EERAID: Energy Efficient Redundant and Inexpensive Disk Array

Dong Li, Peng Gu, Hailong Cai, Jun Wang
Department of Computer Science and Engineering
University of Nebraska-Lincoln, Lincoln, NE 68588-0115
{*li,pgu,hcai,wang*}*@cse.unl.edu*

Abstract

Recent studies show that disk-based storage subsystems account for a non-trivial portion of energy consumption in both low-end and high-end servers. Current energy-efficient solutions work either at a disk drive level or at a storage system cache level without the knowledge of redundant information inside RAID systems and thus have certain limitations.

In this paper, we develop a novel energy-efficient RAID system architecture called EERAID to significantly conserve energy by taking advantage of redundant information. To give a proof-of-concept solution, we develop new energy-efficient, redundancy-aware I/O scheduling and controller-level cache management schemes for EERAID 1 and EERAID 5 respectively. EERAID 1 implements two new policies– Windows Round-Robin (WRR) and Power and Redundancy-Aware Flush (PRF), while EERAID 5 develops two novel schemes–Transformable Read (TRA) and Power and Redundancy-aware Destage (PRD).

Comprehensive trace-driven simulation experiments have been conducted by replaying two real-world server traces and a wide spectrum server-side synthetic traces. Experimental results showed that, 1) for single-speed (conventional) disks, EERAID 1 and EERAID 5 can achieve up to 30% and 11% energy savings respectively, and 2) For multi-speed disks, compared with DRPM, EERAID 1 and EERAID 5 can achieve 22% and 11% extra energy savings. There is little performance degradation or even better performance in EERAID 1 and 5.

1 Introduction

Today energy has become a critical resource in modern computing while motivating an upsurge of activity in industry and academia. Recent studies show that disk-based storage subsystems account for a non-trivial portion of energy consumption in both low-end and high-end servers [2]. Depending on the storage system size, the energy consumed by the storage system can easily surpass that of the rest of the system [20].

There are two major research objects in the power-efficient I/O study: a single disk in mobile devices and laptops, and a multi-disk system such as RAID in server systems and data centers. The focus of this paper is on the latter one, the multi-disk system. For a single disk, most power-efficient solutions spin up or spin down the disk between the high-power mode and low-power mode by dynamically adjusting a time threshold [8, 9, 13, 14]. For a multi-disk system, some recent studies [10, 28, 4] have found that the average idle period is not long enough to spin up or spin down a disk, because the I/O workloads are much more intensive in the server environment. Gurumurthi et al. advocated the use of non-existing multi-speed disks to save energy for server workloads by a DRPM scheduling policy that exploits small-scale spin-up and spin-down periods [10]. The main idea of DRPM is to aggressively fine-tune disk rotation speeds step-by-step for a disk, with the assumption that the faster a disk rotates, the more energy is consumed. Carrera et al. proposed some hybrid storage organizations of laptop disks, server disks and multi-speed disks to save energy [4], and got the conclusion that only multi-speed disks can actually provide energy savings for network servers. Recently Zhu et al. developed some power-aware storage cache management algorithms called PA. Energy is saved by discriminatingly caching the data of those disks with good workload characteristics of power-efficiency in terms of the percentage of cold misses and the cumulative distribution of interval lengths [28]. However, their design is still based on multi-speed disks.

There are some major problems with the above-mentioned schemes. First, they work either at a disk drive level or at a storage system cache level. We cannot see the big picture of the entire multi-disk system at the disk drive level. While working at a storage system cache level, we cannot tell the associative relationship among multiple disks, especially for RAIDs with redundant information. At both levels, each disk is treated independently by mistake, and how disks work collaboratively with each other on redundancy maintenance and load balance is missing. As a result, current solutions cannot perform well and maximize energy savings. For example, in RAID 5 (a typical organization in a multi-disk system), data block is usually linked to more than one disk. To write back a data stripe may involve access to two disks–the data disk and another disk that saves the parity stripe.

Secondly, current solutions choose the multi-speed disk as a building-block of storage systems. Unfortunately it is not clear that the multi-speed disk product will be on the market soon because of its prohibitive manufacturing cost. The reason why they make such a strong assumption is that traditional power-efficient management schemes (TPM) [8, 9, 13, 14, 18] is not feasible for server systems because the average idle period (typically in hundreds of milliseconds) of disks in server systems is too short compared to the disk spin-down or spin-up time (typically in tens of seconds) [10].

We found that current solutions for a single disk have limited ability to stretch the disk idle periods (i.e., optimize the disk access distribution for energy conservation) especially for non-blocking reads. Current solutions based on a multi-disk system have a better but still limited ability to indirectly change the disk access distribution. The problems with the above two solutions are in that they cannot or have a very limited freedom to delay a non-blocking read request and a write request. This motivates us to develop scheduling and cache management policies inside a RAID controller, such that we can change the request intervals directly by transforming requests among different disks. With the help of redundancy, a non-blocking read of a disk can be equally transformed to a read request of another disk without hurting the overall system performance. For example, in a six-disk RAID 5 system, we have 1) a parity group containing stripe 0, 1, 2, 3, 4 and P that are saved in disk 0, 1, 2, 3, 4 and 5 respectively, 2) stripe 2, 3, 4 and P are cached in the RAID controller cache, and 3) there is a read request for stripe 0. To serve such a read, we could either read stripe 0, or read stripe 1 and then calculate stripe 0 on the fly by a XOR calculation (more details are given in Section 4.1). A non-volatile built-in-cache may be used to electively delay writes. By applying such a powerful scheduling and caching management scheme at the RAID controller level, the disk access distribution in a multi-disk system can be ultimately optimized so that near-optimal energy conservation is obtained.

In both low-end and high-end servers, RAIDs have contributed to a large portion of energy consumption of I/O subsystems. We believe that this is a good opportunity to develop new I/O request scheduling and caching management strategies *at the RAID controller level* in order to save maximum energy. In this paper, we build Energy-Efficient RAIDs called **EERAID** (EERAID 1 for RAID 1 and EERAID 5 for RAID 5) by incorporating novel energy-efficient, redundancy-aware I/O request scheduling and RAID controller cache management schemes for two representative RAID organizations, RAID 1 and RAID 5. We assume a non-volatile write-back cache is employed by any RAID controller without explicit explanation in the rest of this paper. One of the most salient benefits of EERAID is to generate idle intervals ideally long enough to spin up/spin down single-speed disks to conserve energy without too much performance compromise. For single-speed disk systems, *spin down* has the same meaning as *shut down* in this paper without specification. Moreover, EERAID is orthogonal to existing power-efficient solutions such as DRPM and PA. Combining with DRPM and PA, EERAID can achieve further energy savings based on multi-speed disks.

To evaluate EERAID, a trace-driven and event-driven simulator based on Disksim [3] is developed. A non-power-aware RAID and a DRPM-based RAID [10] are chosen as the baseline systems for the single-speed disk and multi-speed disks system respectively. By replaying both synthetic and real-life traces, we found that, 1) For single-speed disks, EERAID 1 can achieve up to 30% energy savings and 11% savings by EERAID 5, and 2) For multi-speed disks, compared with DRPM, EERAID 1 can achieve more than 22% energy savings and 11% extra for EERAID 5. In all experiments, there is no or little performance degradation.

The remainder of this paper is organized as follows. In Section 2 we describe the disk power model and introduce our motivation. We describe the design of EERAID 1 and EERAID 5 in Sections 3 and 4 respectively. Our experimental methodology is detailed in Section 5. In Section 6, results and analysis are presented. We discuss the related work in Section 7 and make our conclusions in Section 8.

2 Design Considerations

Modern disks work in several modes including active, idle, standby. The disk is in an active mode when it is servicing requests and rotating at a single full speed. In an idle mode, the disk does not service any request, and the disk arm stops moving while the disk platters keep rotating. Both the platters and disk arm are static when the disk is in a standby mode. In this mode, the disk consumes much less energy than in both active and idle modes because disks rotate at a full speed in active and idle modes. Disks in a standby mode have to spin up to a full speed before servicing a request. In effect, it takes a long time (tens of seconds) to either spin down disks from active to standby or spin up to active from standby. As a result, a large amount of energy is consumed during disk spin-up or spin-down.

There are two design features followed by EERAID. The first consideration is that even small increases in the request interval length of inactive disks can result in significant energy savings. This guides EERAID to preferably dispatch requests to disks in high-power state (e.g., single-speed disks in active mode or multi-speed disks high-speed mode), and discriminatingly transform reads or delay writes for low-power state disks (e.g., single-speed disks in standby mode or multi-speed disks in low speed mode). The idea is to keep disks in low-power mode stay at low-power as long as possible so that both the mode-switch frequency and the aggregate energy consumption of RAID is reduced. For example, in a single-speed disk system, assuming disk A is active and disk B is standby. Disk A may service a request immediately while disk B has to spend tens of seconds in spinning up to a full speed before servicing any request. A typical spin-up period is much longer than a request access time and thus consumes a large amount of energy. If we can transform a read request of disk A to another request of disk B but still reply the same data, then disk B continues to service requests while A stays at a low power mode. Consequently, the aggregate energy consumption for both disk A and B is largely conserved compared to a case without the request transform.

The second design consideration is to make sure the performance is not compromised after applying new EERAID schemes. For example, since TRA transforms some requests to the disk in active mode, we must make sure additional transformable requests do not force the disk overloaded. We analyzed two real-world traces Cello96 and Tpcc in server environments (more information is detailed in Section 5.2) and found that the average idle ratio per disk on are 34% and 44% respectively, which indicates server disks do have enough spare service capacity to afford a few transformable requests. Additional requests transformed by TRA would not compromise the performance at all.

3 EERAID 1

RAID 1, also called mirroring or shadowing, adopts twice as many disks as a non-redundant disk array to maintain 100% redundancy [5]. Many policies are presented on how to dispatch a read request to disks at the RAID 1 controller to obtain high-performance [6]. Some commonly used policies [3] include sending all read requests to a single primary replica, random selection, round-robin, shortest-seek first and shortest-queue first by selecting the replica with the shortest request queue on disk drive and having ties broken by random selection. The shortest queue is adopted in EERAID 1 and EERAID 5 without explicit explanation in the rest of this paper.

However, to the best of our knowledge, none of the above-mentioned solutions takes energy consumption into consideration. A naive scheme is to send all requests to one group such as primary dispatch. This appears to be a good energy-efficient policy because the other group should always be idle (assuming all writes are absorbed in controller cache). However, this may not always be the truth. For the intensive I/O workloads, because the aggregate access time for all requests is substantially stretched, much more energy is actually consumed by this scheme. We develop a new dispatch policy called **Windows Round-Robin** to maximize the energy savings by best exploiting 100% redundancy. The main idea of WRR is to alternatively dispatch every N successive requests (called N request window) to the primary group and the mirror group. Traditional Round-Robin policy can be considered as a special instance of Windows Round-Robin with N equals one. In addition, write requests are satisfied by the controller cache. In an N request window, when the controller cache looks for victims to flush dirty cache lines, it discriminatingly selects a cache line that belongs to the group to which RAID controller is dispatching read requests. We call such a policy **Power and Redundancy-Aware Flush (PRF)**.

When N successive requests are sent all the way to one group (called busy group), the idle period of disks of the other group (called idle group) is largely stretched. As a consequence, idle group get more chances to stay in standby mode for single-speed disks, or spin down to very lower power states for multi-speed disks. Although the busy group consumes little more energy because of more jobs, much more energy is saved from the idle group. Therefore, EERAID 1 is able to drastically reduce the energy consumption for the whole disk array. Next, we describe specific EERAID 1 designs by employing single-speed disk and multi-speed disk separately.

3.1 Single-speed Disk Based EERAID 1

First we investigate a naive Windows Round-Robin implementation. Given an arbitrary number N (e.g. 500), we dispatch N requests to two groups in RAID 1 alternatively without any performance control. The disk spin down policy works like this: spinning down disks of the idle group to standby at the beginning of a N-request window and spinning up disks near the end of the N request window. The idle group will have already ramped up when the N request window is over so that the idle group is ready for servicing requests in advance. But we find that the RAID system performance is severely degraded in experiments. The reason is that in I/O intensive workloads, passing N successive requests all the way to one group may easily overload the busy group, thus significantly stretching the average response time.

To prevent compromising too much performance, N should be adjusted dynamically to adapt to the access pattern change. We improve our WRR design by increasing the value of N step by step based on the performance degradation. First we give a maximum window size N_{max}. Before running WRR, we calculate the average response time T by using the shortest-queue first algorithm for N_{max} requests. Then we select a small number as the initial window size. After the RAID controller sends N requests to one disk group, it sends N requests to the other group. These two N request windows are called one window cycle. By comparing T with the average response time of these 2N requests, the controller decides whether or not to continue WRR with a larger window size. If the performance degradation is less than the predefined threshold p (e.g. 9%), the window size is enlarged by the step size N_{step}. Otherwise, WRR is stopped. Even there is no big change of average response time after many window cycles, we may still stop using WRR and apply shortest-queue first policy for N_{max} requests to get the newest T to reflect the current access pattern. Let C_{max} be the maximum window cycles. Details of WRR are illustrated by Algorithm 1.

Algorithm 1: EERAID 1 (WRR+PRF)

```
(1)  Get T by using shortest-queue first policy
     for $N_{max}$ requests;
(2)  $N = 0$;
(3)  for ( c = 0; $c < C_{max}$; c++ )
(4)  {
(5)    if($N < N_{max}$)$N = N + N_{step}$;
(6)    dispatch N requests to the primary group;
(7)    dispatch N requests to the mirror group;
(8)    calculate the average response time $T_c$
       of these 2*N requests;
(9)    $\Delta T = T_c - T$;
(A)    if($\Delta T > p$) goto (1);
(B)  }
(C)  goto (1);
```

How to choose good values for N and N_{step} is important. Because the spin-down and spin-up time of single-speed disk are fairly long compared to disk request service time, energy savings may be possible only when the lasting period of a N request window is long enough. In other words, disks should be able to spin down only when N is large enough. Usually it takes more than 10 seconds to spin up a single-speed disk from standby to active mode; therefore energy savings can be achieved only if the idle group have enough time to spin down and then spin up before the end of the current N request window. Otherwise disks do not have any chance to stay idle but simply perform spin-downs and spin-ups. In our implementations, given an I/O workload with average request arrival time as 50 ms, WRR spins down the idle group when N is larger than 1000 and ramps them up when there are 300 requests left in the request window. These parameters can be dynamically adjusted according the the workload and disk power model.

Using WRR and PRF in EERAID 1 forms long idle periods on some disks on purpose. To illustrate this, we drew Figure 1[1] to show an accumulative percentage of idle time of the idle group in one request window in EERAID 1. As seen from the picture, in an N request window, one disk group is idle in most of the time.

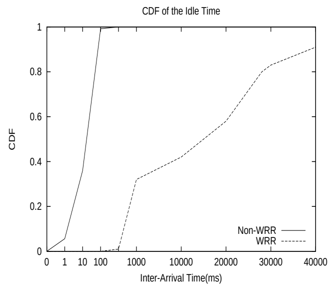

Figure 1: Disk Idle Time CDF

3.2 Multi-speed Disks Based EERAID 1

For multi-speed disks, the time to spin down to a lower speed or to spin up to a full speed from a low speed mode is much shorter than the spin-down and spin-up period between standby and active modes of single-speed disks. For example, in reference [10], an arbitrary multi-speed disk model has 15 different speeds ranging from 12000 RPM to 3600 RPM. It takes only hundreds of milliseconds for a multi-speed disk to spin down to a lower speed and a little more than two seconds from the lowest speed (3600 RPM) to spin up to the full speed (12000 RPM). The DRPM scheduling algorithm is developed to work specifically for multi-speed disks.

Since WRR and PRF are two orthogonal approaches to DRPM, employing WRR and PRF in a multi-speed disk based EERAID 1 can save energy further beyond DRPM (proved in Section 6.1). DRPM allows each disk to spin its speed down at the end of each period (e.g. two seconds) by checking whether the length of its request queue is shorter than the threshold (e.g. zero). The disks spin their speeds down step by step (e.g. 600 RPM for one step). If the queue length is larger than the threshold, the disk will not spin down in the next period. Because WRR and PRF can generate long idle intervals for disks of the idle group, the disks may spin down to a very

[1]The experiment result is collected by feeding real trace 50-ms to the simulator EERAID1-sp. The details of the trace and the simulator are described in Section 5.

low speed in a short time if there is no request on these disks for consecutive periods at all. DRPM ramps disks up only when the performance (average response time) degradation is larger than a high water mark. We make an improvement for performance concern, ramping all disks of the idle group up before the end of current N-request window because the idle group disks will be in the busy group in the next N-request window. In all, the multi-speed disks in EERAID 1 can quickly spin down to a very low speed and attain much longer intervals to stay at this speed.

4 EERAID 5

RAID 5 is a block-interleaved distributed-parity disk array. It provides comparable reliability and availability to RAID 1 but introduces much less storage overhead for redundancy maintenance. WRR is not applicable here since it only works for RAIDs with 100% redundancy. A new redundancy-aware I/O read scheduling policy needs to be developed.

One of the typical disk accesses in RAID 5 is a write update. Generally updating a data stripe needs four disk accesses: reading old data stripe, reading old parity stripe, writing new data stripe and writing new parity stripe that is calculated by exclusive-ORing the first three stripes. It is well-known that this update overhead could degrade the system performance, especially in a small-write intensive environment [16, 23]. Caching is a commonly used solution to address the problem [15]. By adding a non-volatile cache to the RAID controller, disk writes can be satisfied by a write-back cache and then flushed back as a background activity. Updating data an d parity information is denoted as destage [24].

Most of the power aware cache management policies focus on how to select which block to be destaged and how to perform destage more power efficiently. Zhu et.al [28] did not consider scheduling read and write requests separately in their cache management method, and aimed to reshape the interval distribution of disk accesses by assigning more cache space to high-priority disks. As previously discussed, they assume that one cache line is linked to only one disk, which is too simple to reflect the fact in RAID systems.

Taking into account the power consumption effect on both reads and writes, we develop novel energy-efficient, redundancy-aware scheduling schemes in RAID 5 for disk read and write requests respectively. One is called Transformable Read (TRA), which can selectively redirect read requests to other disks to conserve energy while providing the same data service to the user. The other is a RAID controller cache destage algorithm named Power and Redundancy Aware Destage (PRD) that preferably destages the cache blocks of the disks in high power modes. EERAID 5 combines TRA with PRD in such a way that TRA is used to optimize the distribution of disk read access intervals toward energy-efficiency, while PRD is used for disk write access.

4.1 Transformable Read (TRA)

4.1.1 Background

From a storage system user's point of view, the entire RAID 5 system is considered as one big virtual disk. The system I/O requests (with the information such as LBN, request size, read or write) are mapped to block-level requests of the corresponding disks according to the stripe size and layout information. A single system I/O request might be mapped to more than one stripe groups. To develop TRA algorithm we introduce two new terms, a system I/O request called a **Logical Request** and a **Stripe-Group Request** representing a group of disk requests in the same parity group generated by a logical request mapping. It may be noted that one Logical Request may be mapped to more than one Stripe-Group Request. The mapping relationship between Logical request and Stripe-group request is shown in Figure 2, with case (a) demonstrating an one-to-one mapping while case (b) an one-to-two mapping. In case (b), one Logical Request is mapped to two partial Stripe-Group Requests across two different parity groups. To make our idea clear, here we assume that there is no alignment problem. In our experiments we remove this assumption.

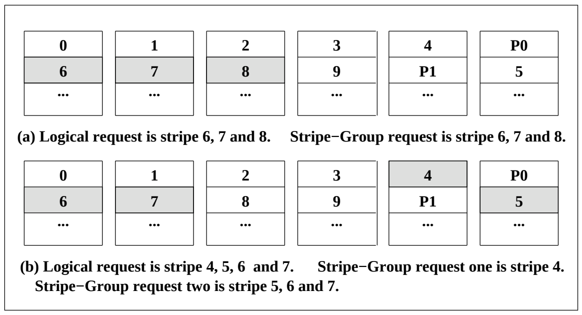
0
1
2
3
4
P0
6
7
8
9
P1
5
...
(a) Logical request is stripe 6, 7 and 8. Stripe−Group request is stripe 6, 7 and 8.
0
1
2
3
4
P0
6
7
8
9
P1
5
...
(b) Logical request is stripe 4, 5, 6 and 7. Stripe−Group request one is stripe 4.
Stripe−Group request two is stripe 5, 6 and 7.

Figure 2: Logical Request and Stripe-Group Request

4.1.2 TRA Algorithm

Given a stripe-group request, TRA replaces one disk read with M new reads of other disks in the same parity group and the requested blocks are constructed by XOR operations on the fly based on redundant information. There are three cases of M: (1) 0, all the stripes in the requested parity group are either cached or requested; (2) 1, we need only one new disk read to replace current read request in the same parity group; or (3) > 1, we need more than one new request to replace current request in the same parity group. Here the maximum value of M is G-1 with G as the parity group size.

In case (1) and (2), the total number of I/Os is unchanged during the TRA process. To estimate how much chances case (1) and (2) hold in practice, we conduct both theoretical and experimental analysis, detailed in the following two sections. In real system, the probability of case 1 and 2 are related to many factors such as cache size and cache management policy, stripe size and data layout, workload characteristics and so on. The sensitivity of TRA without increasing disk access is analyzed in Section 6.3. To simplify our analysis, we consider two numbers–requested stripes and cached stripes per stripe parity group. For example, if one stripe parity group includes stripe 0, 1, 2, 3 and P, stripe 1 and P are cached and stripe 2 is requested in a strip-group request, the number of requested stripes and cached stripes are 1 and 2 respectively.

4.1.3 Theoretical Analysis

To generalize, given a RAID 5 consisting of n stripes in each stripe group, among which C is the set of the stripes currently presented in the cache (let c denote the size of set C, namely $c = |C|$), and the set of stripes in the current Stripe-Group Request is R (similarly, define r as $r = |R|$). Let U be the union of C and R (i.e. $U = C \cup R$, and define u as $u = |U|$). Then the probability that the request can be transformed without increasing the number of disk accesses is $P_{TRA} = P(u \geq n-1)$. Our analysis model is based on the following two different assumptions:

1. The cached stripes are considered discrete, or independent to each other. For example, we think that the following probabilities are equal: $P(C = \{d_1, d_2\}) = P(C = \{d_1, d_3\}) = P(C = \{d_1, p\})$.
2. The requested stripes must be continuous in terms of the stripe number. For example,$P(R = \{d_1, d_2\}) = P(R = \{d_2, d_3\}) = P(R = \{d_3, d_4\}) > 0$ while $P(R = \{d_1, d_3\}) = P(R = \{d_2, d_4\}) = 0$. For example, after mapping the logical request to stripe-group requests, requests with continuous stripes such as $R_1 = \{d_1, d_2\}$ possibly exist while the requests with discrete stripes like $R_2 = \{d_1, d_3\}$ disappear.

To compute the probability $P(u \geq n-1)$, we break down the computation like this: $P(u \geq n-1) = \sum_{r=1}^{n-1} P(r)P(r|u \geq n-1)$ where $P(r)$ is the probability that there are exactly r out of $n-1$ stripes currently requested. Accordingly, $P(r|u \geq n-1)$ means the probability that R is transformable without increasing disk access under this condition.

To further break it down, we have

$$\begin{aligned} P(r|u \geq n-1) &= \sum_{c=n-1-r}^{n} P(c)P(r,c|u \geq n-1) \\ &= \sum_{c=n-1-r}^{n} P(c)P(r,c|u = n-1) \\ &+ \sum_{c=n-1-r}^{n} P(c)P(r,c|u = n) \end{aligned}$$

Here $P(c)$ means the probability that there is exactly c out of n stripes being cached for the corresponding request R, it is irrelevant to the size of the current request R. Accordingly, $P(r,c|u \geq n-1)$ means that under this condition — both the size of the requested set and the size of the cached set are given — the probability that R is transformable without increasing more disk access. This explains why we start from $c = n-1-r$ (if otherwise, $c + r < n-1$, there is no way for us to do any request transform without increasing disk access.). Therefore we have $P(r,c|u = n-1) = \frac{C_{n-r}^{n-r-1} C_r^{c-(n-r-1)}}{C_n^c}$ and
$P(r,c|u = n) = \frac{C_{n-r}^{n-r} C_r^{c-(n-r)}}{C_n^c}$

Finally, a general formula to estimate the chance of TRA is as following.

(I) $P(u \geq n-1) = \sum_{r=1}^{n-1} \{P(|R| = r)\,(A+B)\}$
with $A = \sum_{c=n-r-1}^{n-1} P(|C| = c)\frac{C_{n-r}^{n-r-1} C_r^{c-(n-r-1)}}{C_n^c}$
and $B = \sum_{c=n-r}^{n} P(|C| = c)\frac{C_{n-r}^{n-r} C_r^{c-(n-r)}}{C_n^c}$.

According to our assumption, when n = 5, for the parity stripe group $\{d_1, d_2, d_3, d_4, p\}$, all the possible Stripe-group Requests are listed as $\{d_1\}$, $\{d_2\}$, $\{d_3\}$, $\{d_4\}$, $\{d_1, d_2\}$, $\{d_2, d_3\}$, $\{d_3, d_4\}$, $\{d_1, d_2, d_3\}$, $\{d_2, d_3, d_4\}$, $\{d_1, d_2, d_3, d_4\}$. Supposing each case have the same probability to appear, formula (I) can be simplified as following.

(II) $\frac{2}{2^n n(n-1)} \sum_{r=1}^{n-1} \sum_{c=0}^{n-2} [(n-r)((n-r)C_r^{c-(n-r-1)} + C_r^{c-(n-r)})]$

From formula (II), when n is 4, 5, 6 and 7, we can calculate the probability of TRA without increasing disk access as 35.42%, 33.75%, 30.31% and 26.34% respectively.

4.2 Power and Redundancy Aware Destage (PRD)

In RAID 5 system with a built-in-cache in the RAID controller, disk write access is generated only by destaging blocks out of the controller cache. We develop a cache management scheme called Power and Redundancy Aware Destage (PRD) to collaborate with TRA to optimize the disk access distribution to a further step.

Mishra et al. [17] found that considerable performance improvement can be achieved by caching both data stripe and parity stripe in RAID 5. There are several factors to be considered in developing a destage algorithm, including how to 1) capture most of the rewrites by exploiting the temporal locality in the workload, 2) reduce destage number by aggregating physically close blocks, 3) conduct idleness prediction to reduce the interference between host reads and destage accesses, and 4) reduce destage average time by ordering requests to the disk [24].

Whether the parity stripes are cached in RAID 5, the disk update operation usually involves more than one disk because the parity information is distributed across multiple disks. If the controller cache does not cache the parity stripe, every time when a block is written back, two disks will be accessed, one disk saving this block and the other hosting the block's parity information. If the parity stripe is cached, we can have the following cases during the block destaging (examples of case 3 and 4 are shown in Figure 3):

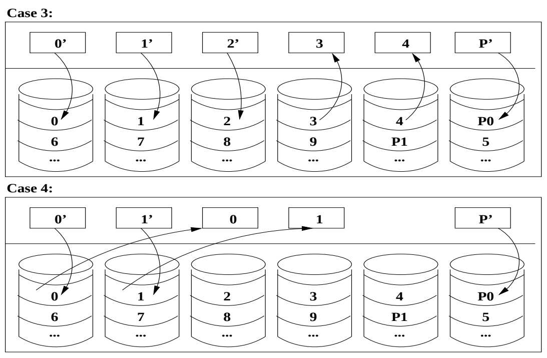

Figure 3: In the example of case 3, stripe 0, 1 and 2 are dirty in the cache, fetch stripe 3 and 4 from disks to generate new parity P', then flush P' and dirty stripe 0, 1 and 2 to disks. In the example of case 4, stripe 0 and 1 are dirty in the cache, fetch old stripe 0 and 1 from disks to generate new parity strip P', then flush P' and dirty stripe 0 and 1 to disks.

1. The block is a data block. The old data stripe and old parity stripe have to be fetched from disks to generate the new parity information. Thus two disks are involved.
2. The block is a data block. The parity stripe is cached and clean and sees a full hit. The old data stripe needs to be fetched from disks to generate the new parity information. Only one disk is involved.
3. The block is a data block and more than half of all the data stripes of the same parity group are dirty in cache. The other data stripes in the same parity are fetched from disks to construct new parity information. More than one disk is involved.
4. The block is a data block and less than half of all the data stripes of the same parity group are dirty in cache. All the old data stripes and the parity stripe are fetched from disks. More than one disk is involved.
5. The block is a data block and all the data stripes of the same parity group are in cache. The parity information are generated immediately, and thus no disk access is needed.
6. The block is a parity block and flushed back to the disk. Only one disk is involved.

We may see other possible cases based on various implementation methods of RAID 5. Previous research on destage algorithms aims to improve performance only, such as least-cost scheduling, high/low mark, linear threshold scheduling [24] and etc. Few of them considered energy consumption as an important trade-off. Although some cache management policies [28, 25] do exist and take the energy consumption as a major concern, none of them works at the RAID controller. Thus none of them see a complete relationship between cached blocks and disks and lead to certain limitation of energy conservation.

The main idea of PRD is to destage blocks in which involved disks are in the low priority group. The low priority group is defined differently in single-speed disk based systems and multi-speed disk based systems (We will discuss them in the following two sections later). PRD works together with existing performance-oriented cache replacement algorithms by considering power as an additional resource trade-off. In our experiments we combine PRD with LRU as the only cache replacement policy.

4.3 Single-speed Disks Based EERAID 5

Our goal is to force the disk access distribution to form long idle intervals on purpose just like what we do in EERAID 1. The basic operation flow of EERAID 5 is to select one disk that is least frequently visited as a high-priority group (high-priority means having good potentials to save energy in near future) and all other disks as a low-priority group (low-priority means having little possibility to save energy in near future), redirect all the requests of high-priority disk to the low-priority group for some request windows through TRA and PRD. Here the request window has the same definition as that in EERAID 1. If the performance has not been degraded too much, repeat the above procedure. It may be noted that the least frequently visited disk is selected according to the original logical request, that is, without considering the requests generated by TRA. Otherwise, the low-priority disk is always the least frequently visited disk. To redirect all read requests of the high-priority disk to other disks, we need to allow M (discussed in Section 4.1) to be any value ranging from 0 to G-1 (G is the parity group size). The worst case is M equals to G-1. Fortunately, the worst case does not have a dominant ratio among all cases especially when the average request size is large. Figure 4 shows the percentage of all cases of Transformable Read that increases the number of disk accesses in the 6-disk RAID 5 and the 8-disk RAID 5 [2]. From Figure 4, we get the average increased disk accesses are 2.56, 1.86 and 1.67 for six disk's workload Small, Half and Full respectively. For 8 disks' Small, Half and Full workloads the numbers are respectively 4.37,2.94 and 2.56. With the consideration of spare service capacity in server disks, TRA will not be likely to hurt the overall system performance at all.

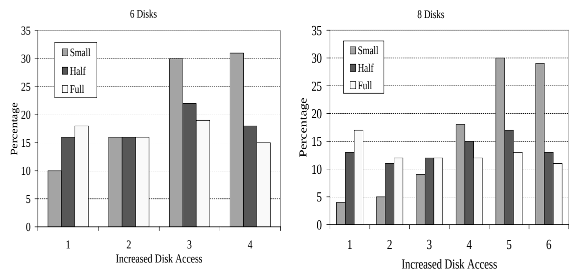

Figure 4: TRA with increasing disk accesses

To develop the TRA algorithm, we investigate the previous research about disk access patterns in server computing. By studying real traces of cello92 and cello99 collected by HP, David A. Pease [19] found that I/O requests were highly unevenly distributed in a multi-disk storage system. Cello was a timesharing system used for simulation, compilation, editing and mail, which had 8 disks in 1992 (cello92) and 20 disks in 1999 (cello99). In cello92, 3 disk drives account for almost 90% of the I/Os while 8 drives for about 86% I/Os in cello99. In both traces, some disks account for less than 1% of the I/Os. Because of the highly skewed access pattern, it is safe to employ TRA to convert some requests of high-priority disks to other data-comparable requests of low-priority disks even the total number of I/Os may increase after the transform.

For the performance concern, the request window size N needs to be dynamically adjusted the same as we do in EERAID 1. How the TRA and PRD work in EERAID 5 is detailed in Algorithm 2, as shown in Table 4.3 where the parameters N, C_{max}, N_{step} and p follow the same definitions in Algorithm 1. We apply the same policy to spin-down disks as that in EERAID 1 employing single-speed disks, spinning down the high-priority disk when N is large enough and ramping it up near the end of the request window.

[2]Figure 4 is drawn by feeding synthetic traces into our storage simulator employing TRA. Small, Half and Full mean the average request size are one stripe, half of the parity group and a whole parity group respectively. The synthetic workload generator is confi gured with 60% read request ratio and 20% sequential requests according to the typical UNIX workload [22]. The RAID controller cache size is 10 MB.

Algorithm 2: EERAID 5 (TRA + PRD)

```
(1)  Get T without using TRA and PRD
     for $N_{max}$ requests;
(2)  $N = 0$;
(3)  for ( c = 0; $c < C_{max}$; c++ )
(4)  {
(5)      if($N < N_{max}$)$N = N + N_{step}$;
(6)      chose the least frequently visited disk as the
         high-priority group;
(7)      set all other disks as the low-priority group
         apply TRA and PRD for N requests;
(8)      calculate the average response time $T_c$
         of the above N requests;
(9)      $\Delta T = T_c - T$;
(A)      if($\Delta T > p$) goto (1);
(B)  }
(C)  goto (1);
```

4.4 Multi-speed Disks based EERAID 5

As we discussed before, multi-speed disks are proposed to save energy even the idle interval period is not as long as tens of seconds. A small idle interval (e.g. hundreds of milliseconds) can still be utilized well by multi-speed disks. Consequently, we do not have to spin down a disk to standby to save energy, since scaling the disk down to a low speed can conserve energy anyway. Based on this principle, we develop a new scheduling scheme by extending the policy used in EERAID 5 employing single-speed disks. In more detail, the definition of a group priority is decided by current power mode of the disk and TRA will not generate extra disk I/Os during the transform, namely M is set as either zero or one. Figure 5 shows an example of how the multi-speed disks take advantage of short idle intervals and how TRA and PRD introduce longer idle intervals in EERAID 5. The X axis shows the system time in seconds while the Y axis shows power state of the disk.

In this example, a multi-speed disk checks its request queue every two seconds. If the request queue length is zero, it will spin down its speed by one step. The multi-speed disk has to ramp up to a full speed before it services any request. At second 0, disk 1 is in a high power mode (high RPM speed) and servicing requests from second 0 to second **6**. Disk 4 is in a low power mode (lower RPM speed). In scenario (a), TRA and PRD are not employed. Assuming a disk read request (r4) of disk 4 arrives near second 2 while a destage request (d4) of disk 4 comes near second 4. Even the disk 4 request queue is empty, it cannot be spun down because it is servicing the request r4. It still can not be spun down at second 4 because of another request d4 arrival. In scenario (b), TRA is employed. Let TRA convert r4 to r1, a read request

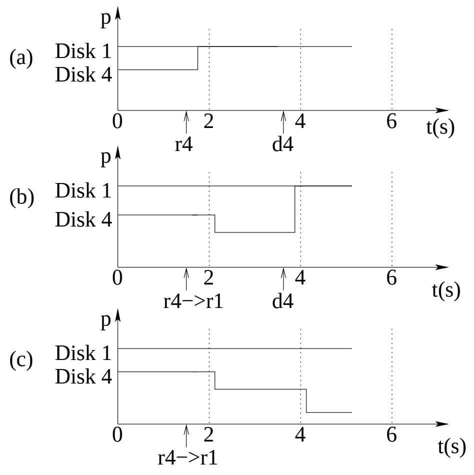

Figure 5: Short Idle Intervals Generated by TRA and PRD

of disk 1, then disk 4 can spin down one step at second 2. But it has to ramp up again to service d4. In scenario (c), both TRA and RPD are employed, ideally, r4 is converted to r1 and d4 is replaced by another destage request of other disks because disk 4 is in a lower power mode compared to other disks (at least lower than that of disk 1). Comparing three scenarios, in an EERAID 5 employing both TRA and PRD, we may possibly spin down disks step by step, keep them to stay at a lower speed for a longer time, reduce the ramp-up overhead and thus save maximum energy.

5 Experimental Methodology

To test the EERAID design, we develop a set of comprehensive trace-driven, disk array simulators based on Disksim [3] incorporating validated power models [1, 10] Two real-world traces and several synthetic traces well emulating server environment are carefully chosen to drive the simulation experiments.

5.1 Disk Power Model

We choose a power model of IBM Ultrastar 36Z15 [1] for the single-speed disk array specification while using the multi-speed power model in reference [10] to set up multi-speed disk array. The detailed parameters are shown in Table 1.

Table 1: Disk Power Model

Parameters	**Value**
Common Disk Parameters	
Individual Disk Capacity	18.4 GB
Max. Disk Rotation Speed	15000 RPM
Idle Power at 15000 RPM	10.2 W
Active Power at 15000 RPM	13.5 W
Seek Power at 15000 RPM	13.5 W
Standby Power	2.5 W
Spin-up Power	13.5 W
Spin-up Time	10.9secs.
Spin-down Time	1.5 secs.
Disk-Arm Scheduling	Elevator
Multi-speed Disk Parameters	
Power Model Type	Quadratic
The number of Power Modes	15
Minimum Disk Rotation Speed	1000 RPM
RPM Step-Size	1000 RPM

5.2 Traces

Different servers have various I/O workloads with diverse characteristics, such as average request arrival interval, request size, request burstness, etc. In order to evaluate real effects of EERAID, we chose to feed different kinds of synthetic workloads into EERAID 1 and EERAID 5 under a wide spectrum of conditions. Based on the server disk trace study of reference [22], in this paper, all workloads consist of 60% read requests and 20% of all requests are sequential without explanation. We set 8 KB as the average request size. According to their average interval-arrive time, these synthetic traces are named 5-ms, 10-ms, 50-ms, 100-ms and 200-ms respectively. In addition, we selected two real-world server-side disk traces to perform further tests. One is from the cello96 trace suite [3] that is collected from the "cello" server over the period from 9 September to 29 November 1996. During that collection time, cello was a K410 class machine (2 CPUs) running HP-UX 10.10, with about 0.5 GB of main memory. The other trace is TPC-C20 [4] running TPC-C database benchmarks with 20 warehouses. In this paper without specification, we configure a six-disk primary group and a six-disk mirror group in RAID 1 and a six-disk RAID 5. However, there are 20 disks and 2 disks in Cello96 and Tpcc traces respectively. When we use Cello96 trace, we select requests out of six busiest disks. For Tpcc trace, we compressed a three-hour period trace into one-hour using round-robin and then produced a six-disk RAID trace. The characteristics of both real traces are listed in Table 2.

Table 2: Trace Characteristics

Trace	Cello96	Tpcc
Average Request Size	4.16 KB	45.96 KB
Average Inter-arrival Time	9.71 ms	25.50 ms
Reads	65.51%	75.07%

[3]http://tesla.hpl.hp.com/public_software
[4]http://tds.cs.byu.edu/tds/index.jsp

5.2.1 Simulators and RAID Configuration

To evaluate different levels of EERAID, we develop four simulators: EERAID1-sp, EERAID5-sp (for EERAID 1 and EERAID 5 employing single-speed disks), EERAID1-mp and EERAID5-mp (for EERAID 1 and EERAID 5 employing multi-speed disks). EERAID1-sp and EERAID5-sp are developed based on Disksim simulators [3] with a single-speed-disk power model and separate EERAID scheduling policies additionally implemented. The conventional RAID deploying single-speed disks is chosen as the baseline for single-speed based EERAID. To test the multi-speed-disk based EERAID, we chose DRPM as a baseline system developed by Gurumurthi et al. [10], which is called PureDRPM in this paper. Similarly, we developed EERAID1-mp and EERAID5-mp based on Disksim [3] and incorporated the multi-speed-disk power model in reference [10], along with separate scheduling policies.

Shortest Queue is deployed in RAID 1 systems. In RAID 5 systems, the stripe size is 8 KB and the data layout is left-symmetric. The RAID controller cache size is 64 MB and both the data stripe and parity stripe are allowed to be cached. The cache replacement algorithm is LRU in PureDRPM while PRD combined with LRU is adopted in EERAID 5. To dynamically adjust the request-window size, we set six to be the maximum window cycle. The maximum window size and step size are 1000 and 200 for traces 50-ms, 100-ms and 200-ms. While for traces 5-ms, 10-ms, Cello96 and Tpcc, the maximum window size and step size are 10000 (2000), 5000 (1000), 5000 (1000), 2000 (500) because they are more I/O intensive. We want to guarantee that EERAID can form idle intervals long enough to shut down and ramp up single-speed disks. The performance degradation threshold is set to be 9%.

6 Experimental Results and Analysis

6.1 Energy Savings

Table 3 shows the results of energy savings of five systems (PureDRPM, EERAID1-sp, EERAID1-mp, EERAID5-sp and EERAID5-mp) compared to the corresponding RAID system without any energy-efficient policy. Here PD, 1-sp, 1-mp, 5-sp and 5-mp represent PureDRPM, EERAID1-sp, EERAID1-mp, EERAID5-sp and EERAID5-mp respectively. The number within a parentheses in columns 1-sp, 1-mp, 5-sp and 5-mp represents the performance (in terms of average request response time) impact calculated by comparing the performance of single-speed disk based EERAID to that of conventional RAID, or comparing multi-speed disks based EERAID to PureDRPM. We will discuss the performance impact in the next section.

From Table 3 we can see that, for single-speed based RAID 1 system, EERAID1-sp can save more energy with the increasing average request inter-arrival time. More than 30% energy is saved in the workload 200-ms while 6% energy can still be conserved in the I/O intensive 5-ms trace though there are not much opportunities to shut down a whole group. For the same reason, EERAID1-sp obtains more energy savings from the real trace Tpcc than it does from Cello96. For multi-speed based RAID 1 system, 50% is no longer to be the upper bound. By applying WRR and PDF, EERAID1-mp can generate much longer idle intervals than PureDRPM since PureDRPM cannot change the request inter-arrival time on purpose. This proves the WRR and PDF in EERAID 1 works extremely well. In all the traces, EERAID1-mp consistently achieves better energy savings than PureDRPM. The extra energy savings is up to 22% with the best gain in the trace 100-ms, compared to PureDRPM. The best energy savings is around 74% in the least I/O intensive trace 200-ms.

Compared with RAID 1, less energy is saved in the single-speed disk based RAID 5 system. Because of the intensive I/O pattern and TRA chance limitation, EERAID5-sp can save at most 11% energy in the lightest workload 200-ms. However, EERAID5-sp can still save energy by 3.6% even for the heaviest workload 5-ms. This indicates the TRA and PRD performs well for EERAID 5. EERAID5-mp is designated to form long idle intervals for current low-power-state disks, but in most cases the average idle interval is not as long as that generated by EERAID1-mp. The reason is that RAID 1 includes a 100% redundancy, which provides a larger scheduling space than that of RAID 5 exhibiting a 1/G redundancy for the redundancy-aware, power-efficient scheduling schemes. Thus, EERAID5-mp achieves less extra energy savings than that of EERAID1-mp. The most extra energy savings obtained by EERAID5-mp is 9%, compared to PureDRPM.

6.2 Performance Impact

EERAID deploys new request scheduling policy (WRR and TRA), cache management policy (PRF and PRD) and disk spin-down scheme (for the single-speed disk). The read request scheduling policies of EERAID always try to direct read requests to some disks (busy group) and let other disks (idle group) have more or longer idle intervals in a request-window. By this way, the workload of the busy group is increased and the response time might be stretched. This is why we want to control the request-window size and adjust it step by step. Like the read request scheduling policies, the cache management policies aim to kick out cache lines linked with busy disks in order to grant the idle group (or low power state) disks longer idle interval. It is possible that this kind

Table 3: Energy Savings and Performance Impact

Trace	RAID 1			RAID 5		
	Single-Speed Disk	Multi-speed Disk		Single-speed Disk	Multi-speed Disk	
	1-sp	PD	1-mp	5-sp	PD	5-mp
5-ms	6.50% (-8.65%)	9.11%	9.13% (-7.14%)	3.61% (-6.26%)	10.06%	10.50% (-3.93%)
10-ms	12.55% (-10.09%)	16.17%	16.24% (-3.30%)	7.93% (-5.64%)	16.27%	16.87% (-1.46%)
50-ms	24.51% (-5.29%)	21.99%	37.85% (-0.38%)	8.81% (-4.13%)	21.45%	37.02% (-0.44%)
100-ms	28.46% (-3.71%)	31.95%	54.47% (+1.27%)	10.31% (-2.94%)	41.54%	44.91% (+2.21%)
200-ms	30.43% (-3.36%)	55.02%	74.63% (+5.11%)	11.04% (-2.32%)	55.03%	64.09% (+4.25%)
Cello96	7.12% (-9.63%)	10.18%	10.51% (+1.18%)	3.06% (-5.69%)	3.77%	4.24% (-3.60%)
Tpcc	20.14% (-4.45%)	15.43%	24.68% (+5.33%)	7.70% (-3.10%)	24.10%	25.79% (+2.87%)

of power-aware replacement policies break the locality of the cached data. Some blocks, which would otherwise seldom be accessed, may have a chance to stay in the cache for a long time because their disks are in idle group (or low power state). These blocks may pollute the cache and thus the system performance may be degraded. Our disk shut-down policies are simple. The disk is shut down to standby when we detect that EERAID will incur a long idle time. We will ramp the disk up when we predict it will not be in the idle group in the near future. However, we cannot guarantee that there is no request coming for the standby disk before we ramp it up. The RAID controller cannot have an infinite cache, so sometimes it has to look for a cache line of the standby disk as victim because the controller cannot find other cache lines of busy disks to evict. We will study the impact of the cache size in Section 6.3. When a standby disk has to ramp up to serve a request, the response time of this request is significantly stretched.

From Table 3, we can see that the performance degradation is controlled well by EERAID 1-sp under light I/O workloads. For example, in a 10-ms workload, the degradation is 10% while with 8% in the trace 5-ms. This is because EERAID 1 has less chances to run WRR in a 5-ms trace period. EERAID1-sp has less performance degradation in Tpcc than Cello96 because Cello96 is a much more intensive workload than Tpcc, with around two times more new requests arrival per second. Because multi-speed disks have much shorter spin-up time than single-speed disks, in I?o intensive workloads 5-ms and 10-ms, EERAID1-mp has less degradation than EERAID1-sp. But for light workloads 50-ms, 100-ms and 200-ms, EERAID1-mp achieves better performance than PureDRPM because the switch frequency between different disk speeds are reduced by WRR. WRR is meant to increase the request burstness and optimize the disk request distribution. The performance degradation of EERAID5-mp is not as much as EERAID5-sp because EERAID5-mp shuts down one disk each time. For both EERAID5-sp and EERAID5-mp, they obtain less performance degradation in Tpcc than that in Cello96 while EERAID5-mp even improves the performance by 3%. The reason is that Tpcc has a longer inter-arrival request time. In addition, Tpcc has a larger average request size and better locality than those of Cello96, which can help facilitate the TRA job.

In summary, the performance of EERAID will not be compromised much under heavy I/O workloads because of our dynamic performance control policy and will be improved by up to 5% under light server workloads.

6.3 Sensitivity study

6.3.1 RAID Size

Since RAID size may be varied by diverse applications, it is important to study how well EERAID1 and EERAID 5 work for different sized RAID. We study the impact of RAID size on the energy savings and performance of EERAID based on single-speed disks. To conduct experiments, we set the RAID size as 4, 6, 8 and 10 disks and choose the synthetic trace 50-ms as workloads. Figure 6 shows the sensitivity results of the RAID size. As we can tell, EERAID1-sp achieves more energy savings and less performance degradation with the increasing RAID size. The reason is that, the average request inter-arrival time of each disk becomes longer in RAIDs with more disks (suppose requests are evenly distributed) and thus EERAID1-sp has more chances to shut down a disk group. While the energy savings is consistently reduced for EERAID5-sp due to the augmented RAID size. In EERAID5-sp, at any point, at most one disk can be shut down. Therefore, the bigger the RAID size, the less energy can be saved. However, the performance degradation is not as sensitive to the RAID size as energy savings. We already know that given a specific workload, the cost of TRA will be more expensive in the larger-sized RAID system. But at the same time, the idle ratio of the disk will also be increased. As a result, the impact of the increasing TRA cost is mitigated.

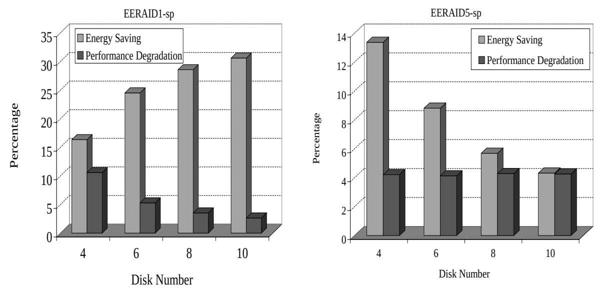

Figure 6: Sensitivity of RAID Size

6.3.2 Cache Size

As discussed in the above section, if the controller has an infinite cache, the effect of cache pollution can be removed. We conducted experiments in conventional RAID 1 systems and found the cache hit rates of Cello96 and Tpcc were 12.95% and 61.17% respectively. By running both traces in EERAID 1, we found the hit rate decreased by 2 5%. By applying an infinite cache to the RAID controller, the cache hit rate of Cello96 and Tpcc are improved to 39.78% and 74.23% respectively. The energy savings of both traces are increased by a factor of 2 to 4. For each disk, the average request interval-arrival time is decreased by the factor of 7.84 for Cello96 and 3.08 for Tpcc. As a result, we can draw the conclusion that, increasing the cache size will result in the stretched average request inter-arrival time.

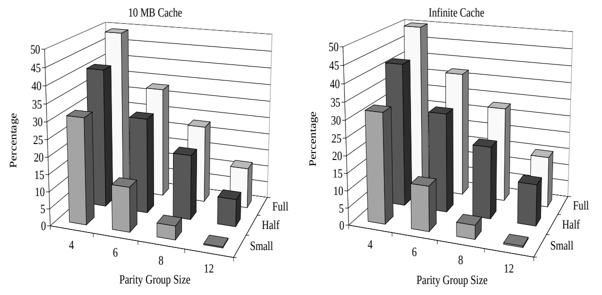

Figure 7: Sensitivity of TRA Chances

In addition, we study whether the cache size has significant impact on the chances of TRA without increasing disk access. We configure a RAID 5 systems with 4, 6 and 8 disks respectively. We select 50-ms as the workload and vary its average request size as Small, Half and Full. All RAID 5 systems have two configuration studies with a 10 MB cache and an infinite cache. We collected results of the chances of TRA (without increasing disk access) under there different cache size, disk numbers and average request sizes in Figure 7. As seen from Figure 7, there is little difference ($< 2\%$) of the results between a RAID 5 with 10 MB cache and the one with an infinite cache. The chance of TRA without increasing disk access is more sensitive to the average request size and parity group size (here it equals the number of disks) instead of the cache size.

6.3.3 Workloads

Other factors that affect the energy savings of EERAID include the read ratio and average request size of the workload. We did the experiment using trace 50-ms and varied the read ratio as 0.2, 0.6 and 1.0 to examine the changes of energy savings. The results are normalized to the result of 50-ms with 0.2 read ratio. In addition, we fix the read ratio as 0.6 and vary the average request size to be Small (8 KB), Half (24 KB) and Full (48 KB). The energy savings results are normalized to that of Small. All the tested RAID systems have the same configuration as that described in Section 5.2.1. Figure 8 gives the changes of energy savings. As seen from the left part

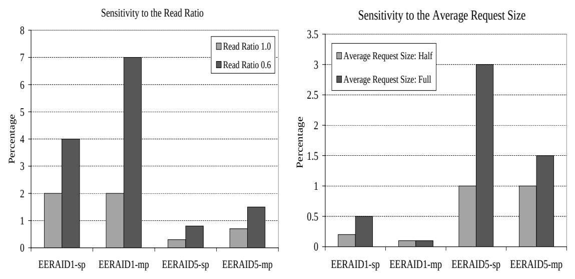

Figure 8: Sensitivity of Read Ratio and Ave. Req. Size

of Figure 8, EERAID 1 is more sensitive to the read ratio than EERAID 5. Because of the limited cache size, the smaller the read ratio, the less disk write operations. When the workload has a higher write ratio, there are more dirty cache lines in the controller cache. The probability of flushing back the cache lines that belong to the idle disk becomes higher and the idle interval period of the idle disk is shorten correspondingly. For the same workload, the probability in EERAID 5 is not as high as that in EERAID 1. The reason is that in EERAID5-sp, there is only one disk in the high priority group and it is possible to find dirty cache lines that belong to the low priority disks. Compared to EERAID 1, the idle intervals in EERAID5-mp are much shorter. The right part of Figure 8 tells us that EERAID 5 is more sensitive to the average request size than EERAID 1. More energy can be conserved as we increase the average request size and thus increasing the chance of TRA without increasing disk access in EERAID 5.

7 Related work

Dynamic disk power management research has been extensively conducted in recent years. A lot of work have been done on investigating how to select thresholds to switch the disk between different power modes [8, 9, 13, 14]. Usually good power-efficient scheduling policies for a single disk are not suitable for the server storage system because the idleness period is too short for the disk to spin down and spin up [4, 11]. Some researchers have studied how to save energy for server disk storage system. Colarelli et al. [7] used "cache disks" to cache active files/blocks, allowing other disks to spin down. How to conserve energy on network servers has also been studied in references [12, 21]. Pinheiro and Bianchini presented a Popular Data Concentration (PDC) scheme to save energy for network servers. PDC aims to skew the load toward a few of all the disks, so that others can be transitioned to low-power modes by migrating frequently accessed data to a subset of the disks [20]. All of the above-mentioned techniques do not work at the RAID system level and thus ignore the details among various disk array organizations. As a result, their approaches either do not work well for making wrong assumptions or could not achieve the best energy-saving. EERAID is able to exploit redundant information to directly optimize the disk access distribution by converting requests from one low-power state disk to other high-power state disk(s). Thus, much more energy savings is attained without compromising system performance.

There are other research works done in disk power modeling. Greenawalt presents a purely analytical model assuming requests arrive according to a Poisson distribution [9]. Zedlewski et al. [26] developed Dempsey, a disk simulation framework used to conduct accurate modeling on disk power consumption, including the power consumption estimation on every disk stage. Some researchers studied how to realize I/O power-efficiency at the operating system level. Zeng et al. developed an ECOSystem in Linux host OS [27], which unified energy accounting over diverse hardware components and enabled fair allocation of available energy among applications. Their focus is on the entire OS level rather than specific I/O devices. Lu et al. [14] presented a design that allowed the applications to be involved in energy management. Several new system calls were introduced to enable applications to inform OS about future hard disk requests. The Coop-I/O approach [25] is presented to reduce the power consumption of devices encompassing all levels in the computer system, from the hardware and OS to a new API used by energy-aware applications. Their design is not compatible to current UNIX-like file systems. EERAID design is orthogonal to the above approaches. With the combination of the above schemes, EERAID can make the storage system more power efficient.

Different levels of RAID performance have been extensively researched [6]. RAID 1 and RAID 5 are two widely used RAID organizations. Chen et al. [6] studied the design issues, the development of mathematical models and examined the performance of different disk array architectures. The performance of several dispatching policies were compared [6]. Write-back caching is a commonly adopted solution to improve the performance RAID 5. Mishra et al. [17] showed a considerable performance improvement can be gained by caching both data stripe and parity stripe in RAID 5 system. Many destage algorithms meant to achieve high-performance have been proposed such as least-cost scheduling, high/low mark and linear threshold scheduling [24]. However, none of them took the energy consumption as a separate trade-off resource.

8 Conclusions

In this paper, we develop novel energy-efficient RAID system architecture called EERAID to significantly save energy by taking advantage of RAID redundant information. To give a proof-of-concept solution, we develop new I/O scheduling and controller-level cache management schemes for EERAID 1 and EERAID 5 respectively. EERAID 1 employs two new policies–Windows Round-Robin (WRR) and Power and Redundancy-Aware Flush (PRF) while EERAID 5 employs another two novel schemes–Transformable Read (TRA) and Power and Redundancy-aware Destage (PRD).

A set of comprehensive trace-driven simulation experiments have been conducted by replaying two real-world server disk traces and wide spectrum server-side synthetic traces. Experimental results showed that 1) For single-speed disks, EERAID 1 can achieve up to 30% energy savings and 11% by EERAID 5, and 2) For multi-speed disks, compared with DRPM, EERAID 1 can achieve 22% extra energy savings and 11% more for EERAID 5. In all experiments, there is either better performance gain or little performance degradation.

References

[1] IBM Hard Disk Drive - Ultrastar 36Z15. http://www.hitachigst.com/hdd/ultra/ul36z15.htm.

[2] Power, heat, and sledgehammer. White paper, maximum Institution Inc., http://www.max-t.com/downloads/whitepapers/SledgehammerPowerHeat20411.pdf, 2002.

[3] J. S. Bucy and G. R. Ganger. The disksim simulation environment version 3.0 reference manual. Technical Report CMU-CS-03-102, Carnegie Mellon University, School of Computer Science, Jan. 2003.

[4] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving disk energy in network servers. In *Proceedings of the 2003 International Conference on Supercomputing (ICS-03)*, pages 86–97, New York, June 23–26 2003. ACM Press.

[5] P. M. Chen, E. L. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID : High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.

[6] S. Chen and D. Towsley. A performance evaluation of RAID architectures. Technical Report UM-CS-1992-067, Departement of Computer Science, University of Masschusetts, Amherst, MA 01003 USA, 1992.

[7] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *SC'2002 Conference CD*, Baltimore, MD, Nov. 2002. IEEE/ACM SIGARCH. pap312.

[8] F. Douglis, P. Krishnan, and B. Bershad. Adaptive disk spin-down policies for mobile computers. In *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing (MLICS'94)*, pages 121–137, Berkeley, CA, USA, Apr. 1995. USENIX Association.

[9] P. Greenawalt. Modeling power management for hard disks. In *Proceedings of the Symposium on Modeling and Simulation of Computer and Telecommunication Systems(MASCOTS 1994)*, pages 62–66, Jan. 1994.

[10] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. DRPM: dynamic speed control for power management in server class disks. In D. DeGroot, editor, *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA-03)*, volume 31, 2 of *Computer Architecture News*, pages 169–181, New York, June 9–11 2003. ACM Press.

[11] S. Gurumurthi, J. Zhang, A. Sivasubramaniam, M. Kandemir, H. Franke, N. Vijaykrishnan, and M. J. Irwin. Interplay of energy and performance for disk arrays running transaction processing workloads. In *Performance Analysis of Systems and Software (ISPASS)*, pages 123–132, Mar. 2003.

[12] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini. Self-confi guring heterogeneous server clusters. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP)*, Sept. 2003.

[13] D. P. Helmbold, D. D. E. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In *Mobile Computing and Networking*, pages 130–142, 1996.

[14] Y.-H. Lu and G. D. Micheli. Adaptive hard disk power management on personal computers. In *Proceedings of the IEEE Great Lakes Symposium*, pages 50–53, Mar. 1999.

[15] J. Menon and J. Cortney. The architecture of a fault-tolerant cached RAID controller. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 76–86, San Diego, California, May 17–19, 1993. ACM SIGARCH and IEEE Computer Society TCCA.

[16] J. Menon and D. Mattson. Performance of disk arrays in transaction processing environments. In *12th International Conference on Distributed Computing Systems (ICDCS '92)*, pages 302–309, Washington, D.C., USA, June 1992. IEEE Computer Society Press.

[17] S. K. Mishra and P. Mohapatra. Performance study of RAID-5 disk arrays with data and parity cache. In *Proceedings of the 25th International Conference on Parallel Processing*, volume I, Architecture, pages I:222–229, Boca Raton, FL, Aug. 1996. CRC Press. Iowa State.

[18] A. E. Papathanasiou and M. L. Scott. Energy effi cient prefetching and caching. In *Proceedings of USENIX Annual Technical Conference*, 2004.

[19] D. A. Pease. Unix disk access patterns revisited. Technical report, Department of Computer Science, Unviersity of California Santa Cruz.

[20] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *Proceedings of the 18th International Conference on Supercomputing*, June 2004.

[21] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power COLP'01*, Sept. 2001.

[22] C. Ruemmler and J. Wilkes. UNIX disk access patterns. In *Usenix Conference*, pages 405–420, Winter 1993.

[23] D. Stodolsky, G. Gibson, and M. Holland. Parity logging: Overcoming the small write problem in redundant disk arrays. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 64–75. IEEE Computer Society Press, May 1993.

[24] A. Varma and Q. Jacobson. Destage algorithms for disk arrays with non-volatile caches. In H. Jin, T. Cortes, and R. Buyya, editors, *High Performance Mass Storage and Parallel I/O: Technologies and Applications*. IEEE/Wiley Press, New York, 2001. chap. 10.

[25] A. Weiel, B. Beutel, and F. Bellosa. Cooperative io - a novel io semantics for energy-aware applications. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, MA, Dec. 2002.

[26] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. Modeling hard-disk power consumption. In *Proceedings of the Second Conference on File and Storage Technologies FAST'03*, pages 217–230, Mar. 2003.

[27] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat. ECOSystem: managing energy as a fi rst class operating system resource. *ACM SIGPLAN Notices*, 37(10):123–132, Oct. 2002.

[28] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing energy consumption of disk storage using power-aware cache management. In *Tenth International Symposium on High Performance Computer Architecture (HPCA-10)*, Madrid, Spain, Feb. 14–18, 2004.