

# Reliability-Aware Energy Management for Hybrid Storage Systems

Wes Felter, Anthony Hylick, and John Carter

IBM Research - Austin  
 {wmf, hylick, retrac}@us.ibm.com

## Abstract

*Modern disk-based storage systems are not energy proportional, because disks consume almost as much power when idle (but spinning) as they do when actively accessing data. We combine a power-aware, solid-state (flash) cache and a reliability-aware disk spin-down mechanism to significantly improve storage energy proportionality without hurting disk reliability, data integrity, or performance.*

*We evaluated the resulting power- and reliability-aware hybrid flash-disk RAID storage array and found that it reduces energy consumption by 85% compared to a similar-cost, similar-performance typical configuration of all SAS drives that are never spun down. Our design also achieves almost 50% energy savings compared to hybrid flash-disk systems tuned for performance or that do not take full advantage of opportunities for safe spin-down. Further, unlike most previous work that exploits spindown to save energy, we limit the rate at which disks are spun down to avoid premature mechanical failures, whereas reliability-unaware spindown algorithms can exceed manufacturer warranted lifetime spindown limits in as little as one year.*

## I. INTRODUCTION

The performance, cost, and reliability of modern computer systems and data centers increasingly are dictated by the management of limited energy budgets. An emerging design goal is for computing devices to be made *energy proportional* [1], the notion that all components within a computing system should be capable of proportional change in power consumption as offered load changes. Modern storage systems are not energy proportional [10]. Most data is stored on disks, which consume almost as much power when idle (but spinning) as they do when being actively accessed. As a result, storage energy scales in proportion to system capacity, not system utilization (load). Storage currently represents 20%-40% of total energy consumption in a typical data center [3], and its relative importance is growing because

the amount of data being stored is growing at a rate faster than Moore's Law and other components (especially processors and the cooling infrastructure) are becoming increasingly energy-efficient.

In recognition of the growing storage energy problem, disk manufacturers have introduced new "green" disks with multiple idle states and autonomous firmware that exploits these lower power modes when the disk is sufficiently idle. For example, the Seagate Constellation ES "green" drives support four idle modes [24] detailed in Table I. Each deeper idle mode consumes less power, but introduces a larger delay to transition back to Active mode, which can impact performance. Table II presents the power in each idle state, the amount of idle time before the disk autonomously enters each idle state, and the amount of time required to transition back to Active.

Many researchers have proposed using solid-state drives (SSDs) as a cache in front of large disk-based storage arrays. SSDs offer greater read/write performance than mechanical disk drives, and are quite energy-efficient given their low idle power needs, but it is typically not cost-effective to replace all capacity with SSDs due to their high per-byte cost compared to disk [18]. Flash caching is typically proposed to improve performance or to reduce system cost by enabling expensive enterprise (15K RPM) disks to be replaced with inexpensive commodity (7200 RPM or slower) disks [8], [22], [26], [27], [30]. Previous researchers have reported power benefits from flash caching that arise because less energy is required to access a block of data from an SSD than from a disk [14], [22], [27]. Further, a large flash cache may increase how often the backing disks are idle, which provides more opportunities for autonomous disk power management mechanisms to save energy. However, as seen in Table II, disks consume almost as much power when idle (but spinning) as they do when actively servicing a read or write request, so simply replacing disk seeks with SSD accesses, without disk spindown, provides only marginal energy benefits, a phenomenon that we quantify in Section V.

To achieve energy proportionality, some researchers have proposed spinning down idle disks [2], [7], [15],

State	Characteristics
Active	Disk spinning at full RPM; electronics active.
Idle A	Read channel and servo control turned off.
Idle B	Idle A plus heads unloaded
Idle C	Idle B plus speed reduced to lower RPM
Standby	Spindle motor turned off (disk spun down)

TABLE I  
DISK POWER STATES (SEAGATE CONSTELLATION)

State	Power (Watts)	Timeout (mins)
Active (Read, Write)	7.95	—
Seeking	7.22	—
Active Idle	5.86	—
Idle A	5.86	2
Idle B	4.51*	4
Idle C	3.55	10
Standby	0.31	15

Transition	Power (Watts)	Transition Time (sec)
Idle A → Active	5.86	0
Idle B → Active	5.86	0.1
Idle C → Active	13.77	0.38
Standby → Active	13.77	6.22

TABLE II  
MODELED DISK CHARACTERISTICS (\* - INTERPOLATED STATE FOR WESTERN DIGITAL DRIVE, SEE SECTION 4.2)

[17], [20], [28], [29], [31]. However, disk spindown has not been widely employed in enterprise storage systems for a variety of reasons. Most important, the physical wear induced by head load/unload cycles reduces the lifetime of disks [24]. Disk manufacturers rate their drives for a maximum number of spindown cycles before the warranty is void. Enterprise disks are designed to spin down only about 25,000 times, while (lower performance, more rugged) laptop disks are rated for roughly 500,000 load-unload cycles. Assuming a three to five year disk lifetime [21], this translates to a maximum spindown rate of only one to six spindowns *per hour*. For example, Seagate Constellation ES “green” disks spin down autonomously only after the disk has been idle for 15 minutes. With the exception of PARAID [28], prior work in this area has ignored the serious reliability problems that can arise from spinning disks up and down too frequently. Also, it takes 6-10 seconds to spin up a disk after the drive motor has been turned off, which impacts performance and can cause brittle software to fail. For these reasons, disk spindown has not been adopted by major storage vendors.

In this paper, we present a reliability-aware hybrid storage system that combines power-aware flash caching and disk spindown. We show that power-aware flash caching enables significantly more opportunities for disk

spindown than is possible without flash caching or with power-oblivious caching (e.g., Sun’s ZFS [26] and NetApp’s WAFL [30]). We introduce a token bucket mechanism to limit the rate at which disks are spun down to ensure that we do not wear out disks prematurely, in contrast with reliability-oblivious spindown mechanisms that can cause disks to exceed their spindown limit in as little as a year. We quantify the benefits of combining both disk power management (spin-down) and flash caching and show how the combined benefit of both approaches is greater than the sum of the benefits from each technique individually. We simulate a power- and reliability-aware hybrid flash-disk RAID storage array and find that it reduces energy consumption by 85% compared to a similar-cost, similar-performance baseline RAID system that uses enterprise SAS drives with no spindown. Our design also reduces energy 45%-47% compared to hybrid flash-disk systems tuned for performance or that do not fully exploit opportunities for safe spindown. Finally, we show that these power savings results are within 5% of a reliability-aware oracle spindown policy. In summary, our work combines the benefits of flash-based caching and disk spindown in a way that *improves storage energy efficiency without impacting disk reliability, data integrity, or performance*.

The rest of this paper is organized as follows: Section II details previous research on disk power management and flash caching. Section III describes the design of our power- and reliability-aware hybrid storage solution. Section IV describes the workloads that we use in our analysis, simulation environment, and experiments. Section V presents our results. Finally, in Section VI we present our conclusions and discuss future work.

## II. RELATED WORK

There has been substantial prior research on using flash-based solid state drives as caches in a storage system to improve performance, some of which also considers the potential energy benefits of flash caches. A second body of work has explored the use of spinning down disks to reduce storage energy consumption. A small number of researchers have explored the combination of these techniques. Our work extends prior research in this area by considering more than two disk power states and by managing disk and SSD reliability. Our work also provides more detailed analysis of the synergy between caching and spindown, illustrates the importance of exploiting all available spindown opportunities, and presents a breakdown of where time and energy are spent in a large variety of storage configurations and energy management policies.

**Improved disks:** Gurumurthi proposes improving the energy efficiency of hard disks by introducing the ability to trade off power and performance by dynamically varying the spindle speed and intra-disk parallelism [11], [23]. These precise mechanisms have not been adopted by disk vendors, but our analysis does consider disks that support variable disk speeds and multiple power states, such as the Seagate Constellation ES.

**Spin-down:** In the early stages of mobile computing, Wilkes outlined an algorithm for predictive power conservation for disks to extend the utility of mobile devices [29]. Due to the cost of spinning up a drive, he proposed an adaptive delay timeout that was a function of previous activity rather than a fixed delay timeout set either by the user or manufacturer. Douglass et al. were some of the first researchers to define the ‘break-even’ time for drive spin-down, the time required to be spun down to save the energy required for the subsequent spin-up [7]. Similar to Wilkes, their work proposes an adaptive spin-down timeout.

More recently, Microsoft researchers observed that disks can be spun down even when they are not completely idle as long as there are no reads, such as occurs when a server is largely idle but continues to write infrequent log messages [17]. In this scenario, disks can be spun down if subsequent (log) writes are “off-loaded” to some other storage device (disk or SSD) that is in an active state; we feel that write caching within a storage system can provide the same benefits but is simpler to implement. Colerelli and Grunwald developed massive arrays of idle disks (MAID) that explicitly manage a large disk array such that a large fraction of the disks are spun-down at any instant in time [6]. By carefully spreading or replicating data across drives in the array, MAID can retain traditional RAID performance at a lower average power. Similar to Microsoft’s write off-loading technique, Pinheiro et al. propose an access mechanism that diverts read and write requests to a redundant copy so that the disk(s) storing the primary copies can remain in lower power states [20]. Pinheiro and Bianchini introduce the notion of popular data concentration (PDC) wherein data that is classified as ‘popular’ is collected onto a subset of storage devices so that the remaining storage may be spun down [19]. Weddle et al. propose PAROID [28], which varies the RAID stripe width at runtime, replicating popular blocks so that some disks can be spun down during periods of low load.

**Flash caching:** In SieveStore, Pritchett and Thottethodi use flash-based SSDs to filter accesses to disks [22]. Their work is based upon the observation that a small

fraction of blocks are responsible for a significant number of total accesses, so a small amount of flash can absorb a significant fraction of I/O requests. Kgil and Mudge present FlashCache, a flash-based cache behind the traditional DRAM buffer cache that allows the DRAM cache size (and thus power requirement) to be reduced, while maintaining high performance by avoiding disk accesses [14]. In EXCES, Useche et al. extend previous work investigating external caching by adding prefetching and buffering of data [27]. Finally, Sun [26], NetApp [30], and EMC [8] have announced commercial storage systems that use flash caching.

**Spin-down + Flash caching:** Lee et al. propose augmenting RAID configurations with a flash-based read cache and write buffer to conserve disk power [15]. They further observed that working set sizes often are small enough to be managed effectively with the flash cache, which allows the disks to be placed in lower power states. Bisson et al. explore the impact of using a non-volatile cache (NVCache) to create more opportunities for disk spin-down by increasing disk idle periods [2]. Their results show significant savings using this approach. Zhu et al. detail several power-aware cache management algorithms that reduce energy consumption and increase I/O performance [31].

**Energy-Aware Caching:** Chen and Zhang demonstrate the need for (DRAM) caches to be energy-aware to produce significant energy-saving disk spindown opportunities [4]. They detail two energy-aware caching policies that increase the burstiness of disk accesses while imposing only a modest performance loss.

**Disk and SSD reliability:** Although there is published work on hard disk reliability, few studies consider the reliability impact of spindown at all. Greenawalt [9] examines the tradeoff between disk spindown timeout and disk lifetime. PAROID [28] considers disk reliability in addition to energy, spreading the wear caused by spindown evenly across disks in the RAID array and conservatively limiting the rate of spindowns to prevent premature disk wear out. We are not aware of any detailed characterization of the effect of spindown on disk reliability; this is an area that deserves more study. We acknowledge a large body of work on wear-leveling for flash; we do not propose any changes or improvements to the state of the art in this area. While most flash reliability management is performed inside SSDs, Soundararajan et al. [25] propose the somewhat counterintuitive idea of reducing flash wear using a hard disk as a write cache; this is a nearly opposite approach to our work.

Additional relevant differences between our work and prior art are discussed throughout the remainder of this paper.

### III. LOW-POWER HYBRID STORAGE

#### A. Reliability-aware Disk Energy Mgmt

Reliability concerns may make it impractical to put the results of prior disk spindown research into practice. For example, Seagate specifies that an enterprise disk may be spun down no more than once every 15 minutes [24], which is much longer than the timeout calculated by a purely energy-optimizing spindown algorithm. All of the power management algorithms that we consider in this paper limit the number of per-disk lifetime spindowns to no more than the manufacturer specified limit, which, for the disks that we consider, translates to no more than one spindown per 15 minutes on average.

As described in Section I, Seagate has developed and is promulgating a standard for energy-efficient “green” disks that can autonomously manage their own idle states to save energy during periods of idle load [24]. Whenever a disk has been idle for a sufficient period of time, these disks autonomously transition to an appropriate idle state. The spindown characteristics of the disks that we model are presented in Table II. The modeled drives represent a combination of Seagate Constellation ES drives (with the most aggressive idle states) and similar Western Digital “green” drives, which are present in our storage server and thus can be used to validate our simulated results.

When enabled, the autonomous power management mode of Seagate Constellation ES drives works as follows. If no requests are received for 2 minutes, the disk enters the Idle A state and disables both the read channel and the servo control, which reduces power consumption marginally. After 4 minutes (total), the disk transitions to the Idle B state, where the heads are unloaded but the disk continues to spin at full speed, which reduces power consumption from 5.9 W to 4.5 W. After 10 minutes (total), the disk enters the Idle C state and roughly halves its rotational speed, which reduces power consumption to 3.5 W. Finally, after 15 minutes, the disk enters the Standby state and turns off the spindle motor, which reduces power to only 0.3 W. Note that spinning up from an idle state requires varying amounts of energy and time; the deeper idle states consume substantially less energy when idle, but require substantially more energy and time to return to the Active state.

For our analysis, we consider six disk energy management policies:

- 1) All-SAS with no spindown (SAS)
- 2) All-SATA with autonomous spindown (SATA)
- 3) Power-aware write-back flash caching (WC)
- 4) Token bucket spindown (TB)
- 5) Unsafe Oracle (UO)
- 6) Safe Oracle (SO)

We describe each of these policies in turn below.

*All-SAS with no spindown (SAS):* Enterprise storage servers traditionally use high-RPM Serial Attached SCSI (SAS) or Fibre Channel (FC) drives due to their high performance and high reliability. These drives are much more expensive (in \$/GB) and less dense than low-RPM commodity disks. Somewhat counterintuitively, SAS disks generally have higher a mean time between failure (MTBF) than commodity drives, but can tolerate fewer spin-ups over their lifetime; they are very reliable when spinning but fragile when power-managed. Note that although low-RPM SAS and high-RPM SATA disks exist, they are not common. Thus, we follow the industry convention of using the term “SAS” to refer to high-RPM disks and “SATA” to refer to low-RPM disks. For the SAS policy, we model a single RAID array consisting of 3.5” 450GB 15K RPM SAS disks (such as the Seagate Cheetah 15K.6) that is never spun down. Throughout our analysis, this policy represents our baseline configuration and energy management policy because it represents the most common enterprise configuration.

*All-SATA with autonomous spindown (SATA):* For our second policy, we replace the enterprise SAS drives with the same number of “green” SATA drives and enable their autonomous power management (spindown) mode. Note that this configuration has substantially more capacity and lower cost than the SAS configuration, but lower performance, and thus is not directly comparable to the all-SAS configuration. Rather, we present the results of this policy to show how much power can be saved by replacing SAS drives with more power-friendly SATA, and to contrast how much additional power savings can be gained by introducing a flash cache.

*Power-aware write-back flash caching (WC):* Several previous researchers have proposed augmenting a traditional flash (SSD) cache with policies that increase the idleness of disks, thereby increasing opportunities to place disks in deeper idle states [2], [15], [28], [31]. It is not uncommon for even very idle systems to periodically update filesystem metadata. If these writes occur less than 15 minutes apart and are written through to disk, then the disks we use *would never be able to spin down* regardless of the cache’s read hit rate.

To improve the power savings of a hybrid flash-disk storage system, we introduce three new power-aware features to the flash caching policy:

- 1) **Write-back caching:** Inspired by PA-LRU [31], we use a write-back policy. Doing so eliminates spinups caused by periodic logging events or other occasional writes. This policy is akin to write offloading [17], except that the writes are written to the cache layer rather than a spun-up disk. Note that since power management continues to be handled autonomously by the disks, we employ a write back policy at all times, not only when we would otherwise write-through to a spun down disk. If we employed a write-through policy for active disks, this would continuously reset each disk’s internal idle timer, thereby causing the disk to not spin down.

To ensure that our write-back caching mechanism does not impact data integrity, we make the flash cache itself redundant. In our analysis, we model the flash cache as a pair of mirrored (RAID-1) SSDs, although in a large storage server with more disks, we would employ a more cost- and power-efficient RAID-5-like scheme.

- 2) **Batched flushes:** To reduce the likelihood that we will need to spin up a disk when we need to evict a dirty block, we proactively write back dirty data whenever a disk is spun up, even if the data is relatively fresh. To minimize performance impact, these flushes are performed in the background when the disk is otherwise idle.
- 3) **Reliability-aware spindown:** Under the WC policy, spindown continues to be handled autonomously by the disk firmware according to the schedule presented in Table II to avoid premature disk failure.

*Flash caching with token bucket spindown (TB):*

Spinning disk drives up and down causes them to fail prematurely due to mechanical wear [21]. To address this problem, we investigated a variety of spindown algorithms that balance the need to maximize opportunities for spindown with the need to avoid premature disk failures. To balance these competing goals, we employ a *token bucket* spindown algorithm that ensures that no useful opportunity to spin down a disk is wasted, but ensures that the number of disk spindowns over its lifetime does not exceed the manufacturer’s warranted limit. Token buckets are a concept first employed by the networking community to perform “traffic shaping” by limiting the rate of admission of packets to a potentially congested shared link [5]. The basic idea of a token bucket mechanism is that tokens are added to a virtual

bucket (represented as an integer) at a fixed rate that matches the average rate at which some activity should occur, e.g., a packet being transmitted in the case of the networking example or a disk spindown in our TB mechanism. Whenever the system wishes to perform the rate-limited activity, it must first remove a token from the bucket or stall until a token is added. This allows the system to have bursts of activity while limiting the average rate to some fixed long-term average.

In our TB policy, the token bucket algorithm takes into consideration the maximum number of spinup/down cycles that the manufacturer specifies a disk can perform over its lifetime. We calculate how often to add a token to the bucket by dividing the planned lifetime of the drive by the total number of cycles for which the drive is rated. For the disks that we model, this translates to adding a token every 15 minutes. Before a disk can be spun down, a token must first be removed from the bucket. If no token is present, the disk is kept spinning until the next token is added, assuming there is no intervening access to the disk that resets its idle timer. The bucket is represented by an integer, and adding/subtracting a token corresponds to incrementing/decrementing the count. The key insight is that the token bucket policy accumulates spindown “credits” whenever there are extended active or idle periods, which is common for many storage environments (e.g., long idle periods at night and over weekends, and long active periods during the day). If a particular disk remains spun up or spun down for thirty minutes, two tokens will accumulate in the (virtual) bucket, which provides credit sufficient for two future spindowns, in addition to the tokens that are added every 15 minutes. Note that the resulting TB policy includes the same flash cache as the WC policy, but replaces the autonomous disk-based spindown policy with one under the control of the storage controller.

Our actual design is somewhat more sophisticated than the basic token bucket mechanism described above. First, as described in Section I, modern disks have more than one idle state, each of which has a unique entry/exit cost and power savings. To exploit all available idle states, we model four separate token buckets, one per idle state (Idle A, Idle B, Idle C, and Standby). In our implementation, the rate at which we add tokens to each bucket is set to be the idle interval required by the reference Seagate drive to autonomously enter that state (i.e., 2 minutes to enter Idle A, etc.). Before we transition to a given idle state, we must first remove a token from the corresponding bucket.

Second, recall that substantial energy is required to transition from an idle state back to the Active state. For example, on our testbed system we measured the

energy consumed to spin a disk up from the Standby state and found that it would need to be spun down for over 15 seconds before a spindown/spinup cycle would have a breakeven energy cost. Any spindown period that lasts under 15 seconds consumes more energy than simply leaving the disk spinning. The other disk idle states have similar, albeit shorter, breakeven times. Our token bucket spindown algorithm uses breakeven data that we collected experimentally to implement a competitive spindown algorithm [16]. Whenever a disk has been idle for twice the breakeven interval for a given idle state, the spindown algorithm will attempt to spin down the disk, subject to the availability of a token in the corresponding bucket. Thus, instead of waiting for 15 minutes of idleness before spinning a disk down, as is done in the preceding solutions, the token bucket spindown algorithm can spin a disk down as early as 30 seconds after it becomes idle if a token is available.

Note that the token bucket algorithm is used only to determine whether a disk can be spun *down*. No token is needed to spin a disk *up*, so disk operations that miss in the flash cache are never delayed due to the token bucket algorithm. Premature spin up may, however, result in the subsequent spin down being delayed until another token has arrived.

*Unsafe oracle (UO)*: To enable us to reason about how close to optimal any of the above policies are, we also simulated a system that combines flash caching with a reliability-unaware (unsafe) oracle spindown policy (UO). For the UO policy, whenever the idle interval between two disk accesses exceeds the breakeven time for a low-power state, the disk enters that state as soon as it becomes idle. This policy represents the lowest possible energy consumption, but may cause premature disk failure due to excessive spindowns.

*Safe oracle (SO)*: The safe oracle (SO) policy is similar to UO, except that it respects the disk manufacturer’s lifetime spindown limits. For our study, this translates into only entering the standby state during the 672 longest idle intervals in the week-long trace. This policy thus achieves the lowest possible energy consumption that does not exceed the warranted spindown rate.

## IV. EVALUATION ENVIRONMENT

### A. Workloads

Until recently, storage research has focused on performance and reliability and has used workloads designed to stress performance. Such workloads usually have little idle time, defeating most attempts at power management. In the real world, storage is used for a variety of

workloads with a wide range of I/O intensity; since heavy workloads such as OLTP databases offer little opportunity for energy savings and archival workloads are well-served by MAID, we turn our attention to medium-duty workloads such as email and file serving. To do so, we use the five most active traces from a set of week-long server block traces published by Microsoft Research [17], [18]: `proj_1`, `proj_2`, `prxy_1`, `usr_1`, and `src1_1`.

### B. Simulated System

We model a small SAN storage system that is comprised of a storage controller, a single RAID-6 array consisting of eight 750 GB 3.5” SATA disks, and (if SSDs are present in the configuration) 2 small SSDs used as a controller-managed cache. A real storage system would have dozens to hundreds of disks; our work should scale linearly with the number of disks. We accurately model the power consumed by the storage elements (disks and SSDs), but do not model the power consumed by the storage controller or network elements, which should be roughly equal for all configurations. In realistic configurations, the power consumed by the storage controller is dwarfed by the disk power. All requests to the modeled storage system are assumed to have come from a separate client, and we do not model any DRAM cache other than that implicitly present in the client. This configuration matches the setup used to collect the Microsoft Research traces described above [17], [18]. We anticipate that the benefit of our energy management policies would be greater in a larger storage system, where a single flash cache could support multiple RAID arrays, each of which could be separately power managed.

For the disks, we model a hypothetical disk with the power and performance characteristics of a Western Digital RE2-GP and the improved power management states found in a Seagate Constellation ES (see Table II). Table 2 presents the relevant characteristics of the modeled disk: (i) the power consumed in each idle state, (ii) the amount of idle time necessary for the autonomous power management mechanisms in the disk to transition to a given idle state, and (iii) the *recovery time* for each state, which is how long the disk takes to transition from the given state back to the active state. We captured the power and recovery time characteristics for the disks from an experimental storage server prototype populated with Western Digital RE2-GP drives. The spin-up power measurements represent the power consumed while the disk is spinning up from the given state to the active state. All power measurements were performed using a highly accurate Yokogawa WT210 power meter. In all

cases, the power measurements are AC power, so they include power supply conversion losses.

For SSDs, we model a SandForce SF-1200. They are 100-GB devices divided into 4-kilobyte blocks. Our cache layer implements an LRU replacement policy, although, as described in Section III, we defer writing dirty blocks back to disk until a disk spins up or is accessed due to a read miss. This ensures that block flushes do not reset idle timers. All cache metadata is stored in the storage controller’s DRAM memory so that the SSD only holds cached data.

### C. Simulator

We developed a simulator that extends the *TRADE* estimator, described in previous work [13], to model the individual disk energy consumption of all the disks in a RAID array. The *TRADE* simulator is a validated disk drive energy estimator based upon accurate accounting of a drive’s internal state. It provides accurate energy accounting without the need for detailed power measurements or drive fingerprinting. The disk drive power models used in the simulator are generated from readily-available disk drive information such as that in Table 2. We extended the *TRADE* estimator to model RAID-6 functionality to support the work described here. Our simulator (*TRAIDE*) maintains an accurate count of the amount of time spent in each power state by each drive in the RAID-6 array, along with each drive’s on-disk cache state and disk head placement.

## V. RESULTS

### A. Performance

The typical response time of our hybrid storage system is under 20 ms, but reading data from a disk that is spun down takes several seconds because the disk must spin up. Some applications, such as mission-critical online transaction processing systems, cannot tolerate such high latency (>1 sec) accesses no matter how rare, so disk spindown should not be employed for storage that these applications access. We do not consider such workloads in this paper. Laptop users commonly tolerate disk spinup latency, so we believe that interactive user-facing workloads such as email and file servers can safely enable aggressive disk power management, including spindown.

The Microsoft Research traces contain the response time for each I/O request. Our simulator uses this information, along with disk and SSD characterization data, to estimate the performance of a hybrid storage system.

Workload	Cache Hit Ratio (%)
proj_1	39
proj_2	52
prxy_1	65
usr_1	67
src1_1	85

TABLE III  
FLASH CACHE READ HIT RATE

We assume that a cache read hit takes 200 us, a cache write takes 400 us, and any access to a spinning disk has the response time recorded in the trace. If an I/O request causes a disk to spin up, the simulator adds the spin-up time to the recorded response time. We are primarily concerned with read response time because most writes have no effect on application performance.

Figure 1 shows the read response time CDF for the SAS (baseline) and TB policies on two representative traces (proj\_2 and src1\_1). The vertical line represents cache read hits - in an actual system the response time would vary somewhat due to queuing that our simulator does not model. A small number of I/Os that miss in the flash cache are faster than a cache hit due to hits in the RAID stripe cache or on-disk DRAM cache. The response time of the hybrid storage system is 5x-10x faster than the baseline disk-only system at virtually every percentile. This performance gain is strong justification for hybrid flash-disk storage designs, which are further justified by the large energy savings illustrated below.

These week-long traces contain tens of millions of I/O requests, yet a disk can only spin up from standby at most 672 times. Thus the number of I/Os delayed by these spin-ups are so few that they are not visible in Figure 1. Even if we make the pessimistic assumption that spinups all occur during an 8-hour business day, disks spend at most 2% of their time spinning up.

The performance gain from flash caching varies significantly based on the workload; e.g., the prxy\_1 trace is comprised mostly of small random reads and has a working set size of roughly 60 GB, which fits entirely within our 100 GB cache, so flash caching reduces its median response time by roughly a factor of 20. The proj\_2, usr\_1, and src1\_1 traces have moderate cache hit rates, allowing a configuration of 8 5400 RPM disks augmented with a flash cache to equal the performance of 16 15K RPM disks. The proj\_1 workload has little reuse, so in this case the flash cache does not contribute enough performance to offset slower disks.

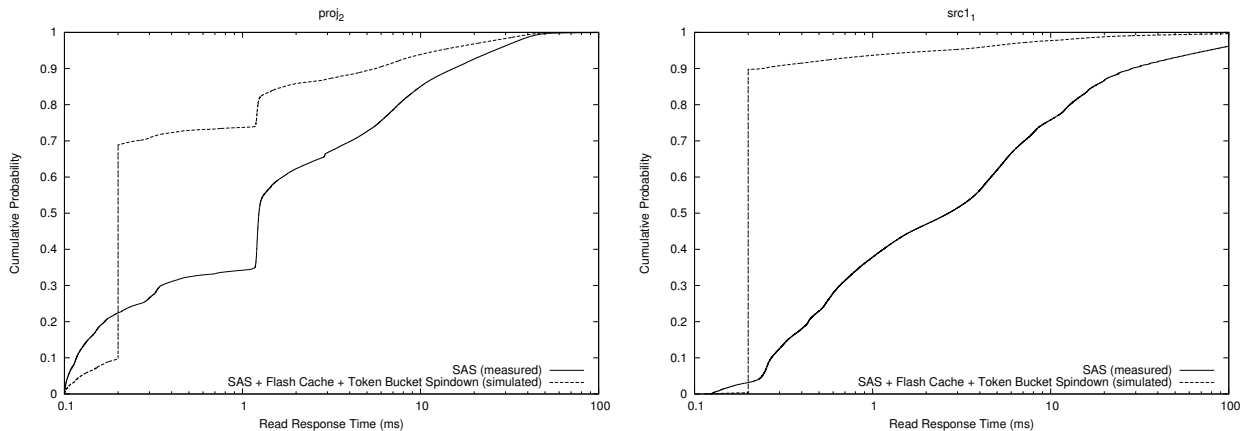


Fig. 1. Performance

We ran a representative set of these experiments on a testbed storage system using the Linux open-source dm-cache [12] module to implement the flash cache. We could only run a small set of configurations and workloads on the testbed due to the time necessary, one week per configuration per workload. The experimental results tightly matched our simulation results, thereby validating our simulation infrastructure.

In hybrid storage systems with a high ratio of disks to SSDs, performance and power can sometimes be at odds. For random I/O, a single SSD can usually outperform dozens of disks while using less power, making flash caching and disk spindown highly desirable, but for sequential I/O it is common for the disks to deliver higher throughput than flash. Thus, servicing sequential I/O from the flash cache tends to reduce power but also reduce performance. We propose that hybrid storage systems should allow the user to choose between power-optimized and performance-optimized modes; in power-optimized mode all I/O would be cached, while in performance-optimized mode sequential I/O would bypass the cache, possibly forcing disks to spin up more frequently.

### B. Exploitation of Lower Power States

Table IV shows the number of times the drives enter each of the power states during the week for the SATA (no flash cache, autonomous spindown) configuration. Without having a cache to insulate the drives from the I/O requests, drives have very few opportunities to enter a lower power state. The trace with the most opportunities to spin down the drives, proj\_2, was only able to do so 60 times over the course of the week. For two of the traces, prxy\_1 and usr\_1, the autonomous disk

power management policies were *never* able to invoke even the most conservative energy saving modes at any time during the week.

	Idle A	Idle B	Idle C	Standby
proj_1	348	97	29	26
proj_2	169	125	26	60
prxy_1	0	0	0	0
usr_1	27	0	0	0
src1_1	385	164	6	10

TABLE IV  
ENTRIES TO EACH DISK STATE (SATA)

In contrast, Table V shows the number of times each power state was entered for the TB (flash cache w/ token bucket spindown) policy for each trace. The table presents the lowest values of all eight disks in the array. For all five of the traces, the TB policy was able to spin down the disks considerably more often than a policy that uses no cache. Two workloads, prxy\_1 and usr\_1, consumed essentially all of the spindown opportunities (672 per week), while power savings on the other three workloads was not token-limited.

	Idle A	Idle B	Idle C	Standby
proj_1	5039	2519	1007	561
proj_2	5034	2042	507	221
prxy_1	5024	2407	1004	669
usr_1	5039	2519	1007	671
src1_1	5039	658	683	535

TABLE V  
ENTRIES TO EACH STATE (TB POLICY)

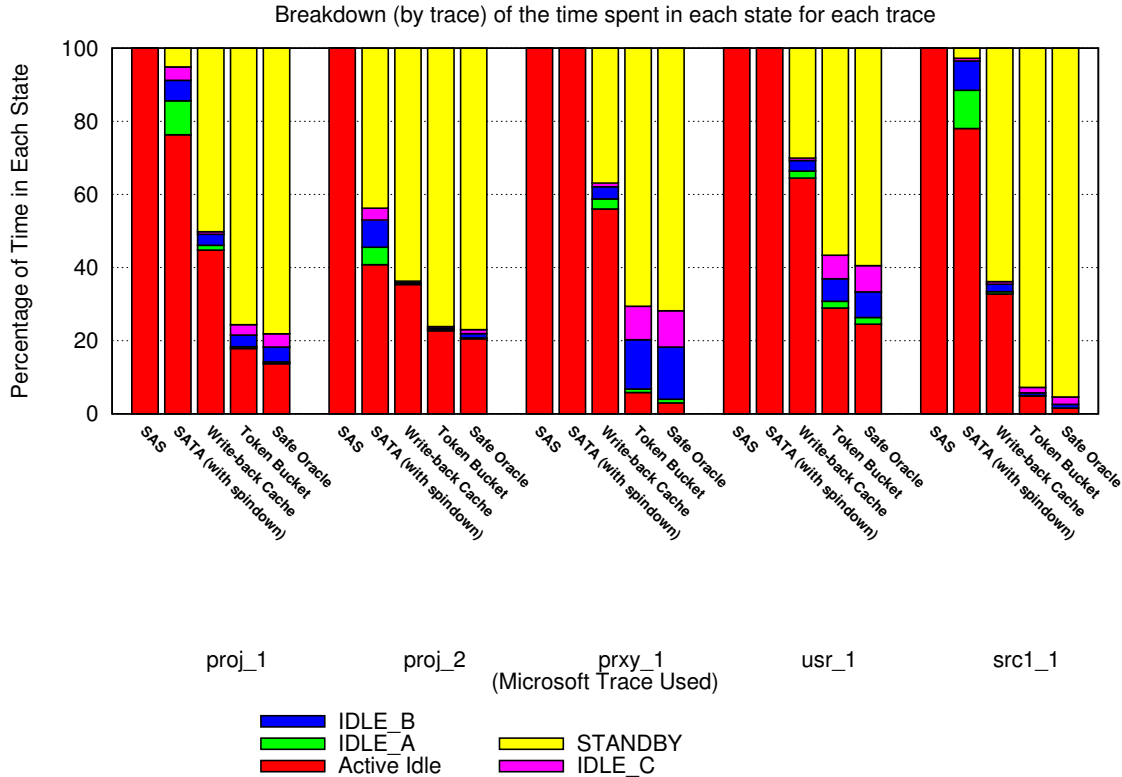


Fig. 2. Percentage of Time spent in each state for all traces and all policies

Figure 2 illustrates what fraction of time each policy spends in each of the five disk power states for each of the five workloads.

By design, SAS spends all of its time in the Active state. The SATA policy, which does not have a flash cache, is only able to spend an average of 13% of its time spindown (Standby); it spends an average of 80% of the time the Active Idle mode (spinning, but not accessing data) despite the presence of autonomous power management. The WC policy is able to spend substantially more time in the Standby state than SATA, but only the TB policy is able to achieve spindown performance close to the Safe Oracle (SO) policy. These results strongly support our theses that to achieve near-optimal power savings, flash caching alone is insufficient and that it is essential not to waste any spindown opportunities.

### C. Energy Consumption

Figure 3 presents the total energy consumption of each policy for each trace, as well as a breakdown of the energy across the different power states. Figure 4 presents the same data with the SAS and SATA results removed so that the remaining results can be seen more clearly.

These workloads have fairly light and bursty I/O demands, which is evident in the small amount of energy consumed in the ‘Active’ state, which corresponds to performing actual read or write operations including the energy required to hold the head in place. Nearly all energy is consumed in states where the drive is not serving I/O requests. Because the Standby state is dramatically lower power than the other states, it contributes little to the total energy consumption even when it dominates the run time. In fact, the percentage of time spent in Standby is a good proxy for the overall energy saved.

The high power consumption of SAS drives and the inability to spin down results in extremely high energy consumption for SAS compared to the other policies. This configuration is not competitive from an energy efficiency standpoint, but may be required for critical enterprise environments that cannot tolerate even rare spinup latencies, e.g., mission-critical OLTP workloads.

Simply replacing SAS drives with lower power SATA drives and enabling autonomous power savings results in a greater than 50% reduction in energy consumption compared to SAS, reducing weeklong energy consumption from over 60 MJ to under 30 MJ. However, almost

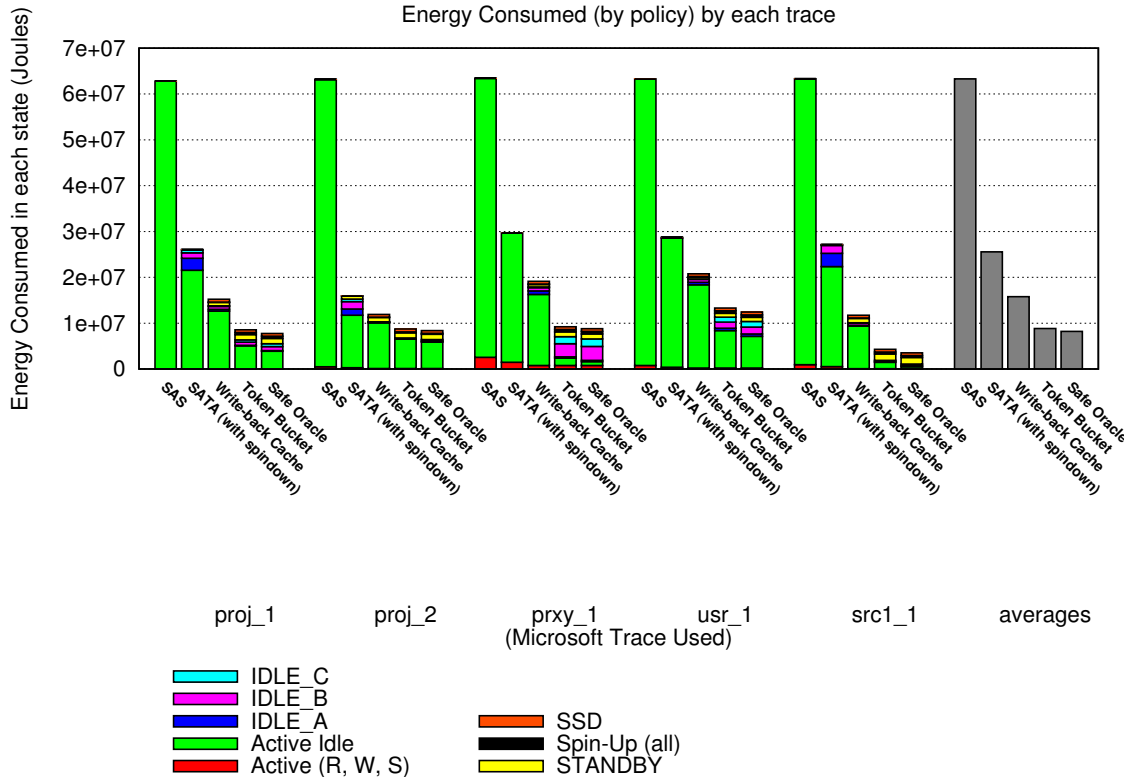


Fig. 3. Energy Consumption for all traces and all policies

of the energy savings come from the inherently lower Active power consumption of these lower RPM SATA drives, not from spindown, and the SATA configuration is lower performance than SAS. We simulated a configuration using SATA disks with power management disabled but we omit this data from the graphs to save space; the energy consumption is virtually the same for all workloads: 29 MJ. For prxy\_1 and usr\_1 in particular, the energy consumption for an all-SATA configuration is the same whether power management is enabled or not.

By delaying and combining writes, the power-aware flash caching (WC) policy is able to exploit the lower disk power states far more effectively than the no-cache SATA configuration. For example, prxy\_1 and usr\_1 contain frequent background writes which keep the disks active in the SATA configuration, but not WC. Although the cost and complexity of a write cache is higher than a read cache due to the need to replicate dirty data to ensure its integrity, these results show that this added complexity pays significant dividends in terms of additional energy savings. For all of the workloads, even those for which flash caching was only marginally successful, the energy savings from flash caching more than offset the energy consumed by the SSDs that were added to the system to implement the caching. The net

result is that the WC policy results in additional average energy savings of roughly 40% compared to the SATA policy.

By better exploiting opportunities to enter deeper idle states, the TB policy is able to spin down much more frequently and thus save substantially more energy than even a less tuned power-aware flash caching policy (WC). On average, TB reduces energy consumption 85% compared to a design based on all SAS drives, 65% compared to a design employing “green” SATA drives, and 45% compared to a traditional power-aware flash caching policy. Note that the energy consumed by spinups is negligible for all policies, including the TB policy that spins disks up and down far more frequently.

Finally, the TB policy performs within 10% of the safe oracle (SO) policy for all of the traces, which indicates there is little benefit to exploring more complex policies.

#### D. Disk and SSD Reliability

In addition to saving energy, any storage power management policy that is likely to be employed in a commercial system must ensure that it does not reduce the useful lifetime of the disks and SSDs. For disks, this

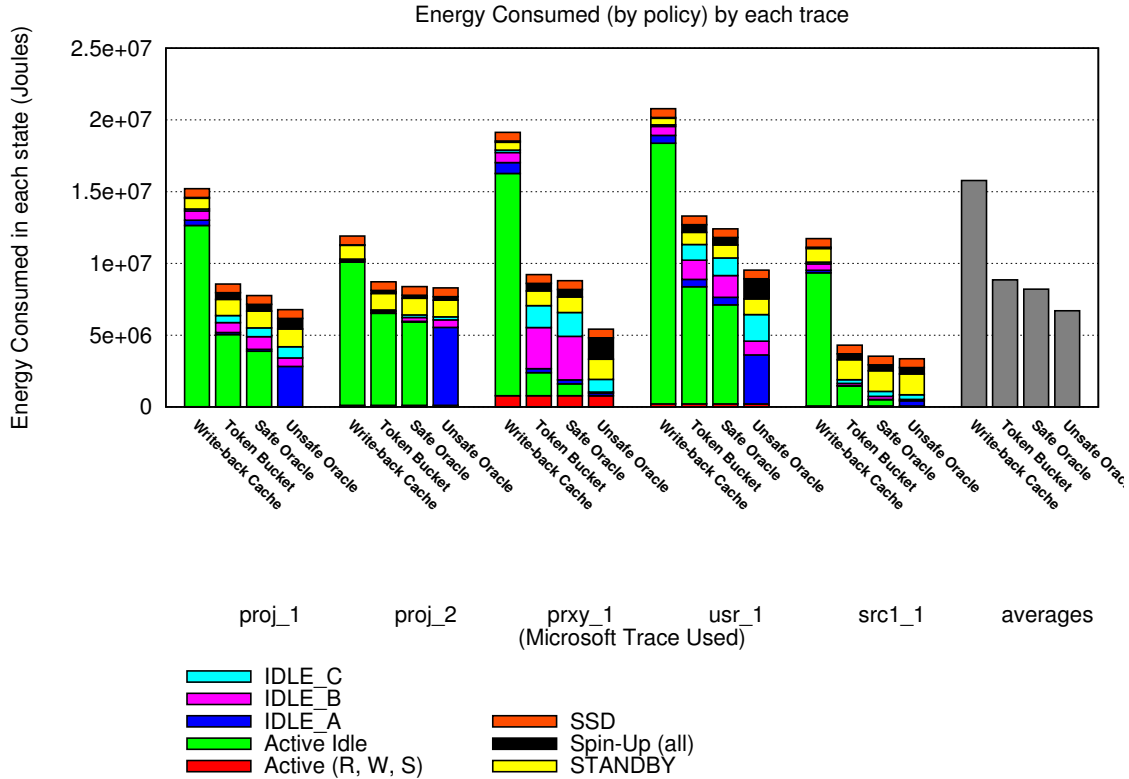


Fig. 4. Zoomed in Energy Consumption Comparison

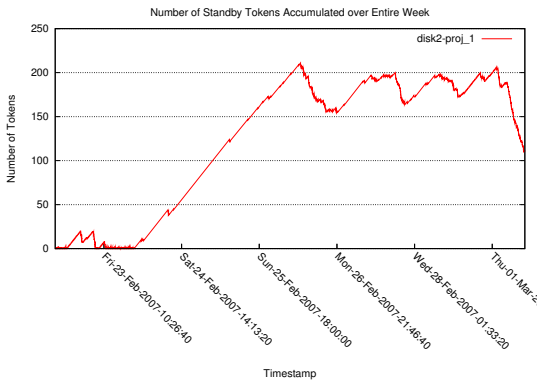


Fig. 5. Tokens Accumulated over Time (proj\_1)

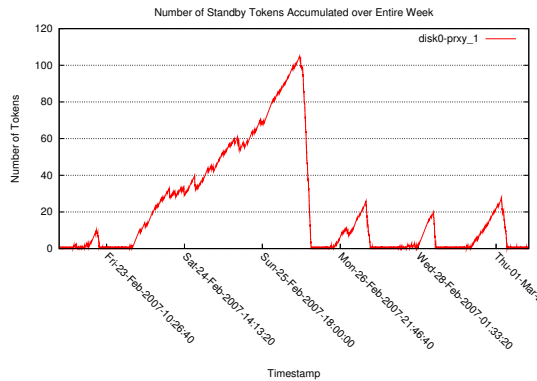


Fig. 6. Tokens Accumulated over Time (prxy\_1)

guarantee is inherent in the design of our TB policy. Figure 5 and Figure 6 illustrate the accumulation of tokens over the week for two traces, proj\_1 and prxy\_1. The proj\_1 trace illustrates a scenario where tokens are generated more quickly than they are consumed, due to prolonged active or idle periods. Workloads like proj\_1 do not induce enough spindowns to impact disk lifetimes. In contrast, the prxy\_1 trace illustrates a workload for which a rate-limiting mechanism is needed to ensure reliability. Tokens are consumed (on average) as fast

as they are generated. Even for this workload, there are long periods of idleness, e.g., nights and weekends, where tokens accumulate (i.e., the graph slopes up), interspersed with opportunities for spindown (where it slopes down).

To determine whether our caching policy would impact the lifetime of the SSDs used in the caching layer, we measured the number of writes per SSD block over the weeklong trace. A 100 GB SSD like the SandForce SF-1200 used in our environment actually has 128 GB

of capacity; only 100 GB is usable because some flash is reserved for wear-leveling, replacement of bad flash blocks, and other FTL overhead. Table VI shows the estimated lifetime of the SSDs when used in a cache configuration for each of the workload traces. These results

	Lifetime (years)
proj_1	474
proj_2	164
prxy_1	31
usr_1	236
src1_1	677

TABLE VI  
SSD LIFETIME (YEARS)

assume perfect wear-leveling, no write amplification, and a flash endurance of 3,000 program/erase cycles, which is typical for an SSD based on commodity 34nm MLC NAND flash. While actual SSD lifetime will be lower due to the extra writes induced by write amplification, our results show that even for the most write-intensive workload (prxy\_1), the SSD lifetime far exceeds the predicted lifetime of a storage array.

To determine whether a rate-limiting mechanism like our token bucket is necessary to ensure disk reliability, we estimated the lifetime of the drives used in the optimal reliability-oblivious (UO) policy. In this analysis, we assume that the drives are rated for 175,000 spinups (every 15 minutes over a 5-year lifetime). As shown in Table VII, ignoring spindown frequency creates a serious risk of premature disk failure for several of these workloads, perhaps as low as one year for prxy\_1.

	Lifetime (years)
proj_1	4
proj_2	14
prxy_1	1
usr_1	2
src1_1	5

TABLE VII  
ESTIMATED HDD LIFETIME WITH UO POLICY

Finally, the difference between the UO and SO policies represents the additional power savings that would be possible if we were able to spin down more often. As seen in Figure 4, for the workloads and drives that we modeled, we could reduce energy consumption by an additional 18%, from an average of 8.2 MJ to 6.7 MJ per week, with more robust disks. Given the highly competitive nature of the disk drive industry, this may provide an opportunity for product differentiation, but

the tradeoff between increased upfront (disk) cost and decreased lifetime operational (energy) cost would need to be studied more closely.

## VI. CONCLUSIONS

Through simulation, we have demonstrated the combined potential of disk power management and flash caching to greatly increase the energy proportionality of storage systems. Our work extends prior research in this area by considering more than two disk power states and by explicitly managing disk reliability using a token bucket algorithm. In addition, we quantitatively show the energy-saving benefits of a flash cache on medium-duty, real-world workloads without negatively impacting request response time.

One of the major contributions of this work is to detail the synergy between flash-caching and disk spindown. On a collection of week-long Microsoft Research traces, an all-SAS configuration consumes 63.3 MJ, while an all-SATA configuration with autonomous spindown consumes 25.6 MJ. Almost all of the energy savings from using SATA drives comes from the inherently lower Active energy of these drives, not spindown. Thus, spindown alone does not suffice to achieve optimal energy savings. Adding a basic power-aware flash cache (WC) saves an additional 38% compared to the baseline SATA configuration, but this design is still far from optimal (SO). Only with the combination of power-aware flash caching, a mechanism that maximizes safe spindown opportunities, and an aggressive spindown mechanism (TB) can you achieve near optimal results, 85% less than all SAS. The TB policy performed within 10% of the safe oracle (SO) policy for all of the traces, which indicates there is little benefit to exploring more complex policies. These results strongly support our theses that flash caching alone is insufficient to achieve near-optimal power savings and that it is essential not to waste any spindown opportunities.

## REFERENCES

- [1] Luiz André Barrosa and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, December 2007.
- [2] Timothy Bisson, Scott A. Brandt, and Darrell D.E. Long. Nvcache: increasing the effectiveness of disk spin-down algorithms with caching. pages 422–432. In Proceedings of the 14th IEEE Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006.
- [3] John Carter and Karthick Rajamani. Designing energy-efficient servers and data centers. *IEEE Computer*, 43(7):76–78, July 2010.
- [4] Feng Chen and Xiaodong Zhang. Caching for bursts (C-Burst): let hard disks sleep well and work energetically. pages 141–146. In Proceedings of the 13th International Symposium on Low Power Electronics and Design, 2008.

- [5] David D. Clark, Scott Shenker, and Lixia Zhang. Supporting real-time applications in an Integrated Services Packet Network: architecture and mechanism. pages 14–26. In Proceedings of the Conference on Communications Architectures and Protocols, August 1993.
- [6] Dennis Colarelli and Dirk Grunwald. Massive arrays of idle disks for storage archives. pages 1–11. In Proceedings of the 2002 ACM/IEEE International Conference on Supercomputing, 2002.
- [7] Fred Douglass, P. Krishnan, and Brian Bershad. Adaptive disk spin-down policies for mobile computers. pages 121–137. In Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing, 1995.
- [8] EMC Corporation. New levels of emc midrange storage efficiency and simplicity accelerate journey to the private cloud, 2010. <http://www.emc.com/about/news/press/2010/20100511-02.htm>.
- [9] Paul Greenawald. Modeling power management for hard disks. pages 62–66. In Proceedings of the 1994 IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 1994.
- [10] Jorge Guerra, Wendy Belluomini, Joseph Glider, Karan Gupta, and Himabindu Pucha. Energy proportionality for storage: Impact and feasibility. HotStorage 2009, 2009.
- [11] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mahmut Kandemir, and Hubertus Franke. DRPM: dynamic speed control for power management in server class disks. In Proceedings of the 30th International Symposium on Computer Architecture (ISCA), 2003.
- [12] Eric Van Hensbergen and Ming Zhao. Dynamic policy disk caching for storage networking, 2006.
- [13] Anthony Hylick and Ripduman Sohan. A methodology for generating disk drive energy models using performance data. In Proceedings of the 2009 Workshop on Power Aware Computing and Systems (HotPower), 2009.
- [14] Taeho Kgil and Trevor Mudge. Flashcache: a NAND flash memory file cache for low power web servers. pages 103–112. In Proceedings of the 2006 Conference on Compilers, Architecture, and Synthesis for Embedded Systems, 2006.
- [15] Hyo J. Lee, Kyu H. Lee, and Sam H. Noh. Augmenting RAID with an SSD for energy relief. In Proceedings of the 2008 Workshop on Power Aware Computing and Systems, 2008.
- [16] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for on-line problems. pages 322–333. In Proceedings of the 20th ACM Symposium on the Theory of Computing (STOC), 1988.
- [17] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write off-loading: practical power management for enterprise storage. In Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST), 2008.
- [18] Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, and Antony Rowstron. Migrating server storage to SSDs: analysis of tradeoffs. In Proceedings of the 4th ACM European Conference on Computer Systems (Eurosys), 2009.
- [19] Eduardo Pinheiro and Ricardo Bianchini. Energy conservation techniques for disk array-based servers. pages 68–78. In Proceedings of the 18th International Conference on Supercomputing, 2004.
- [20] Eduardo Pinheiro, Ricardo Bianchini, and Cezary Dubnicki. Exploiting redundancy to conserve energy in storage systems. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):15–26, June 2006.
- [21] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barrosa. Failure trends in a large disk drive population. In Proceedings of the 5th USENIX Conference on File and Storage Technologies, 2007.
- [22] Timothy Pritchett and Mithuna Thottethodi. Sievestore: a highly-selective, ensemble-level disk cache for cost-performance. pages 163–174. In Proceedings of the 37th International Symposium on Computer Architecture, 2010.
- [23] Sriram Sankar, Sudhanva Gurumurthi, and Mircea R. Stan. Intra-disk parallelism: an idea whose time has come. pages 303–314. In Proceedings of the 35th International Symposium on Computer Architecture (ISCA), 2008.
- [24] Seagate Corporation. Seagate powerchoice technology provides unprecedented hard drive power savings and flexibility, 2010. [http://www.seagate.com/docs/pdf/whitepaper/tp608\\_powerchoice\\_tech\\_provides.pdf](http://www.seagate.com/docs/pdf/whitepaper/tp608_powerchoice_tech_provides.pdf).
- [25] Gokul Soundararajan, Vijayan Prabhakaran, Mahesh Balakrishnan, and Ted Wobber. Extending ssd lifetimes with disk-based write caches. FAST 2010, 2010.
- [26] Sun Microsystems. Solaris zfs enables hybrid storage pools - shatters economic and performance barriers, 2009. [http://www.sun.com/x64/intel/zfs\\_solution\\_brief.pdf](http://www.sun.com/x64/intel/zfs_solution_brief.pdf).
- [27] Luis Useche, Jorge Guerra, Medha Bhadkamkar, Mauricio Alarcon, and Raju Rangaswami. Exces: EXternal Caching in Energy Saving storage systems. In Proceedings of the 14th International Symposium on High-Performance Computer Architecture, 2008.
- [28] Charles Weddle, Mathew Oldham, Jin Qian, An-I Andy Wang, Peter Reiher, and Geoff Kuenning. PARAD: a gear-shifting power-aware raid. *ACM Transactions on Storage*, 3(3):33, October 2007.
- [29] John Wilkes. Predictive power conservation. Technical Report HPL-CSP-92-5, Hewlett-Packard Laboratories, 1992. page 1, February 1992.
- [30] Mark Woods. Optimizing storage performance and cost with intelligent caching, 2010. <http://www.netapp.com/us/library/whitepapers/wp-7107.html>.
- [31] Qingbo Zhu, Francis M. David, Christo F. Devaraj, Zhenmin Li, Yuanyuan Zhou, and Pei Cao. Reducing energy consumption of disk storage using power-aware cache management. pages 118–129. In Proceedings of the 10th International Symposium on High Performance Computer Architecture (HPCA), 2004.