WorkOut: I/O Workload Outsourcing for Boosting RAID Reconstruction Performance

Suzhen Wu[1], *Hong Jiang*[2], *Dan Feng*[1*], *Lei Tian*[12], *Bo Mao*[1]
[1]*Key Laboratory of Data Storage Systems, Ministry of Education of China*
[1]*School of Computer Science & Technology, Huazhong University of Science & Technology*
[2]*Department of Computer Science & Engineering, University of Nebraska-Lincoln*
[*]*Corresponding author: dfeng@hust.edu.cn*
{*suzhen66, maobo.hust*}*@gmail.com,* {*jiang, tian*}*@cse.unl.edu, ltian@hust.edu.cn*

Abstract

User I/O intensity can significantly impact the performance of on-line RAID reconstruction due to contention for the shared disk bandwidth. Based on this observation, this paper proposes a novel scheme, called *WorkOut* (I/O *Work*load *Out*sourcing), to significantly boost RAID reconstruction performance. WorkOut effectively outsources all write requests and popular read requests originally targeted at the degraded RAID set to a surrogate RAID set during reconstruction. Our lightweight prototype implementation of WorkOut and extensive trace-driven and benchmark-driven experiments demonstrate that, compared with existing reconstruction approaches, WorkOut significantly speeds up both the total reconstruction time and the average user response time. Importantly, WorkOut is orthogonal to and can be easily incorporated into any existing reconstruction algorithms. Furthermore, it can be extended to improving the performance of other background support RAID tasks, such as re-synchronization and disk scrubbing.

1 Introduction

As a fundamental technology for reliability and availability, RAID [30] has been widely deployed in modern storage systems. A RAID-structured storage system ensures that data will not be lost when disks fail. One of the key responsibilities of RAID is to recover the data that was on a failed disk, a process known as *RAID reconstruction*.

The performance of RAID reconstruction techniques depends on two factors. First, the time it takes to complete the reconstruction of a failed disk, since longer reconstruction times translate to a longer "window of vulnerability", in which a second disk failure may cause persistent data loss. Second, the impact of the reconstruction process on the foreground workload, i.e., to what degree are user requests affected by the ongoing reconstruction.

Current approaches for RAID reconstruction fall into two different categories: [11, 12]: *off-line reconstruction*, when the RAID devotes all of its resources to performing reconstruction without serving any I/O requests from user applications, and *on-line reconstruction*, when the RAID continues to service user I/O requests during reconstruction.

Off-line reconstruction has the advantage that it's faster than on-line reconstruction, but it is not practical in environments with high availability requirements, as the entire RAID set needs to be taken off-line during reconstruction.

On the other hand, on-line reconstruction allows foreground traffic to continue during reconstruction, but takes longer to complete than off-line reconstruction as the reconstruction process competes with the foreground workload for I/O bandwidth. In our experiments we find that on-line reconstruction (with heavy user I/O workloads) can be as much as 70 times ($70\times$) slower than off-line reconstruction (without user I/O workloads) (see Section 2.2). Moreover, while on-line reconstruction allows foreground workload to be served, the performance of the foreground workload might be significantly reduced. In our experiments, we see cases where the user response time increases by a factor of 3 ($3\times$) during on-line reconstruction(see Section 2.2).

Improving the performance of on-line RAID reconstruction is becoming a growing concern in the light of recent technology trends: reconstruction times are expected to increase in the future, as the capacity of drives grows at a much higher rate than other performance parameters, such as bandwidth, seek time and rotational latency [10]. Moreover, with the ever growing number of drives in data centers, reconstruction might soon become the common mode of operation in large-scale systems rather than the exception [4, 9, 18, 32, 34].

A number of approaches have been proposed to improve the performance of RAID reconstruction, including for example optimizing the reconstruction workflow [12, 22], the reconstruction sequence [3, 41] or the data layout [14, 49]. We note that all these approaches focus on a single RAID set. In this paper we propose a new approach to improving reconstruction performance,

exploiting the fact that most data centers contain a large number of RAID sets.

Inspired by recent work on data migration [2, 24, 46] and write off loading [27], we propose *WorkOut*, a framework to significantly improve on-line reconstruction performance 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 degraded RAID set to a *surrogate RAID set*. The surrogate RAID set can be free space on another live RAID set or a set of spare disks.

The benefits of WorkOut are two-fold. WorkOut reduces the impact of reconstruction on foreground traffic because most user requests can be served from the surrogate RAID set and hence no longer compete with the reconstruction process for disk bandwidth. WorkOut also speeds up the reconstruction process, since more bandwidth on the degraded RAID set can be devoted to the reconstruction process.

In more detail, WorkOut has the following salient features:

(1) WorkOut tackles one of the most important factors adversely affecting reconstruction performance, namely, *I/O intensity*, that, to the best of our knowledge, has not been adequately addressed by the previous studies [3, 41].
(2) WorkOut has a distinctive advantage of improving both reconstruction time and user response time. It is a very effective reconstruction optimization scheme focusing on optimizing write-intensive workloads, a roadblock for many of the existing reconstruction approaches [41].
(3) WorkOut is orthogonal and complementary to and can be easily incorporated into most existing RAID reconstruction approaches to further improve their performance.
(4) In addition to boosting RAID reconstruction performance, WorkOut is very lightweight and can be easily extended to improve the performance of other background tasks, such as re-synchronization [7] and disk scrubbing [36], that are also becoming more frequent and lengthier for the same reasons that reconstruction is becoming more frequent and lengthier.

Extensive trace-driven and benchmark-driven experiments conducted on our lightweight prototype implementation of WorkOut show that WorkOut significantly outperforms the existing reconstruction approaches PR [22] and PRO [42] in both reconstruction time and user response time.

The rest of this paper is organized as follows. Background and motivation are presented in Section 2. We describe the design of WorkOut in Section 3. Performance evaluations of WorkOut based on a prototype implementation are presented in Section 4. We analyze the reliability of WorkOut in Section 5 and present related work in Section 6. We point out directions for future research in Section 7 and summarize the main contributions of the paper in Section 8.

2 Background and Motivation

In this section, we provide some background and key observations that motivate our work and facilitate our presentation of WorkOut in later sections.

2.1 Disk failures in the real world

Recent studies of field data on partial or complete disk failures in large-scale storage systems indicate that disk failures happen at a significant rate [4, 9, 18, 32, 34]. Schroeder & Gibson [34] found that annual disk replacement rates in the real world exceed 1%, with 2%-4% on average and up to 13% in some systems, much higher than 0.88%, the annual failure rates (AFR) specified by the manufacturer's datasheet. Bairavasundaram *et al*. [4] observed that the probability of latent sector errors, which can lead to disk replacement, is 3.45% in their study. Those failure rates, combined with the continuously increasing number of drives in large-scale storage systems, raise concerns that in future storage systems, recovery mode might become the common mode of operation [9].

Another concern arises from recent studies showing a significant amount of correlation in drive failures, indicating that, after one disk fails, another disk failure will likely occur soon [4, 9, 18]. 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% for one hour, 1.0% for 3 hours and 1.4% for 6 hours.

All the above trends make fast recovery from disk failures an increasingly important factor in building storage systems.

2.2 Mutually adversary impact of reconstruction and user I/O requests

During on-line RAID reconstruction, reconstruction requests and user I/O requests compete for the bandwidth of the surviving disks and adversely affect each other. User I/O requests increase the reconstruction time while the reconstruction process increases the user response time.

Figure 1 shows the reconstruction times and user response times of a 5-disk RAID5 set with a stripe unit size of 64KB in three cases: (1) off-line reconstruction, (2) on-line reconstruction at the highest speed (when RAID favors the reconstruction process), and (3) on-line reconstruction at the lowest speed (when RAID favors user

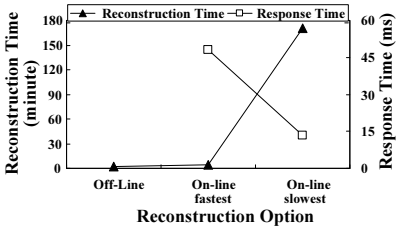


Figure 1: Reconstruction and its performance impact.

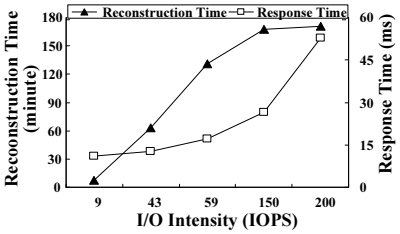


Figure 2: I/O intensity impact on reconstruction.

I/O requests). In this experiment we limit the capacity of each disk to 10GB. User I/O requests are generated by Iometer [17] with 20% sequential and 60%/40% read/write requests of 8KB each. As shown in Figure 1, the user response time increases significantly along with the reconstruction speed, 3 times ($3\times$) more than that in the normal mode. The on-line reconstruction process at the lowest speed takes 70 times ($70\times$) longer than its off-line counterpart.

How reconstruction is performed impacts both the reliability and availability of storage systems [11]. Storage system reliability is formally defined as MTTDL (the mean time to data loss) and increases with decreasing MTTR (the mean time to repair). Ironically, decreasing the MTTR (i.e., speeding up reconstruction) by throttling foreground user requests can lead SLA (Service Level Agreement) violations, which in many environments are also perceived as reduced availability. Ideally, one would like to reduce both the reconstruction time and user response time in order to improve the reliability and availability of RAID-structured storage systems.

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

From the above experiments and analysis, we believe that reducing the amount of user I/O requests directed to the degraded RAID set is an effective approach to simultaneously reducing the reconstruction time and alleviating the user performance degradation, thus improving both reliability and availability. However, naively redirecting *all* requests from the degraded RAID to a surrogate RAID might overload the surrogate RAID and runs the risk that a lot 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

Previous studies indicate that access locality is one of the key web workload characteristics [1, 5, 6] and observe that 10% of files accessed on a web server approximately account for 90% of the requests and 90% of the bytes transferred [1]. Such studies also find that 20%-40% of the files are accessed only once for web workloads [6].

To exploit access locality, caches have been widely employed to improve storage system performance. Storage caches, while proven very effective in capturing workload locality, is so small in capacity compared with the typical storage device that it usually cannot capture all workload locality. Thus the locality underneath the storage cache can still be effectively mined and utilized [23, 41]. For example, based on the study on C-Miner [23] that mines block correlation below the storage cache, correlation-directed prefetching and data layout help reduce the user response time of the baseline case by 12-25%. By utilizing the workload locality at the block level, PRO [41] reduces the reconstruction time by up to 44.7% and the user response time by 3.6-23.9% simultaneously.

Based on these observations, WorkOut only redirects the popular read data to the surrogate RAID set to exploit access locality of read requests bound to the most popular data. *For simplicity, popular data in our design is defined as the data that has been read at least twice during reconstruction*. 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 description of the WorkOut organization and algorithm. The design choice and data consistency issues of WorkOut are discussed at the end of this section.

3.1 Design principles

WorkOut focuses on outsourcing I/O workloads and aims to achieve reliability, availability, extendibility and flexibility, as follows.

Reliability. To reduce the window of vulnerability and thus improve the system reliability, the reconstruction time must be significantly reduced. Since user I/O intensity severely affects the reconstruction process, WorkOut aims to reduce the I/O intensity on the degraded RAID set by redirecting I/O requests away from the degraded RAID set.

Availability. To avoid a drop in user perceived performance and violation of SLAs, the user response time during reconstruction must be significantly reduced. WorkOut strives to achieve this goal by significantly reducing, if not eliminating, the contention between external user I/O requests and internal reconstruction requests, by outsourcing I/O workloads to a surrogate RAID set.

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

Flexibility. Due to the high cost and inconvenience involved in modifying the organization of an existing RAID, it is desirable to completely avoid such modification and instead utilize a separate surrogate RAID set 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 a RAID as the surrogate set ensures that the redirected write data is safe-guarded with redundancy, thus guaranteeing the consistency of the redirected data. How to choose an appropriate surrogate RAID set is based on the requirements on overhead, performance, reliability, and maintainability and trade-offs between them.

3.2 WorkOut architecture overview

Figure 3 shows an overview of WorkOut's architecture. In our design, WorkOut is an augmented module to the RAID controller software operating underneath the storage cache in a system with multiple RAID sets. WorkOut interacts with the reconstruction module, but is implemented independently of it. WorkOut can be incorporated into any RAID controller software, including various reconstruction approaches, and also other background support RAID tasks. In this paper, we focus on the reconstruction process and a discussion on how WorkOut works with some other background support RAID tasks can be found in Section 4.7.

WorkOut consists of five key functional components: Administration Interface, Popular Data Identifier, Surrogate Space Manager, Request Redirector and Reclaimer, as shown in Figure 3. *Administration Interface* provides an interface for system administrators to configure the WorkOut design options. *Popular Data Identifier*

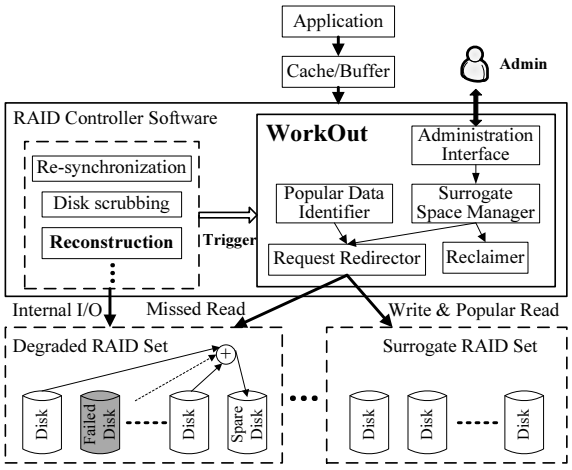


Figure 3: An architecture overview of WorkOut.

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 reconstruction process completes. *Surrogate Space Manager* is responsible for allocating and managing a space on the surrogate RAID set for each current reconstruction process and controlling the data layout of the redirected data in the allocated space.

WorkOut is automatically activated by the reconstruction module when the reconstruction thread is initiated and de-activated when the reclaim process completes. In other words, WorkOut is active throughout the entire reconstruction and reclaim periods. Moreover, the reclaim thread is triggered by the reconstruction module when the reconstruction process completes.

In WorkOut, the idea of a dedicated surrogate RAID set is targeted at a typical data center where the surrogate set can be shared by multiple degraded RAID sets on either a space-division or a time-division basis. The degraded RAID set and surrogate RAID set are not a one-to-one mapping. The device overhead is incurred only during reconstruction of a degraded RAID set and typically amounts to a small fraction of the surrogate set capacity. For example, extensive experiments on our prototype implementation of WorkOut show that, of all the traces and benchmarks, no more than 4% of the capacity of a 4-disk surrogate RAID5 set is used during the WorkOut reconstruction.

Based on the pre-configured parameters by system administrators, the Surrogate Space Manager allocates a disjoint space for each degraded RAID set that requests a surrogate RAID set, thus preventing the redirected data from being overwritten by redirected data from other degraded RAID sets. Noticeably, the space allocated to a degraded RAID set is not fixed and can be expanded. For

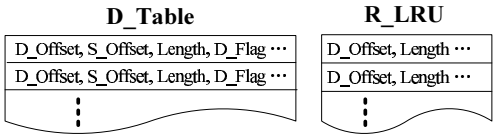


Figure 4: Data structures of WorkOut.

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

3.3 The WorkOut organization and algorithm

WorkOut relies on two key data structures to redirect requests and identify popular data, namely, *D_Table* and *R_LRU*, as shown in Figure 4. D_Table contains the log of all redirected data, including the following four important variables. *D_Offset* and *S_Offset* indicate the offsets of the redirected data in the degraded RAID set and the surrogate RAID set, respectively. *Length* indicates the length of the redirected data and *D_Flag* indicates whether it is the redirected write data from the user application (*D_Flag is set to be true*) or the redirected read data from the degraded RAID set (*D_Flag is set to be false*). R_LRU is an LRU list that stores the information (*D_offset* and *Length* of read data) 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 during reconstruction and does not modify the reconstruction algorithm. How to perform the reconstruction process remains the responsibility of the reconstruction module and depends on the specific reconstruction algorithm, and is thus not described further in this paper.

During reconstruction, all write requests are redirected to the surrogate RAID set after determining whether they should overwrite their previous location or write to a new location according to D_Table. Whereas, for each read request, D_Table is first checked to determine whether the read data is in the surrogate RAID set. If the read request does not hit D_Table, it will be served by the degraded RAID set. If it hits R_LRU, the read data is considered popular and redirected to the surrogate RAID set, and the corresponding data information is inserted into D_Table. If the entire targeted read data is already in the surrogate RAID set, the read request will be served by the surrogate RAID set. Otherwise, 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 both the sets. In order to achieve better performance, the redirected data is laid out sequentially like LFS [33] in the allocated space on the surrogate RAID set.

The redirected write data is only temporarily stored in the surrogate RAID set and thus should be reclaimed back to the newly recovered RAID set (i.e., the formerly degraded RAID set) after the reconstruction process completes. To ensure data consistency, the log of reclaimed data is deleted from D_Table after the write succeeds. Since the redirected read data is already in the degraded RAID set, it need not be reclaimed as long as logs of such data are deleted from D_Table to indicate that the data in the surrogate RAID set is invalid. In order not to affect the performance of the newly recovered RAID set, 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 on the newly recovered RAID set must be checked carefully 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 old data that is still in the surrogate RAID set, the corresponding log in D_Table must be deleted after writing the data to the correct location on the newly recovered RAID set, to prevent the new write data from being overwritten by the reclaimed data. In addition, if a read request hits D_Table and its D_Flag is true, meaning that the up-to-date data of the read request has not been reclaimed back, the read request will be served by the surrogate RAID set.

3.4 Design choices

WorkOut can redirect data to different persistent configurations of storage devices, such as a dedicated surrogate RAID1 set, a dedicated surrogate RAID5 set and a live surrogate RAID set.

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., 2 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., 4 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 redun-

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

Table 1: Characteristic comparisons of three optional surrogate RAID sets used in WorkOut.

dancy, 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 tradeoffs, 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 two aspects: (1) Redirected data must be reliably stored in the surrogate RAID set, (2) The key data structures should be safely stored until the reclaim process completes.

First, 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, such as RAID1 or RAID5. To simplify the design and implementation, the redirected read data is 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 write data that was already redirected should be reclaimed back to the degraded RAID set or redirected to another surrogate RAID set if possible. Our prototype implementation adopts the first option. We will analyze the reliability of WorkOut in Section 5.

Second, since we must ensure never to lose the contents of D_Table during the entire period when WorkOut is activated, it is stored in a NVRAM to prevent data loss in the event of a power supply failure. Fortunately, D_Table is in general very small (see Section 4.8) 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 [8, 15, 16], is roughly the same as the main memory, the write penalty due to D_Table updates can be negligible.

4 Performance Evaluations

In this section, we evaluate the performance of prototype implementation of WorkOut through extensive trace-driven and benchmark-driven experiments.

4.1 Prototype implementation

We have implemented WorkOut by embedding it into the Linux Software RAID (MD) as a built-in module. In order not to impact the RAID performance in normal mode, WorkOut is activated only when the reconstruction process is initiated. During reconstruction, WorkOut tracks 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.

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 1MB/s-200MB/s, MD will favor user I/O requests while ensuring that the reconstruction speed is at least 1MB/s. Under heavy I/O workloads, MD will keep the reconstruction speed at approximately 1MB/s but allows it to be much higher than 1MB/s when 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., 78MB/s in our prototype system). Equivalently, the *minimum reconstruction bandwidth* of XMB/s (e.g., 1MB/s, 10MB/s, 100MB/s) refers to a reconstruction range of XMB/s-200MB/s in MD. When the minimum reconstruction bandwidth is set to 100MB/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, and PRO-powered PR (PRO for short) that is also implemented in MD. PR (Pipeline Reconstruction) [22] takes advantage of the sequential property of track retrievals to pipeline the reading and writing processes. PRO (Popularity-based multi-threaded Reconstruction Optimization) [41, 42] allows the reconstruction process to rebuild the frequently accessed areas prior to other areas.

4.2 Experimental setup and methodology

We conduct our performance evaluation of WorkOut on a platform of server-class hardware with an Intel Xeon 3.0GHz processor and 1GB DDR memory. We use 2 Highpoint RocketRAID 2220 SATA cards to house 15 Seagate ST3250310AS SATA disks. The rotational speed of these disks is 7200 RPM, with a sustained trans-

Trace	Trace Characteristic		
	Write Ratio	IOPS	Aver. Req. Size(KB)
Fin1	67.18%	69	6.2
Fin2	17.61%	125	2.2
Web	0%	113	15.1

Table 2: The trace characteristics.

fer rate of 78MB/s, and the individual disk capacity is 250GB. 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 10GB in the experiments. In our prototype implementation, the main memory is used to substitute a battery-backed RAM for simplicity.

Generally speaking, there are two models for trace replay: open-loop and closed-loop [26, 35]. 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 (and hence the queuing delays) to grow rapidly when system load is high. The opposite is true for 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 (i.e., equal to the number of independent request threads). In this paper, we use both an open-loop model (trace replay with RAIDmeter [41]) and a closed-loop model (TPC-C-like benchmark [43]) to evaluate the performance of WorkOut.

The traces used in our experiments are obtained from the Storage Performance Council [29, 39]. The two financial traces (Fin1 and Fin2) were collected from OLTP applications running at a large financial institution and the WebSearch2 trace (or Web) was collected from a machine running a web search engine. The three traces 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 RAID set (e.g., less than 10% of an 8-disk RAID5 set for the Fin1 trace), their replays may not realistically represent a typical reconstruction scenario where user requests may be spread out over the entire disk address space. To fully and evenly cover the address space of the RAID set, we scale up the address coverage of the I/O requests by multiplying the address of each request with an appropriate scaling factor (constant) without changing the size of each request. While the main adverse impact of this trace scaling is likely to be on those requests that are originally sequential but can subsequently become non-sequential after scaling, the percentage of such sequential requests in the three traces of this study is relatively small at less than 4% [45]. Thus, we believe that the adverse impact of the trace scaling is rather limited in this study and far outweighed by the benefits of the scaling that attempts to represent a more realistic reconstruction scenario. We find in our experiments that the observed trends are similar for the original and the 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 [41, 42] that replays traces at block-level and evaluates the user response time of the storage device. The RAID reconstruction performance is evaluated in terms of the following two metrics: reconstruction time and average user response time during reconstruction.

The TPC-C-like benchmark is implemented with TPCC-UVA [31] and the Postgres database. It generates mixed transactions based on the TPC-C specification [43]. 20 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.

4.3 Trace-driven evaluations

We first conduct experiments on an 8-disk RAID5 set with a stripe unit size of 64KB while running PR, PRO and WorkOut-powered PR and PRO respectively. Table 3 and Table 4, respectively, show the reconstruction time and average user response time under the minimum reconstruction bandwidth of 1MB/s, driven by the three traces. We configure a 4-disk dedicated RAID5 set with a stripe unit size of 64KB as the surrogate RAID set to boost the reconstruction performance of the 8-disk degraded 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, with a reconstruction time of 203.1s vs. 1121.8s for PR and 188.3s vs. 1109.6s for PRO, is due to the fact that 84% of requests (69% of writes plus 15% of reads) are redirected away from the degraded RAID set (see Figure 5), which enables the speed of the on-line reconstruction to approach that of the off-line counterpart. In our experiments, the off-line reconstruction time is 136.4 seconds for PR on the same platform. Moreover, WorkOut out-

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 3: The reconstruction 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

Table 4: The average user response time results.

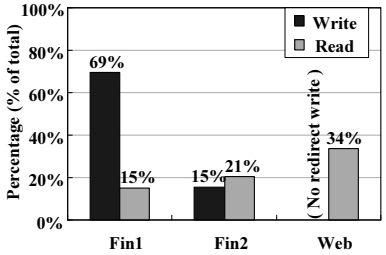
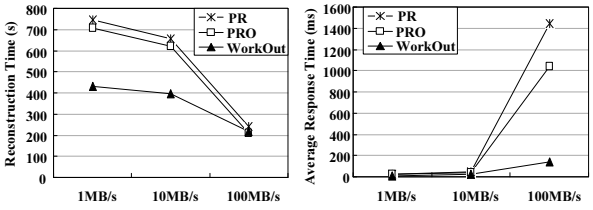


Figure 5: Percentage of redirected requests for WorkOut, under the minimum reconstruction bandwidth of 1MB/s.



(a) Reconstruction time (b) Average user response time

Figure 6: Comparisons of reconstruction times and average user response times with respect to different minimum reconstruction bandwidth (1MB/s, 10MB/s and 100MB/s) driven by the Fin2 trace.

sources 36% and 34% of 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 Figure 5. The response times of redirected requests are no longer affected by the reconstruction process that competes for the available bandwidth with user I/O requests on the degraded RAID set. Second, redirected data is 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.

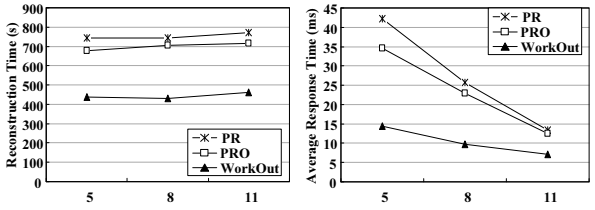
Table 3 and Table 4 show that WorkOut-powered PRO performs similarly to WorkOut-powered PR. The reason is that WorkOut redirects all write requests and popular read requests to the surrogate RAID set, thus reducing the degree of popularity of I/O workloads retained on the degraded RAID set that can be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

4.4 Sensitivity study

WorkOut's performance is likely influenced by several important factors, including the available reconstruction bandwidth, the size of the degraded RAID set, the stripe unit size, 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.

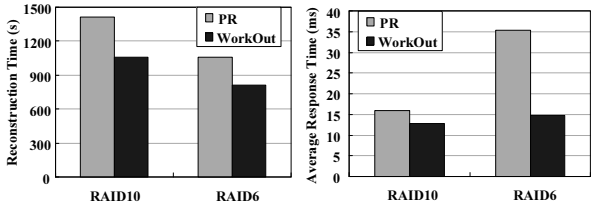
Reconstruction bandwidth. To evaluate how the minimum reconstruction bandwidth affects reconstruction performance, we conduct experiments that measure reconstruction times and average user response times as a function of different minimum reconstruction bandwidth, 1MB/s, 10MB/s and 100MB/s, respectively. Figure 6 plots the experimental results on an 8-disk RAID5 set with a stripe unit size of 64KB.

Figure 6(a)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.



(a) Reconstruction time (b) Average user response time

Figure 7: Comparisons of reconstruction times and average user response times with respect to the number of disks (5, 8, 11) driven by the Fin2 trace.



(a) Reconstruction time (b) Average user response time

Figure 8: Comparisons of reconstruction times and average user response times with respect to different RAID levels (10, 6) driven by the Fin2 trace.

From Figure 6(b), 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 100MB/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 end users.

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 64KB under the minimum reconstruction bandwidth of 1MB/s. Figure 7 shows the experimental results for PR, PRO and WorkOut.

Figure 7(a) and Figure 7(b) 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 a 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.

Stripe unit size. To examine the impact of the stripe unit size, we conduct experiments on an 8-disk RAID5 set with stripe unit sizes of 16KB and 64KB, respectively. The experimental results show that WorkOut outperforms PR and PRO in reconstruction time as well as 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.

RAID level. To evaluate WorkOut with different RAID levels, we conduct experiments on a 4-disk RAID10 set and an 8-disk RAID6 set with the same stripe unit size of 64KB under the minimum reconstruction bandwidth of 1MB/s. In the RAID6 experiments, we measure the reconstruction performance when two disks fail concurrently.

From Figure 8, 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 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 a 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 user response time for RAID10 than for RAID6.

4.5 Different design choices for the surrogate RAID set

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 4-disk RAID5 set). Similar to the experiments conducted in the PARAID [46] and write off-loading [27] studies, we reserve the 10% portion of storage space at the end of the live RAID5 set to store data redirected by WorkOut. The degraded RAID set is an 8-disk RAID5

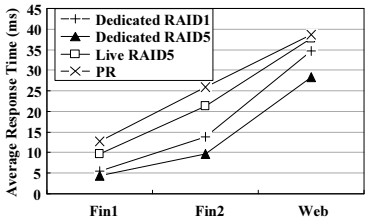


Figure 9: A comparison of average user response times for different types of surrogate RAID set.

set with a stripe unit size of 64KB and under the minimum reconstruction bandwidth of 1MB/s.

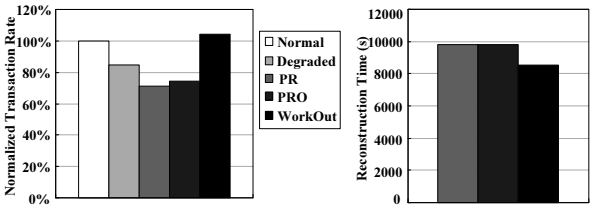
The experimental results show that the reconstruction times achieved by WorkOut are almost the same for the three types of surrogate RAID set and outperform PR as expected, shown in Table 3, since WorkOut outsources the same amount of requests during reconstruction. The results for user response times are somewhat different, as shown in Figure 9. The dedicated surrogate RAID5 set results in the best user response times for the three traces.

From Figure 9, one can see that the dedicated surrogate RAID sets (both RAID1 and RAID5) outperform the live surrogate RAID set in 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 overload on the surrogate RAID set, compared with the dedicated surrogate RAID set, but also destroys some of the sequentiality in 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% 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.

4.6 Benchmark-driven evaluations

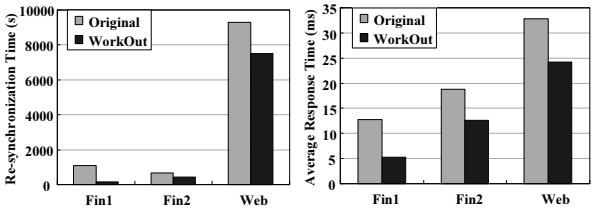
In addition to trace-driven experiments, we also conduct experiments on an 8-disk RAID5 set with a stripe unit size of 64KB under the minimum reconstruction bandwidth of 1MB/s, driven by a TPC-C-like benchmark.

From Figure 10(a), 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 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% 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% respectively.



(a) Transaction Rate (b) Reconstruction time

Figure 10: Comparisons of reconstruction times and transaction rates driven by the TPC-C-like benchmark.



(a) Re-synchronization time (b) Average user response time

Figure 11: Comparisons of re-synchronization times and average user response times during re-synchronization.

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 1MB/s for PR and PRO. As shown in Figure 10(b), the reconstruction times for PR and PRO are similar, at 9835 seconds and 9815 seconds, respectively, while that for WorkOut is 8526 seconds, with approximately 15% improvement over PR and PRO. WorkOut gains much less in 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.

4.7 Re-synchronization with WorkOut

To demonstrate how WorkOut optimizes other background support RAID tasks, such as RAID re-synchronization, we conduct experiments on an 8-disk RAID5 set with a stripe unit size of 64KB under the minimum re-synchronization bandwidth of 1MB/s, driven by the three traces. We configure a dedicated 4-disk RAID5 set with a stripe unit size of 64KB as the surrogate RAID set. The experimental results of re-synchronization times and average user response times during re-synchronization are shown in Figure 11(a) and Figure 11(b), respectively.

Although the re-synchronization process performs somewhat differently from the reconstruction process,

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

4.8 Overhead analysis

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 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 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., 2 disks) is lower than that of the latter (e.g., 4 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 (8-disk data RAID5 set plus 4-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 1MB/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% shorter than that achieved by PR, while the average user response time during the normal period in the 8-disk RAID5 set is 20.1% 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 large-scale and small-scale storage systems.

Memory overhead. To prevent data loss, WorkOut uses non-volatile memory to store D_Table, thus incurring extra memory overhead. The amount of memory consumed is largest when the minimum reconstruction bandwidth is set to 1MB/s, since in this case the reconstruction time is the longest and the amount of redirected data is the largest. In the above experiments on the RAID5 set with individual disk capacity of 10GB, the maximum memory overheads are 0.14MB, 0.62MB and 1.69MB 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 memory size and decrease in cost of non-volatile memories, this memory overhead is arguably reasonable and acceptable to end users.

Implementation overhead. 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 background support RAID 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. We assume that disk failures are independent events following an exponential distribution of rate μ, and repairs follow an exponential distribution of rate ν. For simplicity, we do not consider the latent sector error in the system model.

According to the conclusion about the reliability of RAID5 [50], MTTDL of an 8-disk RAID5 set achieved by PR and PRO is:

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

Figure 12 shows the state transition diagram for a WorkOut-enabled storage system configuration consisting of an 8-disk data RAID5 set and a 4-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 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 4-disk surrogate RAID5 set ready to be used by the 8-disk degraded data RAID5 set. As a result, the state transition diagram only shows the reclaim process but not the reconstruction process of the 4-disk surrogate RAID5 set.

State $<0>$ represents the normal state of the system when its 8 data disks are all operational. A failure of any of the 8 data disks would bring the system to state $<1>$ and a subsequent failure of any of the remaining 7 data disks would result in data loss. A failure of any of the 4 surrogate disks in state $<0>$ does not affect the system

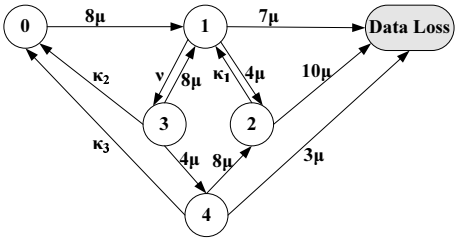


Figure 12: State transition diagram for a WorkOut-enabled storage system configuration consisting of an 8-disk RAID5 set and a 4-disk surrogate RAID5 set. *Note: The 4-disk surrogate RAID5 set does not need to reconstruct the data on the failed disk as long as the redirected write data in the surrogate RAID5 set is reclaimed back to the 8-disk degraded data RAID set.*

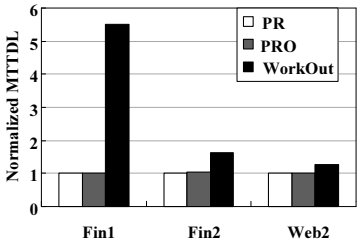


Figure 13: Comparisons of the mean times to data loss. *Note: The normalized baselines are the MTTDLs achieved by PR driven by the three traces respectively.*

reliability of the 8-disk RAID5 set as long as the redirected write data on the former is reclaimed back to the latter and thus it is omitted from the state transition diagram. In state <1>, a failure of any of the 4 surrogate disks would bring the system to state <2>. A second failure in either the 8-disk data RAID5 set (1 out of 7) or the 4-disk surrogate RAID5 set (1 out of 3) in state <2> would result in data loss. In state <2>, WorkOut reclaims the redirected write data back to the 8-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 redirected write data is 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 8-disk data RAID set. Finishing the reconstruction process of the 8-disk data RAID5 set would bring the system from state <1> to state <3>, where the redirected write data has 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 8 data disks would bring the system to state <1>, and a failure of any of the 4 surrogate disks would bring the system to state <4>, where the redirected write data is not protected by redundancy. In state <4>, WorkOut also reclaims the redirected write data back to the 8-disk data RAID set, which bring the system back to state <0> and follows an exponential distribution of rate κ_3. In state <4>, a failure of any of the 8 data disks would bring the system to state <2> and a second disk failure in the 4-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 is reclaimed, it is reasonable to assume that they are equal to a fixed reclaim rate κ, since the amount of 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. Since the expression for MTTDL of Figure 12 is too complex (ratio of two large polynomials) to be displayed here, we present the computed values of MTTDL in the following figure instead.

Figure 13 plots comparisons of 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 one hundred thousand hours, which is a conservative estimate to the values quoted by disk manufactures. Disk repair times, when the individual disk capacity is 250GB, are 25 times the results listed in Table 3, where the capacity of each disk is limited to 10GB. κ is assumed to be equal to the corresponding ν, which is actually overestimated. From Figure 13, one can see that WorkOut increases MTTDL and improves reliability with the decreasing 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 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

Reconstruction algorithms and task scheduling. A large number of different approaches for improving reconstruction performance have been studied. Some of these approaches focus on improved RAID reconstruction algorithms, such as DOR (Disk-Oriented Reconstruction) [12], PR [22], PRO [41] and others [3, 13, 14, 20, 38, 47, 48, 49]. Other approaches focus on better data layout in a RAID set [13, 47, 49]. Moreover, many task scheduling techniques have been proposed to optimize the background applications [25, 40, 44].

While all the the above algorithms focus on improving performance by optimizing the organization of work *within a single RAID set*, our work takes a different approach. The goal behind WorkOut is to increase performance during reconstruction by outsourcing I/O workloads away from the degraded RAID set. Importantly,

WorkOut is orthogonal to and can further improve the above techniques.

Data migration. Our study is related in spirit to write off-loading [27, 28] and data migration [2, 19, 21, 24, 46] techniques, but with distinctively different characteristics. Write off-loading [27] redirects writes from one volume to another, to prolong the idle period for one volume allowing the system to spin down disks for saving energy. Similar in spirit, Everest [28] off-loads writes from overloaded volumes to lightly loaded ones to improve performance during peaks.

Data migration [24] 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 PARAID [46], improving performance by data reallocation (e.g., in the products of EMC's Symmetrix family [2]), for read request offloading in Cuckoo [21] and for user-centric data migration in networked storage systems [19].

In contrast to write off-loading and data migration, WorkOut improves reconstruction performance by, *temporarily* redirecting writes and popular reads during reconstruction and reclaiming the redirected write data back to the newly recovery RAID set after the reconstruction process completes.

7 Future Work

WorkOut is an ongoing research project and we are currently exploring several directions for future work.

Extendibility. In addition to RAID reconstruction and re-synchronization, other background support RAID tasks, such as disk scrubbing and block-level backup and snapshot, could benefit from WorkOut. We plan to conduct detailed experiments to measure the impact of WorkOut on these tasks.

Flexibility. In the current implementation, we configure a reserved space instead of the free space on a live RAID set as the surrogate set, which can be impractical and inflexible. 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 [37]. To make WorkOut more transparent to the file system, and more effectively utilize the free space on a live surrogate RAID set, it would be desirable for WorkOut to obtain the liveness information at the block level. We will explore the live block techniques and apply them in WorkOut to improve its performance.

8 Conclusion

In this paper, for significantly boosting RAID reconstruction performance, 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 to a surrogate RAID set during reconstruction. We present a lightweight prototype of WorkOut implemented 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. Moreover, 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. Importantly, we demonstrate how WorkOut can be easily deployed to improve the performance of other background support RAID tasks such as re-synchronization.

Acknowledgments

We thank our shepherd Bianca Schroeder and the anonymous reviewers for their helpful comments. We also thank Lingfang Zeng, Jianxi Chen and Zhikun Wang for their feedback. This work is supported by the National Basic Research 973 Program of China under Grant No. 2004CB318201, the US NSF under Grant No. CCF-0621526, the China NSFC under Grant No. 60703046 and No. 60873028, the Program for New Century Excellent Talents in University No. NCET-04-0693 and No. NCET-06-0650, the Program for Changjiang Scholars and Innovative Research Team in University No. IRT-0725, the Programme of Introducing Talents of Discipline to Universities (111 Project) No. B07038, SRFDP of Education of China No. 20070487083, and HUST-SRF No.2007Q021B. The work of Lei Tian was done while he was working at the CSE Dept. of UNL.

References

[1] M. Arlitt and C. Williamson. Web Server Workload Characterization: The Search for Invariants. In *SIGMETRICS'96*, May. 1996.

[2] R. Arnan, E. Bachmat, T. K. Lam, and R. Michel. Dynamic Data Reallocation in Disk Arrays. *ACM Transactions on Storage*, 3(1), 2007.

[3] E. Bachmat and J. Schindler. Analysis of Methods for Scheduling Low Priority Disk Drive Tasks. In *SIGMETRICS'02*, Jun. 2002.

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

[5] L. Cherkasova and G. Ciardo. Characterizing Temporal Locality and its Impact on Web Server Performance. Technical Report HPL-2000-82, Hewlett Packard Laboratories, Jul. 2000.

[6] L. Cherkasova and M. Gupta. Analysis of Enterprise Media Server Workloads: Access Patterns, Locality, Content Evolution, and Rates of Change. *IEEE/ACM Transactions on Networking*, 12(5):781–794, Oct. 2004.

[7] T. E. Denehy, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Journal-guided Resynchronization for Software RAID. In *FAST'05*, Dec. 2005.

[8] EMC storage products. `http://www.emc.com/products/category/storage.htm`.

[9] G. Gibson. Reflections on Failure in Post-Terascale Parallel Computing. Keynote. In *ICPP'07*, Sep. 2007.

[10] J. Gray. Rules of Thumb in Data Engineering. Keynote Address. In *ICDE'00*, Feb. 2000.

[11] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Fourth edition, 2006.

[12] M. Holland. *On-Line Data Reconstruction in Redundant Disk Arrays*. PhD thesis, Carnegie Mellon University, Apr. 1994.

[13] M. Holland and G. Gibson. Parity Declustering for Continuous Operation in Redundant Disk Arrays. In *ASPLOS'92*, Oct. 1992.

[14] R. Hou, J. Menon, and Y. Patt. Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array. In *HICSS'93*, 1993.

[15] HP Disk Storage Systems. `http://h18006.www1.hp.com/storage/disk_storage/index.html`.

[16] IBM Disk Storage Systems. `http://www-03.ibm.com/systems/storage/disk/`.

[17] Iometer. `http://sourceforge.net/projects/iometer`.

[18] 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. In *FAST'08*, Feb. 2008.

[19] S. Kang and A. L. N. Reddy. User-Centric Data Migration in Networked Storage Systems. In *IPDPS'08*, Apr. 2008.

[20] H. H. Kari, H. K. Saikkonen, N. Park, and F. Lombardi. Analysis of repair algorithms for mirrored-disk systems. *IEEE Transactions on Reliability*, 46(2):193–200, 1997.

[21] A. J. Klosterman and G. Ganger. Cukoo: Layered clustering for NFS. Technical Report CMU-CS-02-183, Carnegie Mellon University, Oct. 2002.

[22] J. Y.B. Lee and J. C.S. Lui. Automatic Recovery from Disk Failure in Continuous-Media Servers. *IEEE Transactions on Parallel and Distributed Systems*, 13(5):499–515, May. 2002.

[23] Z. Li, Z. Chen, S. M. Srinivasan, and Y. Zhou. C-Miner: Mining Block Correlations in Storage Systems. In *FAST'04*, Mar. 2004.

[24] C. Lu, G. A. Alvarez, and J. Wilkes. Aqueduct: Online Data Migration with Performance Guarantees. In *FAST'02*, Jan. 2002.

[25] 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. In *OSDI'00*, Oct. 2000.

[26] 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. In *FAST'07*, Feb. 2007.

[27] D. Narayanan, A. Donnelly, and A. Rowstron. Write Off-Loading: Practical Power Management for Enterprise Storage. In *FAST'08*, Feb. 2008.

[28] D. Narayanan, A. Donnelly, E. Thereska, S. Elnikety, and A. Rowstron. Everest: Scaling Down Peak Loads Through I/O Off-loading. In *OSDI'08*, Dec. 2008.

[29] OLTP Application I/O and Search Engine I/O. UMass Trace Repository. `http://traces.cs.umass.edu/index.php/Storage/Storage`.

[30] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *SIGMOD'88*, Jun. 1988.

[31] 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.

[32] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure Trends in a Large Disk Drive Population. In *FAST'07*, Feb. 2007.

[33] M. Rosenblum and J. K. Ousterhout. The Design and Implementation of a Log-Structured File System. In *SOSP'92*, Feb. 1992.

[34] B. Schroeder and G. Gibson. Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You? In *FAST'07*, Feb. 2007.

[35] B. Schroeder, A. Wierman, and M. Harchol-Balter. Open Versus Closed: A Cautionary Tale. In *NSDI'06*, May. 2006.

[36] 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 *MASCOTS'04*, Oct. 2004.

[37] M. Sivathanu, L. N. Bairavasundaram, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Life or Death at Block-Level. In *OSDI'04*, Dec. 2004.

[38] 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 *FAST'04*, Mar. 2004.

[39] Storage Performance Council. `http://www.storageperformance.org/home`.

[40] E. Thereska, J. Schindler, J. Bucy, B. Salmon, C. R. Lumb, and G. R. Ganger. A framework for building unobtrusive disk maintenance applications. In *FAST'04*, Apr. 2004.

[41] 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 *FAST'07*, Feb. 2007.

[42] 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. In *MSST'07*, Sep. 2007.

[43] TPC-C specification. `http://www.tpc.org/tpcc/`.

[44] M. Wachs, M. Abd-El-Malek, E. Thereska, and G. R. Ganger. Argon: performance insulation for shared storage servers. In *FAST'07*, Feb. 2007.

[45] M. Wang. *Performance Modeling of Storage Devices using Machine Learning*. PhD thesis, Carnegie Mellon University, Jan. 2006.

[46] C. Weddle, M. Oldham, J. Qian, A. A. Wang, P. Reiher, and G. Kuenning. PARAID: The Gear-Shifting Power-Aware RAID. In *FAST'07*, Feb. 2007.

[47] 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 *FAST'08*, Feb. 2008.

[48] T. Xie and H. Wang. MICRO: A Multilevel Caching-Based Reconstruction Optimization for Mobile Storage Systems. *IEEE Transactions on Computers*, 57(10):1386–1398, 2008.

[49] Q. Xin, E. L. Miller, and T. J. E. Schwarz. Evaluation of Distributed Recovery in Large-Scale Storage Systems. In *HPDC'04*, Jun. 2004.

[50] 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. In *MSST'03*, Apr. 2003.