluation of Distributed Recovery in Large-Scale Storage Systems," *Proc. IEEE Int'l Conf. High performance Distributed Computing (HPDC '04),* June 2004.
[60] Q. Xin, E.L. Miller, T.J.E. Schwarz, D.D.E. Long, S.A. Brandt, and W. Litwin, "Reliability Mechanisms for Very Large Storage Systems," *Proc. IEEE Conf. Mass Storage Systems and Technologies (MSST '03),* Apr. 2003.

Suzhen Wu received the BE degree in computer science and technology and the PhD degree in computer architecture from Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2010, respectively. She is an assistant professor in the Computer Science Department, Xiamen University. Her research interests include computer architecture and storage systems. She has more than 10 publications in journals and international conferences including FAST, IPDPS, MASCOTS, and ICPADS. She is a member of the IEEE.

Hong Jiang received the BSc degree in computer engineering from Huazhong University of Science and Technology, Wuhan, China, in 1982, the MASc degree in computer engineering from the University of Toronto, Canada, in 1987, and the PhD degree in computer science from Texas A&M University, College Station, in 1991. Since August 1991, he has been at the University of Nebraska-Lincoln (UNL), where he served as the vice chair in the Department of Computer Science and Engineering (CSE) from 2001 to 2007 and is a professor of CSE. At UNL, he has graduated 10 PhD students who upon their graduations either landed academic tenure-track positions (e.g., Stevens Institute of Tech., New Mexico Tech., University of Maine, University of Alabama, etc.) or were employed by major US IT corporations (e.g., Microsoft, Seagate, Amazon, etc.). His present research interests include computer architecture, computer storage systems and parallel I/O, parallel/distributed computing, cluster and Grid computing, performance evaluation, real-time systems, middleware, and distributed systems for distance education. He serves as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. He has more than 170 publications in major journals and international Conferences in these areas, including IEEE-TPDS, IEEE-TC, JPDC, ISCA, FAST, ICDCS, IPDPS, OOPLAS, ECOOP, SC, ICS, HPDC, ICPP, etc., and his research has been supported by the US National Science Foundation (NSF), Department of Defense (DOD), and the State of Nebraska. He is a senior member of the IEEE and a member of the ACM.

Dan Feng received the BE, ME, and PhD degrees in computer science and technology in 1991, 1994, and 1997, respectively, from Huazhong University of Science and Technology (HUST), China, where she is a professor and the vice dean in the School of Computer Science and Technology. Her research interests include computer architecture, massive storage systems, and parallel file systems. She has more than 80 publications in journals and international conferences, including JCST, FAST, ICDCS, HPDC, SC, ICS, and ICPP. She is a member of the IEEE and a member of the ACM.

Lei Tian received the BE degree in computer science and technology and the ME and PhD degrees in computer architecture from Huazhong University of Science and Technology (HUST), China, in 2001, 2004, and 2010, respectively. His research interests mainly lie in the following aspects: RAID-structured storage systems, distributed storage systems, and large-scale metadata management. He has more than 20 publications in journals and international conferences including FAST, MSST, ICS, SC, HPDC, and ICDCS. He is a student member of the IEEE.

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 Huazhong University of Science and Technology, Wuhan, China, in 2010. His research interests include computer storage system and architecture and the performance evaluation. He has more than 10 publications in journals and international conferences including FAST, IPDPS, MASCOTS, and ACSAC.

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