PARAID: A Gear-Shifting Power-Aware RAID

Charles Weddle, Mathew Oldham, Jin Qian, An-I Andy Wang,
Florida State University,{weddle, oldham, qian, awang}@cs.fsu.edu
Peter Reiher, *University of California, Los Angeles,reiher@cs.ucla.edu*
Geoff Kuenning, *Harvey Mudd College, geoff@cs.hmc.edu*

Abstract

Reducing power consumption for server computers is important, since increased energy usage causes increased heat dissipation, greater cooling requirements, reduced computational density, and higher operating costs. For a typical data center, storage accounts for 27% of energy consumption. Conventional server-class RAIDs cannot easily reduce power because loads are balanced to use all disks even for light loads.

We have built the Power-Aware RAID (PARAID), which reduces energy use of commodity server-class disks without specialized hardware. PARAID uses a skewed striping pattern to adapt to the system load by varying the number of powered disks. By spinning disks down during light loads, PARAID can reduce power consumption, while still meeting performance demands, by matching the number of powered disks to the system load. Reliability is achieved by limiting disk power cycles and using different RAID encoding schemes. Based on our five-disk prototype, PARAID uses up to 34% less power than conventional RAIDs, while achieving similar performance and reliability.

1 Introduction

The disk remains a significant source of power usage in modern systems. In Web servers, disks typically account for 24% of the power usage; in proxy servers, 77% [CARR03, HUAN03]. Storage devices can account for as much as 27% of the electricity cost in a typical data center [ZHU04]. The energy spent to operate disks also has a cascading effect on other operating costs. Greater energy consumption leads to more heat dissipation, which in turn leads to greater cooling requirements [MOOR05]. The combined effect also limits the density of computer racks, which leads to more space requirements and thus higher operating costs.

Data centers that use large amounts of energy tend to rely on RAID to store much of their data, so improving the energy efficiency of RAID devices is a promising energy-reduction approach for such installations. Achieving power savings on commodity server-class disks is challenging for many reasons: (1) RAID performance and reliability must be retained for a solution to be an acceptable alternative. (2) To reduce power, a server cannot rely on caching and powering off disks during idle times because such opportunities are not as frequent on servers [GURU03, CARR03, ZHU04]. (3) Conventional RAID balances the load across all disks in the array for maximized disk parallelism and performance [PATT88], which means that all disks are spinning even under a light load. To reduce power consumption, we must create opportunities to power off individual disks. (4) Many legacy reliability encoding schemes rely on data and error-recovery blocks distributed among disks in constrained ways to avoid correlated failures. A solution needs to retrofit legacy reliability encoding schemes transparently. (5) Server-class disks are not designed for frequent power cycles, which reduce life expectancy significantly. Therefore, a solution needs to use a limited number of power cycles to achieve significant energy savings.

Some existing approaches use powered-down RAIDs for archives [COLA02] and trade performance for energy savings [PINH01]. Some studies have exploited special hardware such as multi-speed disks [CARR03, LI04, ZHU05]. Although simulation studies show promising energy savings, multi-speed disks are still far from ubiquitous in large-scale deployments [LI04, YAO06]. With the aid of nonvolatile RAM, approaches that use existing server-class drives have been recently made available [LI04, YAO06, PINH06], but the RAID reliability encoding constraints limit the number of spun-down drives (e.g. one for RAID-5).

We have designed, implemented, and measured the Power-Aware RAID (PARAID), which is deployable with commodity server-class disk drives, without special hardware. PARAID introduces a skewed striping pattern that allows RAID devices to use just enough disks to meet the system load. PARAID can vary the number of powered-on disks by *gear-shifting* or switching among sets of disks to reduce power consumption. Compared to a conventional 5-disk RAID, PARAID can reduce power consumption by up to 34%, while maintaining comparable performance and reliability. Moreover, PARAID reuses different RAID levels so that the underlying RAID technology can evolve independently.

Beyond the power savings obtained by PARAID, the process of creating a real energy measurement framework produced some useful insights into the general problem of measuring energy consumption and savings. These insights are also discussed in this paper.

2 Observations

Over-provisioned resources under RAID: Load balancing allows a conventional RAID device to maximize disk parallelism and performance, and ensures that no disk becomes a bottleneck. This uniformity simplifies data management and allows all disks to be accessed in the same way. However, uniform striping is not favorable for energy savings. Load balancing significantly

reduces opportunities to power off disks because all disks in the array need to be powered to serve a file, even if a RAID receives relatively light loads, when fewer powered disks would be sufficient.

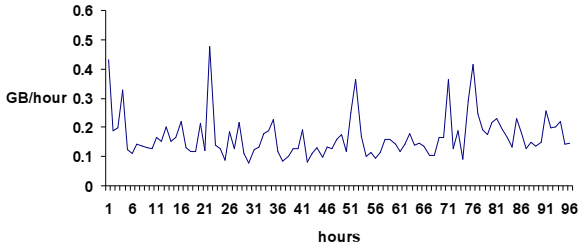


Figure 2.1: UCLA Computer Science Department web server activity from August 11 through August 14, 2006.

Cyclic fluctuating load: Many system loads display cyclic fluctuations [CHAS01]. Figure 2.1 shows the web traffic gathered at the UCLA Computer Science Department across one week. The load fluctuations roughly follow daily cycles. Depending on the types of traffic, different systems may exhibit different fluctuation patterns, with varying ranges of light to heavy loads [IYEN00].

We can exploit these patterns by varying the number of powered disks, while still meeting performance needs and minimizing the number of power switches. A few strategically timed power cycles can achieve significant power savings.

Unused storage space: Storage capacity is outgrowing demand for many computing environments, and various large-scale installations report only 30% to 60% storage allocation [ASAR05, GRAY05, LEVI06]. Researchers have been looking for creative ways to use the free storage (e.g. trading off capacity for performance [YU00] and storing every version of file updates [SANT99]).

Additionally, many companies purchase storage with performance as the top criterion. Therefore, they may need many disks for parallelism to aggregate bandwidth, while the associated space is left largely unused. Further, administrators tend to purchase more space in advance to avoid frequent upgrades. Unused storage can then be used opportunistically for data-block replication to help reduce power consumption.

Performance versus energy optimizations: Performance benefits are realized only when a system is under a heavy load, and may not result in an immediate monetary return. Energy savings, however, are available at once, and could, for example, be invested in more computers. Also, unlike performance, which is purchased in chunks as new machines are acquired, energy savings can be invested immediately and compounded over the lifetime of the computers. Therefore, if a server usually operates below its peak load, optimizing energy efficiency is attractive.

3 Power-Aware RAID

The main design issues for PARAID are how to skew disk striping to allow opportunities for energy savings and how to preserve performance and reliability.

3.1 Skewed Striping for Energy Savings

PARAID exploits unused storage to replicate and stripe data blocks in a skewed fashion, so that disks can be organized into hierarchical overlapping sets of RAIDs. Each set contains a different number of disks, and can serve all requests via either its data blocks or replicated blocks. Each set is analogous to a gear in automobiles, since different numbers of disks offer different levels of parallelism and aggregate disk bandwidth.

The replicated blocks are soft states, in the sense that they can be easily reproduced. Thus, as storage demands rise, replicated blocks can be reclaimed by reducing the number of gears. Unlike memory caches, these soft states persist across reboots.

Figure 3.1.1 shows an example of replicated data blocks persisting in soft states in the unused disk regions. By organizing disks into gears, PARAID can operate in different modes. When operating in gear 1, with disks 1 and 2 powered, disks 3 and 4 can be powered off. As the load increases, PARAID up-shifts into second gear by powering up the third disk.

By adjusting the number of gears and the number of disks in each gear, PARAID provisions disk parallelism and bandwidth so as to follow the fluctuating performance demand curve closely through the day. By creating opportunities to spin down disk drives, PARAID conserves power.

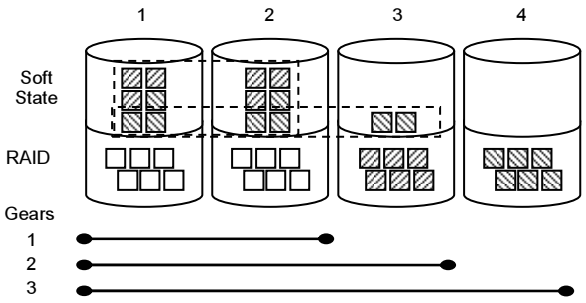


Figure 3.1.1: Skewed striping of replicated blocks in soft state, creating three RAID gears over four disks.

While more gears can match the performance demand curve more closely, the number of gears is constrained by the unused storage available and the need for update propagation when switching gears. To minimize overhead, the gear configuration also needs to consider the number of gears and gear switches.

3.2 Preserving Peak Performance

PARAID matches the peak performance of conventional RAIDs by preserving the original disk layouts when operating at the highest gear. This constraint also

allows PARAID to introduce minimal disturbances to the data path when the highest gear is in use.

In low gears, since PARAID offers less parallelism, the bandwidth offered is less than that of a conventional RAID. Fortunately, the number of requests affected by this performance degradation is significantly smaller compared to those affected during peak hours. Also, as bandwidth demand increases, PARAID will up-shift the gear to increase disk parallelism.

PARAID also can potentially improve performance in low-gear settings. As a gear downshifts, the transfer of data to the soft state from disks about to be spun down warms up the cache, thus reducing the effect of seeking between blocks stored in different gears.

3.3 Retaining Reliability

To retain conventional RAID reliability, PARAID must be able to tolerate disk failures. To accomplish this goal, PARAID needs to supply the data redundancy of conventional RAIDs and address the reduced life expectancy of server-class disks due to power cycles.

PARAID is designed to be a device layer sitting between an arbitrary RAID device and its physical devices. Thus, PARAID inherits the level of data redundancy, striping granularity, and disk layout for the highest gear provided by that RAID. For example, a PARAID device composed with a RAID-5 device would still be able to rebuild a lost disk in the event of disk failure. (The details of failure recovery will be discussed in Section 4.4.)

Because PARAID power-cycles disks to save energy, it must also address a new reliability concern. Power-cycling reduces the MTTF of a disk, which is designed for an expected number of cycles during its lifetime. For example, the disks used in this work have a 20,000-power-cycle rating [FUJI05]. Every time a disk is power-cycled, it comes closer to eventual failure.

PARAID limits the power cycling of the disks by inducing a bimodal distribution of busy and idle disks. The busier disks stay powered on, and the more idle disks often stay off, leaving a set of middle-range disks that are power-cycled more frequently. PARAID can then prolong the MTTF of a PARAID device as a whole by rotating the gear-membership role of the disks and balancing their current number of power cycles.

Further, PARAID limits the power cycles for disks. By rationing power cycles, PARAID can operate with an eye to targeted life expectancy. For example, if the disks have a five-year life expectancy due to the system upgrade policy, and the disks are expected to tolerate 20,000 cycles, then each disk in the array cannot be power cycled more than 10 times a day. Once any of the disks has reached the rationed numbers of power cycles for a given period, PARAID can operate at the highest gear without energy savings. The power-saving mode resumes at the next rationing period.

4 PARAID Components

PARAID has four major components—a block handler, monitor, reliability manager, and disk manager (Figure 4.1)—responsible for handling block I/Os, replication, gear shifting, update propagation, and reliability.

4.1 Disk Layout and Data Flow

PARAID is a new device layer in the conventional software RAID multi-device driver. The block handler under PARAID transparently remaps requests from a conventional RAID device and forwards them to other soft-state RAID devices or individual disk devices.

PARAID currently delegates RAID regions to store replicated soft states for individual gears. The highest gear reuses the original RAID level and disk layout to preserve the peak performance. When the highest gear is active, PARAID forwards requests and replies with minimal disturbance to the data path.

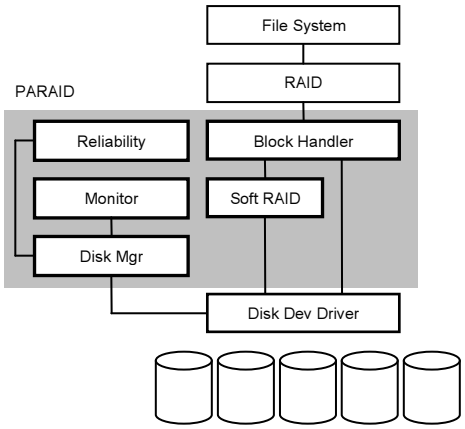


Figure 4.1: PARAID system components.

However, the data and parity blocks of D disks cannot be striped across fewer disks to achieve the same level of redundancy. If we simply assigned the Dth block to one of the still-powered disks, it would be possible for a single drive to lose both a data block and a parity block from the same stripe, while the block stored on the powered-off disk might be out of date.

To provide reliability, the soft-state replicated blocks stored in each gear use the same RAID level. For example, consider a 5-disk RAID-5 (Table 4.1). Gear 2 uses all 5 disks; gear 1 uses 4. When disk 5 is spun down, its blocks must be stored on the remaining 4 disks. This is done by creating a 4-disk soft-state RAID-5 partition; the data and parity blocks from disk 5 are stored in this partition as if they were normal data blocks arriving directly from the application. If necessary, the soft-state partition can be removed to recover space whenever disk 5 is spinning.

The synchronization between disk 5 of gear 2 and the blocks in gear 1 resembles the data flow of RAID1+0. Disk 5 is "mirrored" using RAID-5 on gear 1, with synchronization performed during gear shifts.

By using the underlying RAID-5 code for disk layout and parity handling, the PARAID code is drastically simplified compared to trying to deal with those details internally.

	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
Gear 1	(1-4)	8	12	((1-4),8,12)	
RAID-5	16	20	(16,20, _)	_	
	1	2	3	4	(1-4)
Gear 2	5	6	7	(5-8)	8
RAID-5	9	10	(9-12)	11	12
	13	(13-16)	14	15	16
	(17-20)	17	18	19	20

Table 4.1: PARAID disk layout with one 4-disk gear and one 5-disk gear, each running RAID-5. Each table entry contains either a block number or numbers enclosed with parentheses, denoting a parity block. "_" means an empty block.

For all gears (including the case where all disks are powered), if either a read or write request is sent to a powered disk, the disk simply serves the request. If a request is sent to a powered-off disk, then PARAID will remap the request to a replicated block stored on a powered disk. A remapped update is later propagated to neighboring gears during gear shifts.

The required unused storage depends on the RAID level, the number of gears, and the number of disks in each gear. For RAID-5, $D > 3$ disks, M gears with G_i disks within the ith gear ($1 \le i \le M$, $3 \le G_i < G_{i+1} < G_M = D$) the percentage storage consumption S_i of the total RAID for the ith gear can be solved with M equations:

$$\begin{cases} \sum_{i=1}^{M} S_i = 1 \\ \left(\sum_{j=i}^{M} S_j\right)(G_i - G_{i-1}) = (G_{i-1} - 1)S_{i-1}, i = 2..M \end{cases} \quad (1)$$

For a disk in the lowest gear (Figure 3.1.1, disk 1), the sum of the percentage usage of disk space by each gear must be one. Also, for a gear (Figure 3.1.1, gear 2) to be able to shift to a lower gear (Figure 3.1.1, gear 1), the lower gear must store all the content of the disk(s) (Figure 3.1.1, disk 3) that are about to be spun down, with their parity information created for the lower gear.

PARAID uses around $(D - G_1)/(D - 1)$ of the total RAID-5 storage to store soft states. This estimate is largely based on the number of disks in the lowest gear, not the number of gears or the number of disks in intermediate gears, so the overhead of gear switching and the time spent in each gear will determine optimal gear configurations.

The target percentage of energy savings for an active system (not specific to RAID-5) is described by formula (2), where $P_{standby}$ is the power consumption for a spun-down disk (more details are given in the performance section), and $P_{active/idle}$ is the average power consumption for either a busy disk or an idle disk, to compute disk power savings for busy or idle loads.

Power savings increase with more disks, fewer disks in the lowest gear, and a higher $P_{active}/P_{standby}$ ratio. Since spun-down disks still consume power, it is better to install PARAID with large disks with unused space, rather than buying more disks later.

$$1 - \frac{(D - G_1)P_{standby} + G_1 P_{active/idle}}{DP_{active/idle}} \quad (2)$$

For this paper, an up-shift means switching from a gear with G_i disks to G_{i+1} disks; a downshift, switching from a gear with G_i disks to G_{i-1} disks. A gear switch can be either an up-shift or a downshift.

4.2 Update Propagation

When a powered-off disk misses a write request, it must synchronize the stale data either when powered on or just before the stale information is accessed. If there is a lot of stale data, fully synchronizing a disk can be slow. The on-demand approach updates stale data only when it is accessed, allowing the gear shift to take place much more swiftly, but the full-synchronization approach is simpler to build. The on-demand approach is not applicable for downshifts, since PARAID needs to finish the propagation before spinning down drives.

The disk manager captures outstanding writes to powered-off disks. For full synchronization, the disk manager reissues outstanding writes to the disk when it is powered on, possibly rereading some data from replicated soft states stored in the current gear.

For on-demand synchronization, the PARAID block I/O handler uses a dirty-block list. If a referenced dirty block is not cached, PARAID will retrieve the block from the original gear and return it to the requestor. PARAID will then write that block to the target-gear disks, effectively piggybacking the synchronization step at access time.

The disk manager must track stale block locations for synchronization. This list of dirty blocks is stored on disk in case of system failure and in memory for fast access.

A failed disk can stop the gear-shifting process. Disks can also fail during synchronization. However, the list of outstanding writes is maintained throughout the disk failure and recovery process. Once the failed disk recovers, the synchronization can resume.

The choice of on-demand or full synchronization for up-shifting is configurable. On-demand allows PARAID to be more responsive to sudden request bursts, at the cost of tracking additional writes for unsynchronized disks. The full-synchronization approach may be preferable for few gear shifts and a read-dominated workload, since the number of blocks to be synchronized is small. The full synchronization method is also available for manual maintenance, such as when an administrator would need to have a consistent system state before pulling out a disk.

4.3 Asymmetric Gear-Shifting Policies

The disk manager performs shifts between gears. The PARAID monitor decides when a shift is needed, and the disk manager then performs the actual power cycles.

Switching to a higher gear is aggressive, so that the PARAID device can respond quickly to a sharp and sustained increase in workload. However, the algorithm should be resilient to short bursts, or it will lead to little energy savings. Downshifting needs to be done conservatively, so that wild swings in system activity will not (1) mislead the PARAID device into a gear that cannot handle the requests, or (2) cause rapid oscillations between gears and significantly shorten the life expectancy of disks.

Up-shifts: To decide when to up-shift, the monitor must know whether the current gear has reached a predetermined utilization threshold, in terms of busy RAID milliseconds within a time window. Interestingly, we could not check the disk-busy status directly, since this probe would spin up a powered-down disk. Instead, an active RAID device is marked busy from the point when a request enters the RAID queue to when the completion callback function is invoked. Since multiple RAID requests can overlap, should a request be completed with an elapsed time of t milliseconds, we mark the prior t milliseconds busy.

The threshold and time window are configurable, and are set to 80% (based on prior studies [CARR03]) and 32 seconds (based on empirical experience). The intent is that within the time it takes to spin up the disk and propagate updates, the utilization threshold will not reach 100%. The use of an online algorithm to set thresholds automatically will be future work.

To track the system load, the monitor keeps a moving average of utilization $0 \leq U \leq 1$ for each gear. The purpose of averaging is to filter out short bursts of requests that are frequently seen in real-world workloads. The monitor also keeps a moving standard deviation S. If the utilization plus its standard deviation exceeds the threshold $0 \leq T \leq 1$, an up-shift is performed.

$$U + S > T \qquad \text{(Up-shift condition)}$$

The addition of standard deviation makes up-shift more aggressive; however, since both the moving average and the standard deviation lag behind the actual load, the policy is more responsive to changes that lead to sustained activities.

Downshifts: To decide when to downshift, the utilization of the lower gear $0 \leq U' \leq 1$ needs to be computed, with associated moving standard deviation S'. If their sum is below the threshold T, the lower gear can now handle the resulting load, with associated fluctuations.

$$U' + S' < T \qquad \text{(Downshift condition)}$$

A complication arises when each gear is stored in RAID with parity blocks. Suppose gear 2 contains a 5-disk RAID-5, and the 5th disk is replicated in gear 1 with a 4-disk RAID-5. After a downshift (i.e. spinning down the 5th disk), a write disk request within PARAID will have a 20% chance of accessing the spun-down disk, resulting in a parity update for gear 2, and another parity update for gear 1. Therefore, to compute the downshift threshold, the monitor must track recent write activity and inflate the percentage of write accesses A_{write} to the to-be-spun-down disk(s) by a weight W of 1.5x (specific to RAID-5, where writes to 1 data block and 1 parity block can be increased to 1 data block and 2 parity blocks). Otherwise, the lower gear will be unable to handle the resulting load, and will shift back up. Therefore, U' is computed with the following formula:

$$U' = U \frac{G_i}{G_{i-1}} \left[A_{read} + A_{write} \left(\frac{G_{i-1}}{G_i} + \frac{G_i - G_{i-1}}{G_i} W \right) \right]$$

4.4 Reliability

The reliability manager rations power cycles and exchanges the roles of gear memberships to prolong the life expectancy of the entire PARAID. The reliability manager is also responsible for recovering a PARAID device upon disk failure. When PARAID fails at the highest gear, the recovery is performed by the RAID of the highest gear. When PARAID fails in other gears, the recovery is first performed by the lowest gear containing the failed disk, since the parity computed for disks in that gear is sufficient to recover the soft states stored on the failed disk. The recovered soft-state data then is propagated to the next higher gear before recovering that gear, and so on. In the worst case, the number of bytes needing to be recovered for a single drive failure is the size of a single disk.

Although PARAID may take much longer to recover in the worst case due to cascaded recoveries, the average recovery time can be potentially reduced by recovering only modified content in the intermediary gears and frequent switching to the highest gears. To illustrate, should a PARAID host read-only content, recovery only involves switching to the highest gear and performing the recovery with the underlying RAID once, since no cascaded update propagations are needed. With modified content, PARAID can selectively recover only the modified stripes and stripes used to recover modified stripes at intermediary gears and propagate them to the highest gear, where a full recovery is performed. Assuming that 2% of disk content is modified per day [KUEN97], and PARAID switches to the highest gear 10 times a day, lightweight cascaded recovery is theoretically possible.

One might argue that PARAID can lengthen the recovery time, and thus reduce the availability of PARAID. On the other hand, PARAID reduces power consumption, and the associated heat reduction can

extend drive life by about 1 percent per degree Celsius [HERB06]. Therefore, the tradeoff requires further studies, which is beyond the scope of this paper.

5 Implementation

PARAID was prototyped on Linux (2.6.5), which was chosen for its open source and its software RAID module. The block I/O handler, monitor, disk manager, and reliability manager are built as kernel modules. A PARAID User Administration Tool runs in user space to help manage the PARAID devices. For reliability, data blocks for all gears are protected by the same RAID level. Although we have not implemented drive rotations, our gear-shifting policies and the characteristics of daily work cycles have limited the number of disk power cycles quite well. We have not implemented the mechanisms to recover only modified stripes in intermediary gears to speed up cascaded recovery.

Linux uses the `md` (multiple device) device driver module to build software RAIDs from multiple disks. For the PARAID block handler implementation, we changed the `md` driver to make it PARAID-aware. The data path of the `md` driver is intercepted by the PARAID device layer, so that requests from conventional RAID are redirected to the block queues of PARAID, which remaps and forwards requests to other conventional RAID-device queues.

During initialization, the PARAID-aware `md` module starts a daemon that provides heartbeats to the PARAID device and calls the monitor periodically to decide when to gear-shift. The disk manager controls the power status of disks through the disk device I/O control interface.

As an optimization, to limit the synchronization of content of a powered-off disk only to updated content, the disk manager keeps a per-disk red-black tree of references to outstanding blocks to be updated. This tree is changed whenever an update is made to a clean block on a powered-off disk. The upkeep of this data structure is not CPU-intensive. Currently, the disk manager synchronizes all modified blocks after bringing back powered-off disks, by iterating through the tree for each disk and reissuing all outstanding writes. For each block to be synchronized, the disk manager reads the block from the original gear, and then writes it to the disks being brought back online. Once synchronization is complete, the gear-shifting manager switches to the new gear. Note that the red-black tree is only an optimization. In the case of losing this tree, gear content will be fully propagated. A new tree can be constructed once PARAID gear switches to the highest gear.

Currently, PARAID serves requests from the current gear until the target gear completes synchronization, a conservative method chosen for implementation ease and to assure that no block dependency is violated through update ordering. In the future, we will explore using back pointers [ABDE05] to allow the new gear to be used during update propagation.

For the PARAID monitor, we currently use 32-second time windows to compute moving averages of disk utilization. The choice of this time window is somewhat arbitrary, but it works well for our workloads and can tolerate traffic bursts and dampen the rate of power cycles. Further investigation of the gear-shifting triggering conditions will be future work.

The `mkraid` tool, commonly used by Linux to configure RAIDs, had to be changed to handle making PARAID devices and insertion of entries in `/etc/raidtab`. Additional `raidtab` parameters had to be defined to be able to specify the gears.

PARAID contains 3,392 lines of modified code to the Linux and `Raidtools` source code. Since the PARAID logic is contained mostly in the Linux Software RAID implementation, it should be portable to future Linux kernel versions and software RAID implementations in other operating systems. We inserted four lines into `raid0.c` and `raid5.c` to set a flag to forward the resulting I/O requests to PARAID.

6 Performance Evaluation

Since the study of energy-efficient approaches to RAIDs is relatively recent, most prior work has been done analytically or via simulations. Analytical methods provide a fundamental understanding of systems. Simulation studies allow for the exploration of a vast parameter space to understand system behaviors under a wide range of scenarios. We chose implementation and empirical measurements to see if we could overcome unforeseen physical obstacles and conceptual blind spots to bring us one step closer to a deployable prototype. When we designed, implemented, and evaluated PARAID, we discovered why an empirical study is difficult for systems designed to save energy.

- Prototyping PARAID was the first barrier, since the system had to be stable enough to withstand heavy benchmarking workloads.
- Commercial machines are not designed for energy measurements, so we had to rewire drives, power supplies and probes for power measurements.
- The conceptual behaviors of a system are far from close to its physical behaviors; therefore, we had to adjust our design along the way.
- Most benchmarks and workload generators measure the peak performance of a system at steady state, which is not applicable for measuring energy savings, for which we need to capture daily workload fluctuations.
- For trace replays, since our physical system configuration was largely fixed, we had to try to match different trace environments with our physical environments in terms of the memory size, traffic volume, disk space consumption, and so on.

- Although many research trace replay tools are available, more sophisticated ones tend to involve kernel hooks and specific environments. Incompatibility of kernel versions prevented us from leveraging many research tools.
- Finally, since it cannot be easily automated and cheaply parallelized, measuring energy savings on a server was very time-consuming.

Considering these challenges, we document our experimental settings to obtain our results. We demonstrate the power savings and the performance characteristics of PARAID by replaying a web trace (Section 6.1) and the Cello99 trace [HP06] (Section 7). The web workload contains 98% reads and is representative of a very large class of useful workloads. The Cello99 workload is I/O intensive, and consists of 42% writes. We used the PostMark benchmark [KATC97] (Section 8) to demonstrate PARAID's performance under peak load. To demonstrate that PARAID can reuse different RAID levels, PARAID was configured with RAID-0 for the Web workload, and RAID-5 for the Cello99 workload. The PostMark benchmark stresses the gear-shifting overhead. All experiments were conducted five times. Error curves were removed from graphs for clarity. Generally, the standard deviations are within 5% of the measured values, with the exceptions of latency and bandwidth numbers, which tend to be highly variable.

6.1 Web Trace Replay Framework

The measurement framework consisted of a Windows XP client and a Linux 2.6.5 server. The client performed trace playback and lightweight gathering of measurement results, and the server hosted a web server running on a RAID storage device [FUJI06] (Table and Figure 6.1.1). On the server, one disk was used for bootstrapping, and five disks were used to experiment with different RAIDs. The client and server were connected directly by a CAT-6 crossover cable to avoid interference from extraneous network traffic.

	Server	Client
Processor	Intel Xeon 2.8 Ghz	Intel Pentium 4 2.8 Ghz
Memory	512 Mbytes	1 Gbytes
Network	Gigabit Ethernet	Gigabit Ethernet
Disks [FUJI06]	Fujitsu MAP3367 36.7Gbytes 15K RPM SCSI Ultra 320 8MB on-disk cache 1 disk for booting 5 disks for RAID experiments Power consumption: 9.6 W (active) 6.5 W idle (spinning) 2.9 W standby (spun-down, empirically measured)	Seagate Barracuda ST3160023AS 160 Gbytes 7200 RPM SATA

Table 6.1.1: Hardware specifications.

To measure the power of the disks, we used an Agilent 34970A digital multimeter. Each disk probe was connected to the multimeter on a unique channel, and the multimeter sent averaged data to the client once per second per channel via a universal serial bus.

To measure the power of a disk, we inserted a 0.1-Ω resistor in series in the power-supply line (Figure 6.1.2). The multimeter measured the voltage drop across the resistor, V_r. The current I through the resistor—which is also the current used by the disk—can be calculated as V_r/R. Given the voltage drop across the disk, V_d, its power consumption is then V_d times I.

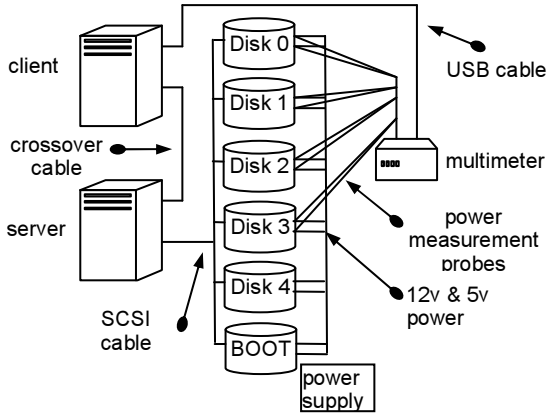


Figure 6.1.1: The measurement framework.

In the measurement system, we removed each disk from the server and introduced a resistor into its +12V and +5V power lines. The +12V line supplied power to the spindle motor; the +5V line provided power to the disk electronics. The SCSI cable was connected directly to the motherboard, allowing the cable to maintain the same performance as if the disks were connected to the SCSI hot swappable backplane in the server.

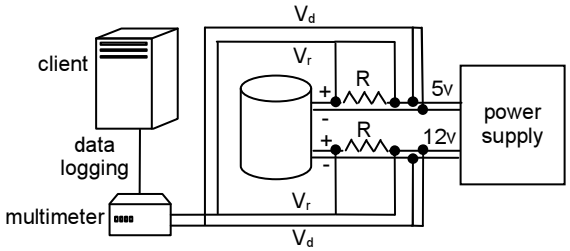


Figure 6.1.2: The resistor inserted in series between the power supply and the disk adapter.

On the client, the Agilent Multimeter software logged the data using Microsoft Excel XP. The multi-threaded trace driver, implemented in Java 1.5, was designed to replay web access log traces and collect performance numbers. Associated requests generated from the same IP address are each handled by a separate thread, to emulate users clicking through web pages. The trace driver also collected server-side and end-to-end performance numbers.

The server hosted an Apache 2.0.52 web server on top of an ext2 file system operating over a RAID storage device that is described in Table 6.1.1.

6.2 Web Server Workload

Workload characteristics affect PARAID's ability to save energy. Under a constant high load, PARAID will not have opportunities to downshift and save energy. Under a constant light load, trivial techniques like turning everything on and off can be used to save energy. In practice, workloads tend to show cyclic fluctuations. The chosen workload needs to capture these fluctuations to demonstrate PARAID's energy savings.

We chose a web server workload from the UCLA Computer Science Department. Since the web content is stored in a decentralized fashion via NFS mounts, we only report the hardware configuration of the top-level web server, which is a Sun Ultra-2 with 256 Mbytes of RAM, 200 Mhz UltraSPARC I CPU, one 2-Gbyte system disk and one 18-Gbyte external SCSI disk, running Apache 1.3.27. Activity was captured from August 10, 2006 to August 16, 2006. Various NFS file systems contained approximately 32 Gbytes of data and ~500K files. We recreated the file system based on the referenced files in the trace. For each full path referenced, every directory in the full path and the file was created according to the order of replay. The file blocks stored on the web server were refilled with random bits. Also, the replay did not include dynamic file content, which accounts for relatively few references in this trace.

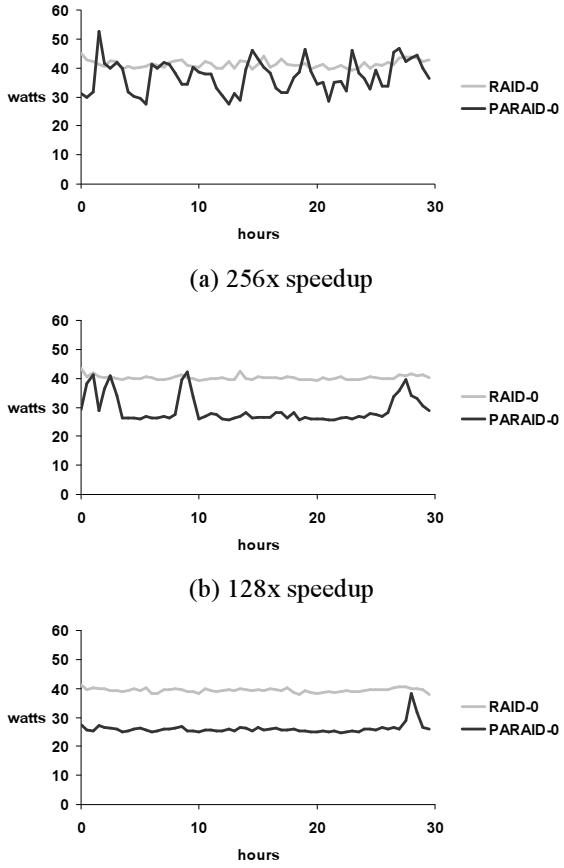
We chose a 30-hour trace starting from 6 PM, August 12, 2006. The duration included 95K requests, with 4.2 Gbytes of data, of which 255 Mbytes are unique. Although the workload is light, it captures the essence of read-mostly cyclic loads and sheds light on PARAID system behaviors, gear-shifting overhead, and the practical implementation limits on power savings.

6.3 Web Trace Replay Experimental Settings

PARAID was compared with a RAID-0 device. The PARAID device used 5 disks, with 2 disks in gear 1, and 5 disks in gear 2. Both client and server were rebooted before each experiment, and PARAID was configured to start with the lowest gear, with gear content pre-populated. The client replayed trace log entries to the server. Due to the hardware mismatch and light trace workload, the collected trace was accelerated at different speeds to illustrate the range of possible savings with different levels of workloads. Experiments included a 256x speedup, which is close to a zero-think-time model, translating to 241 requests/second. With this reference point, we lowered the speedup factor to 128x and 64x, which correspond to 121 and 60 requests/second. All three loads offer few opportunities for the entire 5-disk RAID to be power-switched as a whole. Timing dependent on human interactions, such as the time between user mouse clicks on links (i.e. reference intervals by the same IP) was not accelerated.

6.4 Power Savings

Figure 6.4.1 compares the power consumption of PARAID and RAID-0. Due to the effects of averaging, power spikes are not visible.



(a) 256x speedup

(b) 128x speedup

(c) 64x speedup

Figure 6.4.1: Power consumption for web replay.

PARAID demonstrates a 34% overall savings (ratios of areas under the curves) at 64x. The results approximately match the 33 - 42% range based on equation (2), indicating that further load reduction will yield limited energy benefits. However, turning off 3 out of 5 drives achieves nowhere near 60% energy savings, for PARAID or other RAID systems that save power by spinning down disks. Powering off a disk only stopped it from spinning its platter and therefore only the 12V line was shut off. Power was still needed for the 5V line that powered the electronics, so that it could listen for a power-up command and pass commands along the daisy-chained SCSI cable.

Based on our measurements, spinning up a disk can consume 20-24W. Also, a spun-down disk still consumes 2.9W, noticeably higher than the 1.0W to 2.5W extracted from various datasheets and used in many simulations [GURU03, HUAN03, PINH04, ZHU04, ZHU04B, ZHU05]. The results show that variations in physical characteristics can change the expected energy

savings significantly. In our case, if we replace our Fujitsu [FUJI06] with the commonly cited IBM Ultrastar 36Z15 [IBM06], we anticipate an additional 5% energy savings.

The second observation is that the traffic pattern observed in the web log does not correlate well with the disk power consumption. Although this finding reveals more about the nature of caching than the energy benefits of PARAID, it does suggest the value of further investigations into the relationship between server-level activities and after-cache device-level activities.

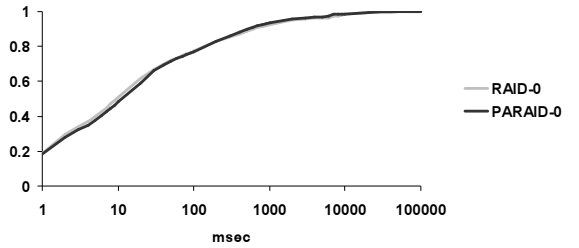
Table 6.4.1 summarizes the overall PARAID energy savings.

Speed-up	Power savings
256x (241 req/sec)	10%
128x (121 req/sec)	28%
64x (60 req/sec)	34%

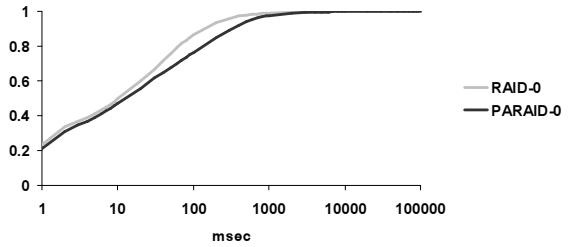
Table 6.4.1: Percent energy saved for web replay.

6.5 Performance

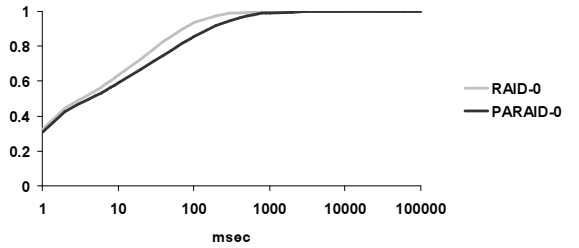
Latency: Figure 6.5.3 shows the CDFs of per-request latency, which measures the time from the last byte of the request sent from the client to the first byte of data received at the client.



(a) 256x speedup



(b) 128x speedup



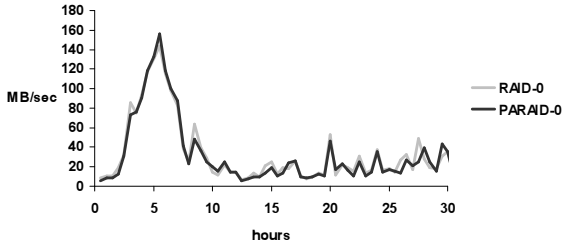
(c) 64x speedup

Figure 6.5.3: Latency for web replay.

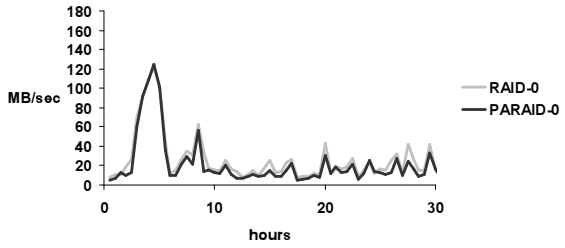
As expected, when playing back the trace at high speed, PARAID spent more time at the high gear and used the original RAID-5 disk layout, and the latency CDFs matched closely. The average latency is within 2.7% (~840ms). The data path overhead of PARAID is negligible (Section 8).

When the load was light at 64x, PARAID spent most of the time at the lower gear. PARAID-0 had to use 2 disks to consolidate requests for 5 disks. As a result, the average latency PARAID-0 was 80ms compared to 33ms of RAID-0. However, a web end user should not notice the response time difference during light loads.

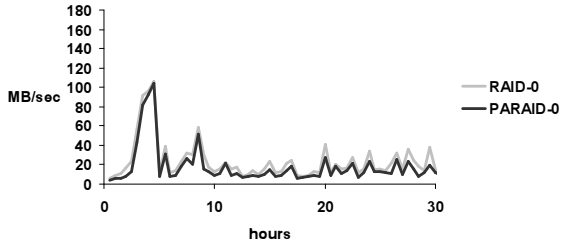
Bandwidth: Figure 6.5.4 shows the bandwidth over time, which measures the number of bytes transferred in a 30-minute interval, divided by the time the client spent waiting for any request to complete within the same interval.



(a) 256x speedup



(b) 128x speedup



(c) 64x speedup

Figure 6.5.4: Bandwidth for web replay.

As expected, when PARAID operates mostly in low gear, having fewer active disks leads to lower bandwidth numbers during light loads (Figure 6.5.4 (c)), 24 MB/sec as opposed to 31 MB/sec for RAID-0. However, during the time intervals when PARAID operates in high gear, the peak load bandwidth matches well with the original RAID (within 1.3% of 32MB/sec).

Note that due to time-based data alignment and averaging effects, Figure 6.5.4 (a) only shows a close bandwidth match when PARAID's high-gear performance dominates within a time bracket. Section 7 will also explore request-based alignment to demonstrate bandwidth matching.

Gear-switching statistics: Table 6.5.1 summarizes various PARAID gear-switching statistics for the web replay experiment. Clearly, PARAID spends more time in the low gear as the intensity of workload decreases with the replay speed. Also, each gear switch introduces up to 0.1% extra system I/Os. Interestingly, frequent gear switches can reduce the per-switch cost down to 0.041%, since less time is available for updates to accumulate at a given gear.

	256x	128x	64x
Number of gear switches	15.2	8.0	2.0
% time spent in low gear	52%	88%	98%
% extra I/Os for update propagations	0.63%	0.37%	0.21%

Table 6.5.1: PARAID gear-switching statistics for web replay.

7 HP Cello99 Replay

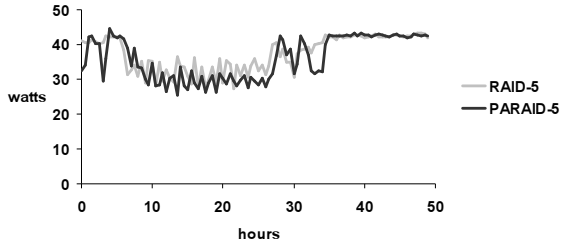
The HP Cello99 trace [HP06] is a SCSI-controller-level trace collected by the HP Storage Research Lab from January 14 to December 31, 1999. The Cello99 data represents IO-intensive activity with writes, which is in contrast to the read-mostly UCLA web with lighter traffic. The traced machine had 4 PA-RISC CPUs, and some devices are md devices, so we had to extract a trace that neither overwhelms our system nor produces too little traffic. The `spc` formatted trace file was generated from the Cello99 data using `SRT2txt`, a program that comes with the HP Cello99 data. The generated trace file was further trimmed so that only the activity associated with `lun 2` was used. Also, we looked for traces with cyclic behaviors. The extracted trace contains 50 hours beginning on September 12, 1999, consisting of ~1.5M requests, totaling 12 GB (stored in 110K unique blocks).

7.1 Cello99 Experimental Settings

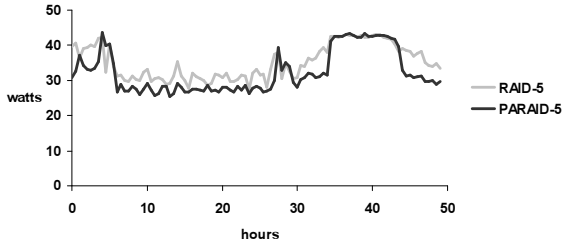
PARAID was compared this time with a RAID-5 device. We used a 3-disk gear and 5-disk gear, each reusing the RAID-5 disk layout and reliability mechanisms. The Cello99 trace was replayed on the server at 128x, 64x, and 32x speedup factors to vary the intensity of workloads, corresponding to 1020, 548, and 274 requests/second. The energy measurement framework is the same as depicted in Figure 6.1.1. The server was rebooted before each run, with PARAID configured to start in the lowest gear.

7.2 Power Savings

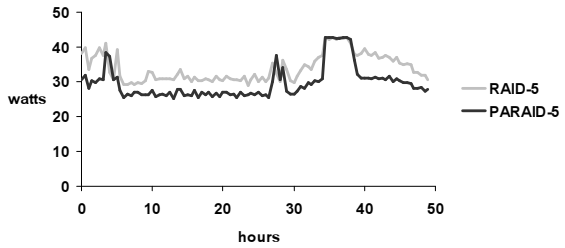
Figure 7.2.1 compares the power consumptions of PARAID and RAID-5. PARAID demonstrates a single- point-in-time savings of 30% at 128x speedup (~13 hours into the replay) and a 13% overall power savings at 32x speedup. Equation (2) suggests a power saving range of 22 - 28%. Adjusted by the time spent at the high gear (no energy savings), PARAID should have saved 17 - 22% at 32x, 14 - 18% at 64x, and 10 - 13% at 128x. Based on Table 7.2.1, PARAID gear switches, update propagations, and the additional parity computation incur about 4 – 10% of energy overhead, a future goal for optimization. Nevertheless, despite the heavy load of 270 – 1000 requests/second, PARAID can still conserve up to 13% of power.



(a) 128x speedup



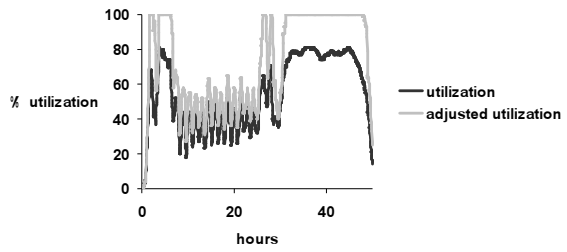
(b) 64x speedup



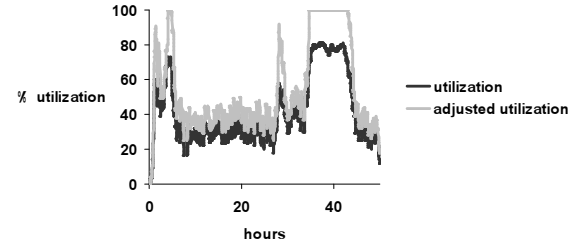
(c) 32x speedup

Figure 7.2.1: Power consumption for Cello99.

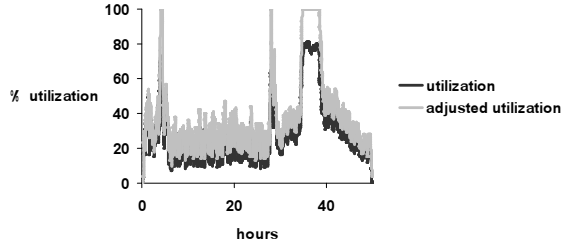
Figure 7.2.2 shows how gears are shifted based on the current gear utilization, on the percentage of busy gear seconds within a 32-second window, and adjusted utilization, as if the workload is using the low gear. PARAID consolidates the load spread among 5 disks to 3 disks, so that disks 4 and 5 can be spun off, while disks 1 to 3 can operate at 10 – 40% utilization. The graph also reconfirms the lack of opportunities to power-switch the entire RAID for power savings.



(a) 128x speedup



(b) 64x speedup



(c) 32x speedup

Figure 7.2.2: Gear utilization for Cello99 replay. Utilization measures the percentage of busy time of the current gear. Adjusted utilization measures the percentage of busy time of the low gear if the workload is applied to the low gear.

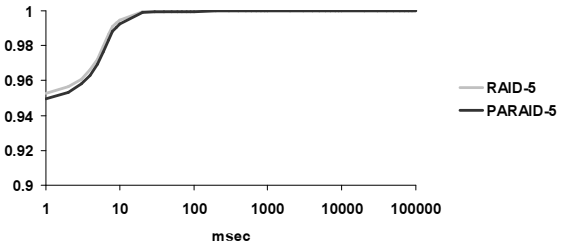
Speed-up	Power savings
128x (1024 req/sec)	3.5%
64x (548 req/sec)	8.2%
32x (274 req/sec)	13%

Table 7.2.1: Percent energy saved for Cello99.

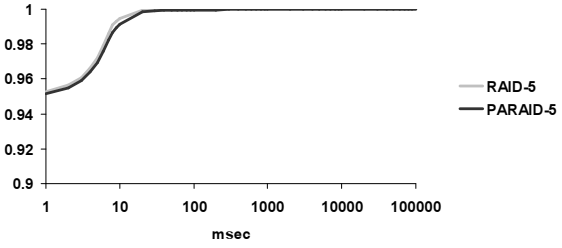
7.3 Performance

Completion time: Figure 7.3.1 shows the CDFs of completion time (from the time of PARAID forwarding a request to the moment the corresponding complete callback function is invoked). Latency is more difficult to measure since blocks are served out of order, and individual blocks from various disks need to be demuxed to the corresponding multi-block request to gather latency information. Therefore, completion time, which is also the worst-case bound for latency, is used.

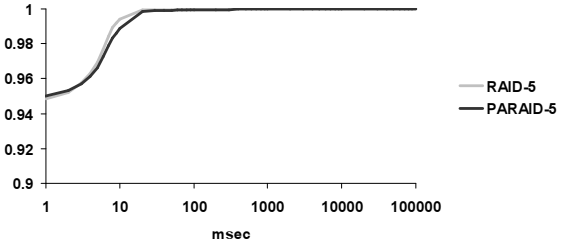
Unlike the latency CDFs from the web trace, the completion time CDFs of Cello99 showed very similar trends between PARAID and RAID-5, and Figure 7.3.1 presents only the high 90 percentile. At 32x, since PARAID spends more time at the lower gear, its latency is 26% slower than RAID-5 (1.8ms vs. 1.4ms).



(a) 128x speedup



(b) 64x speedup



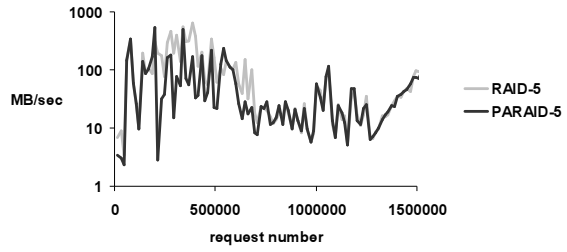
(c) 32x speedup

Figure 7.3.1: Completion time for Cello99.

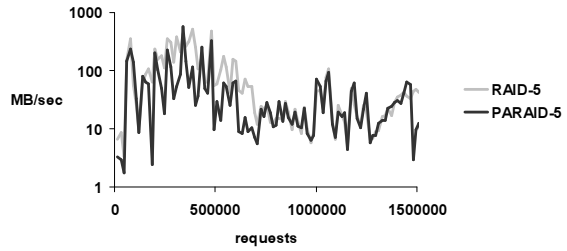
We examined the decompositions of I/Os. Although only 51% of bytes are accessed at high gear, they account for 97% of unique bytes. During the light-load periods, such as between the 6th and 27th hour, only 29Mbytes of unique data were referenced. Given that each powered disk can use 5Mbytes of on-disk cache, the bandwidth degradation of PARAID at low gear is significantly dampened by low-level caches. Therefore, the shape of the completion time CDFs is dominated by the high-gear operation, which uses the same RAID-5.

Figure 7.3.2 shows the bandwidth comparisons between PARAID and RAID-5. Note that these graphs are aligned by request numbers to emphasize that 60% of requests that occur during the peak load have the same bandwidth. Whenever PARAID is at high gear, the peak bandwidth is within 1% of RAID-5 (23 MB/sec). The low average bandwidth at high load periods reflects small average request sizes. During periods of light loads, the high bandwidth of both PARAID and RAID-5 reaffirms the enhanced role of low-level caches during light loads. Since PARAID did not use the SCSI controller, which contains additional cache,

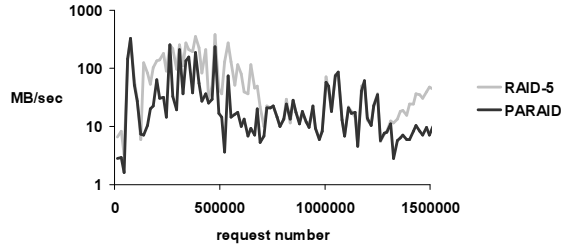
the bandwidth degradation of PARAID at low gear is likely to be further dampened. When PARAID operates at low replay speed and spends most of its time in the low gear, the average bandwidth degrades as expected (12 MB/sec vs. 21 MB/sec for RAID-5).



(a) 128x speedup



(b) 64x speedup



(c) 32x speedup

Figure 7.3.2: Bandwidth for Cello99.

Gear-switching statistics: Table 7.3.1 summarizes various PARAID gear-switching statistics for the Cello99 replay experiment. Again, PARAID spends more time in the low gear with reduced workload with decreasing playback speed. Due to heavy updates, each gear switch needs to incur an extra 1.3% to 3.9% of system I/Os. Fortunately, gear shifting occurs either before the system becomes highly loaded or is about to downshift due to the upcoming period of light loads. Therefore, these extra I/Os can be effectively absorbed by PARAID with spare I/O capacity, which may otherwise be left unused.

	128x	64x	32x
Number of gear switches	6.0	5.6	5.4
% time spent in low gear	47%	74%	88%
% extra I/Os for update propagations	8.0%	15%	21%

Table 7.3.1: PARAID gear-switching statistics for Cello99.

8 PostMark Benchmark

The PostMark synthetic benchmark generates ISP-style workloads that stress a storage device's peak performance for its read- and write-intensive activity [KATC97]. Running PostMark with PARAID starting at the lowest gear can be indicative of the overhead and latency of gear shifts during a request burst. The PostMark Benchmark was run directly on the server. PARAID propagated updates synchronously during gear shifts.

Figure 8.1 presents PostMark results comparing the elapsed times of RAID 5, PARAID starting with the highest gear, and PARAID starting with the lowest gear under three different benchmark configurations.

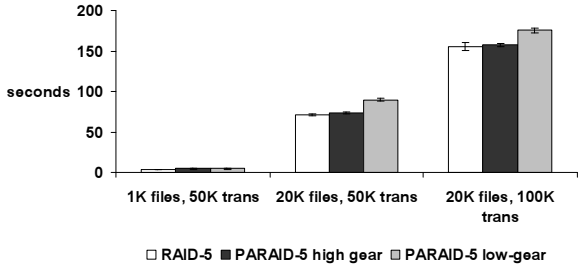


Figure 8.1: PostMark results for a RAID-5 device compared to a PARAID device starting in the highest gear and starting in the lowest gear.

For different PostMark configurations, PARAID starting with the highest gear demonstrates performance similar to RAID 5, which reflects the preserved layout of underlying RAID and minimal disturbances to the `md` data path. Figure 8.2 shows that as expected, PARAID does not save energy at the highest gear. PARAID energy savings is primarily from low gear settings.

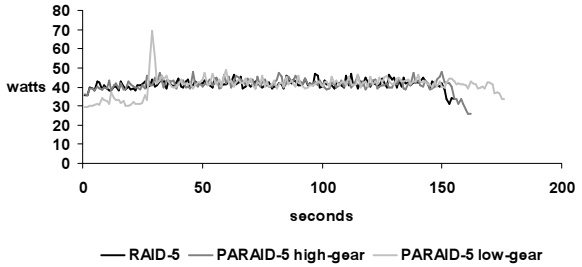


Figure 8.2: The PostMark power consumption results for a RAID-5 device compared to a PARAID device starting in the highest gear and starting in the lowest gear. The experiment contains 20K files and 100K transactions.

Figure 8.1 also compares the performance of RAID-5 with PARAID starting in the lowest gear. It demonstrates how the current up-shift policy prevents PARAID from being responsive to short bursts. The slowdown factor is about 13% due to up-shift overhead. The most responsive approach is to up-shift whenever a

burst is encountered. However, this would cause too many gear shifts throughout a day. Our observations suggest that daily cyclic workloads cause few gear shifts, so this overhead is unnoticeable. We plan to explore online algorithms to improve the responsiveness to burst loads while minimizing the number of gear shifts.

Table 8.1 demonstrates that PARAID in either configuration incurs similar CPU and system overhead when compared to RAID-5.

	Mean % CPU	Mean % System
RAID-5	3.24%	41.18%
PARAID high gear	3.11%	41.60%
PARAID low gear	3.08%	41.93%

Table 8.1: PostMark CPU and system overhead for RAID-5, PARAID starting in the highest gear and PARAID starting in the lowest gear. The experiment contains 20K files and 100K transactions.

9 Related Work

Most energy-reduction studies have addressed mobile computing [DOUG95, HELM96]. Recently, energy reduction for servers has also generated interest. Various approaches range from the hardware and RAID levels to the file system and server levels.

Reducing power consumption in hard disks: Carrera, et al. [CARR03] suggest using hypothetical two-speed disks. During periods of high load, the disk runs at maximum speed and power. During periods of light load, the disk spins at a lower speed and possibly idles. They report simulated disk energy savings between 15% to 22% for web servers and proxy servers, with throughput degradation of less than 5%.

FS2 [HUAN05] replicates blocks on a single disk to improve performance and reduce energy consumption via reducing seek time. FS2 reports up to 34% improvement in performance. The computed disk power consumption for per disk access also shows a 71% reduction. Since FS2 does not attempt to spin down disks, and since PARAID has spare storage for disks in high gears due to skewed striping (Figure 3.1.1), FS2 can be used on disks in high gears to extend PARAID's power savings.

Energy-efficient RAIDs: Hibernator [ZHU05] aims to reduce the energy consumption of RAIDs without degrading performance through the use of multi-speed disks. According to demand, data blocks are placed at different tiers of disks spinning at different speeds. A novel disk-block distribution scheme moves disk content among tiers to match disk speeds. When performance guarantees are violated, Hibernator spins disks at full speed to meet the demand. In simulation, Hibernator shows up to 65% energy savings.

Unlike Hibernator, PARAID is designed for existing server-class disks, and the minimum deployment granularity can be a small RAID on a typical server. Also, legacy systems can deploy PARAID via a software patch. As one consequence, some of the PARAID disks running at the lowest gear have few power-saving options. The future ubiquity of multi-speed disks will allow PARAID to explore further energy savings when running at the lowest gear.

MAID [COLA02] assumes that the majority of the data is being kept primarily for archival purposes, and its energy savings are based on the migration of this inactive majority to rarely used disks that fill a role similar to tape archives. PARAID, on the other hand, assumes that all data must be available at a high speed at all times. PARAID's techniques could be used on MAID's relatively few active disks to further improve the performance of that system.

Popular data concentration (PDC) [PINH04] saves energy by observing the relative popularity of data. PDC puts the most popular data on the first disk, the second most popular on the second disk, etc. Disks are powered off in PDC based on an idleness threshold. Without striping, PDC does not exploit disk parallelism.

In the absence of disk striping, the power-aware cache-management policy (PA-LRU) [ZHU04] saves power by caching data blocks from the less active disks. Lengthening the access interval for less active disks allows them to be powered off for longer durations. Partition-based cache-management policy (PB-LRU) [ZHU04B] divides the cache into separate partitions for each disk. Each partition is managed separately by a replacement algorithm such as LRU. PB-LRU provides energy savings of 16%, similar to that of PA-LRU.

EERAID [LI04] and its variant, RIMAC [YAO06], assume the use of a nonvolatile cache at the disk-controller level and the knowledge of cache content to conserve energy in RAIDs. Both lengthen disk idle periods by using nonvolatile disk controller cache to delay writes and computing parity or data-block content on the fly. Both spin down at most one disk for RAID-5, which limits their power savings.

[PINH06] generalizes RIMAC to erasure encoding schemes and demonstrates energy savings up to 61% in simulated tests. This approach defines and separates the primary data from the redundant data and stores them on separate disks. Then, the system makes redundant data unavailable at times to save energy. Writes are buffered via nonvolatile RAM.

Energy-aware storage systems: BlueFS [NIGH05], a distributed file system, uses a flexible cache hierarchy to decide when and where to access data based on the energy characteristics of each device. Through empirical measurements, BlueFS achieved a 55% reduction in file system energy usage. Adding PARAID to BlueFS can improve energy benefits.

The *Conquest-2* file system [XU03] uses nonvolatile RAM to store small files to save energy consumed by disks. PARAID can be readily used as a counterpart to serve large files while conserving energy.

Saving power in server clusters: Chase, et al. [CHAS01] and Pinheiro, et al. [PINH01] have

developed methods for energy-conscious server switching to improve the efficiency of server clusters at low request loads. They have reported energy reductions rangin from 29% to 43% for Webserver workloads.

PARAID can be combined with the server paradigm, so that over-provisioned servers used to cushion highly bursty loads or pre-powered to anticipate load increases can turn off many PARAID drives. Since powering on disks is much faster than booting servers, PARAID incurs less latency to respond to traffic bursts.

When traffic loads involve a mixture of reads and writes, disk switching in PARAID provides localized data movement and reduces stress on the network infrastructure. Also, PARAID can be deployed on individual machines without distributed coordination.

Other alternatives: Instead of implementing PARAID, one might use HP AutoRAID's ability to reconfigure to emulate PARAID's behavior [WILK95]. However, one fundamental difference is that reconfiguring a RAID with D disks to D - 1 disks under AutoRAID requires restriping all content stored on D disks, while PARAID can restripe the content from a partial stripe, in this case 1 disk.

10 Ongoing Work

PARAID is still a work in progress. First, although PARAID exploits cyclic fluctuations of workload to conserve energy, our experience with workloads suggests that it is difficult to predict the level of benefit based on the traffic volume, the number of requests, the number of unique bytes, the peak-to-trough traffic ratios, and the percentage of reads and writes. We are interested in measuring PARAID with diverse workloads to develop further understandings of PARAID's behavior. Also, we plan to test PARAID with other types of workloads, such as on-line transaction processing [UMAS06].

Currently, PARAID is not optimized. The selection of the number of gears, the number of disks in each gear, and gear-shifting policies are somewhat arbitrary. Since empirical measurement is unsuitable for exploring a large parameter space, we are constructing a PARAID-validated simulation for this purpose, which will allow more exploration of parameters. At the same time, we are investigating analytical approaches to develop online algorithms with provable optimality.

We will modify our disk synchronization scheme to explore asynchronous update propagation, allowing newly powered-on drives to serve requests immediately. We plan to implement selective recovery schemes for intermediary gears to speed up cascaded recovery (Currently, PARAID-5, as used in this paper, recovers 2.7x slower than RAID-5.), and also to incorporate the S.M.A.R.T tools [TOOL05] for disk health monitoring, allowing more informed decisions on rationing power cycles, and rotation of the gear-membership of disks. Finally, we plan to mirror a PARAID server to FSU's department server for live testing, deploy PARAID in a real-world environment, and compare PARAID with other energy-saving alternatives.

11 Lessons Learned

The idea of PARAID was born as a simple concept to mimic the analogy of gear-shifting, which conserves fuel in vehicles. However, turning this concept into a kernel component for practical deployment has been much more difficult than we anticipated.

First, our early design and prototype of PARAID involved cloning and modifying RAID-0. As a result, we had to bear the burden of inventing replication-based reliability mechanisms to match different RAID levels. Our second-generation design can reuse the RAID encoding scheme, making the evolution of new RAID levels independent of PARAID. Although the resulting energy savings and performance characteristics were comparable in both implementations, PARAID's structural complexity, development time, and deployment potential improved in the new design.

Second, measuring energy consumption is difficult because of data-alignment problems and a lack of integrated tools. With continuous logging, aligning data sets is largely manual. For multi-threaded experiments and physical disks, the alignment of data sets near the end of the experiment is significantly poorer than it was at the beginning. Early results obtained from averages were not explicable, since unaligned square waves can be averaged into non-square shapes.

Third, measuring systems under normal loads is harder than under peak loads. Replaying traces as quickly as possible was not an option, and we had to explore different speedup factors to see how PARAID reacts to loads changes. Since server loads have constant streams of requests, we cannot simply skip idle periods [PEEK05], because such opportunities are relatively infrequent. Worse, consolidated workloads are carried by fewer powered-on components with less parallelism, further lengthening the measurement time.

Fourth, modern systems are complex. As modern hardware and operating systems use more complex optimizations, our perception of system behaviors increasingly deviates from their actual behaviors. Memory caching can reduce disk activity, while file systems can increase the burstiness of RAID traffic arrivals due to delayed write-back policies. Disks are powered with spikes of current, making it difficult to compute power consumption as the areas under the spike. Disk drives can still consume a significant amount of power even when they are spun down.

Fifth, matching the trace environment to our benchmarking environment is difficult. If we use a memory size larger than that of the trace machine, we may encounter very light disk activity. The opposite can saturate the disks and achieve no power savings.

Cyclic workload patterns before the cache may poorly reflect the patterns after the cache. Additionally, traces might not have been made using RAIDs, some traces might be old, and the RAID geometry might not match our hardware settings. The base system might have multiple CPUs, which makes it difficult to judge whether a single modern CPU is powerful enough. Although the research community is well aware of these problems, the solutions still seem to be achieved largely by trial and error.

12 Conclusion

PARAID is a storage system designed to save energy for large computing installations that currently rely upon RAID systems to provide fast, reliable data storage. PARAID reuses standard RAID-levels without special hardware, while decreasing their energy use by 34%. Since PARAID is not currently optimized, and since we measured only 5 drives (among which at least 2 are always powered), we believe that an optimized version of PARAID with many disks could achieve significantly more energy savings.

A second important conclusion arises from the research described in this paper. Actual implementation and measurement of energy-savings systems is vital, since many complex factors such as caching policies, memory pressure, buffered writes, file-system-specific disk layouts, disk arm scheduling, and many physical characteristics of disk drives are difficult to capture fully and validate simultaneously using only simulation. Also, implementations need to address compatibility with legacy systems, the use of commodity hardware, and empirical evaluation techniques, all of which are necessary for practical deployments.

Unfortunately, our research also shows that there are considerable challenges to performing such experiments. We overcame several unforeseen difficulties in obtaining our test results, and had to invent techniques to do so. This experience suggests the value of developing standard methods of measuring the energy consumption of computer systems and their components under various conditions. We believe this is another fruitful area for study.

Acknowledgements

We thank our shepherd Keith Smith and the anonymous reviewers for their invaluable insights. Additionally, we thank Jason Flinn, Daniel Peek, Margo Seltzer, Daniel Ellard, Ningning Zhu, HP, and Sun Microsystems for providing accesses to various tools and traces. Cory Fox, Noriel Lu, Sean Toh, Daniel Beech, and Carl Owenby also have contributed to PARAID. This work is sponsored by NSF CNS-0410896. Opinions, findings, and conclusions or recommendations expressed in this document do not necessarily reflect the views of the NSF, FSU, UCLA, Harvey Mudd College, or the U.S. Government.

References

[ABDE05] M. Abd-El-Malek, W.V. Courtright II, C. Cranor, G.R. Ganger, J. Hendricks, A.J. Klosterman, M. Mesnier, M. Prasad, B. Salmon, R. R. Sambasivan, S. Sinnamohideen, J.D. Strunk, E. Thereska, M. Wachs, J.J. Wylie, URSA Minor: Versatile Cluster-based Storage. *Proceedings of the 4th USENIX Conference on File and Storage Technology*, 2005.

[ASAR05] T. Asaro. An Introduction to Thin Provisioning. *Storage Technology News*, 2005. http://searchstorage.techtarget.com/columnItem/0,294698,sid5_gci1134713,00.html.

[CARR03] E. Carrera, E. Pinheiro, R. Bianchini, Conserving Disk Energy in Network Servers, *Proceedings of the 17th Annucal ACM International Conference on Super Computers*, 2003.

[CHAS01] J. Chase, D. Anderson, P. Thakar, A. Vahdat, R. Doyal, Managing Energy and Server Resources in Hosting Centers, *Proceedings of the 18th ACM Symposium on Operating System Principles*, 2001.

[COLA02] D. Colarelli, D. Grunwald, Massive Arrays of Idle Disks For Storage Archives, *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, 2002.

[DOUG95] F. Douglis, P. Krishnan, B. Bershad Adaptive Disk Spin-down Policies for Mobile Computers *Proceedings of the 2nd USENIX Symposium on Mobile and Location-Independent Computing, 1995.*

[FUJI06] Fujitsu, MAP 10K RPM, 2006. http://www.fujitsu.com/us/services/computing/storage/hdd/discontinued/map-10k-rpm.html.

[GRAY05] J. Gray, Keynote Address Greetings from a Filesystem User, *the 4th USENIX Conference on File and Storage Technologies,* 2005.

[GURU03] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, H. Franke, DRPM: Dynamic Speed Control for Power Management in Server Class Disks, *Proceedings of the International Symposium on Computer Architecture*, 2003.

[HELM96] D.P. Helmbold, D.D.E. Long, B. Sherrod, A dynamic disk spin-down technique for mobile computing, *Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking (MobiCon'06), 1996.*

[HERB06] G. Herbst. Hitachi's Drive Temperature Indicator Processor (Drive-TIP) Helps Ensure High Drive Reliability, http://www.hitachigst.com/hdd/technolo/drivetemp/drivetemp.htm, 2006.

[HP06] HP Labs, Tools and Traces, 2006. http://www.hpl.hp.com/research/ssp/software/

[HUAN03] H. Huang, P. Pillai, K.G. Shin, Design and Implementation of Power Aware Virtual Memory, *Proceedings of the 2003 USENIX Annual Technical Conference*, 2003.

[HUAN05] H. Huang, W. Hung, K.G. Shin: FS2: Dynamic Data Replication in Free Disk Space for Im-

proving Disk Performance and Energy Consumption. *Proceedings of the 20th Symposium on Operating Systems Principles*, 2005.

[IBM06] IBM, IBM Hard Disk Drive—Ultrastar 36Z15. http://www.hitachigst.com/hdd/ultra/ul36z15.htm.

[IYEN00] A. Iyengar, J. Challenger, D. Dias, P. Dantzig, High-performance Web Site Design Techniques, *IEEE Internet Computing*, 4(2):17–26, March 2000.

[KATC97] J. Katcher, PostMark: A New File System Benchmark, Technical Report TR3022, Network Appliance Inc., 1997

[KUEN97] G.H. Kuenning, G.J. Popek. Automated Hoarding for Mobile Computers. *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, 1997.

[LEVI06] M. Levin. Storage Management Disciplines are Declining. *Computer Economics*. 2006. http://www.computereconomics.com/article.cfm?id=1129.

[LI04] D. Li, P. Gu, H. Cai, J. Wang. EERAID: Energy Efficient Redundant and Inexpensive Disk Array. *Proceedings of the 11th ACM SIGOPS European Workshop*, 2004.

[MANL98] S. Manley, M. Seltzer, M. Courage, A Self-Scaling and Self-Configuring Benchmark for Web Servers**,** *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, 1998.

[MILL93] E. Miller, R. Katz, An analysis of file migration in a Unix supercomputing environments, *Proceedings of the 1993 USENIX Winter Technical Conference*, 1993.

[MOOR05] J. Moore, J. Chase, P. Ranganathan, R. Sharma, Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers, *Proceedings of the 2005 USENIX Annual Technical Conference*, 2005.

[NIGH05] E.B. Nightingale, J. Flinn, Energy-Efficiency and Storage Flexibility in the Blue File System, *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, 2005.

[PATT88] D.A. Patterson, G. Gibson, RH Katz, A case for redundant arrays of inexpensive disks (RAID). *ACM SIGMOD International Conference on Management of Data*, 1(3):109-116, June 1988.

[PEEK05] D. Peek, J. Flinn, Drive-Thru: Fast, Accurate Evaluation of Storage Power Management, *Proceedings of the 2005 USENIX Annual Technical Conference*, 2005.

[PINH01] E. Pinheiro, R. Bianchini, E. V. Carrera, T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. *Proceedings of the Workshop on Compilers and Operating Systems for Low Power*, 2001.

[PINH04] E. Pinheiro, R. Bianchini, Energy Conservation Techniques for Disk Array-Based Servers, *Proceedings of the 18th Annual ACM International Conference on Supercomputing*, 2004.

[PINH06] E. Pinheiro, R. Bianchini, C. Dubnicki. Exploiting Redundancy to Conserve Energy in Storage Systems. *Proceedings of Sigmetrics/Performance*, 2006.

[RFC01] RFC-3174 - US Secure Hash Algorithm 1, 2001. http://www.faqs.org/rfcs/rfc3174.html

[TOOL05] SANTools, Inc. 2005. http://www.santools.com/smartmon.html

[SANT99] D.S. Santry, M.J. Feeley, N.C. Hutchinson, A.C. Veitch, R.W. Carton, J. Ofir, Deciding when to forget in the Elephant File System, *Proceedings of the 17th ACM Symposium on Operating Systems Principles*, 1999.

[UMAS06] UMass Trace Repository, Storage Traces, 2006. http://signl.cs.umass.edu/repository/walk.php?cat=Storage.

[WILK95] J. Wilkes, R. Golding, C. Staelin, T. Sullivan. The HP AutoRAID Hierarchical Storage System. *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, 1995.

[XU03] R. Xu, A. Wang, G. Kuenning, P. Reiher, G. Popek, Conquest: Combining Battery-Backed RAM and Threshold-Based Storage Scheme to Conserve Power, *Work in Progress Report, 19th Symposium on Operating Systems Principles (SOSP)*, 2003.

[YAO06] X. Yao, J. Wang. RIMAC: A Novel Redundancy-based Hierarhical Cache Architecture for Energy Efficient, High Performance Storage Systems. *Proceedings of the EuroSys*, 2006.

[YU00] X. Yu, B. Gum, Y. Chen, R. Wang, K. Li, A. Krishnamurthy, T. Anderson, Trading Capacity for Performance in a Disk Array, *Proceedings of the 4th Symposium on Operating Systems Design and Implementation*, 2000.

[ZHU04] Q. Zhu, F.M. David, C. Devaraj, Z. Li, Y. Zhou, P. Cao, Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management, *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, 2004.

[ZHU04B] Q. Zhu, A. Shanker, Y. Zhou, PB-LRU: A Self-Tuning Power Aware Storage Cache Replacement Algorithm for Conserving Disk Energy, *Proceedings of the 18th Annual ACM International Conference on Supercomputing*, 2004.

[ZHU05] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, J. Wilkes, Hibernator: Helping Disk Arrays Sleep through the Winter, *Proceedings of the 20th ACM Symposium on Operating Systems Principles*, 2005.