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

Abstract

User I/O intensity has a significant impact on the on-line RAID reconstruction performance by virtue of disk bandwidth contention. Based on our observation, this paper proposes a novel scheme, called *WorkOut* (I/O *Work*load *Out*sourcing), to significantly boost the RAID reconstruction performance. WorkOut effectively outsources I/O workloads by temporarily redirecting all write requests and popular read requests originally targeted at the degraded RAID set to a surrogate RAID set during reconstruction, thus significantly speeding up the reconstruction process and alleviating the user performance degradation. WorkOut is orthogonal to and can be easily incorporated into any existing reconstruction algorithms. Furthermore, it is applicable to improving the performance of other background support RAID tasks such as re-synchronization and disk scrubbing.

Our lightweight prototype implementation of WorkOut, driven by three representative real-life traces and a TPC-C-like benchmark, demonstrates that, compared with the existing reconstruction approaches PR and PRO, WorkOut speeds up the reconstruction time by a factor of up to 2.87, with an average of 2.09, and speeds up the average user response time by a factor of up to 5.89, with an average of 2.88, simultaneously.

1 Introduction

As a fundamental technology for reliability and availability, RAID [29] has been widely deployed in modern storage systems. A redundancy-based 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*.

RAID reconstruction is commonly performed in two different ways [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. While the former entails stopping all applications completely, making it inappropriate for most RAID-structured storage systems that must provide 24/7 nonstop services to the users, the latter is often limited to using some fractions of the total bandwidth of the RAID system, making it much slower than the former.

Advances in the storage technology have significantly increased the capacity and reduced the cost of hard disks, while other performance parameters, such as bandwidth, seek time and rotational latency, have improved much more slowly [10] and individual disk failure rates have remained largely unchanged [4, 8, 19, 31, 33], thus significantly increasing the length and frequency of reconstruction and resulting in a longer "window of vulnerability", the time span in which a second disk failure may cause persistent data loss. Moreover, this time span is adversely affected by user I/O requests that arrive during reconstruction. For example, the reconstruction time of the on-line approach (with user I/O requests) can be as much as 70 times ($70\times$) longer than that of the off-line approach (without user I/O requests) (see Section 2.2).

On the other hand, the extended reconstruction period, in which the RAID-based server operates in the degraded mode as a result of the user I/O requests having to compete for disk bandwidth with the reconstruction process, will more likely violate the performance goals and Service Level Agreement (SLA) [11]. For example, the reconstruction process can increase the user response time by a factor of 3 ($3\times$) (see Section 2.2). Importantly, the transition from the state of *service accomplishment*, where the service is delivered as specified in SLA, to the state of *service interruption*, where the delivered service is different from and below that specified in SLA, is also defined as a *failure* [11], which constitutes a noticeable fraction of storage subsystem failures [19].

Therefore, the on-line reconstruction performance affects the reliability and availability of RAID-structured storage systems directly and significantly. On-line reconstruction algorithms not only demand the ability to recover from disk failures without data loss, but also must (1) speed up the reconstruction process maximally to reduce the window of vulnerability and (2) impose minimal impact on user performance, simultaneously [14].

Most existing effective RAID reconstruction approaches mainly focus on optimizing reconstruction workflow [12, 23], reconstruction sequence [3, 38] or data layout [15, 43] to improve the reconstruction performance in a single RAID set. In a typical on-line reconstruction process, performing the internal reconstruction I/O requests and external user I/O requests simultaneously leads to disk bandwidth contention and frequent long seeks to and from the multiple separate data areas. Reducing the amount of user I/O requests directed to the degraded RAID set, we believe, is an effective approach to reducing the reconstruction time and alleviating the performance degradation simultaneously. This also hap-

pens to be the reason why the off-line reconstruction approach is much faster than its on-line counterpart.

Inspired by the state-of-the-art data migration [2, 25, 41] and write off-loading [27] approaches, we propose *WorkOut* (I/O *Work*load *Out*sourcing) to effectively utilize a set of spare disks or the free space in one of the many RAID sets in a large-scale storage system, which we call a *surrogate RAID set*, to store the data relocated by a request redirection technique that outsources a significant amount of user I/O requests away from the degraded RAID set during reconstruction. 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, thus significantly reducing the reconstruction time since much more disk bandwidth can then be allocated to the reconstruction process and alleviating the user performance degradation caused by the reconstruction process.

In the context of the on-line RAID reconstruction technologies, WorkOut has the following salient features:

(1) WorkOut tackles one of the most important factors adversely affecting the reconstruction performance, namely, *I/O intensity*, that, to the best of our knowledge, has not been adequately addressed by the previous studies [3, 38] and achieves a reconstruction performance approaching that of the off-line reconstruction approach for write-intensive applications, by continuously and effectively serving the external user I/O requests during reconstruction.

(2) WorkOut has a distinctive advantage of improving both reconstruction time and user response time. It is a very effective reconstruction optimization scheme that focuses on optimizing write-intensive workloads that have been considered a roadblock for many of the existing reconstruction approaches [38].

(3) WorkOut is orthogonal and complementary to and can be easily incorporated into most existing RAID reconstruction approaches to further improve their reconstruction performance.

(4) In addition to boosting the RAID reconstruction performance, WorkOut is very lightweight and can be easily extended to improve the performance of other background support RAID tasks, such as resynchronization and disk scrubbing, 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 outperforms the existing reconstruction approaches PR [23] and PRO [39], significantly in both reconstruction time and user response time, simultaneously.

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 through a prototype implementation are presented in Section 4 and related work is presented in Section 5. We point out directions for future research in Section 6, and conclude the paper in Section 7 by summarizing the main contributions of this paper.

2 Background and Motivation

In this section we provide the necessary background knowledge and key observations that serve to motivate our proposed research and facilitate our presentation of WorkOut in later sections.

2.1 Disk failures in the real world

Recently, extensive studies on statistical analysis, trend predications and impact evaluations of partial or complete disk failures in large-scale storage systems have revealed that disk failures are becoming the norm rather than the exception [4, 8, 19, 31, 33]. Schroeder & Gibson [33] 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 that can lead to disk replacement is 3.45% in their study. Due to the high failure rate and the increasing scale in disk-based storage systems that may contain hundreds or thousands of disks, storage systems in the near future may be perpetually repairing multiple failed disks in which recovery becomes the state of storage [8].

In addition, partial or complete disk failures exhibit a significant amount of spatial and temporal locality, indicating that, after one disk failure, another disk failure will likely occur soon [4, 8, 19]. Gibson [8] points out that the probability of a second disk failure in a RAID-structured storage system during reconstruction increases significantly along with the reconstruction time: approximately 0.5% for one hour, 1.0% for 3 hours and 1.4% for 6 hours.

Frequent or long-term downtime and data loss are obviously intolerable to the end users. Thus, given such high partial or complete disk failure rates and their significant economic impact to the users, fast recovery from disk failures becomes increasingly important for building reliable storage systems.

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

During the on-line RAID reconstruction period, reconstruction I/O requests and user I/O requests compete for the bandwidth of surviving disks and adversely affect each other. User I/O requests increase the reconstruc-

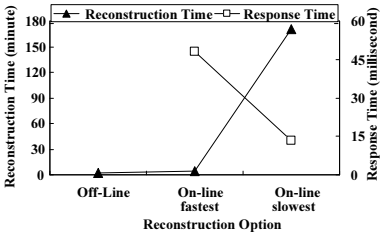


Figure 1: Reconstruction and its performance impact.

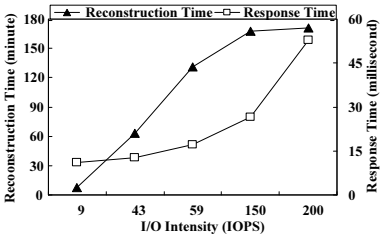


Figure 2: I/O intensity impact on reconstruction.

tion time while the reconstruction process increases the user response time.

Figure 1 shows the reconstruction times and user response times of a RAID5 disk array consisting of 5 disks with a stripe unit size of 64KB in three cases: off-line reconstruction, on-line reconstruction at the fastest speed (when RAID favors the reconstruction process) and at the slowest speed (when RAID favors the user I/O requests), respectively. In the experiment we limit the capacity of each disk to 10GB. The user I/O requests are generated by Iometer [18] with 20% sequential, 60%/40% read/write and 8KB request size [32]. As shown in Figure 1, the user response time increases significantly with the increasing reconstruction speed, 3 times more than that in the normal mode, and the on-line reconstruction process requires an amount of time that is up to 70 times more than its off-line counterpart.

How reconstruction is performed impacts both the reliability and availability of the storage system [11]. Storage system reliability is formally defined as the mean time to data loss (MTTDL) that is a function of the mean time to failure (MTTF) and the mean time to repair (MTTR). For example, MTTDL of a RAID5 disk array with D disks is given as follows [44]:

$$MTTDL_{RAID5} = \frac{(2D-1)\mu + \nu}{D(D-1)\mu^2} \qquad (1)$$

where μ is the failure rate of hard disks (equal to $\frac{1}{MTTF}$) and ν is the repair rate (equal to $\frac{1}{MTTR}$).

It must be noted that MTTR is typically no more than hundreds of hours while MTTF is typically many orders of magnitude larger than MTTR. Consequently, MTTDL (or the system reliability) increases approximately linearly with ν (or with decreasing MTTR). The system availability, formally defined as $\frac{MTTF}{MTTF+MTTR}$ [11], improves with decreasing MTTR. Importantly, increasing the user response time significantly will likely lead to a violation of SLA, an event that is considered a system failure [11], thus worsening the system reliability and availability [5, 11]. Therefore, reducing the reconstruction time and user response time simultaneously is critical for improving the reliability and availability of RAID-structured storage systems, especially as the RAID reconstruction is becoming more frequent and lengthier in large-scale storage systems.

On the other hand, user I/O intensity has a significant impact on the on-line RAID reconstruction performance, as show in Figure 2. The experimental setup is the same as that in Figure 1, except that we imposed different I/O request intensities. Moreover, the RAID reconstruction process is set to yield to user I/O requests (i.e., RAID favors the user I/O requests). From Figure 2, one can see that both the reconstruction time and user response time increase with IOPS. When IOPS of user I/O requests reaches its maximum of 200, both the reconstruction time and average user response time have also reached their respective maximum, with the former being 20.9 times its minimum value while the latter being 3.76 times its minimum value when IOPS of user I/O requests is at its minimum of 9.

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 reducing the reconstruction time and alleviating the user performance degradation simultaneously, thus improving the reliability and availability of RAID-structured storage systems.

2.3 Workload locality

Studies indicate that access locality is one of the key web workload characteristics [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]. Studies also find that there is a significant portion of the files, 28%-49%, are only accessed once [6].

As a result of access locality, caches have been widely employed to improve the storage system performance. The storage cache, 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 [9, 24, 38]. For example, based on the study on C-Miner [24] that mines block correlation below the storage cache, correlation-directed prefetching and data layout reduce the user response time by 12-25% compared with the baseline case. By utilizing the workload locality at the block level, PRO [38] reduces the reconstruction time by up to 44.7% and the user response time by 3.6-23.9% simultaneously.

Based on these observations, redirecting all read data will increase the chances that the redirected data will never be accessed in the future, choke the request queue on the surrogate RAID set and increase its overhead. Therefore, WorkOut only redirects the popular read data to the surrogate RAID set by exploiting the access locality so that future read requests to these popular data can be served by the surrogate RAID set. *Popular data in this paper is defined as the data that has 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 architecture 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, flexibility, and extendibility with data consistency guarantee, 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 avoid I/O workloads as many as possible by redirecting it away from the degraded RAID set.

Availability. To alleviate the user performance degradation and thus meet SLA, 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 means of outsourcing I/O workloads to a surrogate RAID set.

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 our current design, the surrogate RAID set can be a dedicated surrogate RAID1 set, a dedicated surrogate RAID5 set or a live surrogate RAID set that uses the free space of an operational (live) RAID set. How to choose an appropriate surrogate RAID set is based on the space overhead, performance, reliability, and maintainability requirements and trade-offs.

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 extended to improve the performance of these RAID tasks.

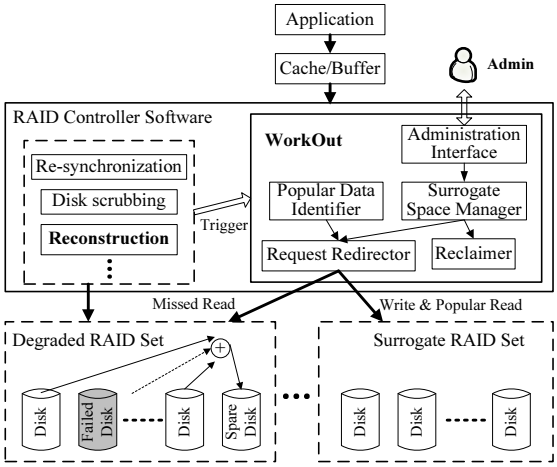


Figure 3: An architecture overview of WorkOut.

Data consistency. Redirecting the write data to a storage device with a single point of failure is unreliable since data will be lost if a failure occurs in this device. Therefore, redundant storage devices, such as a RAID set, are more appropriate for WorkOut to store the redirected data. Moreover, redirected data from different degraded RAID sets to the same surrogate RAID set must not overwrite each other, for otherwise it can be very dangerous and thus will not be acceptable to end users.

3.2 WorkOut architecture overview

Figure 3 shows an architecture overview of WorkOut. In our current design, WorkOut, working underneath the storage cache, is an augmented module to the RAID controller software and works with but is independent of the reconstruction module, so WorkOut can be incorporated into any RAID controller software with not only various reconstruction approaches but also other background support RAID tasks. In this paper, however, 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* 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 to the degraded RAID set after the reconstruction process completes. *Surrogate Space Manager* is responsible for allocating and managing a space in the surrogate RAID set for each current reconstruction process and controlling the data layout of the redirected data in the allocated space.

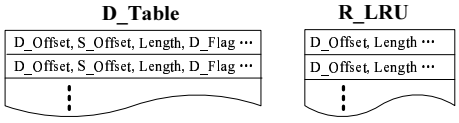


Figure 4: Data structures of WorkOut.

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 works through the entire reconstruction and reclaim periods. Moreover, the reclaim thread is triggered by the reconstruction module when the reconstruction process completes.

In WorkOut, a physical surrogate RAID set can be shared by multiple degraded RAID sets simultaneously, so space in it should be effectively managed. Based on the parameters pre-configured 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 but could 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 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 systems with multiple concurrent degraded RAID sets are 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* (a log table that manages the redirected data) and *R_LRU* (an LRU-style list that identifies the most recent reads), 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 redirected data and *D_Flag* indicates whether the redirected data is the redirected write data from the user application (*D_Flag is set to be true*) or the redirected data from the degraded RAID set to the surrogate 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.

Since WorkOut focuses on outsourcing user I/O workloads during reconstruction and does not modify the existing reconstruction algorithm, how to perform the reconstruction requests remains the job of the reconstruction module, which is dependent upon the specific reconstruction algorithm and thus not shown 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 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. Whether the fetched read data is popular and should be redirected to the surrogate RAID set are indicated by R_LRU. If a read request 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, that is, it partially hits D_Table, the read request will be split and served by both the sets. In order to make the best of the allocated space for the redirected data and improve the performance, the redirected data is laid out as sequentially as possible in 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. Since the redirected read data is already in the degraded RAID set, it needs not be reclaimed as long as logs of such data are removed 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 3.5.

During the reclaim period, all requests to the newly recovered RAID set must be checked carefully in D_Table for the purpose of data consistency. If a write request hits D_Table and its D_Flag is true, meaning that it rewrites the old data that is still in the surrogate RAID set, the corresponding log in D_Table must be deleted before 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.

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

Optional surrogate RAID set	Space 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

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 space overhead, 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 can be deployed to store the redirected data. The space management is simple while the space overhead is relatively high.

A live surrogate RAID set. WorkOut can utilize the free space of a 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 space overhead, performance, reliability and maintainability requirements 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 three aspects: (1) Redirected data must be reliably stored in the surrogate RAID set, (2) User reads must fetch the up-to-date data, and (3) 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.

During the reclaim period, the redirected data in the surrogate RAID set is protected in exactly the same way as normal data is protected in any RAID set of the same level (RAID1 or RAID5). Given the reasonable assumption that the reliability of the surrogate RAID set (RAID1 or RAID5) is at least as good as that of the newly recovered RAID set, the redirected data will not be any more likely to be lost (due to unrecoverable failures) on the former than on the latter. Therefore, it is reasonable to exclude the reclaim time from the reconstruction time in the reliability evaluation of WorkOut.

Second, 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 is popular or modified by a previous write request during reconstruction, 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 is partially modified) to keep the fetched data always up-to-date, until all the redirected write data is reclaimed.

Finally, since the content of D_Table cannot be lost during the entire period when WorkOut is activated, D_Table must be stored in a non-volatile memory 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.

4 Performance Evaluations

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

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 the 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 the user I/O requests as much as possible while it ensures 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 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., 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 obtainable for most disks, MD utilizes any disk bandwidth available for the reconstruction process.

To better examine the WorkOut performance on the 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) [23] takes advantage of the sequential property of track retrievals to pipeline the reading and writing processes. PRO (Popularity-based multi-threaded Reconstruction Optimization) [38, 39] 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 transfer 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, which does not noticeably affect the conclusions of this paper. In our prototype implementation, we use the main memory to substitute a battery-backed RAM for simplicity and for the fact that D_Table's space overhead is very small and the latter's performance is roughly the same as the former. Moreover, since battery-backed RAM has become a *de facto* standard form of NVRAM for storage controllers [7, 16, 17], the write penalty due to D_Table updates can be negligible.

Generally speaking, there are two models for trace replay in trace-driven experiments: open-loop and closed-loop [26, 34]. In an open-loop model, new I/O arrivals are independent of I/O completions, while in an closed-loop model, new I/O arrivals are only triggered by I/O completions. In this paper, we use both the open-loop model (trace replay with RAIDmeter [38, 39]) and the closed-loop model (TPC-C-like benchmark [40]) to evaluate the WorkOut performance. While 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 (the queuing delay) to grow rapidly when the storage system is slow and the request rate is high, the opposite is true for the latter as the request 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).

Table 2: The trace characteristics.

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

The traces used in our experiments are obtained from the Storage Performance Council [28, 37]. The two financial traces (or 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, while the read-dominated Web trace exhibits the locality property prominently. Since the Web trace is too intense to be tolerated by our degraded RAID set, we only use part of the Web trace attributed to device zero while the part due to device one and two is ignored. Moreover, to fully and evenly cover the address space of the RAID set from the three traces that have limited footprints, 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.

The trace-driven evaluation is based on RAIDmeter [38, 39] that is a block-level trace replay software with functions of replaying traces and evaluating 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.

We also use a TPC-C-like benchmark that is implemented with TPCC-UVA [30] and the Postgres database and generates mixed transactions based on the TPC-C specification [40] to evaluate the WorkOut performance. 20 warehouses are built on the Postgres database with the ext3 file system in the degraded RAID set. Transactions, such as PAYMENT, NEW_ORDER, DELIVERY, generate read and write requests at the RAID level. To evaluate the performance by various reconstruction schemes, we

Table 3: The reconstruction time results.

Traces	Reconstruction Time (second)						
	Off-line	PR	WorkOut+PR	speedup	PRO	WorkOut+PRO	speedup
Fin1	136.4	1121.75	203.13	5.52	1109.62	188.26	5.89
Fin2		745.19	453.32	1.64	705.79	431.24	1.64
Web		9935.6	7623.22	1.30	9888.27	7851.36	1.26

Table 4: The average user response time results.

Traces	Average User Response Time during Reconstruction (millisecond)							
	Normal	Degraded	PR	WorkOut+PR	speedup	PRO	WorkOut+PRO	speedup
Fin1	7.92	9.52	12.71	4.43	2.87	9.83	4.58	2.15
Fin2	8.13	13.36	25.8	9.69	2.66	22.97	10.19	2.25
Web	18.46	26.95	38.57	28.35	1.36	35.58	29.12	1.22

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 a RAID5 disk array consisting of 8 disks with a stripe unit size of 64KB, while running PR, PRO and WorkOut-powered PR and PRO respectively, to evaluate reconstruction times and average user response times driven by the three traces, under the minimum reconstruction bandwidth of 1MB/s, as shown in Table 3 and Table 4, respectively. We configure a dedicated RAID5 disk array consisting of 4 disks with a stripe unit size of 64KB 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, with a reconstruction time of 203.13s vs. 1121.75s for PR and 188.26s vs. 1109.62s for PRO, is because of the fact that 84% (69% of writes plus 15% of reads) of the requests 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 in the same platform. Moreover, WorkOut outsources 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.

From Table 4, one can see 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. The average user response times achieved by WorkOut are even better than that in the normal or degraded mode for the Fin1 and Fin2 traces. The reasons why WorkOut achieves significant improvement on response times are threefold. First, a significant amount of requests are redirected away from the degraded RAID set during reconstruction, as shown in Figure 5. The response times of requests redirected to the surrogate RAID set 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 as sequentially as possible in the surrogate RAID set, thus further speeding up the user response time. Third, since many requests are redirected away from the degraded RAID set, the I/O queue on the degraded RAID set is reduced 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 much lower than that without WorkOut, especially for the Fin1 trace.

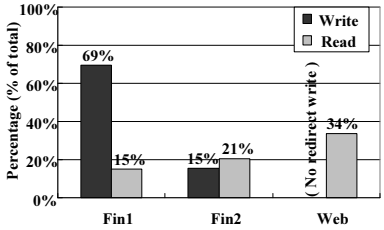


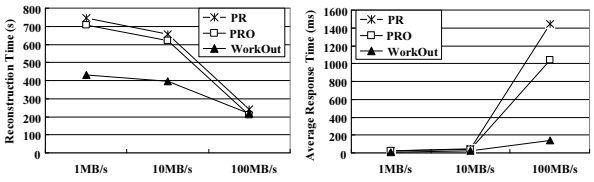
Figure 5: Percentage of redirected requests for WorkOut, under the minimum reconstruction bandwidth of 1MB/s.

From Table 3 and Table 4, one can also see that WorkOut-powered PRO performs similarly to WorkOut-powered PR. The reason is that WorkOut has redirected all write requests and popular read requests to the surrogate RAID set, thus depleting the amount of popularity of I/O workloads remained in the degraded RAID set that can be exploited by PRO. Based on this observation, in the following experiments, we only compare WorkOut-powered PR (or WorkOut) with PR and PRO.

4.4 Sensitivity study

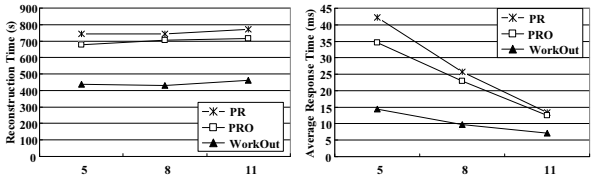
The WorkOut performance is influenced by several important factors, including available reconstruction bandwidth, the size of the degraded RAID set, the stripe unit size, and the RAID level. In what follows we study their sensitivity to and impact on WorkOut quantitatively through trace-driven results of the Fin2 trace due to space limit and the fact that 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 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 a RAID5 disk



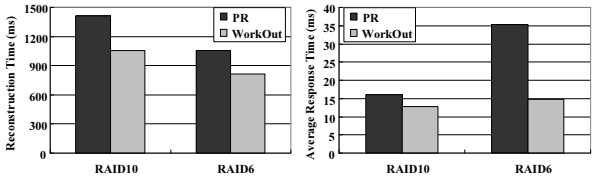
(a) Reconstruction time (b) Average user response time

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.



(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.

array consisting of 8 disks with a stripe unit size of 64KB driven by the Fin2 trace.

From Figure 6(a), one can see that, with a lower minimum reconstruction bandwidth, WorkOut speeds up the reconstruction time more significantly than with a higher minimum reconstruction bandwidth. The reason is that the reconstruction process exploits all the available disk bandwidth when the reconstruction bandwidth is higher, thus leaving very small room for the reconstruction time to be improved.

From Figure 6(b), in contrast, the user response time increases rapidly with the increasing minimum reconstruction bandwidth both for 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, meaning that 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, for PR and PRO, the user response time during reconstruction is so long that it will likely violate SLA and thus become unacceptable to the 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 disk arrays consisting of different number of disks (5, 8 and 11) with a stripe unit size of 64KB under the minimum reconstruction bandwidth of 1MB/s, driven by the Fin2 trace. Figure 7 shows the experimental results with respect to the different numbers of disks for PR, PRO and WorkOut.

From Figure 7(a) and 7(b), one can see that, for all the three approaches, the reconstruction time increases while the user response time decreases with the increase in the number of disks of the degraded RAID set. The reason is that more disks in a RAID set not only imply a larger RAID group size and thus more disk read operations by a construction request, 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 a RAID5 disk array consisting of 8 disks with stripe unit sizes of 16KB and 64KB respectively, driven by the Fin2 trace. The experimental results show that WorkOut outperforms PR and PRO in both the reconstruction time and average user response time for both the two 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 RAID10 disk array consisting of 4 disks and a RAID6 disk array consisting of 8 disks with the same stripe unit size of 64KB under the minimum reconstruction bandwidth of 1MB/s, driven by the Fin2 trace. 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 disk arrays. The difference in the amount of improvement on reconstruction time and user response time is caused by the different user I/O intensities on the RAID10 and RAID6 disk arrays that have different numbers of disks. The user I/O intensity on individual disks in the RAID10 disk array is obviously higher than that in the RAID6 disk array, thus making the reconstruction time in the RAID10 disk array higher than that in the RAID6 disk array.

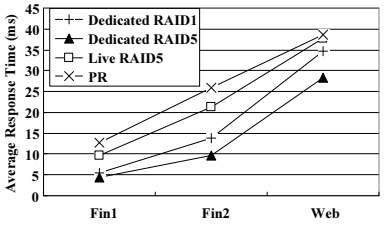
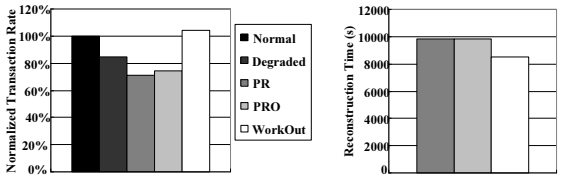


Figure 9: A comparison of average user response times for different types of surrogate RAID set.



(a) Transaction Rate (b) Reconstruction time

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

On the other hand, since each read request to the failed disks in a RAID6 disk array 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 alleviated by WorkOut due to its external I/O outsourcing. For the RAID10 disk array, however, the situation is quite different, where the read data can be directly returned from the surviving disks instead of needing to be rebuilt on-the-fly, which explains why WorkOut does not provide the same amount of improvement on the user response time for RAID10 as it does 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 RAID5 disk array consisting of 4 disks). Similar to the experiments conducted in the PARAID [41] and write off-loading [27] studies, we reserve the 10% portion of storage space at the end of the live RAID5 set to be used by WorkOut to store the redirected data. The degraded RAID set is a RAID5 disk array composed of 8 disks with a stripe unit size of 64KB and under the minimum reconstruction bandwidth of 1MB/s.

The experimental results show that the reconstruction times of WorkOut are almost the same for the three types of surrogate RAID sets and outperform PR as expected, shown in Table 3, since WorkOut outsources the same amount of requests during reconstruction. Whereas, the 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.

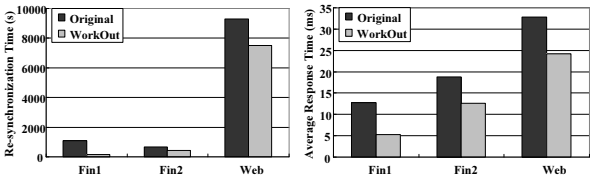
From Figure 9, one can see that the dedicated surrogate RAID sets (both RAID1 and RAID5) outperform the live surrogate RAID set in user response time. The reason is that all disk resources in the former are dedicated to serving the redirected requests, while in the latter the redirected requests must compete for the resources with the native I/O requests on the live RAID set, causing the disk heads to seek back and forth to serve the two different types of requests, and thus increasing the response time of the redirected requests. 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 a RAID5 disk array consisting of 8 disks with a stripe unit size of 64KB driven by the TPC-C-like benchmark under the minimum reconstruction bandwidth of 1MB/s.

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. As 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.

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 of WorkOut is 8526 seconds, with approximately 15% improvement over PR and PRO. The main reason why WorkOut gains much less in reconstruction time with the benchmark-driven experiments than with the trace-driven experiments lies in the fact that the very high I/O intensity of the benchmark application constantly pushes the disk array 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.



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

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

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 a RAID5 disk array consisting of 8 disks 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 RAID5 disk array consisting of 4 disks with a stripe unit size of 64KB as the surrogate RAID set. The evaluation 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, the 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

Memory overhead. To prevent data loss, WorkOut uses non-volatile memory to store D_Table, thus incurring extra memory overhead. When the minimum reconstruction bandwidth is set to 1MB/s, the reconstruction time is the longest and the amount of redirected data is the largest, thus consuming correspondingly the largest amount of memory. In the above experiments on the RAID5 disk array 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 cache size and decrease in cost of non-volatile memories, this memory overhead is arguably reasonable and acceptable to the 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 disk array layout, they are easy to be shared by different RAID levels. Moreover, the added code is not interactive with the reconstruction module, so it is easy to be modified to optimize the other background support RAID tasks, only needing to modify the corresponding triggered flag of WorkOut. Due to this independent characteristic of the WorkOut module, it is portable to other software RAID implementations in other operating systems.

5 Related Work

Reconstruction algorithms for RAID-structured storage systems, such as DOR (Disk-Oriented Reconstruction) [12], PR [23], PRO [38] and others [3, 13, 15, 21, 42, 43], have been extensively studied. However, they mostly focus on optimizing the reconstruction workflow [12, 23] or reconstruction sequence [3, 38] inside a single RAID set and try to achieve a trade-off between the reconstruction bandwidth and the I/O serving bandwidth to satisfy the requirements of end users. By utilizing the file system's semantic knowledge, Sivathanu et al. proposed a live-block recovery method in D-GRAID [36] that only reconstructs live data to the hot spare disk from the viewpoint of file systems.

By reorganizing the data layout of RAID, the reconstruction performance can be improved. Parity declustering [13] decreases the parity group size to boost scalable RAID rebuild rates. The client-driven rebuild approach [42] 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 consisting of hundreds of thousands of disk drives, FARM [43] exploits excess disk capacity to reduce the recovery time, but still faces the bandwidth competition between the recovery and user I/O requests.

WorkOut, however, focuses on outsourcing I/O workloads away from the degraded RAID set during reconstruction, and, to the best of our knowledge, optimizes the write-intensive workloads that are by and large ignored by existing reconstruction approaches [38]. Moreover, WorkOut is orthogonal to and can be easily incorporated into most existing reconstruction approaches to accelerate the reconstruction process and alleviate the performance degradation simultaneously. More importantly, it can also be extended to improve the performance of other background support RAID tasks.

Our study is inspired by the write off-loading [27] and data migration [2, 20, 22, 25, 41] techniques but with distinctively different characteristics as shown in Table 5. First, write off-loading [27] redirects write requests at the block level from one volume to another, to significantly prolong the idle period for one volume disks to spin down for a longer time to improve energy efficiency. Second, data migration [25], defined as moving data from one

Table 5: A comparison of systems related to WorkOut.

Scheme	Purpose	Environment	Data Migration		
			What	When	Where
Write off-loading [27]	Energy efficiency	Enterprise storage systems	Write data	Write-dominated period	From spin-down volumes to spin-up volumes
PARAID [41]	Energy efficiency	Single RAID set	All data	Gear shifting	From spin-down disks to active disks
Data reallocation [2]	Performance	Multidisk systems	All data	Dynamic to workloads	From busy disk/volumes to less busy ones
Cuckoo [22]	Performance	Clustering servers	Frequently-read, rarely-updated files	N/A	From busy servers onto others
User-centric migration [20]	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 during reconstruction	From a degraded RAID set to a surrogate RAID set

storage device to another for the purpose of load balancing (or load concentration), failure recovery, system expansion, or other reasons, has been used to improve the energy efficiency [41] and the storage system performance, such as data reallocation in the products of EMC's Symmetrix family of disk array [2], read request offloading in Cuckoo [22] and the user-centric data migration in networked storage systems [20].

Different from write off-loading and data migration, WorkOut, for the purpose of improving the on-line reconstruction performance, temporarily redirects the write data and popular read data from the degraded RAID set to a surrogate RAID set during reconstruction and reclaims the redirected write data back to the newly recovery RAID set after the reconstruction process completes.

6 Future Work

WorkOut is an ongoing research project. There is still a rich design space to explore and room to further optimize WorkOut. Possible directions for future work include, but are not limited to the following.

Extendibility. Due to the complicated failure characteristics of RAID-structured storage systems, in addition to the RAID reconstruction and re-synchronization, some other background support RAID tasks, such as disk scrubbing and block-level backup and snapshot, are performed to prevent data loss and protect data integrity. Very similar to the case of the RAID reconstruction, external I/O workloads have a great impact on the performance of these support RAID tasks, and vice versa. We will conduct experiments to measure the impact of WorkOut on these tasks under more benchmarks.

Free space on a live surrogate RAID set. In the current implementation, we configure a reserved space instead of the free space in a live RAID set as the surrogate RAID set, which can be impractical and inflexible. Utilizing the free space in 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. To make WorkOut more transparent to the file system, and more effectively utilize the free space in 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 [35] and apply them in WorkOut to improve its performance.

7 Conclusion

In this paper, we propose a novel scheme, called *WorkOut* (I/O *Work*load *Out*sourcing), for significantly boosting the RAID reconstruction performance, which uses a request redirection technique to outsource a significant amount of I/O requests away from the degraded RAID set to a surrogate RAID set during reconstruction, thus significantly speeding up the reconstruction time and user response time.

We carried out a comprehensive performance evaluation of WorkOut by implementing a lightweight prototype of WorkOut in the Linux software RAID and conducting a number of trace-driven and benchmark-driven experiments. The experimental results demonstrate that, compared with the existing reconstruction algorithms PR and PRO, WorkOut speeds up the reconstruction time by a factor of up to 2.87, with an average of 2.09, and speeds up the average user response time by a factor of up to 5.89, with an average of 2.88, simultaneously.

In conclusion, this paper makes the following main contributions:

- We propose WorkOut to outsource I/O workloads away from the degraded RAID set during reconstruction, thus significantly improving the on-line reconstruction performance.
- We conduct comprehensive experiments on our lightweight prototype implementation to evaluate the WorkOut performance and its sensitivity to a number of workloads and system parameters.
- We provide useful design insight and guidance for storage system designers and administrators by exploiting three WorkOut design options based on their space overhead, performance, reliability, and maintainability 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.

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*, 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] A. Brown and D. A. Patterson. Towards Availability Benchmarks: A Case Study of Software RAID Systems. In *USENIX'00*, Jun. 2000.

[6] 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.

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

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

[9] B. S. Gill. On Multi-level Exclusive Caching: Offline Optimality and Why promotions are better than demotions. In *FAST'08*, Feb. 2008.

[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] M. Holland, G. Gibson, and D. P. Siewiorek. Architectures and Algorithms for On-Line Failure Recovery in Redundant Disk Arrays. *Journal of Distributed and Parallel Databases*, 2(3):295–335, Jul. 1994.

[15] 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.

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

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

[18] Iometer. http://sourceforge.net/projects/iometer.

[19] 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.

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

[21] 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.

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

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

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

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

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

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

[30] 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.

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

[32] A. Riska and E. Riedel. Disk Drive Level Workload Characterization. In *USENIX'06*, Jun. 2006.

[33] 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.

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

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

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

[41] 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.

[42] 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.

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

[44] 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.