2010 International Conference on Distributed Computing Systems

RoLo: A Rotated Logging Storage Architecture for Enterprise Data Centers

Yinliang Yue[†‡], Lei Tian[†‡§], Hong Jiang[§], Fang Wang[†‡], Dan Feng[†‡], Quan Zhang[†‡], Pan Zeng[†‡]
[†] School of Computer Science & Technology, Huazhong University of Science & Technology
[‡] Wuhan National Lab for Optoelectronics
[§] Department of Computer Science & Engineering, University of Nebraska-Lincoln
{yueyinliang, leitian.hust}@gmail.com, jiang@cse.unl.edu, {wangfang, dfeng}@hust.edu.cn
Corresponding Author: Fang Wang

***Abstract*—We propose *RoLo* (*Ro*tated *Lo*gging), a new logging architecture for RAID10 systems for enhanced energy efficiency, performance and reliability. By spreading destaging I/O activities among short idle time slots and proactively reclaiming the stale logging space, RoLo rotates loggers among a logical logging space pool formed collectively from the free storage space available among mirrored disks. Therefore, without the extra dedicated log disks and the corresponding centralized logging, RoLo eliminates the additional hardware and energy costs, potential single point of failure and performance bottleneck. Furthermore, RoLo prolongs the lifecycle of the disks and improves the system's energy efficiency by reducing the disk spin up/down frequency. We develop three flavors of RoLo, that is, RoLo-E/R/P, to emphasize energy efficiency, reliability, and performance respectively. Extensive trace-driven evaluations demonstrate the advantages of the three RoLo schemes over both a RAID10 system with centralized logging architecture and a typical RAID10 system.**

I. Introduction

Recent studies report that energy costs may increase from 10% of the TCO (Total Cost of Ownership) of data centers to over 50% in the next few years [14]. Storage subsystems are a major contributor to the energy consumption of data centers. For a typical data center, the disk-based storage subsystem can consume 27% of the total energy and this fraction tends to increase rapidly as storage requirements rise by 60% annually [37].

Hierarchical storage architectures are widely adopted in the modern storage systems to judiciously cache/buffer some data blocks for enhanced I/O performance or energy efficiency [8] [10] [17] [21] [38]. Write requests tend to dominate disk I/O activities, since most read requests are absorbed by multi-level storage caches, while modified data blocks must be written to their targeted disks eventually to ensure data integrity. Logging write requests is one of the main approaches to improving energy efficiency and performance of storage subsystems [8] [10] [17] [21] [22] [27] [31]. Temporarily redirecting write requests to log disks enables the write-targeted disks to switch to and remain in the low-power state for a longer period of time to save energy. Since frequently spinning up/down disks will negatively impact the disk lifecycle and energy efficiency, only when the capacity occupancy of the log disks reaches the predefined threshold can the inconsistent data blocks on write-targeted disks be updated, a process known as *destaging*. We refer to the period during which write requests are redirected to log disks as *logging period*, and the period during which the inconsistent data blocks on write-targeted disks are updated as *destaging period*. There are several inherent shortcomings associated with the conventional centralized logging architecture. On the one hand, the centralized destaging, which incurs bursty I/O activities, prevents the system's energy efficiency from being further improved by simply increasing the available logging space (see section II). On the other hand, the extra dedicated log disks not only incur additional hardware and energy costs, but also become a potential performance bottleneck and single point of failure. Unfortunately, to the best of our knowledge, none of the existing logging schemes has overcome these shortcomings, especially for the RAID10 storage architecture that is widely deployed in modern data centers.

In this paper, we propose a new logging architecture RoLo (*Ro*tated *Lo*gging), which combines *decentralized destaging* with *rotated logging*, for the RAID10 storage architecture. RoLo organizes the available free space of all mirrored disks into a logical logging space pool, and employs decentralized destaging to spread and dilute destaging I/O activities among short idle time slots during each logging period, thus proactively and non-intrusively reclaiming the stale logging space. With this simultaneous consumption and reclamation of the logger capacity, logging can be continuously rotated among the logical logging space pool that becomes a virtually unlimited logging capacity. Hence, by circularly reusing logical logging space of the existing mirrored disks instead of extra dedicated log disks, RoLo eliminates the additional hardware and energy costs, potential single point of failure and performance bottleneck of the conventional centralized logging architecture. Furthermore and most importantly, RoLo can prolong the lifecycle of write-targeted disks and improve the system's energy efficiency by reducing the disk spin up/down frequency.

More specifically, three flavors of RoLo are developed in

1063-6927/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDCS.2010.22

this study, namely, a performance-oriented RoLo (RoLo-P), a reliability-oriented RoLo (RoLo-R) and an energy-oriented RoLo (RoLo-E), to suit the needs of different application environments. In RoLo-P, all the primary disks are set as ACTIVE all the time to guarantee read performance, while one or more *mirrored disks* are set as on-duty log disks at a time. In RoLo-R, all the primary disks are also set as ACTIVE like RoLo-P, while one or more *mirrored disk pairs* are set as on-duty log disks at a time to further lower the probability of data loss. RoLo-E utilizes one or more *mirrored disk pairs* to buffer all the write data blocks and popular read data blocks, and thus spin down as many inactive mirrored disk pairs as possible to significantly reduce power consumption.

Extensive trace-driven evaluations demonstrate the distinctive advantages of these three RoLo schemes. RoLo-E achieves up to 81.7% energy saving over a RAID10 system, although at the expense of incurring wide performance swings at times. RoLo-P and RoLo-R save up to 47.2% power than the RAID10 system at the cost of 0.71% ∼5.09% performance degradation. More importantly, when compared with GRAID [21], a representative RAID10 system with the centralized logging architecture, RoLo-P/R save up to 11.81% energy and RoLo-E saves up to 69.37% energy. Among the three RoLo schemes, RoLo-P provides the best I/O performance in most cases while RoLo-R achieves the highest reliability in terms of a combined measure of MTTDL and disk-spin frequency at the cost of 3.77% ∼ 4.35% performance degradation over RoLo-P (see Table IV for more detailed results).

The rest of this paper is organized as follows. Background and motivation are presented in Section II. We describe the design of RoLo in Section III and analyze its reliability in Section IV. Performance evaluations through extensive trace-driven experiments are presented in Section V. We present related work in Section VI and conclude the paper in Section VII by summarizing the main contributions of this paper and pointing out the future work.

II. Background and Motivation

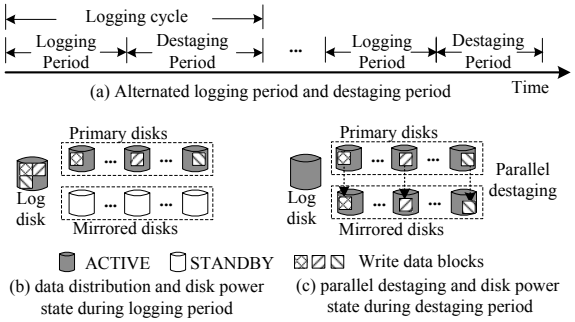


Figure 1. Working principles of centralized logging and destaging in RAID10 storage architecture

Conventional logging architectures usually employ one or more extra dedicated disks as logger to redirect write requests among write-targeted disk(s) to log disk(s). Logger can improve not only system performance [17], but also energy efficiency of storage systems [10][21]. For example, GRAID [21], which augments one extra dedicated log disk and spins down redundant mirrored disks to save energy, is one representative logging architecture for RAID10 systems. In such an architecture, shown in Figure 1(a), each *logging cycle* is composed of a *logging period* and its corresponding *destaging period*. During the logging period, all the mirrored disks are set to the STANDBY state, and the two copies of each write data are written to the corresponding location in primary disks and sequential location in the log disk respectively (see Figure 1(b)). The destaging process is triggered once the occupancy level of the log disk reaches a predefined threshold value (e.g., 80%). During the destaging period, all the mirrored disks are set to the ACTIVE state and *all the inconsistent data blocks in mirrored disks are updated in parallel from the corresponding primary disks instead of log disk(s)* (see Figure 1(c)) for better destaging performance.

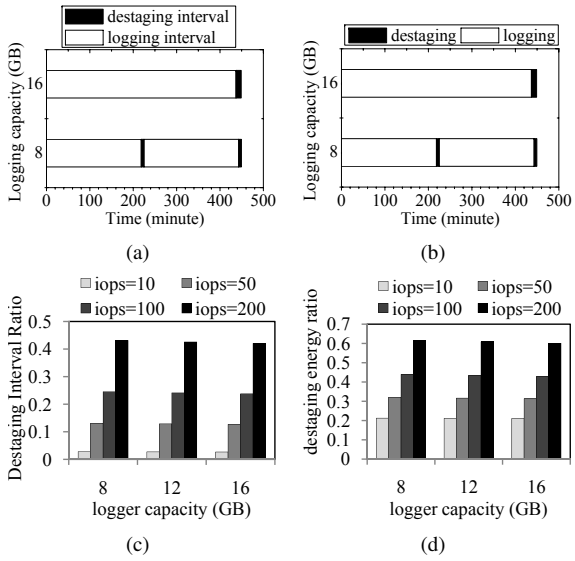
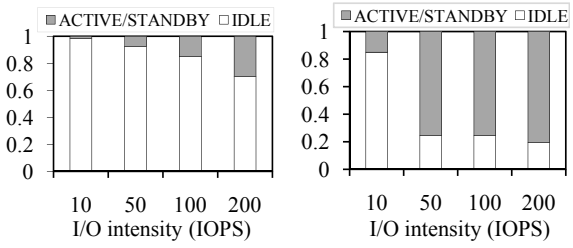


Figure 2. Impact of logger capacity on (a) time interval, (b) energy consumption, (c) destaging interval ratio, and (d) destaging energy ratio

To identify the performance and energy bottleneck of the conventional logging architecture, we define *destaging interval ratio* and *destaging energy ratio* to be the fractions respectively of time spent and energy consumed by destaging during each logging cycle. Clearly, the first ratio measures performance impact of destaging since destaging I/Os compete for disk bandwidth with user I/Os, while the second reflects the energy impact of destaging. We built a DiskSim-based [2] simulator to simulate a RAID10 system

with the centralized logging architecture, having 10 pairs of mirrored disks and one extra dedicated log disk with a stripe unit size of 64KB. User I/O workloads are generated with 70% random and 100% write requests of 64KB each. These parameters are reasonable, since we are only concerned with the write I/O activities, and the extensive experimental results demonstrate that I/O sequentiality has no effect on the conclusions. We then conducted extensive experiments to study the centralized logging architecture and obtained the following important observations.

Observation *Simply increasing the logging space alone will not decrease destaging interval ratio and destaging energy ratio, as shown in Figures 2(c) and 2(d).*
Reasons: The increased amount of logged data in a larger logger will prolong both the logging period and destaging period proportionally and thus increase their corresponding energy consumption simultaneously, as shown in Figures 2(a) and 2(b).
Implications: Simply increasing the available logging space is not a viable power saving solution and this inability to improve energy efficiency by increasing logging space stems from the centralized destaging strategy of the conventional logging architecture.



(a) For primary disks (b) For log disks

Figure 3. The proportion of IDLE duration and that of ACTIVE and STANDBY duration under different I/O intensities

There are two types of "free" resources that have been exploited in storage subsystems of typical data centers: *unused storage space* and *idle time slots*. In Symantec's 2008 State of the Data Center survey [30], it was found that data centers typically utilize 50% of their storage capacity. Mark Levin [20] points out that on average the disk storage utilization is 56.6% and 46.4% for UNIX and Windows environments respectively under locally attached storage deployment, and this proportion increases to 75.5% and 55.8% under SAN deployment. Short idle time slots are abundant for both primary disks and log disk(s) during logging periods under light workloads as indicated by our experimental results, shown in Figure 3, and other studies [24]. Most idle time slots are much shorter than the break-even time for modern disks to spin down to save power [24].

While either idle time slots or free storage space has been exploited to improve performance [18], reliability [24], or energy efficiency [33] of storage systems, these resources remain to be effectively and fully tapped to optimize the performance and energy efficiency of storage systems with a logging architecture.

Given the important observations above and the availability of the aforementioned free resources, we are motivated to propose a new rotated logging architecture, RoLo. It integrates the unused free space of redundant mirrored disks in a RAID10 system into a large logical logging space pool, which is circularly recyclable by exploiting the short idle time slots for decentralized destaging to improve both performance and energy efficiency of the system. In other words, one or a few mirrored disks take turns to serve as on-duty log disks, while off-duty mirrored disks can be spun down to save energy. The basic model of RoLo is shown in Figure 4. At the same time, idle time slots are exploited to spread and dilute the bursty destaging I/O activities. With this decentralized destaging strategy, the stale logging space can be proactively and non-intrusively reclaimed, thus enabling logging to be unlimitedly rotated among the logical logging space pool. The design of the RoLo architecture is detailed next.

III. RoLo Architecture and Design

A. The basic RoLo idea

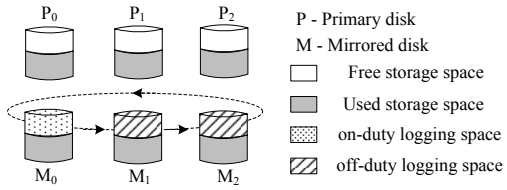


Figure 4. The basic model of RoLo

Rotated Logging:

Any new data is written to two disks in order to prevent data loss, with one copy written to the primary disk as in the standard RAID10 fashion while the second copy is sequentially written to the on-duty logging space. When the available free storage space of the on-duty log disk decreases to a predefined threshold with the logged data, the on-duty logger is rotated to the next mirrored disk, and this will end the current logging period and start a new one. Each logger rotation triggers a new destaging process for the newly on-duty log disk to update the inconsistent data blocks from its corresponding primary disk by spreading destaging I/O activities among the short idle time slots, as shown in Figures 5(a), (c) and (d). For instance, M_0 is used as the on-duty log disk in logging period T_0, and the second copies of newly written data in T_0, i.e., $\mathrm{D}_0\mathrm{T}_0$, $\mathrm{D}_1\mathrm{T}_0$ and $\mathrm{D}_2\mathrm{T}_0$, are written to M_0. Similarly, when entering logging period T_1, $\mathrm{D}_0\mathrm{T}_1$, $\mathrm{D}_1\mathrm{T}_1$ and $\mathrm{D}_2\mathrm{T}_1$ are written to M_1, and $\mathrm{D}_0\mathrm{T}_2$, $\mathrm{D}_1\mathrm{T}_2$ and $\mathrm{D}_2\mathrm{T}_2$ are written to M_2 during T_2. The dashed lines with arrows and data layout presented in Figures 5(b), (c) and (d) show the basic principles of the logger rotation mechanism.

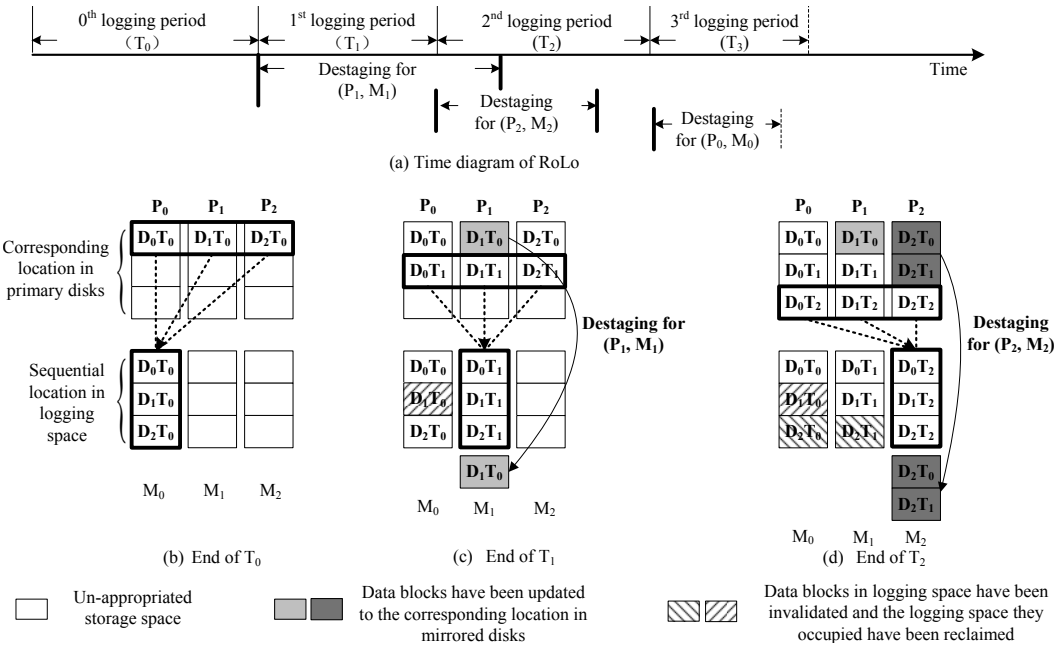


Figure 5. The decentralized destaging process of RoLo. $\mathrm{P_i}$ presents the $\mathrm{i^{th}}$ primary disk, and $\mathrm{M_i}$ represents the $\mathrm{P_i}$'s mirrored disk. The $\mathrm{D_mT_n}$ represents the newly written data for the $\mathrm{m^{th}}$ mirrored disk pair $(\mathrm{P_m}, \mathrm{M_m})$ during the $\mathrm{n^{th}}$ logging period $\mathrm{T_n}$.

Decentralized Destaging:

Spreading destaging I/O activities among idle time slots during logging periods is the basic idea of decentralized destaging. A new destaging process is triggered only when the logger rotates to a new log disk and it will not finish until all the inconsistent data blocks in the mirrored disks have been updated. For example, the destaging process for $(\mathrm{P_1}, \mathrm{M_1})$ is triggered immediately when the logger rotates to $\mathrm{M_1}$. The only responsibility of this destaging process is to update all the inconsistent data blocks from $\mathrm{P_1}$ to $\mathrm{M_1}$. As shown in Figure 5(c), only when all the inconsistent data blocks in $\mathrm{M_1}$ have been updated from $\mathrm{D_1T_0}$ in $\mathrm{P_1}$, can this destaging process be terminated. Note that data block $\mathrm{D_1T_0}$ in $\mathrm{M_0}$ becomes invalid and the corresponding stale logging space it occupies is reclaimed when the destaging process for $(\mathrm{P_1}, \mathrm{M_1})$ finishes. Similarly, the destaging process for disk pair $(\mathrm{P_2}, \mathrm{M_2})$ is triggered immediately after $\mathrm{M_2}$ is selected as the on-duty log disk. Data blocks $\mathrm{D_2T_0}$ and $\mathrm{D_2T_1}$ are updated from $\mathrm{P_2}$ to $\mathrm{M_2}$, and the logging space occupied by $\mathrm{D_2T_0}$ and $\mathrm{D_2T_1}$ in $\mathrm{M_0}$ and $\mathrm{M_1}$ are reclaimed after the destaging process for $(\mathrm{P_2}, \mathrm{M_2})$ is completed. The solid lines with arrows and rectangles with twills in Figure 5(c) and (d) show the decentralized destaging and proactive reclaiming mechanism respectively. Since most of the stale logging space of $\mathrm{M_0}$ has been proactively reclaimed during periods $\mathrm{T_1}$ and $\mathrm{T_2}$, the logger can be rotated onto $\mathrm{M_0}$ from $\mathrm{M_2}$ again. Note that the logging space occupied by $\mathrm{D_0T_0}$ in $\mathrm{M_0}$ will be reclaimed in $\mathrm{T_3}$, during which destaging I/O activities are issued for $(\mathrm{P_0}, \mathrm{M_0})$.

Note that the length of individual logging period or destaging period varies since it depends on the foreground user write intensity and available on-duty logging space. Heavy foreground user write intensity not only shortens the logging period, but also consumes more disk bandwidth and leaves less available free disk bandwidth for destaging I/O activities, which in turn leads to longer destaging periods. Further, less available free space on the on-duty log disk also shortens the logging period. Even though the logger has been rotated to a new log disk, the unfinished destaging process for the last log disk will continue until all the inconsistent data blocks in the last log disk are updated, as shown by the comparison between the 1^{st} logging period and destaging for $(\mathrm{P_1}, \mathrm{M_1})$ in Figure 5(a). Also, the destaging process will be terminated before logger's rotating to a new log disk (as shown by the comparison between the 2^{nd} logging period and destaging for $(\mathrm{P_2}, \mathrm{M_2})$ in Figure 5(a)), if it obtains more free disk bandwidth.

Note that the priority of the background destaging I/O activities is always lower than that of the foreground user I/O activities, and only free disk bandwidth is utilized by the background destaging I/O activities. Thus the background destaging I/O activities have little, if any, impact on the foreground user I/O performance, no matter when the destaging process is terminated.

B. RoLo-P/R/E

Based on the above principles of logger rotation and decentralized destaging, three flavors of RoLo are proposed in this paper, named the Performance-oriented RoLo (RoLo-

P), the Reliability-oriented RoLo (RoLo-R) and the Energy-oriented RoLo (RoLo-E) for respectively specific application scenarios.

1) RoLo-P: In RoLo-P, all the primary disks are set to the ACTIVE/IDLE state to guarantee that all the read requests are serviced without any disk spin-up latency, i.e., the latency caused by the disk's state switching from STANDBY to ACTIVE. The disk spin-up latency, which is often more than ten seconds for enterprise-level hard drives [4], is unacceptable for most applications.

In RoLo-P, one or a few *mirrored disks* take turns to serve as on-duty log disks, while off-duty mirrored disks can be spun down to save energy, as shown in section III-A. Note that there are *two* copies of each new data block. One copy is written to the corresponding location in the primary disk, and the other copy is written to the sequential location in the logging space of the on-duty log disk(s), such as M_0.

2) RoLo-R: Mindful of the fact that the reliability of RoLo-P is slightly lower than that of a RAID10 system (see section IV), RoLo-R is proposed to provide higher reliability by designating one or a few *mirrored disk pairs* as the on-duty logger at a time and writing *three* copies for each new data block. One copy is written to the corresponding location in the targeted primary disk, and the other two copies are written to the two disks in one mirrored disk pair that serve as the on-duty logger, e.g., $(\mathrm{P}_0, \mathrm{M}_0)$, respectively. Note that all the primary disks are also set to the ACTIVE/IDLE state for the same purpose as in RoLo-P.

3) RoLo-E: Motivated by the fact that there are no or very few read requests for some applications, such as storing checkpointing data sets in high performance computing environments [9] and online transaction applications [11], we believe that it is not necessary to keep all the primary disks ACTIVE/IDLE all the time. Thus we propose RoLo-E to utilize one or several *mirrored disk pairs*, such as $(\mathrm{P}_0, \mathrm{M}_0)$, as log disks at a time and spin down all the other mirrored disk pairs to the low-power STANDBY state. *Two* copies of each new data block are written to both the logging space of the primary disk and that of the mirrored disk respectively. To further alleviate the long response time of some reads, we cache popular read data blocks in the on-duty logging space to avoid the passive and expensive disk spin up/down caused by read misses.

When all the logging space has been filled up, all the disks in the STANDBY state should be spun up for centralized destaging. Since only the on-duty log disks are set to ACTIVE for write requests, there is no decentralized destaging during the logging period. Note that RoLo-E is only suitable for write-dominant workloads. This rotated logging architecture without the advantages of decentralized destaging and proactive reclamation of logging space can still be very useful.

C. Disk Failure Recovery

Disk failures can occur either in primary disks or mirrored disks. For the sake of simplicity and without loss of generality, only *one mirrored disk* in RoLo-P or *one mirrored disk pair* in RoLo-R/E is assumed to be set as the on-duty log disk(s) at any given time. All the other mirrored disks in RoLo-P/R or mirrored disk pairs in RoLo-E are set to STANDBY to save energy.

When one disk fails, only the disks that are essential for data recovery are spun up. In RoLo-E, only the mirrored disk (or primary disk) are "*silently*" spun up in case of a primary disk (or mirrored disk) failure, since the mirrored disk (or primary disk) has already been set in the ACTIVE state. In RoLo-P/R, the failure of an on-duty log mirrored disk, e.g., M_0, will trigger the "*silent*" spinning up of its corresponding primary disk, e.g., P_0, which is ACTIVE all the time. A failure of any primary disk in RoLo-P/R will not only "*silently*" spin up its corresponding mirrored disk but also awaken several other mirrored disks, which are the on-duty log disks during the previous several logging periods according to the rotated logging mechanism.

D. Scalability

The unique features of the RoLo architecture have the following beneficial impacts on system scalability.
Alleviated performance bottleneck: Since the peak bandwidth of on-duty log disk(s) can be interleaved with the I/O workloads by exploiting short idle time slots and adjusting the number of on-duty log disks, the performance bottleneck imposed by centralized logging can be significantly alleviated.
Elimination of single point of failure: Since another mirrored disk can be immediately selected as the new on-duty log disk to replace a failing or failed on-duty log disk to guarantee a non-interrupted logging service, the single point of failure in the dedicated log disk imposed by the centralized logging architecture is eliminated.

E. Implementation Issues

Free space management: We divide the mirrored disk space into two logical parts: *data region* and *logger region*. Since the logger region is not necessarily sequential in logical address and may be split into multiple sub regions, we adopt two linked lists, called *used logger region list* and *unused logger region list*, to keep track of the addresses of used logger region and unused logger region respectively. If the existing data region is full, one unused logger region will be freed from the unused logger region list to expand the data region. In order to keep the continuity of unused logger region, a background thread will try to combine the multiple data regions into one sequential data region. This work can be done during the logger rotation, for the occupied logger space is reclaimed along with the logger rotation, as presented in section III-A. If the destination

address of the newly written data has been occupied by the logger data, another available address will be assigned to this newly written data and the corresponding address information will be tracked. The offloaded data will be moved to its destination address later. If free space of all the mirrored disks decreases to a predefined threshold, say, 5%, RoLo will be de-activated and all the disks in the RAID10 disk array must be spun up. As long as there is sufficient free space, RoLo can be re-activated.

IV. Reliability Analysis

To estimate the reliability of RoLo-P/R/E, we adopt the Mean Time To Data Loss (MTTDL) metric that is widely used in the reliability analysis of storage systems. Our system model consists of a disk array of four disks (i.e., two mirrored disk pairs, P_0, M_0, P_1 and M_1). When a disk fails, a repair process is immediately initiated for data recovery. We assume that disk failures are independent events following an exponential distribution of rate λ, and the repairs follow an exponential distribution of rate μ. For the sake of simplicity and without loss of generality, we assume that only *one* mirrored disk is used as an on-duty log disk at a time in RoLo-P and only *one* mirrored disk pair is used as on-duty log disks in RoLo-R/E. According to the conclusion on the MTTDL results of RAID10 [35], the MTTDL of RAID10 consisting of four disks is:

$$MTTDL_{RAID10-4} \approx \frac{3\lambda+\mu}{4\lambda^2} \quad (1)$$

and the MTTDL of GRAID consisting of four data disks and one dedicated log disk is [21]:

$$MTTDL_{GRAID-5} \approx \frac{17\lambda+2\mu}{12\lambda^2} \quad (2)$$

Recent studies show a significant amount of correlation in drive failures, indicating that after one disk fails another disk failure will likely occur soon [6] [19]. Therefore, Equations (1) and (2) tend to overestimate MTTDL. Since modeling this failure correlation is both very difficult and secondary to the presentation of RoLo, we will not address this issue in this paper. Similarly, we will not consider the impact of latent sector errors on the system model for simplicity.

In the state transition diagram shown in Figure 6, Figure 7 and Figure 8, each transition (an arrow) due to a disk failure is labeled by the transition rate λ and the triggering event, $F(X_i)$, where *F* signifies "failure" and X_i denotes the failed disk. Note that multiple X_i s indicate that any one of the disks can trigger the transition.

Reliability of RoLo-P

Figure 6 shows the state transition probability diagram for a RoLo-P disk array consisting of four disks.

The MTTDL of RoLo-P consisting of four disks is:

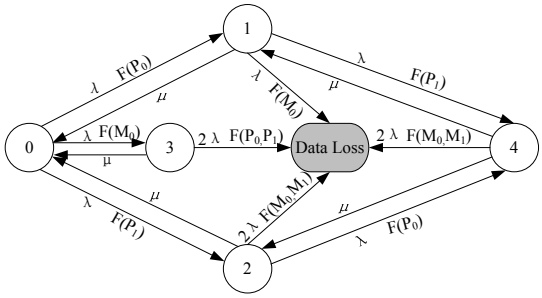


Figure 6. State transition probability diagram for a RoLo-P disk array consisting of four disks

$$MTTDL_{RoLo-P-4} \approx \frac{10\lambda+\mu}{5\lambda^2} \quad (3)$$

Reliability of RoLo-R

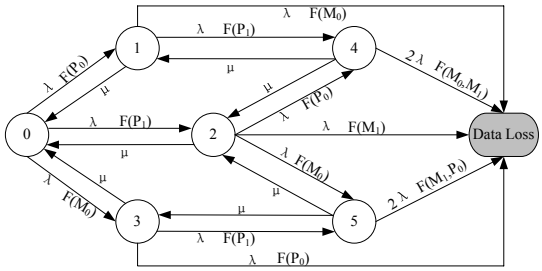


Figure 7. State transition probability diagram for a RoLo-R disk array consisting of four disks

Figure 7 shows the state transition probability diagram for a RoLo-R disk array consisting of four disks.

The MTTDL of RoLo-R consisting of four disks is:

$$MTTDL_{RoLo-R-4} \approx \frac{15\lambda+2\mu}{6\lambda^2} \quad (4)$$

Reliability of RoLo-E

Since only one mirrored disk pair is used as log disks in RoLo-E at any given time, we derive the state transition probability diagram in Figure 8:

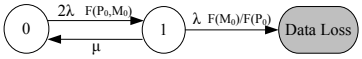


Figure 8. State transition probability diagram for a RoLo-E disk array consisting of four disks

The MTTDL of RoLo-E consisting of four disks is:

$$MTTDL_{RoLo-E-4} \approx \frac{3\lambda+\mu}{2\lambda^2} \quad (5)$$

Reliability Comparison

Since the frequency at which a disk spins up/down plays a critical role in the lifetime of the disk and its failure rate

Table I
NUMBER OF DISKS SPIN UP/DOWN OF DIFFERENT SCHEMES UNDER $src2\_2$ AND $proj\_0$

Trace	RAID10	GRAID	RoLo-P/R	RoLo-E
src2_2	0	40	4	357
proj_0	0	120	12	2874

λ [3][4], we must combine MTTDL with this frequency to evaluate the reliability of RoLo.

Table I shows the measured numbers of times disks are spun up/down among the schemes studied under the two traces of $src2\_2$ and $proj\_0$. There is no disk spinning up/down in the standard RAID10 systems since all the disks in RAID10 systems are kept ACTIVE. The number of times disks are spun up/down in RoLo-P or RoLo-R is only 10% of that of GRAID. The reason is that all the mirrored disks must be spun up for destaging when the log disk in GRAID reaches its capacity. On the contrary, in either RoLo-P or RoLo-R, only the next on-duty mirrored disk is spun up for destaging during the logger rotation when the current on-duty log disk reaches its capacity.

Note that although many read requests have been serviced by the current active cache disks in RoLo-E, the disk spun up/down number is much larger than that of GRAID, RoLo-P or RoLo-R due to the multiple read miss requests that force spun-down disks to spin back up. It is reasonable to believe that the much higher disk-spin frequency can substantially increase the disk failure rate λ of RoLo-E and thus decrease its MTTDL [3], which is inversely proportional to λ. It is easy to conclude that RoLo-P and RoLo-R can significantly decrease disk-spin frequency of conventional centralized logging architectures such as GRAID and thus prolong the lifetime of disks. On the other hand, we argue that RoLo-E is only suitable for write-dominated workloads from a reliability's viewpoint.

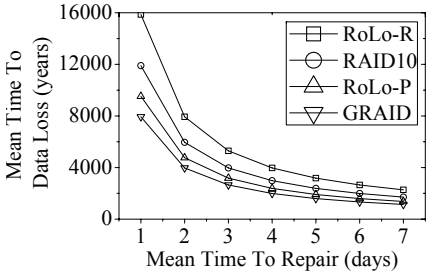


Figure 9. Mean Time to Data Loss achieved by different disk array levels. Note that a higher MTTDL indicates a higher reliability

Figure 9 plots MTTDL as a function of MTTR (Mean Time To Repair) for RAID10, GRAID, RoLo-P and RoLo-R respectively. The disk failure rate λ is assumed to be one failure every one hundred thousand hours, which is slightly less than one failure every eleven years and more conservative than the values quoted by disk manufacturers [4]. MTTR is expressed in terms of days and MTTDL is expressed in terms of years. Due to the fact that there are three copies of each write data written to three independent disks in RoLo-R, it outperforms RAID10 in terms of MTTDL by up to 33%. Meanwhile, MTTDL of RAID10 is higher than that of RoLo-P and GRAID by up to 20% and 33% respectively, in which the second copies of all the write data are written to a single log disk. The reason why MTTDL of RoLo-P is higher than that of GRAID is that only a small subset of the relevant mirrored disks are spun up for the recovery of the failure of any primary disk in RoLo-P, while all the mirrored disks must be spun up for the recovery of the failure of any primary disk in GRAID.

Since the fact that the spin frequency of RoLo-P/R is only 1/10 of that of GRAID would imply a higher (or lower) MTTF for the former (or the latter), the MTTDL of RoLo-P/R (or GRAID) plotted in Figure 9 is possibly underestimated relative to GRAID (or overestimated relative to RoLo-P/R). However, the impact of disk spin up/down frequency on the disk failure rate has not been quantified and thus not reflected in Figure 9.

Note that, from Equations (1) and (5), MTTDL of RoLo-E is *n* times that of RAID10, where *n* is the number of mirrored disk pairs (2 for this case). However, this can be grossly misleading since it does not reflect the much higher disk-spin frequency of the former. Nevertheless, the MTTDL measure can be meaningful for RoLo-E only if RoLo-E is used in all-write workloads, such as on-line or off-line backup, where there are only $\frac{1}{n}$ disks kept ACTIVE, and the disk failure rate λ is comparable with that of RAID10.

V. PERFORMANCE EVALUATIONS

A. Evaluation Methodology

In order to evaluate the performance and energy efficiency of RoLo, we extend DiskSim [2] with a disk power model used in Dempsey [36], whose accuracy has been proven in the evaluations of power consumption. In addition to the augmentation of DiskSim with RoLo, we also implemented the RAID10 architecture as a baseline in our evaluations, as well as GRAID [21], which is a representative of centralized logging architecture with extra dedicated log disks as a comparable alternative to RoLo. In GRAID, all the mirrored disks are kept STANDBY to save energy and all the second copies of write data are redirected to the extra dedicated log disk.

Unless otherwise stated, the log disk capacity of GRAID is configured to be 16GB, and free space of each disk in RoLo is configured to be 8GB by default, based on the assumption that 50% of disk capacity is always free in most cases as mentioned in Section I. In our evaluations, we use average response time to evaluate the performance of all the schemes. A RAID10 disk array consisting of 20$\sim$ 40 disks is adopted in our simulation, a configuration that is commonly deployed in high-end computing or large-scale

Table II
DISK AND RAID CONFIGURATION PARAMETERS

Disk Parameter	***Value***
Disk Model	IBM Ultrastar 36Z15
Capacity /Rotation Speed	18.4GB/15000 RPM
Average Seek/Rotation Time	3.4ms/2ms
Sustained Transfer Rate	55 MB/s
Power Parameters	***Value***
Power(Active,Idle,Standby)	13.5W,10.2W,2.5W
Spin down/up energy	13J/135J
Spin down/up time	1.5s/10.9s
RAID Configuration	***Value***
RAID level	RAID10
Stripe Unit Size	16KB, 32KB, 64KB
Number of Disks	20,30,40(21,31,41 for GRAID)
Free Disk Space	8GB,6GB,4GB(16GB for GRAID)

Table III
A SUMMARY OF CHARACTERISTICS OF $src2\_2$ AND $proj\_0$

Trace	Write Ratio	IOPS	Avg.Req Size	Write Capacity
src2_2	99.62%	78.80	63.64KB	33GB
proj_0	94.90%	23.89	51.42KB	99.3GB

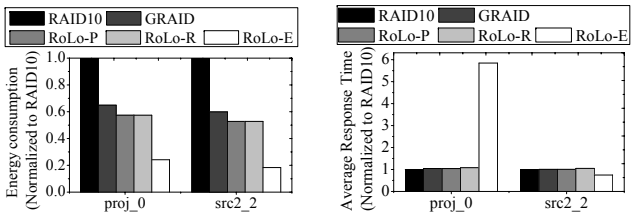
data centers to deliver peak I/O bandwidth [16]. A widely used high performance SCSI disk, IBM Ultrastar 36Z15 [4], is used throughout our experiments. Its main specifications and RAID configuration parameters are listed in Table II.

The traces used in our experiments are collected from a production data center in Microsoft Research UK with a total of 36 different traces [27]. Since the main design goal of RoLo is to significantly benefit write-intensive applications in energy-efficiency, reliability and performance, we select 2 of the 36 traces, i.e., $src2\_2$ and $proj\_0$, that have the highest write-intensity for use in the main portion of our experiments and sensitivity studies to show how different design parameters may impact RoLo. We choose five additional traces, i.e.,mds_0, $wdev\_0$, web_1, $rsrch\_2$ and hm_1, to show the effectiveness of RoLo under non-write-intensive workloads. The traces chosen for our experimental study have different read/write ratios, IOPS (I/O Per Second), and average request sizes, to represent multiple types of workloads in real enterprise-level data centers, such as source control (*src*), project directories (*proj*), media server (*mds*), test web server (*wdev*), web/SQL server (*web*), research projects (*rsrch*) and hardware monitoring (*hm*) [27]. The summary of the characteristics of these seven traces are listed in Table III and Table VI respectively.

B. Trace-driven Evaluations

We first conduct our experiments on a RAID10 disk array consisting of 40 disks with a fixed stripe unit size of 64KB to evaluate the energy efficiency and performance of RoLo.

Figure 10 compares the energy consumption and average response time of RoLo with the standard RAID10 and GRAID schemes under $src2\_2$ and $proj\_0$. From Figure 10(a), one can see that the energy consumptions of RoLo-P and RoLo-R are almost the same under both traces. The reason is that the only difference between RoLo-P and RoLo-R is an extra write operation on the primary log disk for write requests in RoLo-R. Since the primary disks in RoLo-P and RoLo-R are set ACTIVE all the time, an extra write operation has negligible impact on energy consumption. One can see that RoLo-P and RoLo-R consume 42.6% and 47.2% less energy than RAID10 under $proj\_0$ and $src2\_2$ respectively. RoLo-E consumes the least amount of energy under both $proj\_0$ and $src2\_2$, 75.8% and 81.7% less than RAID10 under $proj\_0$ and $src2\_2$ respectively.



(a) energy consumption (b) average response time

Figure 10. Energy consumption and average response time, normalized to RAID10, under different traces

In addition, RoLo-P and RoLo-R consume less energy than GRAID under $proj\_0$ and $src2\_2$ by 11.50% and 11.81% respectively. This is because the logical logger space consisting of the free storage space of disks in the former two is much larger than that of GRAID so that the idle periods of disks can be much longer than that of GRAID. RoLo-E consumes less energy than GRAID under $proj\_0$ and $src2\_2$ by 62.70% and 69.37% respectively since only one mirrored disk pair is kept ACTIVE and used as logger in RoLo-E.

Figure 10(b) shows a comparison of average response times of RAID10, GRAID, RoLo-P, RoLo-R and RoLo-E under both $src2\_2$ and $proj\_0$. The average response times of GRAID and RoLo-P are nearly the same under both $src2\_2$ and $proj\_0$ because the basic model of RoLo-P is the same as that of GRAID except for the logger management policy. One can see that the average response time of RoLo-P is slightly greater than RAID10 under $proj\_0$ and $src2\_2$ by 4.18% and 0.71% respectively, while RoLo-R underperforms RAID10 under $proj\_0$ and $src2\_2$ by 8.11% and 5.09% respectively.

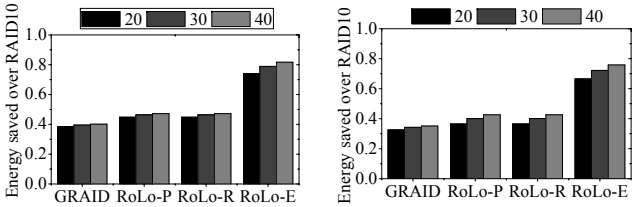
The reason behind RoLo-R's inferior performance to RoLo-P, by 3.77% and 4.35% under $proj\_0$ and $src2\_2$ respectively, is because the former issues three copies of each written data to provide higher reliability while the latter issues only two such copies. It is interesting to notice the polarization of the average response time of RoLo-E under $proj\_0$ and $src2\_2$. To investigate the main reason behind this performance disparity of RoLo-E under the two traces, we analyze the corresponding read hit rates during

Table IV
COMPARISON RESULTS AMONG RAID10, GRAID, ROLO-P, ROLO-R AND ROLO-E

	Energy saved over				Performance gained over				Reliability compared with	
	RAID10		GRAID		RAID10		GRAID		RAID10	GRAID
	src2_2	proj_0	src2_2	proj_0	src2_2	proj_0	src2_2	proj_0		
RoLo-P	47.2%	42.6%	11.81%	11.50%	-0.71%	-4.18%	0%	0%	Lower	Higher
RoLo-R	47.2%	42.6%	11.81%	11.50%	-5.09%	-8.11%	-4.35%	-3.77%	Higher	Higher
RoLo-E	81.7%	75.8%	69.37%	62.70%	75%	-584%	75.18%	-557%	Higher	Higher

Table V
A SUMMARY OF CHARACTERISTICS OF ROLO-E UNDER $src2\_2$ AND $proj\_0$

Trace	Read Ratio	Read Hit Rate	Burstiness	Performance Gained over RAID10
proj_0	5.10%	26.67%	Very Low	-584%
src2_2	0.38%	90.59%	Very High	75%



(a) under $src2\_2$ (b) under $proj\_0$

Figure 11. Energy saved over RAID10 as a function of the number of disks under $src2\_2$ and $proj\_0$

our experiments. Table V shows that the read ratio of $src2\_2$ approaches to 0 while its read hit rate is over 90%, which sharply contrasts to $proj\_0$ that has a read ratio about 14 times that of $src2\_2$ and a read hit rate of less than 30%. This means that $proj\_0$ results in much more read misses than $src2\_2$, where each missed read request translates into a disk spin-up operation and incurs an expensive latency penalty of 1000-10000 times that of a read hit.

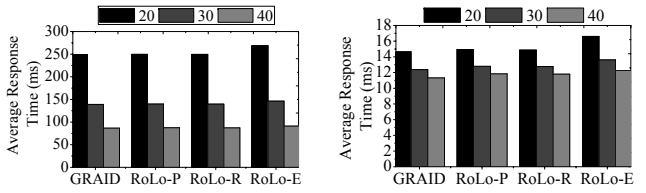
The comparison results among RAID10, GRAID, RoLo-P, RoLo-R and RoLo-E are listed in Table IV.

C. Sensitivity Study

To examine the impacts of the number of disks, the stripe unit size, the available free storage space, the disk sizes and write intensiveness, we conduct a series of sensitivity studies through experiments. Since RoLo-E will be most suitable for write-dominated workloads where performance and energy efficiency are the main concerns, as discussed earlier, its response time measures will not be included in this sensitivity study.

Number of Disks : To examine the impact of the number of disks, we conduct experiments on different numbers of disks (20, 30 and 40 for RoLo and 20+1, 30+1 and 40+1 for GRAID) in a RAID10 disk array with a stripe unit size of 64KB.

As all the primary disks of the RoLo-P, RoLo-R and



(a) under $src2\_2$ (b) under $proj\_0$

Figure 12. Average response time as a function of the number of disks under different traces

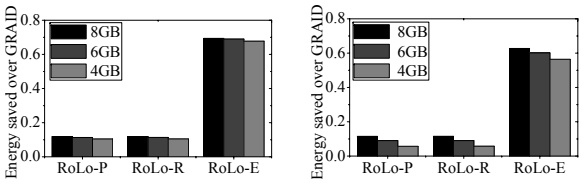
GRAID schemes are kept ACTIVE, it is expected that more energy is consumed with an increase in the number of disks. Figure 11 shows that the amount of energy saved by GRAID, RoLo-P, RoLo-R and RoLo-E from the RAID10 scheme increases by 1.82%, 2.41%, 2.41% and 7.77% respectively as the number of disks increases from 20 to 40 under $src2\_2$. These energy-saving increases improve to 2.58%, 6.10%, 6.10% and 9.26% respectively under $proj\_0$. We can conclude that proportionally more energy can be saved as the number of disks increases for GRAID, RoLo-P, RoLo-R and RoLo-E. Note that the amount of the increased energy savings by RoLo-P, RoLo-R and RoLo-E are much larger than that by GRAID as the disk array size increases.

From Figure 12 we can conclude that the average response times of RAID10, GRAID, RoLo-P and RoLo-R decrease as the disk array expands in size, as a result of the increased disk access parallelism.

Stripe Unit Size:To examine the sensitivity of RoLo to the stripe unit size, we conduct experiments on a 40-disk RAID10 disk array with stripe unit sizes of 16KB, 32KB and 64KB, respectively. The experimental results, not shown here for space constraint, reveal that, except for RoLo-E that is noticeably sensitive to stripe unit size under $src2\_2$, none of the schemes is sensitive at all to stripe unit size in terms of energy efficiency. To explain the sensitivity of RoLo-E to stripe unit size under $src2\_2$, we find from the trace characteristics that the average read request size is 68.08KB and 17.84KB for $src2\_2$ and $proj\_0$ respectively. When the stripe unit size is set to 16KB under $src2\_2$, most of the read requests are split into more than one sub-requests and more disks have to be spun up. The number of disks spun up for the read miss requests are reduced as stripe unit size increases, which explains why energy saved by RoLo-E from RAID10 increases as the stripe unit size increases

from 16KB to 64KB.

Free Storage Space: To evaluate RoLo with different amounts of free disk space, we conduct experiments on a 40-disk RAID10 array with a stripe unit size of 64KB.



(a) under $src2\_2$ (b) under $proj\_0$

Figure 13. Energy saved over GRAID as a function of the amount of free storage space under $src2\_2$ and $proj\_0$

Figure 13 shows that the amounts of energy saved by RoLo-P, RoLo-R and RoLo-E from GRAID slightly decrease with a reduction in the free storage space. The reason is that the capacity of the active logging space, which is proportional to the free space available on the on-duty logger, decreases with a reduction in the free space of every disk. This in turn means a shorter logging period that requires more frequent rotations of the logger, i.e., more frequent disk spin-up/down.

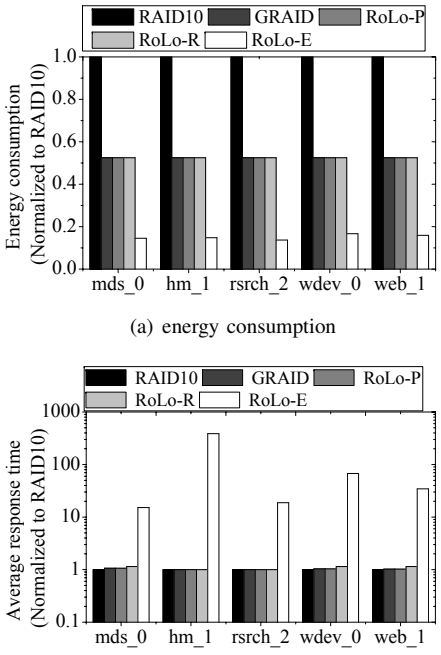
The experimental results show that the average response times of RoLo-P, RoLo-R and RoLo-E are almost unchanged as the free storage space changes, suggesting that the average response time of RoLo is not sensitive to the amount of available free storage space. The reason is that the background destaging I/O activities with a lower priority have no or little impact on the foreground I/O performance, even though the decreased free storage space shortens the logging period.

Disk Sizes: To examine the sensitivity of the energy saving effectiveness to the size of disks, we set disk capacity of GRAID as 16GB, 8GB, 4GB and that of RoLo-P/R as 8GB, 4GB, 2GB correspondingly to keep the fixed free storage space ratio as 50%. The experimental results, not shown here for space constraint, reveal that the energy saving effectiveness of RoLo over GRAID does not vary with the disk capacity under the condition of unalterable disk I/O performance. This phenomenon is consistent with the observations made in Section II. Consequently, we can conclude that the energy saving effectiveness of RoLo over GRAID varies with the number of disks and free storage space rather than the disk size under fixed free storage space ratio. The energy saving effectiveness of RoLo over GRAID under different disk models, such as Seagate Cheetah 15K.5 [1], with larger capacity and particular performance and power parameters will be studied as our future work.

Write Intensiveness: We conduct experiments on a 40-disk RAID10 array with a stripe unit size of 64KB to evaluate the effectiveness of RoLo in non-write-intensive workloads with five traces with different write intensities, mds_0, $wdev\_0$, web_1, $rsrch\_2$ and hm_1. The characteristics of these five traces are listed in Table VI.

Table VI
A SUMMARY OF CHARACTERISTICS OF mds_0, $wdev\_0$, web_1, $rsrch\_2$ AND hm_1

Trace	Write Ratio	IOPS	Avg.Req Size	Write Capacity
mds_0	88.11%	2.00	9.20KB	7.0GB
wdev_0	79.92%	1.89	9.08KB	7.15GB
web_1	45.89%	0.27	29.07KB	664MB
rsrch_2	34.31%	0.35	4.08KB	295MB
hm_1	4.66%	1.02	15.16KB	553MB



(a) energy consumption

(b) average response time

Figure 14. Energy consumption and average response time, normalized to RAID10, under different traces

Figure 14 shows that the energy consumptions of RoLo-P and RoLo-R are the same as that of GRAID and the average response time of RoLo-R is worse than those of RoLo-P and GRAID under mds_0, $wdev\_0$, web_1, $rsrch\_2$ and hm_1 by 0.74% ∼7.31%. It indicates that when RoLo is deployed in non-write-intensive application environments, its negative impact, if any, is negligible.

VI. Related Work

Logging techniques Logging techniques have been widely used in hierarchical storage architectures to improve performance or energy efficiency of storage systems. DCD [17] uses a small log disk as a secondary disk cache beneath a memory cache to optimize write performance. Logging RAID [8] bundles small writes into large writes to overcome the small-write performance problem in parity-based disk arrays. GRAID [21] concentrates the second copy of write data blocks onto an extra log disk to prolong the idle period of mirrored disks to save energy. Different from the above

existing logging architectures with a fixed and dedicated logger, RoLo rotates the logger among the logical logging space pool formed by free storage space, thus avoiding the dedicated and extra log disks. RoLo is also different from Log-structured File System [28] in how the write data blocks are updated to the targeted permanent location.

Destaging algorithms When and which data blocks to destage are the main concerns of destaging schemes. Different from the fixed periodic update policy in traditional UNIX systems [25], modern destaging schemes used in *MEMORY/NVRAM-DISK* multi-level storage architectures tend to maintain a good tradeoff between exploiting temporal locality and spatial locality for destaging and preventing frequent write buffer overflow [5] [7][13] [23] [26][32] . However, RoLo is used in log-disk based storage systems, for which the capacity of log disk(s) is much larger than that of expensive small capacity NVRAM. RoLo is similar to the previous schemes in scheduling the destaging process as a background task and steals idle periods for destaging. However, in order to prolong the idle periods of as many mirrored disks as possible, RoLo carries out decentralized destaging along with rotated logging to alleviate or eliminate the negative impact on energy efficiency caused by centralized destaging in conventional centralized logging architecture, e.g., GRAID. During the decentralized destaging of RoLo, spatial locality is exploited to bundle as many data blocks with successive location as possible in one destaging I/O operation.

Energy efficient schemes Single-hard-disk energy efficiency optimization schemes, such as Dynamic RPM (DRPM) [15], Intra-Disk Parallelism (IDP) [29] and FS2 [18], are orthogonal to and thus can help extend the power savings of RoLo, which is optimized for write-dominant workloads. Free space is exploited in FS2 and PARAID [33] to save energy, where FS2 utilizes free space to replicate some data blocks to minimize the disk positioning time while PARAID uses it to gather all active data onto a small number of disks in a RAID. In contrast, RoLo utilizes free space to form a large logical logging space.

VII. Conclusion

In this paper, we propose a rotated logging RAID10 storage architecture RoLo, which combines decentralized destaging with rotated logging, to overcome the shortcomings of conventional centralized logging architecture. By spreading destaging I/O activities among short idle time slots and proactively reclaiming the stale logging space, RoLo rotates loggers among a logical logging space pool formed collectively from the free storage space available among mirrored disks, and thus not only achieves high scalability and enhanced energy efficiency, but also eliminates the additional hardware and energy costs, potential single point of failure and performance bottleneck caused by the extra dedicated log disks. We design three flavors of RoLo, i.e., RoLo-E/R/P, for respectively specific application scenarios and analytically comparison on their reliability. The detailed trace-driven experimental evaluations demonstrate their advantages over both a centralized logging RAID10 system and a typical RAID10 system. A study on the feasibility and efficiency of RoLo deployed in parity-based storage systems will be conducted as our future work.

VIII. Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments in reviewing this paper. We also thank Dushyanth Narayanan from Microsoft Research Ltd. for providing MSR Cambridge traces.

This work is partially supported by the National High Technology Research and Development Program (863 Program) of China under Grant No.2009AA01A401, 2009AA01A402, Changjiang innovative group of Education of China No. IRT0725, China NSFC under Grant No. 60933002 and No. 60873028, US NSF under Grant No. NSF-IIS-0916859, NSF-CCF-0937993 and NSF-CCF-0621526.

References

[1] Cheetah 15k.5 Hard Drives. http://www.seagate.com/www/en-us/products/servers/cheetah/cheetah15k.5/.

[2] *The DiskSim Simulation Environment Version 4.0 Reference Manual*. http://www.pdl.cmu.edu/DiskSim/index.h-tml.

[3] *Idema Standards, Specification of Hard Disk Drive Reliability, Document Number r2-98*. In *http://www.idema.org/*.

[4] *Ultrastar 36Z15 Datasheet*. http://www.hitachigst.com/hdd/ultra/ul36z15.htm.

[5] M. Alonso and V. Santonja. *A New Destage Algorithm for Disk Cache: DOME*. In *EUROMICRO*, pages 1416–1423, 1999.

[6] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler. *An Analysis of Latent Sector Errors in Disk Drives*. In *SIGMETRICS*, pages 289–300, 2007.

[7] A. Batsakis, R. C. Burns, A. Kanevsky, J. Lentini, and T. Talpey. *AWOL: An adaptive Write Optimizations Layer*. In *FAST*, pages 67–80, 2008.

[8] Y. Chen, W. W. Hsu, and H. C. Young. *Logging RAID - An Approach to Fast, Reliable, and Low-cost Disk arrays*. In *Euro-Par*, pages 1302–1312, 2000.

[9] E. N. Elnozahy and J. S. Plank. *Checkpointing for Peta-scale Systems: A Look into the Future of Practical Rollback-recovery*. *IEEE Transactions on Dependable and Secure Computing*, 1(2):97–108, April-June 2004.

[10] L. Ganesh, H. Weatherspoon, M. Balakrishnan, and K. Birman. *Optimizing Power Consumption in Large Scale Storage Systems*. In *HotOS*, 2007.

[11] Y. Ge, C. Wang, X. Shen, and H. Young. *A Database Scale-out Solution for Emerging Write-intensive Commercial Workloads*. *Operating Systems Review*, 42(1):102–103, 2008.

[12] S. Ghemawat, H. Gobioff, and S.-T. Leung. *The Google File System*. In *SOSP*, pages 29–43, 2003.

[13] B. S. Gill and D. S. Modha. *WOW: Wise Ordering for Writes - Combining Spatial and Temporal Locality in Non-Volatile Caches*. In *FAST*, 2005.

[14] L. Goasduff and C. Forsling. *Gartner urges it and business leaders to wake up to it's energy crisis*. In *http://www.gartner.com/it/page.jsp?id=496819*, 2006.

[15] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. *DRPM: Dynamic Speed Control for Power Management in Server Class Disks*. In *ISCA'03*, pages 169–181, New York, NY, USA, 2003.

[16] S. Gurumurthi, J. Zhang, A. Sivasubramaniam, M. T. Kandemir, H. Franke, N. Vijaykrishnan, and M. J. Irwin. *Interplay of Energy and Performance for Disk Arrays Running Transaction Processing Workloads*. In *ISPASS*, pages 123–132, 2003.

[17] Y. Hu and Q. Yang. *DCD - Disk Caching Disk: A New Approach for Boosting I/O Performance*. In *ISCA*, pages 169–178, 1996.

[18] H. Huang, W. Hung, and K. G. Shin. *FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption*. In *SOSP*, pages 263–276, 2005.

[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*, pages 111–125, 2008.

[20] M. Levin. *Storage Management Disciplines are Declining*. In *http://www.computereconomics.com/article.cfm?id=1129*, 2006.

[21] B. Mao, D. Feng, S. Wu, L. Zeng, J. Chen, and H. Jiang. *GRAID: A Green RAID Storage Architecture with Improved Energy Efficiency and Reliability*. In *MASCOTS*, pages 113–120, 2008.

[22] J. Menon. A Performance Comparison of RAID-5 and Log-Structured Arrays. In *HPDC*, page 167, 1995.

[23] J. Menon and J. Cortney. *The Architecture of A Fault-tolerant Cached RAID Controller*. In *ISCA*, pages 76–86, 1993.

[24] N. Mi, A. Riska, Q. Zhang, E. Smirni, and E. Riedel. *Efficient Management of Idleness in Storage Systems*. *TOS*, 5(2), 2009.

[25] J. C. Mogul. *A Better Update Policy*. In *USENIX Summer*, pages 99–111, 1994.

[26] Y. J. Nam and C. Park. *An Adaptive High-Low Water Mark Destage Algorithm for Cached RAID5*. In *PRDC*, pages 177–184, 2002.

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

[28] M. Rosenblum and J. K. Ousterhout. *The Design and Implementation of a Log-Structured File System. ACM Trans. Comput. Syst.*, 10(1):26–52, 1992.

[29] S. Sankar, S. Gurumurthi, and M. R. Stan. *Intra-Disk Parallelism: An Idea Whose Time has Come*. In *ISCA'08*, pages 303–314, Washington, DC, USA, 2008.

[30] P. Steege. *50% Storage Utilization: Are Data Centers Half Empty or Half Full?* In *http://storageeffect.media.seagate.com/2009/0-1/storage-effect/50-storage-utilization-are-datacenters-half-empty-or-half-full/*, 2009.

[31] D. Stodolsky, G. A. Gibson, and M. Holland. *Parity Logging Overcoming the Small Write Problem in Redundant Disk Arrays*. In *ISCA*, pages 64–75, 1993.

[32] A. Varma and Q. Jacobson. *Destage Algorithms for Disk Arrays with Nonvolatile Caches*. *IEEE Trans. Computers*, 47(2):228–235, 1998.

[33] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. L. Reiher, and G. H. Kuenning. *PARAID: A Gear-Shifting Power-Aware RAID*. *TOS*, 3(3), 2007.

[34] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. *Ceph: A Scalable, High-Performance Distributed File System*. In *OSDI*, pages 307–320, 2006.

[35] Q. Xin, E. L. Miller, T. Schwarz, D. D. E. Long, S. A. Brandt, and W. Litwin. *Reliability Mechanisms for Very Large Storage Systems*. In *MSST'03*, Washington, DC, USA, 2003.

[36] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. *Modeling Hard-Disk Power Consumption*. In *FAST'03*, pages 217–230, Berkeley, CA, USA, 2003.

[37] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. *Hibernator: Helping Disk Arrays Sleep Through the Winter*. In *SOSP'05*, pages 1–14, 2005.

[38] Q. Zhu and Y. Zhou. *Power-Aware Storage Cache Management*. *IEEE Trans. Comput.*, 54(5):587–602, 2005.