Power-Reduction Techniques for Data-Center Storage Systems

TOM BOSTOEN and SAPE MULLENDER, Alcatel-Lucent Bell Labs
YOLANDE BERBERS, Katholieke Universiteit Leuven

As data-intensive, network-based applications proliferate, the power consumed by the data-center storage subsystem surges. This survey summarizes, organizes, and integrates a decade of research on power-aware enterprise storage systems. All of the existing power-reduction techniques are classified according to the disk-power factor and storage-stack layer addressed. A majority of power-reduction techniques is based on dynamic power management. We also consider alternative methods that reduce disk access time, conserve space, or exploit energy-efficient storage hardware. For every energy-conservation technique, the fundamental trade-offs between power, capacity, performance, and dependability are uncovered. With this survey, we intend to stimulate integration of different power-reduction techniques in new energy-efficient file and storage systems.

Categories and Subject Descriptors: C.4 [**Computer Systems Organization**]: Performance of Systems; D.4.2 [**Operating Systems**]: Storage Management; D.4.3 [**Operating Systems**]: File Systems Management; E.2 [**Data**]: Data Storage Representations; E.5 [**Data**]: Files; H.3.2 [**Information Storage and Retrieval**]: Information Storage—*File Organization*

General Terms: Design, Algorithms, Performance, Reliability

Additional Key Words and Phrases: Cloud storage, data center, disk drive, energy efficiency, power reduction

ACM Reference Format:
Bostoen, T., Mullender, S., and Berbers, Y. 2013. Power-reduction techniques for data-center storage systems. ACM Comput. Surv. 45, 3, Article 33 (June 2013), 38 pages.
DOI: http://dx.doi.org/10.1145/2480741.2480750

1. INTRODUCTION

With the advent of data-intensive, network-based applications and services, data centers around the world consume a significant and rapidly growing amount of electricity. In the U.S. alone, all data centers together are projected to consume 100 TWh per year by 2011, which costs more than $10billion at a common price of $100 per MWh [Kaushik et al. 2010]. Since 2005, data-center power consumption in the U.S. has more than doubled, from 40 TWh [Zhu et al. 2005]. The energy consumed by data centers represents 1–2% of the total U.S. power consumption [Sehgal et al. 2010]. The cost of energy is a growing concern for data-center operators, as it may constitute half of the data center's total cost of ownership [Joukov and Sipek 2008]. Data storage alone is responsible for about 25 to 35% of data-center power consumption [Kim and Rotem

This work is supported by the Flanders Agency for Innovation by Science and Technology (IWT), grant IWT 100690.
Authors' addresses: T. Bostoen and S. Mullender, Alcatel-Lucent Bell Labs, Copernicuslaan 50, B-2018 Antwerp, Belgium; email: {Tom.Bostoen, Sape.Mullender}@alcatel-lucent.com; Y. Berbers, Computer Science Department, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee (Leuven), Belgium; email: yolande.berbers@cs.kuleuven.be.

DOI: http://dx.doi.org/10.1145/2480741.2480750

2011]. In addition to the power required for the storage itself, energy is required for cooling the data center. Cooling is necessary because of the high energy-dissipation density (150–200 Watt per square foot) and is accountable for one third of the power consumed by the data center [Zhu et al. 2005]. Moreover, not only the economic cost of energy should be considered, but also its environmental impact in terms of carbon dioxide (CO_2) emissions. In the U.S., the generation of 1 kWh of electricity results in 0.7 kg of emitted CO_2 [Zong et al. 2007].

It is expected that the share of the storage subsystem in the data-center power consumption will increase further because storage capacity requirements are rising annually by around 60% [Pinheiro et al. 2006]. Digital data are forecast to occupy 1.8 zettabytes by 2011 [Kaushik et al. 2010]. The storage subsystem consumes a large share of power because it primarily consists of hard disk drives, which require mechanical movement for their operation. Taking the rapidly improving performance of microprocessors into account, data-intensive network servers need an increasing number of high-performance (fast-spinning) disks to avoid I/O becoming the bottleneck. Also, the transition from tape backups to online (disk-based) backups leads to an elevated power consumption [Storer et al. 2008].

To enable sustainable growth of data-center storage, researchers started looking for methods to reduce the power consumption of the storage subsystem. This new research area was defined about a decade ago: the first publication we found on this topic dates from 2002 [Colarelli and Grunwald 2002]. In the 1990s, researchers had already spent significant effort on making hard disks for portable computers more energy efficient. This endeavor was driven by consumer desire for extended battery life. However, this research did not find its way into the data center until the beginning of the 21st century, when concerns grew about rising data-center power consumption. Section 6 maps all of the established power-reduction techniques on a timeline.

Note that we refer to power reduction as a savings in energy consumption, rather than a reduction in instantaneous power. In fact, we use the terms power and energy interchangeably, as many publications in this domain do.

Whilst the sheer number of publications justifies a comprehensive and structured overview, to the best of our knowledge, no exhaustive survey of this domain exists. For our survey, we consider all articles on power-saving techniques for data-center storage, regardless of the specific storage subsystem addressed. Thus, the techniques described may apply to the disk drive itself, the disk array, the storage server, the distributed file system, etc. Section 6 also classifies all of the techniques according to the layer in the storage stack they address. We describe power-reduction techniques targeted at any type of workload ranging from light to heavy and from read-dominated to write-dominated. We particularly pay attention to the fundamental trade-offs between power consumption, capacity, performance (response time and throughput), and dependability (reliability and availability). This survey focuses on the storage and file-system software rather than the hardware aspects of storage devices and media. However, we do consider innovative storage and file-system designs that save power by taking advantage of new storage hardware. We cover techniques that specifically target power reduction as well as performance-improvement methods that increase energy efficiency as a side benefit, but the focus is on the former.

After a decade of research, a large number of power-saving techniques for data-center storage has been conceived. However, most of the techniques were analyzed, simulated, and experimented with in relative isolation. Methods were compared, but seldom an attempt was made to combine techniques with the goal of accumulating energy savings. With this survey, we would like to stimulate synthesis in the design of new energy-efficient file and storage systems, uniting power-saving techniques and identifying conflicting ones.

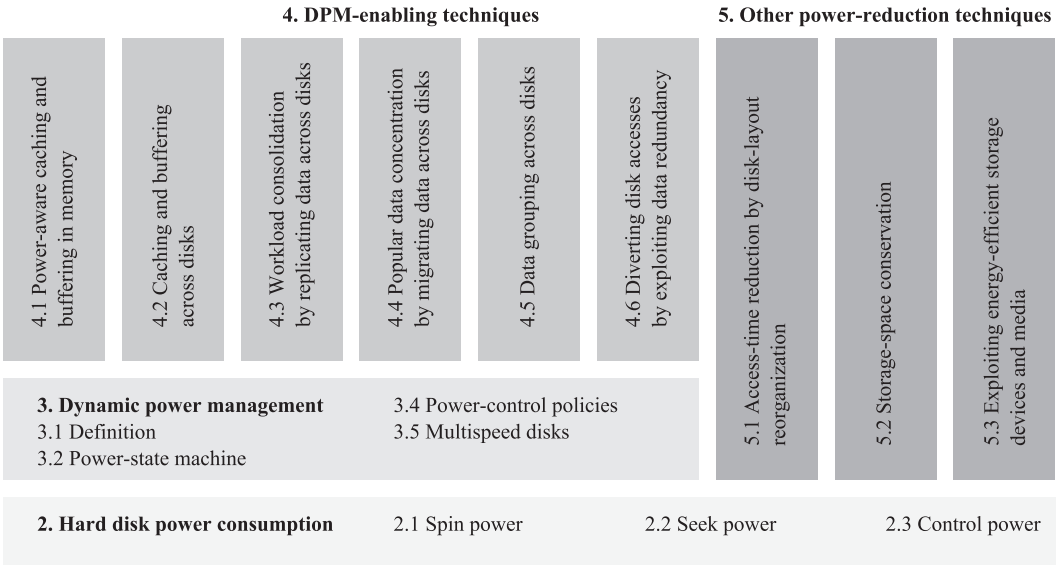


Fig. 1. Survey organization.

Surveys of related domains exist. Venkatachalam and Franz [2005] survey power-reduction techniques for microprocessor systems. With such a broad scope, however, only a small part of their survey is dedicated to power-reduction techniques for disk drives. Deng [2011] gives an overview of the future of disk drives in all of their aspects, but his article barely touches upon power-reduction techniques. Some publications analyze existing power-reduction techniques for storage systems from a specific angle. However, these lack the breadth and depth to serve as a real survey. Guerra et al. [2010] discuss how storage can be made energy-proportional by applying existing power-reduction techniques. Chrobak [2010] discusses online problems in power management for memory (including disk drives).

This survey is structured as shown in Figure 1. In Section 2, we analyze the power consumption of a hard-disk drive, the primary building block of any data-center storage system. In succession, we describe a disk's spin, seek, and control power. We identify the impact of potential power-saving design changes on disk capacity and performance.

In Sections 3 and 4, we describe techniques that have power reduction as the primary goal, while considering the disk characteristics as a given. *Dynamic power management* (DPM), described in Section 3, is the basis for such techniques. DPM spins down an idle disk when, ideally, the idle time is at least sufficiently long to compensate for the power-state transition costs. However, in an enterprise environment, the idle time between two disk accesses is, in most cases, too short for DPM to save energy. Therefore, DPM may alternatively reduce the rotational speed of a disk under light load, but such a speed-reduction requires a *multispeed* disk. Fortunately, techniques were devised that allow for applying DPM to conventional server disks as well. Such techniques reduce the number of disk accesses or concentrate such accesses over time and across disks to enlarge idle time intervals. Section 4 describes such DPM-enabling techniques, which may be subdivided into one of the six categories presented in Figure 1. The major part of this survey addresses such DPM-enabling techniques because these are specific to enterprise storage systems.

In Section 5.1, we describe performance-improvement techniques that reduce power consumption as a side benefit. As opposed to DPM, which minimizes power consumption

of an idle disk, such techniques reduce the energy dissipation of an active disk. These methods shorten the disk access time by reorganizing the data layout on disk. Section 5.2 describes space-conservation techniques that can also reduce power consumption, namely data compression and deduplication. In Section 5.3, we present how energy-efficient storage devices and media may be exploited to save energy. We consider, in succession, laptop disks, multiactuator disks, hybrid disks, and solid-state disks.

Finally, Section 6 zooms out from the details of the individual techniques to the level of the research domain as a whole, to conclude.

2. HARD-DISK POWER CONSUMPTION

The core of secondary storage systems consists of disk drives. Therefore, to fathom and organize the domain of power-reduction techniques for storage systems, it is important to understand for which purposes a disk needs power. The total power consumed by a disk drive P_{dk} is given by Equation (1).

$$P_{dk} = P_{sp} + P_{sk} + P_{ct}. \tag{1}$$

As an approximation, one commonly assumes that the spindle motor power P_{sp}, seek power P_{sk}, and power P_{ct} required for the operation of the interface and control logic each represent one third of the total disk power P_{dk} [Freitas and Wilcke 2008]. The power supply to the spindle motor may be considered constant over time. The interface and control logic continuously draw power for their operation, but more power is required during the actual data transfer. However, the additional power drawn for transferring the data is significant only for larger I/O requests. The power consumed by seeking varies with the I/O workload: when the disk is idle, no seek power is required. On average, about 2/3 of the total disk power consumption is fixed, while the remainder varies with the workload [Allalouf et al. 2009].

2.1. Spin Power

A hard disk consists of one or more platters coated with a thin magnetic layer for storing data. These platters continuously spin when the disk is powered on. To sustain a constant spinning speed, the disk needs to overcome air friction. This is addressed by the disk motor, which consumes a major part (P_{sp}) of the power supplied to the disk.

P_{sp} depends on the number of platters N_{pl}, the platter diameter d (typically expressed in inches), and the rotational or angular speed ω (expressed in revolutions per minute, i.e., RPM), as given by Equation (2) [Gurumurthi et al. 2005].

$$P_{sp} \propto N_{pl} d^{4.6} \omega^{2.8}. \tag{2}$$

From Equation (2), one can derive three ways to decrease spin power consumption: use disks with fewer platters, decrease the platter diameter, or reduce the rotational speed. We explore these options and their impact on capacity and performance in the next sections.

2.1.1. Using Fewer Platters. The capacity C of a disk drive is proportional to its number of surfaces N_{sf}, the square of its platter diameter d, and its areal bit density σ_b, typically expressed in bits per square inch (BPSI), as given in Equation (3) [Gurumurthi et al. 2005].

$$C \propto N_{sf} d^2 \sigma_b. \tag{3}$$

The maximum number of surfaces is twice the number of platters. When the number of platters is reduced by a certain factor, the power consumption decreases by the same factor, but the capacity declines as well, as may be observed from Equations (2) and (3) given that $N_{sf} \propto N_{pl}$. Consequently, the power consumed per bit does not decrease when the number of platters is reduced.

2.1.2. Reducing the Platter Diameter. When the platter diameter is reduced to save power, this is done at the expense of disk capacity, as may be observed from Equation (3). However, the spin power per bit is still proportional to $d^{2.6}$, which means that it makes sense to reduce the platter diameter to improve energy efficiency, at least as far as the contribution of spin power is concerned. However, one cannot conclude that disk drives with a smaller diameter are overall more energy-efficient per bit, because spin power represents only a part of the total disk power. Indeed, the control power does not decrease when the platter diameter becomes smaller. Note that the capacity loss resulting from a reduction of the platter diameter can be compensated by an increase of the areal density, without negatively impacting the consumed spin power.

2.1.3. Reducing the Rotational Speed. The performance of a disk drive is measured in terms of *response time* and *throughput*. Response time T_{rp} is the sum of the *queueing time* T_q, during which a request waits in the queue to get served by the disk, and the *access time* (or *service time*) T_{acs}, during which the request is served by the disk, that is, $T_{rp} = T_q + T_{acs}$. When the request interarrival times and the access times are exponentially distributed, the expected queueing time T_q equals $\rho T_{acs}/(1-\rho)$, where ρ is the disk utilization. The access time can be decomposed in *seek time* T_{sk}, *rotational latency* T_{rt}, and *transfer time* T_{tf}, as given by Equation (4) [Deng 2011].

$$T_{acs} = T_{sk} + T_{rt} + T_{tf}. \tag{4}$$

Seek time represents the time it takes to move the disk head to the right track and is further explained in Section 2.2 because it doesn't depend on rotational speed. Rotational latency (expressed in ms) is the time required to rotate the right sector under the head and is inversely proportional to the rotational speed (expressed in revolutions per minute, i.e., RPM). On average, rotational latency is half the time the disk needs for one revolution, as given by Equation (5) [Deng 2011].

$$T_{rt} = \frac{1}{2}\frac{60 \times 10^3}{\omega}. \tag{5}$$

Transfer time (expressed in ms) is the duration of the actual data transfer. It is a function of the request size s_{rq} (expressed in KiB) and the transfer rate R_{tf} (expressed in MiB/s), as given in Equation (6) [Deng 2011]. The transfer rate, also called internal data rate (IDR) or peak bandwidth, depends on the rotational speed (expressed in RPM), track radius r_{trk} (expressed in inches), and linear bit density λ_b (expressed in bits per inch, i.e., BPI). The outermost track has the highest transfer rate because it stores the most data while the rotational speed is the same for all tracks.

$$T_{tf} = \frac{s_{rq}}{1024}\frac{1000}{R_{tf}} \text{ where } R_{tf} = \frac{\omega}{60}\frac{2\pi r_{trk}\lambda_b}{8 \times 1024 \times 1024}. \tag{6}$$

Finally, the throughput (expressed in requests per second) is defined as $R_{rq} = 1000/T_{acs}$. Thus, when the rotational speed is reduced, the rotational latency increases, as well as the transfer time. This implies an increase of the access time, the queueing time, and hence the response time. The throughput is also negatively impacted by decreasing the rotational speed. Although a speed reduction worsens performance, disk manufacturers have introduced reduced-speed enterprise disks. These disks may be deployed when power reduction is a priority.

2.2. Seek Power

Data are read from or written to a platter by a disk head. When the disk head is not positioned at the right track for reading or writing, it needs to be moved there. This movement is called a *seek* and it is handled by the access arm to which the head is

attached. The arm is moved by the voice-coil actuator consuming power P_{sk} during this movement. In contrast to the spinning motor, the actuator does not consume power continuously but only when a disk seek occurs. On average, a disk seek requires power to accelerate (P_{acl}), decelerate (P_{dcl}), and settle (P_{st}) the disk head, as given in Equation (7).

$$P_{sk} = P_{acl} + P_{dcl} + P_{st}. \tag{7}$$

For an average seek, there's no *coasting* (traveling at constant speed), because the average seek distance d_{sk} is 1/3 of the data zone, which is too short to reach the maximum nominal speed [Deng 2011]. We assume that acceleration equals deceleration, that is, $a_{acl} = a_{dcl} = a$ [Deng 2011]. On average, the power required for acceleration and deceleration is proportional to the acceleration and the average seek distance, as given by Equation (8).

$$P_{acl} = P_{dcl} \propto ad_{sk}. \tag{8}$$

From Equation (8), one can derive two ways to decrease seek power consumption: decrease the seek acceleration or the average seek distance.

2.2.1. Reducing the Seek Acceleration. When one decreases the seek acceleration, a performance reduction is expected because response time and throughput depend on the seek time T_{sk}, as explained in Section 2.1.3, and seek time depends on seek acceleration, as given by Equation (9) [Deng 2011].

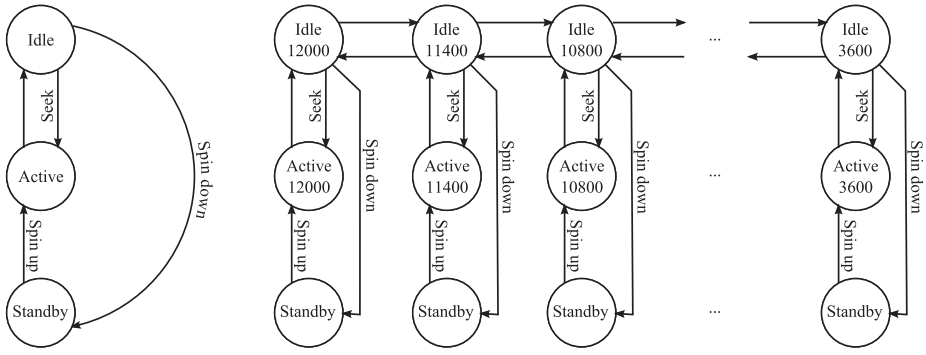
$$T_{sk} = 2\sqrt{\frac{d_{sk}}{a}} + T_{st}. \tag{9}$$

When the seek acceleration is decreased by a factor of 4, so is power, but seek time only doubles, as observed when comparing Equations (8) and (9). However, seek time is commonly considered too important to trade for power. Nevertheless, disk manufacturers invented a method to reduce acceleration without impacting the overall access time. This method is called *just-in-time seek* [Seagate 2000]. Instead of moving the head as fast as possible to the right track and then waiting for the requested sector to rotate towards the head position, it is more energy-efficient to move the head with a reduced acceleration that brings it to the right track only just-in-time, that is, the requested sector passes under the head exactly when the latter arrives at the right track.

2.2.2. Average Seek-Distance Reduction. A reduction of the average seek distance not only lowers the power consumption but also reduces seek time, as Equation (9) indicates. First, a reduction of the seek distance may be achieved by shrinking the platter diameter which impacts capacity, as described in Section 2.1.2. Second, the seek distance may also be reduced by increasing the areal density. Third, the layout of data on disk determines the seek distance. Section 5.1 explores how the seek distance can be decreased by optimizing the data placement on disk.

2.3. Control Power

The disk continuously consumes power P_{ct} for the operation of its interface and control logic. The disk controller maps logical to physical disk addresses, transfers data from the platters to the peripheral bus when reading and vice versa when writing, and manages the integrated cache [Deng 2011]. The disk controller consumes power when the disk is on, even when it is idle. However, when actual data transfer occurs, the power consumed by the interface and control logic surges. This variable part of the disk power consumption is only significant for large I/O requests [Allalouf et al. 2009].



(a) Conventional. (b) Multispeed.

Fig. 2. Power-state machine.

3. DYNAMIC POWER MANAGEMENT

In this section, we start the survey of actual power-reduction techniques with a description of *dynamic power management* (DPM) and its application to conventional server disks (§3.1, §3.2, and §3.3) and newly-proposed *multispeed* disks (§3.4). We define DPM in Section 3.1. DPM is based on a *power-state machine* (§3.2) and *power-control policy* (§3.3).

3.1. Definition

Dynamic power management is a technique for reducing power consumption by turning off system components or decreasing their performance when they are idle or underutilized [Benini and Micheli 2000]. For disk drives, DPM is also known as *resource hibernation* [Venkatachalam and Franz 2005] or *disk power management* [Zhu and Zhou 2005]. DPM can make a system *energy-proportional*, which means the power consumed is in proportion to the amount of work performed [Barroso and Hölzle 2007]. Without DPM, a hard-disk drive is far from energy-proportional. Only the seek power and power required for the actual data transfer scale perfectly with the workload. The disk motor, on the other hand, keeps the platters spinning at a constant speed, and the disk controller remains active, independently of the rate of I/O requests. These factors represent, on average, two thirds of the total disk power budget.

3.2. Power-State Machine

A modern disk drive may be considered a *power-manageable component* (PMC) because it has multiple *modes of operation* and corresponding *power states* for trading power and performance. A PMC may be modeled as a *power-state machine* (PSM). Transitions between power states typically incur a cost in terms of energy and latency [Benini and Micheli 2000].

Figure 2(a) depicts a three-state disk-drive PSM [Gurumurthi et al. 2003a]. Actual data transfer occurs in the *active* mode. In this state, the platters spin and the control and interface logic is active, consuming the active power P_{act}. When the disk is just waiting for I/O requests to arrive, it is in the *idle* mode, drawing the idle power P_{id}. In this state, the disk continues rotating. The disk controller consumes less power in the idle mode than in the active one because no data transfer occurs. When a request arrives, a seek is typically required before the disk can transition from the idle to the active mode. Seeking requires considerable power, represented by P_{sk}, because of the mechanical movement involved, as explained in Section 2.2.

DPM adds a third state, namely the *standby* mode, in which the spindle is at rest and the heads are parked, resulting in power savings. In the standby state, the power P_{sb} is consumed. When the disk is idle, it may be put in standby mode by lifting the head off the platters into a safe location, also called *unloading*, and spinning the disk down, which costs time T_{dn} and energy E_{dn}. If the disk is in standby when a new I/O request arrives, it needs to spin up its platters and load its heads, which costs time T_{up} and energy E_{up}. The disk spin-up time ranges from a few seconds to tens of seconds, which is 3 to 4 orders of magnitude larger than the access time ($\sim$ milliseconds).

In addition to potential energy waste and deteriorated performance, power-state transitions lead to degraded disk reliability. The *duty-cycle rating* is the number of spin-downs a disk can withstand before the failure risk on drive spin-up exceeds 50 %. For server-class disks, a duty-cycle rating of only 50,000 is typical because the disk head is moved to a landing zone on the platter when the disk is turned off. For laptop disks, a duty-cycle rating of 500,000 is common because the disk head comes to rest off the side of the platter, reducing wear caused by stiction effects [Bisson et al. 2007].

3.3. Power-Control Policies

Because of the disk spin-down energy cost $E_{dn}+E_{up}$, a disk has to remain in the standby mode for a time T_{sb}^{min} at least until $E_{dn}+T_{sb}^{min}P_{sb}+E_{up} = P_{id}(T_{dn}+T_{sb}^{min}+T_{up})$ [Lu et al. 2000]. If the actual standby time T_{sb} is larger than this minimum standby time T_{sb}^{min}, then a disk spin-down saves energy. The minimum idle time that allows the disk to remain in the standby mode during the minimum standby time is called the *break-even time* T_{be} and is given by Equation (10) [Lu et al. 2000].

$$T_{be} = T_{dn} + T_{sb}^{min} + T_{up} = \frac{E_{dn} + E_{up} - P_{sb}(T_{dn} + T_{up})}{P_{id} - P_{sb}}. \tag{10}$$

The control of the power-state machine is typically assigned to a *power manager*, which implements the power-control procedure or *policy* based on the measured or assumed workload characteristics. The most popular power-control policies are based on thresholds. Threshold-based disk spin-down policies are also referred to as *traditional power management* (TPM). Although devised for laptops, these policies may be considered a baseline for server-specific techniques. The thresholds may be fixed, variable (e.g. depending on the time of day), or adaptive (e.g., adjusted based on the time-varying actual workload and resulting performance impact) [Golding et al. 1995].

A threshold-based disk spin-down policy relies on a timer that is reset just after the service of an I/O request is completed. The timer runs as long as the disk remains idle. If the value of the timer exceeds a configured threshold τ, then the disk transitions to the standby mode [Douglis et al. 1994]. A threshold-based disk spin-down policy is usually combined with the trivial spin-up policy of turning on the disk motor when a new I/O request arrives. This obviously leads to a significant performance deterioration because the access time is increased by the spin-up time. The effectiveness of this policy depends on the threshold value τ chosen [Benini and Micheli 2000; Lu et al. 2000].

If we assume that the I/O request arrival pattern is completely unknown, then we have to resort to *competitive analysis* to find the best threshold value. When τ is set to the break-even time T_{be}, then the threshold-based disk spin-down policy is said to be 2-competitive, and its competitive ratio equals 2. This means that such online, timer-based procedure consumes at most twice the power of the optimal offline policy for any pattern of I/O request arrivals [Irani et al. 2005].

3.4. Multispeed Disks

Under a typical data-center workload, the average idle time between I/O requests is not long enough to justify a disk spin-down. Gurumurthi et al. [2003b, 2003a] and

Carrera et al. [2003] solve this problem by modulating the disk speed according to the disk load. Gurumurthi et al. call their technique *dynamic rotations per minute* (DRPM), whereas Carrera et al. refer to their technique as the *multispeed* approach. As explained in Section 2.1.3, by decreasing the rotational speed, power consumption may be reduced at the expense of performance. A speed reduction saves less power than a full spin-down, but the break-even time for a speed reduction is smaller as well thanks to a smaller power-state transition cost in terms of energy and latency. Moreover, when a limited performance degradation is acceptable, one may decide not to increase the speed again at the arrival of the next request, but service it at the reduced speed. A multispeed disk improves the energy proportionality of the disk subsystem, as it allows a flexible trade-off between power and performance according to the load imposed to the disk.

Figure 2(b) shows the power-state machine for a multispeed disk (with 15 speed levels from 3,600 RPM to 12,000 RPM) that applies an online heuristic power-control policy [Gurumurthi et al. 2003b]. In this case, some performance degradation is deemed acceptable. Therefore, requests may be serviced at a reduced speed. The power-control policy is implemented partially at the level of the disk controller, partially at the level of the array controller. The disk controller checks periodically how many requests reside in its queue. If the queue is empty, then it may reduce its speed by one level, down to the minimum speed set by the array controller. The array controller measures the increase in response time periodically. If the response-time increase is larger than a predetermined upper tolerance, then the array controller sets the minimum speed at which every disk is allowed to operate, to the full speed. If the response-time increase is smaller than a configurable lower tolerance, then the minimum speed is decreased a number of levels as a function of the ratio of the response-time increase to the lower tolerance.

While Gurumurthi et al. [2003a, 2003b] and Carrera et al. [2003] introduced DRPM as a concept in 2003, to date no multispeed disks exist that can serve requests at a reduced speed, at least not without an offline reconfiguration of the disk [Okada et al. 2000]. However, disk manufacturers already do offer energy-efficient disk drives that exhibit a reduced-speed power state [Seagate 2010]. When the idle time exceeds a threshold, the rotational speed is reduced. When a new I/O request arrives, the disk increases its speed again to the maximum before serving the request. This type of disk is purpose-built for *nearline* data-center storage of write-once/read-maybe archival workloads, which require a performance level between expensive high-performance online storage and inexpensive high-capacity offline storage. Although DRPM has not been fully industrialized yet, it is used as a baseline in a number of publications that target enterprise-storage power savings [Li and Wang 2004a; Zhu et al. 2004; Pinheiro and Bianchini 2004; Carrera et al. 2003; Xie 2008].

4. DPM-ENABLING TECHNIQUES

Typical data-center workloads are characterized by periods of idleness too short to realize power savings by means of traditional power management, except for periods of light load, for example, during the night or weekend [Gurumurthi et al. 2003c; Carrera et al. 2003]. Alternatively to multispeed disks, which adapt DPM to cope with short idle periods, *DPM-enabling* techniques exist which target an extension of the idle periods.

Energy savings can be maximized by distributing I/O requests over time and across multiple disks in an array as follows. First, the arrival rate of I/O requests to the disk or disk array needs to be minimized such that the mean idle time is maximized. Second, the idle-time variance needs to be maximized over time and across disks. On the one hand, an increase of the idle time variance may lead to disk contention and thus to a larger response time for the most-frequently accessed disks. On the other hand, a

Table I. Classification of DPM-Enabling Techniques

Category	Techniques	Section
Power-aware caching and buffering in memory	PA/PB-LRU, PA prefetch, EEFS, WBEU, WTDU, DSC-WB, Coop-I/O	4.1
Caching and buffering across disks	MAID, WO, (PRE)-BUD, LFS	4.2
Workload consolidation by replicating data across disks	PARAID, FREP, SRCMap	4.3
Popular data concentration by migrating data across disks	PDC, Nomad FS, Hibernator, SEA, GreenHDFS, Lightning	4.4
Data grouping across disks	SDP, OE ME	4.5
Diverting disk accesses by exploiting data redundancy	DIV, EERAID, RIMAC, eRAID, DG, PA code, Rabbit, Sierra, CD, AN, GreenFS	4.6

larger idle time variance may result in fewer disk spin-ups, which improves response time. Moreover, when idle time variance increases, fewer power-state transitions are required, which results in improved disk reliability. In this section, we describe all of the DPM-enabling techniques according to the classification introduced in Table I.

4.1. Power-Aware Caching and Buffering in Memory

We describe, in succession, how conventional caching and buffering enables DPM (§ 4.1.1), power-aware cache replacement (§ 4.1.2), power-aware prefetching (§ 4.1.3), energy-efficient cache write policies (§ 4.1.4), and application-enabled batching of read requests (§ 4.1.5).

4.1.1. Caching and Buffering in Memory Saves Energy. To improve performance, data blocks read from disk are *cached* in memory because memory is several orders of magnitude faster than a disk and recently-read blocks have a high probability of being read again soon after. This workload characteristic is called *temporal* locality. For every cache hit, a disk access is avoided. This means that caching results in longer disk idle times, facilitating DPM. So, although caching was introduced to improve performance, it also for allows saving disk power [Douglis et al. 1995]. This disk-power conservation comes at the expense of the memory space used as cache and its associated energy consumption. Fortunately, an effective cache typically requires a space in memory that is only a small fraction of the total disk capacity thanks to the *locality of reference*, which means only a small fraction of the data stored on the disk drive is accessed during a relatively short time interval.

Modified data blocks, which need to be written to disk for persistent storage, are *buffered* in memory such that the application performance is not impacted by the large disk response time. When a buffered block is modified again, a disk access is avoided. This increases the time the disk is idle, creating more opportunities for saving power using DPM. Disk power is saved in exchange for data durability, because if the power supply to the memory is interrupted before the data are written to the disk, then the data are lost [Douglis et al. 1995]. The buffer space is typically part of the memory cache. Buffering resembles a *write-back* caching policy, which writes a modified data block to disk only when it is evicted from the cache [Zhu and Zhou 2005]. A write-back policy is more energy-efficient than a *write-through* caching policy, as the latter writes a modified data block to disk immediately.

4.1.2. Power-Aware Cache Replacement Policies. A modern storage system based on an array of disks commonly features a large cache, potentially based on nonvolatile memory. The storage-cache replacement policy determines which data block is replaced when a cache miss occurs. It is designed for optimal performance by minimizing the number of disk accesses. Although an optimal-performance cache replacement algorithm

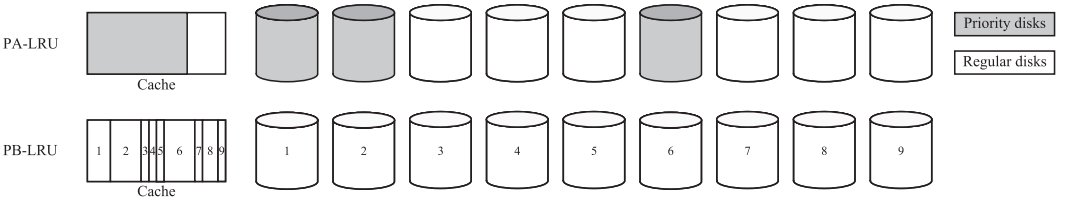


Fig. 3. Power-aware cache replacement policies: power-aware LRU and partition-based LRU.

reduces energy consumption, it does not necessarily lead to minimum power consumption, because it doesn't take into account the distribution of requests over time and across disks. Zhu and Zhou [2004, 2005] introduce two online *power-aware* (PA) cache replacement algorithms: *power-aware LRU* (PA-LRU) and *partition-based LRU* (PB-LRU). Both algorithms trade power for performance: energy is saved in exchange for a reduced cache hit ratio.

PA-LRU classifies the disks in the array in two categories, as shown in Figure 3: regular and priority. This cache replacement algorithm prioritizes disks that exhibit a large percentage of long idle periods and a high ratio of capacity misses to cold misses. PA-LRU selects the least-recently used regular-disk block for eviction. Only when no blocks from regular disks reside in the cache, the least-recently used priority-disk block is evicted. Thus, the cache tends to hold more blocks from priority disks than from regular disks. As a consequence, PA-LRU directs I/O requests mostly towards the regular disks. This means that the priority disks may increase their mean idle time, while for the regular disks, the idle time is reduced, on average. The higher idle time variance results in larger energy savings.

PB-LRU is an online, power-aware cache replacement algorithm that divides the storage cache into individually managed partitions. Every disk has its own partition of the cache with LRU as the block-replacement algorithm, as shown in Figure 3. The partitions are sized such that energy consumption is minimized. The algorithm tends to make partitions larger for relatively inactive disks, while assigning smaller partitions to disks with a higher load. This results in a larger cache-miss rate variance over the disks and hence more energy saved.

4.1.3. Power-Aware (PA) Prefetching. To improve performance, data blocks close to read data blocks are *prefetched* to memory because those blocks have a high probability of being read soon after. This workload characteristic is called *spatial* locality. However, traditional prefetching doesn't save energy, since a disk access is only advanced, not avoided. Nevertheless, since prefetching can change the timing of disk accesses, it has the potential to increase idle time variance, thus saving energy.

Any prefetching algorithm has to determine when to fetch which block and which block to replace. Papathanasiou and Scott [2004a] propose a scheme that always prefetches after completion of a fetch if blocks are available for replacement. In addition, the proposed policy never interrupts an idle period by a prefetch operation except if required to avoid a performance reduction. Deciding which blocks to prefetch is based on how a file is accessed (sequentially, in a loop, or randomly) and when (first and last access time). Prefetched blocks need to be stored in the cache, which requires the eviction of cached blocks. If a *prefetch miss* occurs when the disk is spun down, then the prefetching depth (i.e., the number of blocks prefetched) is increased. On the other hand, if a disk needs to be spun up because of an *eviction miss*, the prefetching depth is decreased. An eviction miss is a miss on a block that was evicted to make room for a prefetched block.

EEFS. Li and Wang [2004b] introduce the so-called *energy-efficient file system* (EEFS) which applies power-aware prefetching at the file rather than block level. EEFS is based on the observation that when a small file is accessed, the probability is high that related small files will be accessed as well soon after. Therefore, when a file is accessed, EEFS prefetches its affiliated files.

In fact, EEFS consists of two file systems: a traditional UFS (Unix File System) and a new group-based file system (GFS). Only small (< 64 KiB) grouped files are stored in GFS. Larger files or files that don't belong to any group are stored in UFS. Periodically, EEFS migrates files between UFS and GFS. GFS stores every group of small related files on the disk in one or more consecutive *clusters*. This layout of files on the disk limits the additional energy required for prefetching to what is required for the actual file transfer: neither additional seek time nor rotational latency are incurred.

4.1.4. Energy-Efficient Cache Write Policies. Zhu and Zhou [2005] present two cache write policies that achieve higher energy savings than the write-back policy: *write-back with eager update* (WBEU) and *write-through with deferred update* (WTDU). WBEU writes all modified data blocks to disk whenever the target disk is active. In this way, WBEU reduces the number of disk spin-ups due to the number of modified data blocks in the cache exceeding a threshold. WTDU writes modified data blocks temporarily to a log whenever the target disk is in the standby mode. The log may reside on any persistent storage device, such as NVRAM or a dedicated log disk.

Weissel et al. [2002] proposes a *drive-specific cooperative write-back* (DSC-WB) policy, which increases idle time variance by adjusting the timing of dirty block updates for Unix-like systems. It is based on four strategies. First, all dirty blocks need to be updated, not only the ones for which the dirty-buffer life span has elapsed. Second, an update has to be performed as soon as the disk becomes active. Third, each disk needs to be updated separately. Finally, an update has to be performed before a disk spins down. This policy resembles WBEU. Hence, power-aware (PA) buffering of modified data blocks in memory facilitates batching the actual writes to disk, which increases idle time variance.

4.1.5. Application-Enabled Batching of Read Requests. Weissel et al. [2002] suggest batching read requests as well, as far as applications exhibit the required flexibility. A write request may involve a read request if the block that requires modification needs to be fetched from the disk. Therefore, application-enabled batching is also beneficial for write requests. Weissel et al. introduce an operating system interface for *cooperative* I/O, which allows the application to specify a timeout value and to set a cancel flag for every I/O request. The operating system tries to abort or defer requests when a disk is in the standby mode. A request may be deferred until its timeout elapses. In this way, requests are batched, increasing idle time variance. The periodic autosave function of a text editor is an example of a deferrable and abortable write operation. Another example is a video player read buffer, which may be filled with deferrable, but not abortable, read operations.

4.2. Caching and Buffering Across Disks

This section differs from Section 4.1 in the sense that data get buffered and cached across disks rather than in RAM. We describe in succession MAID (§ 4.2.1) and write off-loading (§ 4.2.2).

4.2.1. Massive Array of Idle Disks. Colarelli and Grunwald [2002] propose replacing tape libraries with disk arrays for archival storage, which has become feasible thanks to the declining cost of commodity disk drives. Using disks for archival storage improves performance and simplifies management. However, a traditional RAID consumes, per

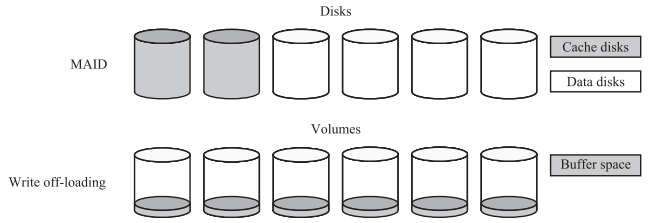


Fig. 4. Caching and buffering across disks: MAID and write off-loading.

bit, at least ten times more than a tape library. Colarelli and Grunwald argue that the level of performance and dependability offered by RAID is not required for archival storage.

Therefore, Colarelli and Grunwald [2002] put forward the idea of a *massive array of idle disks* (MAID). A MAID is a disk array without redundancy for data reliability, but with replication of recently-used data on a number of reserved *cache* disks, as shown in Figure 4. The regular data disks apply DPM, while the cache disks remain always spun up. In fact, for the targeted write-once/read-maybe archival workload, DPM is capable of realizing energy savings even without the use of cache disks. However, when the workload exhibits sufficient temporal locality, read requests are concentrated towards the cache disks. This concentration keeps the regular data disks idle for a longer period, such that the energy savings increase. The cache drives enable additional power savings at the expense of disk capacity.

The cache drives also buffer the writes. A new or modified data block is immediately written to a cache drive, but only written to the target data disk when it transitions from standby to active, typically because of a read miss.

(PRE)-BUD Framework. Zong et al. [2007] propose diverting writes to one or more dedicated buffer disks. They introduce an architecture they call the *BUD framework*, which is composed of a RAM buffer, buffer disks, data disks, and an energy-aware buffer disk controller, similar to MAID. In general, multiple buffer disks may be deployed such that multiple writes can be handled concurrently. For the same reason, every data disk is given an equal partition on every buffer disk. A large write ($>$ 10 MiB) is immediately written to the right data disk. A small write is first buffered in RAM, while the controller finds an idle buffer disk such that the write can be buffered reliably. Manzanares et al. [2008] extend the BUD framework with a prefetching mechanism they call *PRE-BUD*, which replicates the most popular data onto the buffer disks such that most reads can be served from those disks.

4.2.2. Write Off-Loading. Narayanan et al. [2008] build further on the idea of buffering writes on disk to avoid disk spin-ups. They consider this more important than caching on disk for an enterprise data-center workload. This type of workload is, typically, write-dominated during relatively long periods of time because the in-memory cache can effectively address most of the reads.

The redirection of writes from a logical volume in standby to an active volume is called *write off-loading* (WO). A logical volume may be considered a virtual disk, which relies on one or more physical disks, potentially configured as RAID. On every volume, a partition is reserved for temporary storage of off-loaded blocks, as shown in Figure 4. An off-loaded block is copied to its target volume when this volume becomes active because of a read-cache miss. Then, the off-loaded block may be reclaimed. Writeoff-loading increases the idle time variance over the volumes at the expense of some capacity. Narayanan et al. suggest that blocks may also be off-loaded to other persistent

storage, such as NVRAM, when available. Write off-loading resembles the technique of deferred updates touched upon in Section 4.1.4.

Log-Structured File System. Ganesh et al. [2007] show that a log-structured file system (LFS) by itself enables DPM. An LFS is designed for a write-dominated workload, which is common because memory caches are effectively absorbing most of the reads. An LFS writes all updates to disk sequentially in an append-only log. Therefore, ideally, without cache-read misses, all disk accesses would be concentrated to just one disk. Consequently, all other disks could be spun down. An LFS may be considered as write off-loading to the extreme, where the log is the actual target: LFS eliminates the copying to other on-disk data structures.

4.3. Workload Consolidation by Replicating Data across Disks

Under light to moderate load, a subset of disks may be spun down while sustaining full data availability by replicating the disabled disks' data on the disks that remain active and diverting accesses from the inactive to the active disks. In this section, we survey such workload-consolidation techniques, which also reduce the number of power-state transitions, resulting in improved disk reliability and average response time. We describe in succession PARAID (§ 4.3.1), which applies workload consolidation to a RAID, and FREP (§ 4.3.2) and SRCMap (§ 4.3.3), which consolidate the workload on a cluster of nodes (or volumes).

4.3.1. Power-Aware RAID. In many cases, cyclic load fluctuations exist: the load on the storage system is relatively light during a certain recurring period. Conventional RAID wastes energy under light load because data are striped across all disks. This load balancing requires all disks to remain active, even when the load is light. In addition, storage capacity is wasted. Typically, only 30 % to 60 % of storage space is allocated [Weddle et al. 2007].

Weddle et al. [2007] exploit the unused storage space for data replication such that one or more disks of a RAID can be put in standby mode to save power. They call this replication technique *power-aware RAID* (PARAID). Figure 5 illustrates how PARAID replicates data in the case of a four-disk RAID. This replication results in a skewed striping pattern. For a light load, PARAID operates in *first gear*, in which only two disks are active. In first gear, requests for disk three or four can be redirected to disk one or two thanks to the replicated data. Changing the number of active disks is called *gear shifting*. When the load increases, PARAID shifts to higher gears. When disk three is spun up, all of its stale data need to be replaced by means of a synchronization method. The number of spun-down disks depends on the system utilization. Downshifting is done more conservatively than upshifting to prevent power-state transitions from occurring too frequently. Moreover, PARAID rations the number of duty cycles per period for every disk and rotates the gearmembership of the disks to balance their duty cycles.

4.3.2. Fractional Replication for Energy Proportionality. Inspired by PARAID [Weddle et al. 2007], Kim and Rotem [2011] propose replicating data across nodes to facilitate node deactivation when the system load decreases. They call their data-replication technique *fractional replication for energy proportionality* (FREP). They define a *node* to be a group of disks, potentially organized as a RAID. DPM is applied at the level of the node rather than the individual disk. Node deactivation means spinning down all of its disks. Kim and Rotem mainly target read-many/write-rare workloads because many data centers use write off-loading to cluster writes.

Figure 5 illustrates the data-replication strategy employed by FREP. Kim and Rotem make a distinction between *covering set* (CS) nodes, which consist of continuously spinning disks that hold a replica of all of the available data, and non-CS nodes, which

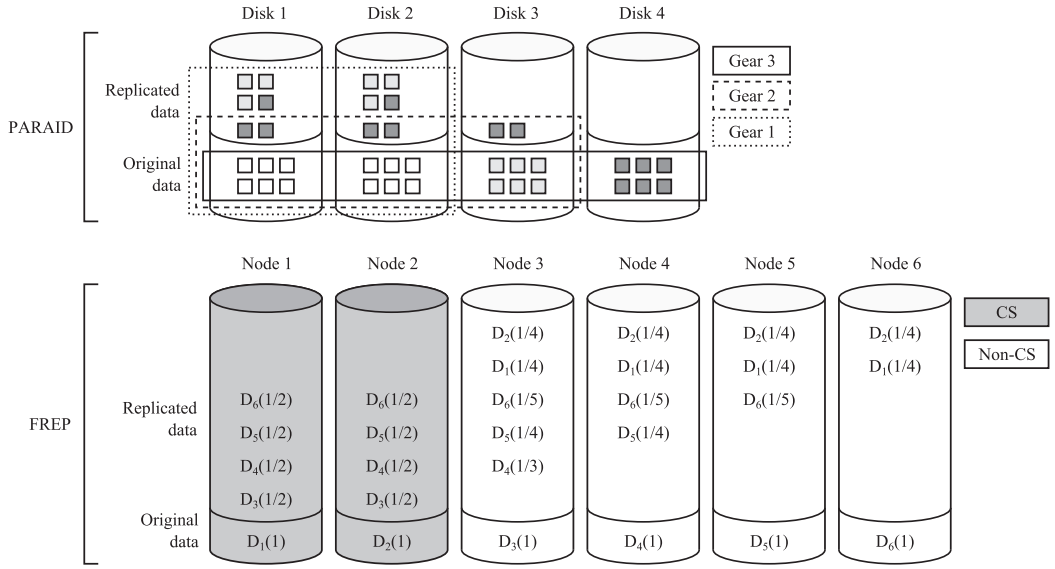


Fig. 5. Workload consolidation by replicating data across disks: PARAID [Weddle et al. 2007] and FREP [Kim and Rotem 2011].

may be deactivated depending on the load. FREP applies gear-shifting at the level of the node. $D_i(a/b)$ denotes an a/b fraction of the original data D_i of node i. Every CS node replicates an equal fraction of the data of each non-CS node to enable the lowest gear, which consists of the CS nodes only. Each non-CS node replicates an equal fraction of the data of each CS node for fault tolerance and load-balancing purposes. Additionally, non-CS nodes replicate data from other non-CS nodes in a skewed way to enable load balancing when operating at a non-highest gear. A fraction $1/(i-1)$ of the data of node i ($i > m+1$) is replicated on every node j ($m < j < i$), where m represents the number of CS nodes.

FREP requires a storage capacity that is larger than twice, but smaller than three times, the size of the original data. The minimum fraction of the total number of available nodes that are required for the covering set is equal to the storage utilization. This minimum corresponds to the lowest possible gear, which results in the minimum power consumption. Thus, the higher the excess capacity, the higher the potential energy savings. FREP allows for balancing the load well, except when only a small number of non-CS nodes are active.

4.3.3. SRCMap. Verma et al. [2010] propose consolidating the workload across physical volumes using a technique they call *sample-replicate-consolidate mapping* (SRCMap). They assume a physical volume to be composed of a RAID. SRCMap samples a subset of blocks, namely the working set, from each physical volume. This working set is replicated on other physical volumes. SRCMap only keeps the minimum number of physical volumes required by the workload turned on. Any I/O request towards an inactive volume is diverted to a replica on an active volume. SRCMap reconfigures the power state of a volume, if needed, only once every two hours, to protect disk reliability. SRCMap exhibits the flexibility to deactivate any number of physical volumes.

SRCMap is different from PARAID in that only the working set, not the full volume, gets replicated. In this way, SRCMap incurs less space overhead. Consequently, every volume can be replicated on multiple other volumes, and every volume can hold replicas of multiple other volumes. Because a replica consists of only the working set, a read

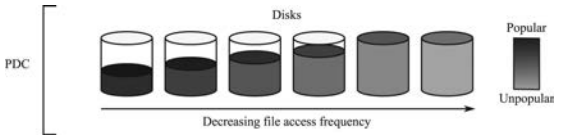


Fig. 6. Popular data concentration.

miss may occur. In such a case, the primary volume is enabled to serve the read and update the active replica. Other secondary replicas are synchronized in the background. Because SRCMap replicates only the working set, it may also be considered a caching technique (cf. § 4.2).

4.4. Popular Data Concentration by Migrating Data across Disks

In this section, we describe power-reduction techniques that distribute data across disks according to the data popularity. All techniques presented in this section are based on this idea of *popular data concentration* (PDC), as proposed by Pinheiro and Bianchini [2004]. In Section 4.4.1, we introduce PDC and its implementation in NomadFS. In Section 4.4.2, we describe techniques that combine PDC and RAID: Hibernator and SEA. In Section 4.4.3, we present applications of PDC to distributed file systems: GreenHDFS and Lightning.

4.4.1. Popular Data Concentration. In general, the problem to be solved is how to allocate a set of m files $\{f_1, f_2, \ldots, f_m\}$ on a set of n disks $\{D_1, D_2, \ldots, D_n\}$ and how to adjust the data placement over time when file popularity changes. A file f_i is characterized by a size s_i and an access rate λ_i. The disks D_j are characterized by a capacity C and a maximum load L, which is defined as a fraction of the peak bandwidth R_{tf}. The load of a file f_i on a disk D_j may be expressed as $l_i = \lambda_i s_i / R_{tf}$. We assume that a file is accessed as a whole [Xie 2008] such that seek time and rotational latency may be ignored. The allocation of files to the disks is always subject to the constraints given in Equation (11) [Otoo et al. 2009].

$$\sum_{f_i \in D_j} s_i \leq C \text{ and } \sum_{f_i \in D_j} l_i \leq L. \tag{11}$$

The access rate of file f_i may be expressed as $\lambda_i = p_i \lambda$, where p_i represents the relative probability for accessing this particular file and λ is the rate at which requests arrive. The file-access rates follow a Zipf distribution: $p_i \propto r_i^{-\alpha}$, with r_i the rank of the file in terms of popularity. The skew parameter $\theta = 1 - \alpha = \log(A/100)/\log(B/100)$ determines the percentage of accesses A that are directed to the percentage of files B. If the skew parameter θ is small (close to zero), then a relatively large percentage of accesses A are directed to a relatively small percentage of files B. Such a skewed file-access frequency distribution is typical for network-server workloads. Such a distribution often causes the per-file *disk-access* frequency to be distributed in a skewed way as well, albeit with a smaller α [Pinheiro and Bianchini 2004]. The file access rate and file size are, typically, inversely correlated [Xie 2008; Otoo et al. 2009].

PDC [Pinheiro and Bianchini 2004] exploits the skewed per-file disk-access frequency distribution. This technique sorts the files according to the frequency with which they are accessed. The files are placed across the ordered disks of an array according to their access frequency, as Figure 6 shows. To avoid a performance hit on the disks with the most popular files, PDC migrates files onto a disk only as far as this disk can support the expected load and subject to its capacity limit, as expressed by Equation (11). The capacity C of the disks holding the most popular data may not be fully utilized in order

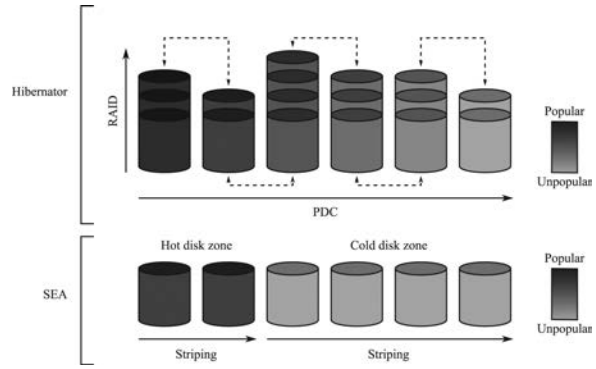


Fig. 7. Combining PDC, RAID, and DRPM: Hibernator and SEA.

to avoid a performance degradation. The migration of files across the disks needs to be applied periodically.

Pinheiro and Bianchini also present a power-aware file system, called *Nomad FS*, which applies PDC. For the sake of simplicity, Nomad FS doesn't provide striping or any form of data redundancy. File metadata are stored on the disk holding the most popular data. File popularity is tracked incrementally every time a file access requires a disk access, using the multiqueue (MQ) algorithm.

4.4.2. Combining PDC, RAID, and DRPM.

Hibernator. Zhu et al. [2005] combine PDC with RAID and DRPM to realize a power-aware storage system they call *Hibernator*. To realize this combination, they introduce a new power-control policy for DRPM, called *coarse-grain response* (CR), which adapts the speed of the disks in an array periodically. Figure 7 shows how Hibernator is composed of groups of disks, called tiers. All of the disks of each tier spin at the same speed, and the data are migrated across tiers such that the most popular data are allocated to the first tier and the least-frequently accessed data to the last tier. Each tier has a RAID data layout for performance and fault tolerance.

At the beginning of every epoch, the CR algorithm determines for every disk the speed that minimizes the predicted power consumption subject to the constraint that the predicted average response time remains smaller than a certain threshold, which is specified through a service-level agreement (SLA). Hence, the disks (and the data they hold) may move from one tier to another, as indicated by the arrows in Figure 6. If the data on a disk are accessed more frequently, the CR algorithm migrates such a disk to the higher-speed tiers and vice versa. This type of data migration across tiers is coarse-grained because it involves the migration of entire disks. Moreover, such disk migration occurs infrequently (once every few hours), because of its negative impact on disk reliability.

Therefore, Hibernator complements the CR algorithm with a *temperature-based* algorithm for *small-scale reorganization* of the data, which runs continuously. Hibernator features 256-KiB relocation blocks (RBs). These blocks are migrated across the disks according to their temperature. The temperature t of an RB is a measure for the access frequency of an RB. It is adjusted after every k accesses as $t_{new} = (1 - \xi)t_{old} + \xi k/T_{last}$, where t_{old} is the previous temperature and T_{last} is the time period of the last k accesses.

When the CR algorithm adds a new disk to a tier, then Hibernator applies a *large-scale reorganization* of the tier's RBs to even out the average temperature of the RBs over all of the disks of the tier by means of a *randomized shuffling* (RS) algorithm.

SEA. Xie [2008] presents a *striping-based energy-aware* (SEA) data-placement strategy, which combines the main ideas of PDC, RAID, and DRPM. SEA is different from Hibernator because it considers only two tiers, a *hot* and *cold* disk zone, as shown in Figure 7. Moreover, SEA only considers two-speed disks, and the speed of every disk is only configured once, when the system is started. Hot disks spin at high speed, whereas cold disks spin at low speed. In addition, only an offline data-placement algorithm is provided.

SEA ranks all of the files according to their popularity. The $(1-\theta)m$ highest-ranked files are considered popular, where θ is the skew parameter, as defined in Section 4.4.1. The other θm files are classified as unpopular. SEA allocates the set of popular files F_h to the hot disks and the set of unpopular files F_c to the cold disks. The ratio γ of the number of hot to cold disks is $\gamma = \sum_{f_i \in F_h} l_i / \sum_{f_j \in F_c} l_j$. SEA stripes all popular files over the hot disks and the unpopular files over the cold disks, similar to RAID-0. Optionally, parity blocks are interleaved with regular data blocks for fault tolerance, similar to RAID-5.

4.4.3. Multizoned Distributed File System.

GreenHDFS. The Hadoop distributed file system (HDFS) [Shvachko et al. 2010] stripes files and replicates the data stripes across a cluster of servers to provide high performance and dependability as required by data-intensive computation. A typical server has four directly-attached 1-TiB disks [Kaushik and Bhandarkar 2010]. The disks don't need to be configured as a RAID because of the replication of data blocks across servers in the cluster. HDFS's data placement based on striping and replication tends to distribute data accesses uniformly over all of the servers of the cluster. Such a uniform distribution impedes DPM.

Therefore, Kaushik and Bhandarkar [2010] introduce GreenHDFS, which integrates the idea of popular data concentration into HDFS. GreenHDFS groups the servers in a Hadoop cluster in two zones: a hot zone hosting popular data and a cold zone where unpopular data are placed. Only the servers in the cold zone apply DPM. In the cold zone, files are not striped such that when a file needs to be read from disk, only one server needs to be activated. Files not recently accessed are migrated to the cold zone. A file may be moved back to the hot zone after it has been accessed a certain number of times.

Kaushik et al. [2011] extend GreenHDFS with a *predictor* component that predicts the lifespan, size, and popularity of every file based on the directory in which the file resides. The predicted file attributes facilitate proactive file placement, migration, and replication.

Lightning. Kaushik et al. [2010] present a design for an energy-aware distributed file system for cloud storage, called *Lightning*, that closely resembles GreenHDFS. Lightning features four zones instead of two: two hot zones (of which one is based on solid-state drives) and two cold zones. A file is allocated to a zone according to its classification. Small read-only files are stored in the first hot zone, based on SSDs. Lightning assigns popular or highly-rated files to the second hot zone. Less popular or lowly-rated files are placed in the first cold zone. Finally, the second cold zone holds least-frequently accessed data, such as backups, archivals, etc. Data classification is based on hints received from the application as well as the file characteristics observed by the file system. File striping is only applied in the second hot zone. Lightning enables DPM only in the cold zones.

4.5. Data Grouping across Disks

Wildani and Miller [2010] group similar data across disks to reduce the number of disk spin-ups in the case of *archival-by-accident* workloads. Such a workload emerges when

active data gradually become passive and is, typically, write-once, read-maybe, except for a changing *hot* area that exhibits a significant number of reads and overwrites. Personal data have a tendency to become an archival-by-accident workload. Wildani and Miller propose to group files according to semantic and incidental labels, such as time stamp, file-system placement, author, file type, etc. They call this technique *semantic data placement* (SDP). It assumes that similar data are temporally related. SDP targets clustering of disk accesses towards a minimal set of disks, especially when a small part of an archival workload suddenly becomes active.

Essary and Amer [2008] present a technique they call *predictive data grouping*, which groups temporally-related data on disk to reduce seek distance and rotational latency, as described in Section 5.1.3. However, they suggest using this technique also to reduce the number of disk spin-ups by applying predictive data grouping across multiple disks or even storage servers.

It is worth noting that data grouping across disks may involve data migration as well as data replication in case the same data are allocated to different groups. We found relatively few publications on energy conservation by grouping related data across disks. Therefore, we believe this class of power-reduction techniques deserves intensified research.

4.6. Diverting Disk Accesses by Exploiting Data Redundancy

This section is different from Section 4.2 because, here, existing data redundancy is exploited rather than new redundancy added. The purpose of the redundancy may be improved throughput, reliability, or availability. Moreover, the redundancy may not only exist in the form of replicated data, but also as parity data and erasure codes. A redundancy configuration is characterized by its (n, m) tuple: every data block is striped, replicated, or encoded into n fragments of which only m ($\leq n$) are required to reconstruct the original block [Pinheiro et al. 2006]. In Section 4.6.1, we describe the technique of *diverted accesses* (DIV), which is applied to RAID in Section 4.6.2 and to distributed file systems in Section 4.6.3.

4.6.1. Diverted Accesses. Pinheiro et al. [2006] introduce a power-reduction technique they call *diverted accesses*, which may be applied to any storage system with an (n, m) redundancy configuration. DIV separates the redundant data from the original data, as shown in Figure 8. We assume the total number of disks to equal n. In the case of erasure coding, it may not be possible to classify fragments as original. However, m fragments of every data block, which allow for reconstruction of the original data, may be allocated to a subset of m disks.

Under light to moderate load, DIV directs read requests only to the m disks holding the original data fragments. Write requests are immediately directed to the m disks holding the original data fragments but only forwarded to the $n - m$ disks where the redundant data fragments are stored when the write buffer is full. The writes of the $n - m$ redundant data fragments are buffered, for example, in NVRAM at $n - m$ storage devices. As such, DIV concentrates disk accesses towards the subset of disks holding the original data.

4.6.2. Applying DIV to a RAID.

EERAID. Li and Wang [2004a] introduce the idea of diverted accesses in the context of RAID-1 and RAID-5. They call their storage system architecture *energy-efficient RAID* (EERAID).

RAID-1, also called mirroring, has a $(2m, m)$ redundancy configuration. The primary replica and mirror replica are allocated to a separate set of disks, as illustrated in Figure 8. For maximum performance, the workload is balanced between the two replicas.

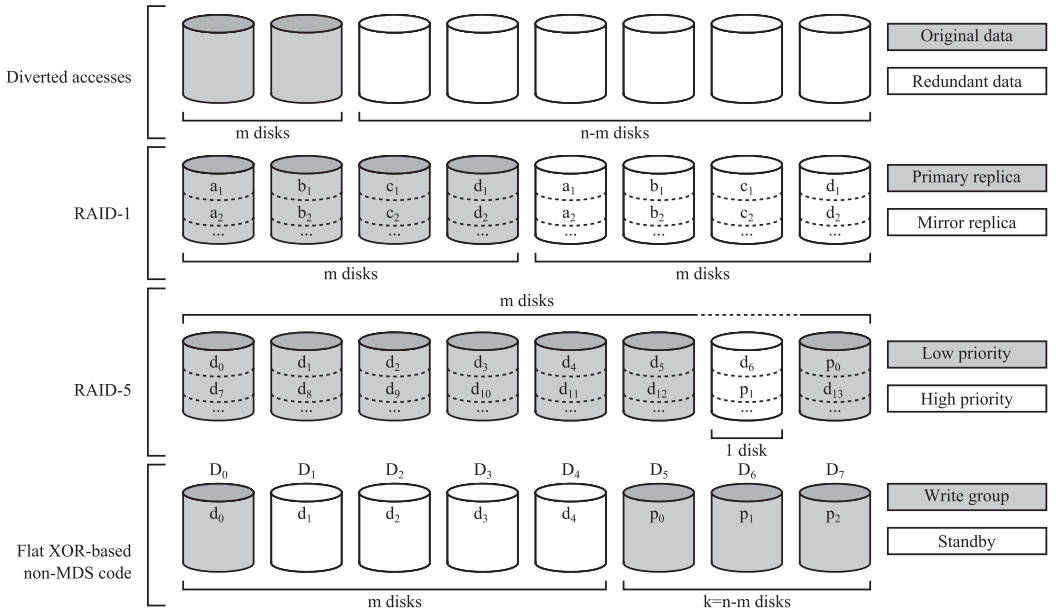


Fig. 8. Diverted accesses and application to RAID-1, RAID-5, and flat XOR-based non-MDS codes [Pinheiro et al. 2006; Li and Wang 2004a; Yao and Wang 2006; Wang et al. 2008; Greenan et al. 2008].

EERAID proposes a *windows round-robin* (WRR) policy for dispatching read requests. This policy forwards N successive read requests alternately to the primary replica and the mirror replica. Writes are buffered according to a so-called *power and redundancy-aware flush* (PRF) policy. This cache write-back policy flushes buffered, modified blocks of a certain replica only during the N-request window when WRR directs reads to that replica. N is adjusted to maximize power savings while limiting the response-time degradation. EERAID spins down the disks holding the replica to which WRR doesn't forward requests if N is large enough and spins up these disks before the end of the N-request window.

RAID-5 has an $(m + 1, m)$ redundancy configuration. RAID-5 partitions data in chunks, which are successively placed across the disks of the array, as illustrated in Figure 8. For every m such data blocks d_i, RAID-5 adds a parity block p_j. These parity blocks are evenly distributed across the array. The set of blocks over which the parity block is calculated is called a *stripe* or *parity group*. A chunk is also called *stripe unit* [Yao and Wang 2006].

EERAID selects one disk of the array as the *high-priority disk*, whereas the subset of the other disks is called the *low-priority group*, as shown in Figure 8. The goal is to maximize the idle time of the high-priority disk. Therefore, EERAID proposes a *transformable read* (TRA) policy that redirects read requests from the high-priority disk to the low-priority group. TRA reconstructs a requested high-priority disk block by XORing all of the other blocks of the stripe. Writes are buffered according to a so-called *power and redundancy-aware destage* (PRD) policy. Destaging involves updating a data block and corresponding parity block. PRD only *destages* blocks for which the disks required for the destaging are part of the low-priority group. TRA and PRD operate jointly during an N-request window. Before the start of a new window, the high-priority disk is selected based on the original, untransformed requests. The size N of the window is adjusted in the same way as for RAID-1. EERAID spins down the

high-priority disk if N is sufficiently large and spins up this disk before the end of the N-request window.

Other Applications of DIV to RAID. Yao and Wang [2006] refine the techniques introduced as part of EERAID, specifically for RAID-5. They propose a two-level cache architecture called RIMAC, which consists of the storage cache and the cache of the RAID controller. RIMAC caches the data stripe units exclusively in the storage cache, whereas the parity stripe units are only cached in the RAID controller cache. Wang et al. [2008] propose a new power-control policy for RAID-1, called *eRAID*, that facilitates trading performance for power in a controlled way. They provide models to predict the degradation of the response time and throughput as a function of the number of spun-down mirror disks. Their policy spins down the maximum number of mirror disks, keeping the predicted performance degradation within predefined limits. Lu et al. [2007] propose a set of new data layouts they call *DiskGroup* (DG), based on RAID-1, that provides an opportunity to increase DIV-based power savings depending on the workload.

Power-Aware (PA) Coding. For an (n, m) redundancy configuration, *maximum distance separable* (MDS) erasure codes defined across a group of disks can withstand $k = n - m$ disk failures, whereas non-MDS codes can only tolerate strictly fewer than k disk failures. However, MDS codes require at least m active disks to reconstruct the data of a single disk in standby mode, given m original-data disks, whereas non-MDS codes require, typically, fewer than m active disks for such reconstruction. Therefore, Greenan et al. [2008] argue that more power can be saved by applying non-MDS erasure codes across disks instead of the traditional RAID-5 redundancy scheme, which is based on MDS erasure coding. Such *power-aware coding* facilitates power savings at the cost of reduced fault tolerance.

Greenan et al. [2008] propose using a flat XOR-based non-MDS code, for which Figure 8 provides an example, characterized by an $(8, 5)$ redundancy configuration. A flat code maps each ith code-word symbol to the ith disk. Each code word of a systematic code consists of m data symbols and $k = n - m$ parity symbols, and an XOR-based code is a systematic code with parity symbols p_j that are calculated by XORing one or more of the data symbols d_i. For example, $p_0 = d_0 \oplus d_1 \oplus d_2$, $p_1 = d_0 \oplus d_1 \oplus d_3$, and $p_2 = d_0 \oplus d_2 \oplus d_3 \oplus d_4$. If, for example, only D_0, D_5, D_6, and D_7 are enabled, as shown in Figure 8, then d_4 may be reconstructed as $d_0 \oplus p_0 \oplus p_1 \oplus p_2$, if requested. A comparable MDS code with an $(8, 5)$ redundancy configuration would not allow reconstruction of d_4 in this case, because this would require at least $m = 5$ active disks.

4.6.3. Applying DIV to a Distributed File System.

Rabbit. In Section 4.4.3, we explained why HDFS may be considered energy inefficient and how PDC, when applied to HDFS, may enable power savings. In this section, we revisit HDFS's lack of power proportionality and describe how DIV may be applied to HDFS to facilitate DPM. HDFS ensures that r replicas of every block of data are stored across the cluster. By default, $r = 3$ [Leverich and Kozyrakis 2010]. However, HDFS places replicas randomly across all nodes of the cluster, only subject to two invariants. First, at most one replica of a data block is stored on any one node. Second, at least one replica of a data block must be stored on another rack than the other replicas.

Taking this data-placement strategy into account, DIV only allows putting $r - 1$ nodes in standby. Therefore, Leverich and Kozyrakis [2010] propose adding one block-replication invariant: at least one replica of a data block must be stored in a subset of nodes called the *covering set* (CS). As such, the covering set may be considered for grouping the original data. This grouping facilitates DIV, which allows disabling the non-CS nodes when utilization is low.

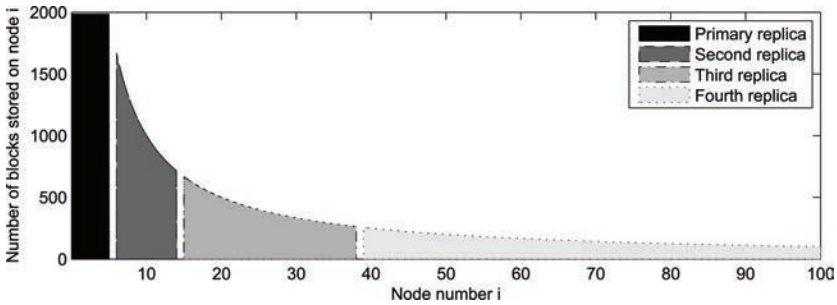


Fig. 9. Rabbit's equal-work data layout [Amur et al. 2010].

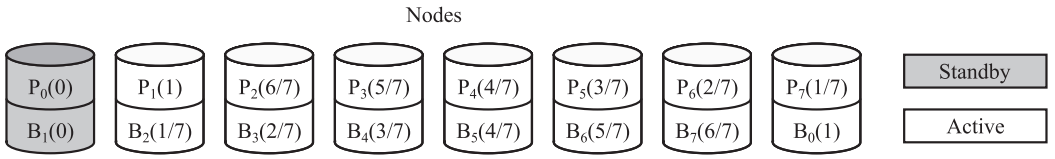


Fig. 10. Chained declustering example [Lang et al. 2010].

Amur et al. [2010] present a *power-proportional distributed file system* (PPDFS), called *Rabbit*, based on HDFS. Rabbit elaborates the idea of a covering set, called *primary set* by Amur et al. Figure 9 illustrates how Rabbit lays out data across the nodes of a cluster. Rabbit places r different replicas of the data on r different subsets of nodes. This segregation of original data and redundant data enables DIV. The primary replica of every block of the dataset is randomly placed on one of the p nodes of the primary set. Every primary node stores B/p blocks, where B represents the total number of blocks of the dataset. Figure 9 gives an example for $B = 10^4$, $p = 5$, and $r = 4$. The cluster consists of $N = 100$ nodes. The consumed power may be minimized, while ensuring data availability, by turning all non-primary nodes off.

Rabbit introduces the idea of an *expansion chain*, which defines the order in which nodes may be activated to scale up performance. The ith ($i > p$) node of the expansion chain stores B/i blocks, which results in *ideal* power proportionality. Power proportionality is ideal if $R_i/P_i = R_N/P_N$ for any $i \in \{p, \ldots, N\}$, where R_i represents the combined throughput of i active nodes and P_i the total power consumed by i active nodes. This *equal-work* data layout ensures that the load can be evenly shared between all active nodes. The *spread* s of a dataset, which equals the number of nodes required to store all of its replicas, has a lower bound: $s \geq pe^{r-1}$. For the example in Figure 9, $p = 5$ was chosen such that the spread spans the entire cluster, given $r = 4$.

Sierra. Whereas Rabbit focuses on read-only workloads, *Sierra* [Thereska et al. 2011] targets general workloads, including read and writes, by integrating write off-loading, as described in Section 4.2.2. The baseline architecture of Sierra resembles Rabbit's. Its data layout, called *power-aware grouping*, is simpler than Rabbit's. The different replicas are on separate subsets of nodes, similar to Rabbit's data layout. These subsets are called gear groups. Whereas Rabbit allows increasing throughput with the granularity of a server, Sierra doesn't provide this fine-grained scaling.

Chained Declustering. Lang et al. [2010] propose a data layout different from the one of Rabbit and Sierra. They use an established data-replication scheme, called *chained declustering* (CD), in combination with DIV. Figure 10 illustrates CD for eight nodes and two replicas. The first replica is called *primary* (P) and the second replica *backup*

(B). The data are striped twice across the nodes. CD tolerates up to $N/2$ node failures, with N the total number of nodes, if the failed nodes are not adjacent. In the context of power reduction, CD guarantees data availability when up to $N/2$ nodes are turned off. Figure 10 shows how the load (between brackets) may be redistributed when, for example, one node is disabled. CD facilitates balancing the workload across the active nodes. In the example of Figure 10, every active node serves 8/7 of the original number of requests.

Auxiliary Nodes. Harnik et al. [2009] apply DIV to existing distributed file systems used for cloud storage, such as HDFS, without changing the data layout, as opposed to Rabbit, Sierra, and CD. Let N be the number of storage nodes, B the number of data blocks, r the replication factor, and q the number of nodes that are disabled under light load to save power. Assume that the DFS employs a random data placement function. Harnik et al. prove that, in such a case, for every subset of nodes of size Nq that is disabled, the expected number of unavailable data blocks is approximately Bq^r. To keep all data blocks available when Nq nodes are disabled, they introduce Nq^r/r *auxiliary nodes* (AN), which store one additional replica for all data blocks of which all original replicas are stored on nodes that may be turned off. These auxiliary nodes are always turned on. The value of q that maximizes the power saving is $\alpha_{low}^{1/(r-1)}$, with α_{low} being the fraction of time the load is sufficiently low to turn off the subset of Nq nodes.

GreenFS. Joukov and Sipek [2008] reduce the power consumption of desktop and laptop disks in the enterprise environment. They increase the idleness of these disks by reversing the roles of such local disks and the remote backup server. The local disk stores a backup of the primary replica of the data stored on the remote server. The backup on the local disk is only used when the network is not available or has insufficient bandwidth. Moreover, every client is equipped with a flash memory device for buffering updates (when the remote server is not available) and caching frequently-accessed data for improved performance. They implemented this idea as part of a *fan-out stackable* file system, called *GreenFS*, which is mounted over the local disks, flash memory devices, and the remote server.

5. OTHER POWER-REDUCTION TECHNIQUES

In this section, we describe power-reduction techniques based on a reduction of the disk access time (§ 5.1), on storage-space conservation (§ 5.2), and on new energy-efficient storage devices and media (§ 5.3).

5.1. Access-Time Reduction by Disk-Layout Reorganization

Whereas DPM targets power savings by putting an *idle* disk in the standby mode, other techniques reduce energy consumption when a disk is in the *active* state, possibly heavily loaded. The latter techniques reduce the average seek distance and rotational latency by improved I/O scheduling or a reorganization of the disk layout. Such access-time reduction techniques improve the response time and throughput. In fact, these techniques were primarily conceived to improve performance. They lower energy consumption as a side benefit. We focus on the publications that relate access-time reduction to power reduction. The access-time reduction techniques may be classified in three categories: caching across disk tracks (§ 5.1.1), popular data concentration by migrating data across disk tracks (§ 5.1.2), and data grouping across disk tracks (§ 5.1.3). Note the similarity to three of the classes of Table I.

5.1.1. Caching across Disk Tracks. Traditional file systems, such as FFS (Fast File System), place related data and metadata blocks close together to limit the seek distance. However, the average seek distance may deteriorate when the system is shared

concurrently by multiple users, database data are stored in a large file, or shared libraries are used. Huang et al. [2005] present the so-called *free-space file system* (FS^2), which replicates data across the disk to reduce seek distance and rotational latency. FS^2 is based on the same observation as PARAID (described in Section 4.3.1), namely that disk capacity is underutilized. The available free disk space may be exploited for data replication to improve performance and save power.

As a background activity, FS^2 first searches a *hot region* on disk, which is a small area where the blocks are accessed frequently. Blocks outside this region that are frequently accessed together with blocks inside this region are copied there to shorten the seek distance. The blocks are replicated in the order in which they were accessed earlier to reduce rotational latency. When the targeted hot region has sufficient contiguous free space, the replicas are placed sequentially on disk. For every request, the system accesses the replicated data instead of the original data if the replicas of the requested blocks are contiguous on disk and the disk head is closer to the replicated blocks than the original ones.

5.1.2. Popular Data Concentration by Migrating Data across Disk Tracks. Huang et al. compare FS^2 with earlier systems, such as the *hierarchical file system* (HFS) [Apple 2004] and the *smart file system* (SFS) [Staelin and Garcia-Molina 1991], which migrate frequently-accessed data to a reserved area on disk. They also refer to other early techniques that adapt the disk layout to minimize the seek distance. These techniques move the most popular data to the middle of the disk. This so-called *organ-pipe* heuristic was proven to be optimal for random disk accesses.

5.1.3. Data Grouping across Disk Tracks. EEFS [Li and Wang 2004a], described in Section 4.1.3, groups small temporally-related files on disk to facilitate prefetching. This grouping results in a reduction of the access time. EEFS employs an online file grouping algorithm based on the *recent popularity* (RP) file successor prediction model. If the size of an identified group of files is larger than a predetermined maximum, then the algorithm breaks the group up in subgroups by means of a depth-first search algorithm. The group size is limited to avoid diminishing energy savings when too many files are prefetched. The algorithm ensures that file groups don't overlap to avoid consistency-control overhead, which would be incurred if multiple copies of a file were stored.

Essary and Amer [2008] describe *predictive data grouping* as a means to reduce the access latency and, consequently, power consumption. Their algorithm resembles the one integrated in EEFS but breaks up groups in subgroups, balancing depth-first search and breadth-first search. They call this balancing *optimal expansion, maximized expectation* (OE ME). Also, Essary and Amer allow groups to overlap, which implies data replication. Moreover, they group data blocks rather than files. Finally, a group is sized such that it fits onto a disk track. Liao et al. [2011] improve the data-grouping technique of Essary and Amer by putting the groups on the disk in the order that minimizes the seek time. Also, they eliminate data replication to avoid the complexity associated with maintaining data consistency. They call their technique *ISRA-based grouping*, where ISRA stands for *immediate successor relationship amount*.

5.2. Storage-Space Conservation

In this section, we describe two techniques for storing data in a space-efficient way: data compression and data deduplication. Such techniques reduce the required disk capacity for data storage by eliminating unnecessary redundancy. Consequently, the data can be stored on fewer disks. Whereas the primary objective of space-conservation techniques is saving disk capacity, energy consumption is reduced as a side benefit. Such techniques are the opposite of the replication-based techniques described in

Sections 4.2 and 5.1.1 as far as their impact on disk space utilization is concerned. Space-conservation techniques are most suited for archival storage, because archived data are size-intensive rather than load-intensive. Because space conservation is a broad research domain by itself, we focus on the publications that link space conservation to power reduction.

5.2.1. Data Compression. Kothiyal et al. [2009] analyze in which cases data compression may reduce the energy consumed for data transfer to and from disk. They consider different software-based data-compression tools available on Linux: *compress* based on the *Lempel-Ziv-Welch* (LZW) algorithm, *gzip* and *lzop* based on the *LZ77* algorithm, and *bzip2* based on the *Burrows Wheeler Transform* (BWT). Four different file types are considered, in ascending order of their redundancy level: *zero*, *text*, *binary*, and *random*.

Data compression doesn't always result in energy savings because the compression itself (before writing) and the decompression (after reading) consumes additional power. Kothiyal et al. establish that the energy consumption is proportional to the time required for the I/O operation and the compression/decompression. The duration of the I/O depends on the amount of data to be read or written. The compression algorithm determines the trade-off between compression ratio and speed.

The compression ratio also depends on the file type. For zero files, data compression is beneficial for almost all of the considered data-compression algorithms. For random files, on the contrary, data compression increases power consumption for all algorithms. For text and binary files, in some cases, the combination of compressing and writing consumes more power than plain writing, whereas reading followed by decompression saves power when compared to plain reading. In these cases, the ratio of reads to writes characterizing the workload determines whether data compression makes sense. Except in the case of lzop, this ratio is typically too low for saving power. Therefore, Kothiyal et al. recommend using lzop. However, this recommendation is only valid for actively-used data, because energy saved while data are passively stored is not taken into account.

5.2.2. Data Deduplication. Whereas data compression eliminates redundancy internal to an object, such as a file or a data block, data *deduplication* suppresses redundancy caused by identical objects [Kulkarni et al. 2004]. Joukov and Sipek [2008] integrate deduplication in the design of their energy-efficient file system *GreenFS* (§ 4.6.3), because in a typical enterprise environment, a lot of files are replicated across local disks. Under such conditions, data deduplication may reduce required disk space by a factor of three.

Joukov and Sipek refer to Hong et al. [2004], among others, for more details about data deduplication. Inspired by *Venti* [Quinlan and Dorward 2002], a system for archival storage, Hong et al. present a *duplicate-data elimination* (DDE) technique targeted for online file systems. DDE operates as a background process in order to minimize its performance impact. When a client writes data on a disk, it sends the hashes of all written blocks to the storage server. The hashes are calculated using a collision-resistant *SHA-1* hash function. The server coalesces blocks with the same hash, also called *fingerprint*, by adjusting corresponding file-block allocation maps. When clients want to modify a block, they need to copy it first, because the block may be referenced by other files. This technique is called *copy-on-write*.

Data deduplication tends to spread temporally-related blocks, for example, the blocks of a file, across the disk tracks and even multiple disks. Consequently, more seeks and accesses to more disks may be required to serve a read, increasing the access time (cf. § 5.1) and decreasing the idle time variance (cf. Section 4). Further research is required

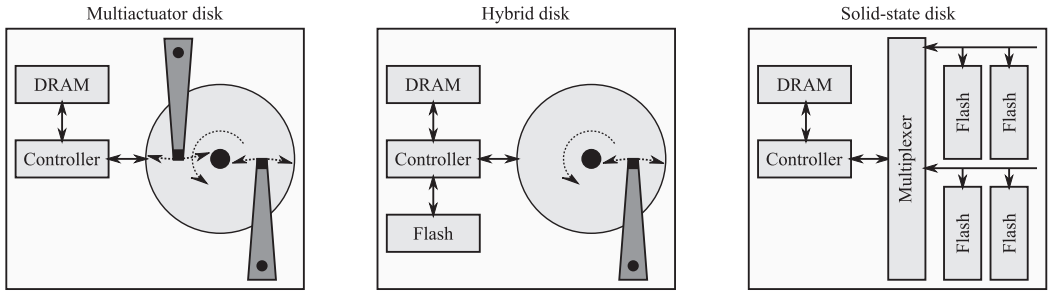


Fig. 11. Overview of energy-efficient disk drive architectures.

to establish whether and under which conditions data deduplication may lead to power savings.

5.3. Exploiting Energy-Efficient Storage Devices and Media

In this section, we describe how storage devices and media other than the conventional server disk may be exploited to reduce power consumption. An overview of new disk-drive architectures, storage devices, and storage media may be found in Deng [2011]. Figure 11 illustrates the three types of new energy-efficient disk-drive architectures, which are described in this section. However, we first describe how a low-power laptop disk may be exploited to build energy-conserving enterprise storage systems.

5.3.1. Laptop Disks. Carrera et al. [2003] explore replacing server disks by lower-performance, lower-power laptop disks. Because performance and dependability need to be maintained, one server disk has to be replaced by a RAID composed of four laptop disks. With respect to performance, only throughput can be maintained, because the response time of an individual request served by a laptop disk cannot be improved by adding laptop disks. Fortunately, throughput may be considered more important than response time, because the latter may be ignored often, taking into account wide-area network latency. However, with a replacement rate of four-to-one, Carrera et al. assess the power savings to be negligible. Based on the characteristics of more recent disk models, Papathanasiou and Scott [2004b] calculate, however, that a replacement ratio of three-to-one suffices to maintain performance and dependability. Based on simulation, they observe energy savings of more than 50 %. The main disadvantage of replacing a server disk with three laptop disks is the initial cost of a laptop disk, which is comparable to that of a server disk.

Because Carrera et al. [2003] disregard the replacement of server disks by laptop disks, they explore the possibility of combining of every server disk with one laptop disk of the same size, which mirrors the server disk. Only the laptop disk is active when the load is light, whereas only the server disk is activated under heavy load. When switching from the laptop disk to the server disk and vice versa, both disks remain active during a short transition period to allow synchronization.

5.3.2. Multiactuator Disks. Sankar et al. [2008] argue that the number of disks in a high-performance storage system is determined by requirements of performance rather than capacity. This leads to an underutilization of the available disk space and thus a waste of energy. Therefore, Sankar et al. propose to improve disk performance such that disks can be completely filled, and hence, a minimum number of disks is required. They suggest eliminating the performance bottleneck by introducing intradisk parallelism, for which they put forward the $D_kA_lS_mH_n$ taxonomy, where k, l, m, and n represent the degree of parallelism in, respectively, the disk stack (D), arm assembly (A), surface (S),

and head (H). They establish that rotational latency is the primary performance bottleneck. Therefore, they propose introducing parallelism along the actuator dimension, which corresponds to design points characterized by $D_1A_nS_1H_1$. Figure 11 provides an example for $n = 2$. They simplify the design of a multiactuator disk by not allowing more than one access arm to move simultaneously and by letting only one head at a time transfer data. These restrictions ensure that such multiactuator disks have peak power consumption comparable to that of conventional disks. The disk scheduler selects for every I/O request the access arm that minimizes the positioning time.

5.3.3. Hybrid Disks. Deng [2011] describes a *hybrid disk* as the combination of a conventional disk with NAND flash memory, which serves as a second-level cache, as shown in Figure 11.

NAND Flash Memory. NAND flash memory is nonvolatile memory that is accessed in a similar way as a block device, such as a disk. A NAND flash memory device contains in a single *package* (or *channel*) multiple *dies* (also called *chips* or *ways*). A die is composed of multiple *planes* (or *banks*) of blocks. Every block consists of multiple *pages*, and a page is divided into multiple *subpages* of 512 bytes, the size of a sector of a hard disk drive. Reading and writing (also called *programming*) occurs at the page level. However, a page cannot be rewritten: it needs to be erased first. Erasing needs to be done at the block level. NAND flash memory can be erased only a limited number of times, typically ranging from 10,000 to 100,000. NAND flash memory consumes relatively little power because it doesn't require mechanical movement. Mohan et al. [2010] propose an analytical power model for NAND flash memory. Grupp et al. [2009] empirically characterized the power consumption of 11 different NAND flash memory devices of five different manufacturers. They provide measurements of the idle power as well as the peak power, average power, and per-operation energy for reading, programming, and erasing.

NVCache. Bisson et al. 2006, 2007 describe how a hybrid disk, which integrates a so-called *NVCache* device based on flash memory, may be used to save power. They implement a threshold-based disk spin-down policy that exploits the NVCache in four ways.

First, when the disk is in the standby mode, write requests are directed to the NVCache. This means that the timer employed by the disk spin-down algorithm has to be reset only when a read access occurs, because a write access doesn't cause a disk spin-up. Second, every read request is first passed to the NVCache, but in most cases, a read miss occurs, thus the read needs to be served by the disk. When a read miss occurs, the read data are stored in a partition of the NVCache such that a disk spin-up due to the same read miss is avoided in the future. Third, Bisson et al. propose spinning up the disk in anticipation of an I/O request that cannot be served by the NVCache. For example, for writes, the optimal anticipatory spin-up time can be calculated based on the available number of free NVCache sectors and the rate at which such sectors are written. Fourth, writes to the NVCache are throttled to avoid wearing out the flash memory prematurely.

The NVCache enables DPM in a similar way as the techniques described in Section 4.1 but based on low-power, nonvolatile flash memory. Thus, writes can be buffered without jeopardizing reliability.

Pergamum. Storer et al. [2008] present an energy-efficient archival storage system, which consists of a network of low-power storage appliances, called Pergamum *tomes*. A tome resembles a hybrid disk. It is composed of a commodity disk, flash memory, a low-power CPU, and a network port, but no RAM. Pergamum's architecture facilitates distributed control of the storage system, eliminating the need for power-hungry

storage servers and RAID controllers. Pergamum targets to keep 95 % of the disks in the standby mode, while ensuring reasonable performance. Therefore, the flash memory is exploited in three ways. First, it is used to buffer writes when the disk is spun down. Second, data signatures are stored in the flash memory such that interdisk data verification can be performed even when the disk is inactive. Interdisk redundancy resembles RAID-5. Algebraic signatures are used such that the signatures exhibit the same relationships as the underlying data. This means the signatures, instead of the data, can be used for interdisk data verification. Third, metadata are stored in the flash memory, again to increase the average disk idle time.

5.3.4. Solid-State Disks.

SSD Power Consumption. A solid-state disk (SSD) is a storage device composed of NAND flash memory devices (see Section 5.3.3) that emulates a hard disk drive. Figure 11 illustrates the architecture of an SSD, which consists of a host interface (left out of the figure), controller, DRAM, multiplexer, and NAND flash memory chips. The SSD of Figure 11 exhibits two *channels* (NAND flash memory packages) and two *ways* (NAND flash memory dies or chips) per channel. The channels can be accessed in parallel. The SSD controller is composed of a processor, a flash controller (one for every channel), a buffer (DRAM) controller, internal memory (SRAM), an ECC (error correcting code) module, and the flash translation layer (FTL) [Yoo et al. 2011]. The FTL emulates a disk drive by exposing the flash memory as an array of logical sectors by means of *address translation*. It converts logical-sector reads and writes requested by the file system into flash-memory, physical-page reads and writes and block erasures. The FTL hides the write-after-erase complexity of flash memory by means of *garbage collection*. When a page needs to be modified, the new data are written to a free page and the old page is invalidated. When the number of invalid pages of a block exceeds a threshold, the garbage collector copies the remaining valid pages to free pages and then reclaims the invalid pages by erasing the block. In addition, the FTL provides *wear-leveling*, which targets an even distribution of the erasures across all of the blocks to cope with the limited write endurance of NAND flash memory.

Because an SSD has no mechanical parts and is based on low-power NAND flash memory, it is more energy efficient than a regular disk drive [Park et al. 2009]. An SSD does not require *spin* power nor *seek* power because it is composed of electronic components only. Like a regular disk drive, an SSD does also require *control* power when it is idle. When the SSD processes an I/O request, the required control power is higher. Because of the flash translation layer, the SSD controller typically embeds a more powerful processor than an HDD controller. In addition, the DRAM buffer may need to be relatively large, especially for a page-mapping FTL, which maps any logical page to any physical page. Therefore, the idle power of an SSD may not be all that much smaller than that of an HDD [Seo et al. 2008].

An SSD exhibits a smaller response time than an HDD because serving an I/O request requires neither a seek nor a disk rotation. Typically, an SSD also has a higher throughput than an HDD because I/O requests may be addressed by multiple NAND flash memory devices in parallel by means of data striping. However, the instantaneous power consumption increases linearly with the number of *ways* simultaneously active. Therefore, Yoo et al. [2011] propose making the maximum tolerable instantaneous power a configuration parameter of the solid-state drive such that the host can control the power/performance trade-off.

An SSD has a much better performance than a traditional disk drive, except for random writes. In addition, random writes cost more energy because of the write-after-erase complexity of NAND flash memory. Therefore, an SSD may be less

energy-efficient than an HDD for workloads dominated by random writes. It is also worth noting that SSDs provide less capacity and cost more.

Solid-state disks may also provide a standby mode in which electronic components are turned off to enable DPM [Park et al. 2009]. The power-state transition costs are much lower for an SSD than an HDD because there's no disk that needs to spin down and spin up. As a consequence, the break-even time is of the order of milliseconds rather than seconds. Therefore, SSDs can save energy by applying DPM, even for an enterprise workload characterized by short idle times. In addition, similar to DRPM for HDDs, dynamic voltage/frequency scaling (DVFS) allows an SSD to operate at different power/performance levels according to the fluctuating workload [Lee and Kim 2010].

Exploiting Solid-State Disks for Energy-Efficient Enterprise Storage. Deng [2011] suggests combining SSDs and conventional disks in a storage cluster and concentrating the popular data on the SSDs. This proposal resembles the design of Lightning, described in Section 4.4.3. Guerra et al. [2011] propose using a multitier storage system composed of high-performance disks (e.g., SAS), low-power solid-state disks providing a superior random-access rate, and low-cost high-capacity SATA disks. Their system, called EDT (*extent-based dynamic tiering*), is composed of a *configuration adviser* (CA) and a *dynamic tier manager* (DTM). The CA calculates the number of disks for every tier that minimizes the acquisition cost while providing adequate capacity and performance to support a given workload taking into account its fluctuations over time. The DTM periodically migrates data extents across the tiers to address changes in the extents' performance requirements while minimizing power consumption. As opposed to the workload-consolidation techniques described in Section 4.3, DTM consolidates the workload on a minimum number of disks by data migration rather than replication. As opposed to the multitier systems described in Section 4.4.3, DTM only spins down unused disks to avoid performance impact.

Lee et al. [2008] propose equipping a RAID-5 with an SSD-based cache, which allows for buffering writes and serving reads of most-recently used data without spinning disks up (cf. § 4.1.1). Using an SSD as a cache is similar as using flash memory as a cache (cf. § 4.1.2). As a cache, an SSD is better than a hard disk because an SSD provides more performance per Watt [Narayanan et al. 2009]. Because the SSD is based on nonvolatile flash memory, it's more dependable than RAM, especially for buffering writes (cf. § 4.1.1).

Narayanan et al. [2009] investigate to which extent it is currently economical to transition enterprise storage to SSDs. They argue that this depends on the throughput and capacity required by the specific workload. On the one hand, an SSD typically offers a higher throughput per Watt than a disk drive. On the other hand, an SSD has a similar capacity per Watt as an enterprise disk. Moreover, a low-power disk drive offers more capacity per Watt than a typical SSD. Thus, to minimize power consumption, SSDs should be used for load-intensive data, not for size-intensive data. At current price levels, Narayanan et al. claim that the energy savings are still outweighed by the hardware costs. Nevertheless, Caulfield et al. [2009] present *Gordon*, a system architecture for massively-parallel, data-centric computation that relies fully on solid-state drives.

6. CONCLUSIONS

To conclude, we zoom out from the details of the specific techniques to the level of the research domain as a whole.

Storage Stack. We classify, in Table II, all of the existing power-reduction techniques for data-center storage according to the storage-stack layer they were originally targeted at. The more specific a technique, the more it is tied to a specific layer of the

Table II. Classification of Power-Reduction Techniques According to the Storage-Stack Layer to Which They are Applied

Storage-stack Layer	Techniques
Storage-server cluster	Rabbit, Sierra, GreenHDFS, Lightning, GreenFS, WO, FREP, SRCMap, AN, CD, DDE, EDT, Gordon
RAID	EERAID, RIMAC, eRAID, DG, PARAID, Hibernator, SEA, PA code, Pergamum
JBOD (non-RAID)	DRPM, MAID, PDC, DIV, NomadFS, (PRE-)BUD, PA/PB-LRU, LFS, SDP, OE ME
Single disk	TPM, cache, PA prefetch, PA buffer, OE ME, ISRA, FS^2, HFS, SFS, EEFS, Coop-I/O, laptop, hybrid, SSD, multiactuator, compression

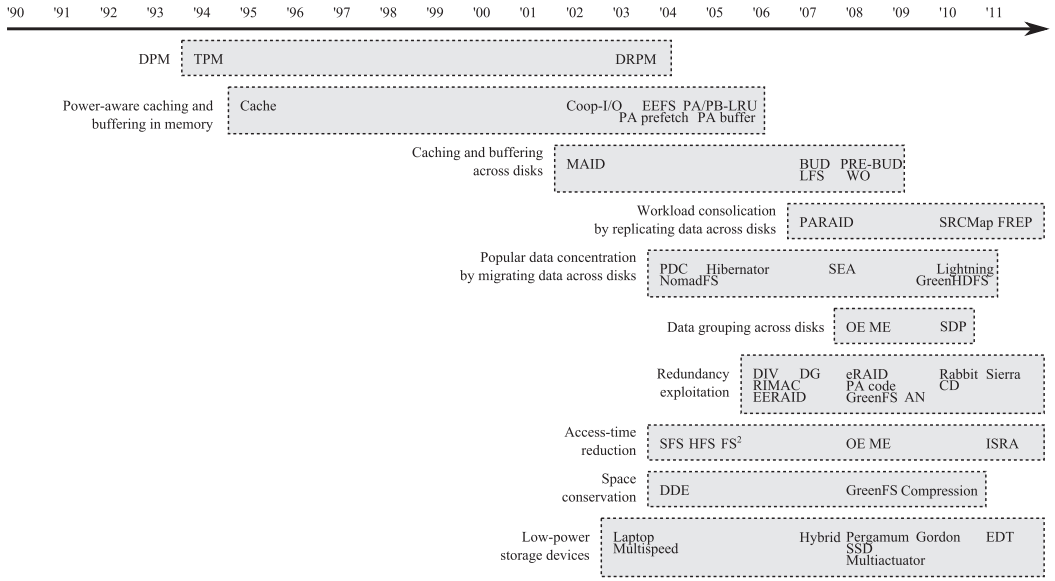


Fig. 12. Power-reduction techniques mapped to a timeline.

storage stack. Conversely, the more general, the higher the likelihood it can be applied to different layers. For example, PDC is a general technique that can be applied at the level of an individual disk (e.g., HFS), a RAID (e.g., Hibernator), and a distributed file system (e.g., GreenHDFS). A general technique may be the basis for a whole class of more specific, elaborated techniques, as is the case for PDC (cf. Table I).

Timeline. Figure 12 maps all of the power-reduction techniques, per class, on a timeline. Typically, a more general technique is conceived first, in the context of an individual disk or JBOD (just a bunch of disks). In a second stage, the technique is combined with or applied to RAID. In a third stage, the technique is integrated in a distributed file system on a cluster of storage servers. For example, DIV was introduced as a general technique for a JBOD in 2006. From 2006 to 2008, DIV was applied to RAID (e.g., eRAID), and from 2009 onwards, it was also used to build energy-aware DFSs (e.g., Rabbit). We envision a fourth stage with a focus on the design of new energy-efficient distributed file systems that combines techniques from as many different classes as possible to maximize power savings. Combining power-reduction techniques will give rise to new research challenges. For example, data deduplication tends to spread out disk accesses across disks, which impedes DPM.

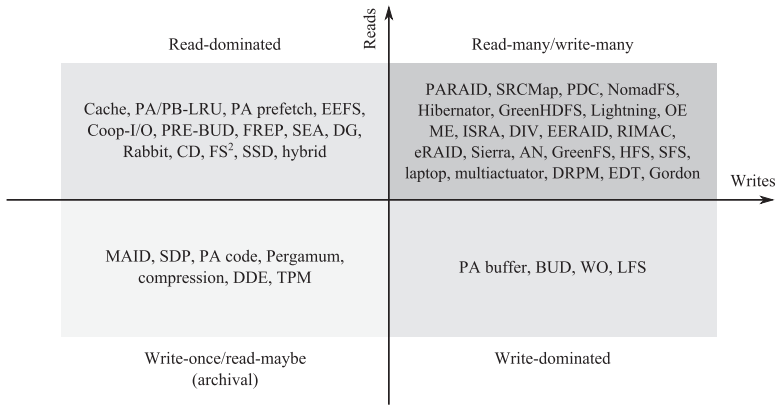


Fig. 13. Power-reduction techniques classified according to targeted workload.

Table III. Classification of Power-Reduction Techniques According to Their Impact on Performance

Impact	Cause	Techniques
Increased mean and max response time	Disk spin-up required to serve I/O request if disk is spun down	TPM, cache (with TPM), MAID, Lightning, GreenHDFS, Pergamum, PA/PB-LRU, NomadFS, PDC, SDP, hybrid
Increased mean and max response time	Fewer disk spin-ups required to serve I/O requests than for plain TPM	WO, LFS, PA buffer, PA prefetch, Coop-I/O, (PRE-)BUD, PA code
Increased mean response time	Disk spin-ups only for increasing bandwidth to accomodate higher load	PARAID, FREP, Sierra, Rabbit, EERAID, RIMAC, eRAID, GreenFS, DIV, DG, SRCMap, AN, CD
Increased mean response time	Only disk *speed*-ups for increasing bandwidth but requests served at lower speed under lighter load	laptop, DRPM, Hibernator, SEA
Increased mean response time	Computational overhead	compression, DDE
Reduced mean response time	Reduced seek and rotational latency	FS^2, HFS, SFS, EEFS, OE ME, ISRA, multiactuator
Reduced mean response time	Substitution of disk access by DRAM or SSD access	cache (without TPM), SSD, EDT, Gordon

Workload. Many of the existing power-reduction techniques target a specific type of workload. Figure 13 maps every technique into one of four quadrants according to the targeted read and write request arrival rate. The first quadrant corresponds to a read-many/write-many workload, the second quadrant to a read-dominated workload, the third quadrant to a write-once/read-maybe (archival) workload, and the fourth quadrant to a write-dominated workload. Since the workload may vary over time, a combination of techniques is generally required. For example, SRCMap, which replicates data to facilitate workload consolidation, integrates write off-loading to cope with workloads that are not read-dominated.

Finally, we classify the power-reduction techniques according to their impact on performance, dependability, and capacity.

Performance. Table III classifies the power-reduction techniques according to their impact on performance. Traditional power management (threshold-based disk spin-down policies) increases the response time of a disk from milliseconds to up to tens of seconds when the disk is spun down. The negative performance impact of TPM limits

its application to backup and archival storage systems. TPM-enabling techniques reduce the number of disk accesses or cluster disk accesses over time and across disks such that the length of disk idle periods increases. As a consequence, disks may exhibit longer and/or more standby periods such that more energy is saved. When the number of standby periods increases, the number of disk spin-ups increases with a negative impact on performance as a result. When their length increases, standby periods may join such that the number of disk spin-ups actually decreases with a positive impact on performance as a result. Some TPM-enabling techniques only prolong standby periods. Such techniques enable TPM exclusively by avoiding spinning up a standby disk, for example, by buffering writes to standby disks, delaying or aborting reads to standby disks, transforming reads to standby disks to reads to spinning disks (e.g., power-aware coding), and prefetching from spinning disks. Other TPM-enabling techniques mitigate or worsen the negative performance impact of TPM depending on the workload. Moreover, techniques that concentrate disk accesses over time and across disks increase the risk of disk contention. However, this risk can be mitigated by limiting the concentration of the workload. For example, PDC ensures that disks storing popular data are not overloaded. Because of TPM's negative performance impact, multizoned distributed file systems, such as GreenHDFS and Lightning, only apply TPM in the *cold* zone (see § 4.4.3).

Because of the limited applicability of threshold-based disk spin-down policies, other power-reduction techniques are based on *load-directed* (LD) power control policies, that is, the power-state of the disks is periodically adapted according to workload fluctuations. Such techniques ensure that at least one replica of the data is available on active disks such that an I/O request is never delayed because of a disk spin-up. Such availability can be achieved by replicating data of inactive disks on active disks (cf. § 4.3) or by spinning down only disks containing redundant data (cf. § 4.6). It is worth noting that SRCMap only replicates the working set of the spun-down disks. Therefore, a disk spin-up is still required for fewer than 0.003 % of all read requests for the workload under consideration [Verma et al. 2010]. Techniques based on LD power control limit the response-time increase or throughput reduction such that a predetermined service-level agreement is respected. Therefore, such techniques may be applied to primary storage systems. In addition, we make a distinction between techniques based on conventional disks that scale throughput by activating more or fewer disks (so-called *gear-scheduling*) and techniques based on multispeed disks (cf. § 4.4.2), which scale throughput by adapting the disk speed.

Space-conservation techniques typically incur a computational overhead, which has a negative impact on performance. Access-time reduction techniques save power while increasing performance. New energy-efficient disk drive architectures, such as multiactuator disks and SSDs, also improve performance. However, hybrid disks may incur disk spin-up delays because such disks implement threshold-based power control. Laptop disks operate at a lower speed such that their response time is larger and throughput smaller. However, in Section 5.3.1, we explain how the negative impact on throughput can be eliminated.

Dependability. As far as dependability is concerned (Table IV), DPM tends to decrease disk reliability because of the wear caused by every disk spin-down. A server-class disk typically has a duty-cycle rating of 50,000 and an expected lifetime of five years. As a consequence, such a disk can sustain a disk spin-down frequency of about one per hour. We classify the DPM-based power-reduction techniques according to the expected frequency of disk spin-downs similar as for the performance-impact analysis. To limit the negative impact of disk spin-downs on dependability (and performance), coarse-grained load-directed power control is preferable to threshold-based

Table IV. Classification of Power-Reduction Techniques According to Their Impact on Dependability

Impact	Cause	Techniques
Reduced disk reliability	Disk spin-up required to serve I/O request if disk is spun down	TPM, cache, MAID, Lightning, GreenHDFS, Pergamum, PA/PB-LRU, NomadFS, PDC, SDP, hybrid
Reduced disk reliability	Fewer disk spin-ups required to serve I/O requests than for plain TPM	WO, LFS, PA buffer, PA prefetch, Coop-I/O, (PRE-)BUD, PA code
Limited reduction of disk reliability	Disk spin-ups (or speed-ups) only for increasing bandwidth to accomodate higher load	PARAID, FREP, Sierra, Rabbit, EERAID, RIMAC, eRAID, GreenFS, DIV, DG, SRCMap, AN, CD, DRPM, Hibernator, SEA
Reduced data reliability	Volatile memory used for buffering	PA buffer
Limited reduction of data reliability	Number of erase cycles of NAND flash memory may exceed erase-cycle rating for write-intensive workload	SSD, Lightning, EDT, Gordon, hybrid, Pergamum, GreenFS
No impact		FS^2, ISRA, HFS, SFS, OE ME, EEFS, multiactuator, laptop, compression, DDE

disk spin-down policies. For example, SRCMap gear-shifts on a time scale of hours to avoid negative impact on the disk lifetime. Ideally, dynamic *power* management transforms into dynamic *data* management. Some of the DPM-based techniques, such as PARAID and GreenFS, include a disk spin-down throttling mechanism, which ensures that the duty-cycle rating is not exceeded in the course of the normal disk lifetime. Pergamum adds interdisk redundancy to overcome the increased risk of disk failures, which results in a mean time to data loss of 1,400 years. GreenHDFS, which enables DPM only in the so-called cold zone, exhibits an ignorable risk of exceeding the duty-cycle rating during a five-year disk lifetime for the workload considered [Kaushik and Bhandarkar 2010].

Buffering in DRAM negatively impacts reliability because data may be lost if the power supply is interrupted. However, this reliability cost may be avoided by buffering in nonvolatile RAM, battery-backed or based on NAND flash memory. The application of NAND flash memory to enterprise storage systems is not completely risk-free as far as dependability is concerned because of its limited program/erase endurance. For example, Bisson et al. [2007] propose write-throttling to avoid failure of the NAND flash memory buffer (NVCache) integrated in a hybrid disk. GreenFS applies single-level cell (SLC) NAND flash memory technology, which exhibits a ten-times-higher write endurance than multilevel cell (MLC) NAND flash memory, for write-intensive workloads.

Finally, when replacing a server-class disk with multiple laptop disks to avoid a throughput reduction, Carrera et al. [2003] suggest organizing the laptop disks as a RAID-1 or RAID-5 so as not to jeopardize reliability.

Note that power-reduction techniques that combine the application of NAND flash memory with disk power management appear twice in Table IV.

Capacity. When it comes to capacity (Table V), classes of DPM-enabling techniques based on data replication, such as caching in memory and on disk and workload consolidation, trade power for storage space. A server-class disk is replaced by an array of laptop disks organized as a RAID-1 or RAID-5, which adds redundant data to safeguard data reliability. Power-aware codes withstand fewer disk failures but require fewer active disks to reconstruct data from a standby disk. Thus, the redundant data added by a power-aware code is partially exploited for saving power rather than increasing reliability. Auxiliary nodes add redundant data to keep all data blocks available when

Table V. Classification of Power-Reduction Techniques According to Their Impact on Capacity

Impact	Cause	Techniques
Fewer disks required	Increased capacity utilization	multiactuator
Less space required	Redundancy elimination	compression, DDE
No impact		DIV, EERAID, RIMAC, eRAID, DG, Sierra, Rabbit, GreenFS, CD, HFS, SFS, SSD, LFS, EEFS, Coop-I/O, TPM, DRPM, SDP, ISRA, BUD, PA buffer, WO, Gordon, EDT, Pergamum
More space required	Working set replication	MAID, PRE-BUD, SRCMap, cache, FS^2, PA prefetch, PA/PB-LRU, hybrid
Much more space required	Redundancy addition (more than working set)	PARAID, FREP, laptop, PA code, AN, OE ME
More disks required	Decreased capacity utilization to avoid disk contention	PDC, NomadFS, Hibernator, SEA, Lightning, GreenHDFS, laptop

applying DIV without changing the data layout of an existing distributed file system such as HDFS. OE ME groups data but allows overlap, which implies data replication. Surprisingly, techniques based on popular data concentration, which only *migrate* data, may incur a capacity loss as well because the disks that store the most popular data cannot be filled completely to avoid disk contention. We classify laptop disks also in this category since data needs to be striped over multiple laptop disks to match the throughput of a single server-class disk.

On the other hand, space-conservation techniques may contribute to power savings, especially in the case of archival storage, but more research is required to fathom the trade-offs between power and space conservation. Multiactuator disks facilitate a consolidation of the workload on fewer disks.

Closing Remarks. We envision the ultimate power-aware enterprise storage system integrating the different classes of power-reduction techniques, covering all of the storage-stack layers and addressing diverse workloads. Our envisioned system offers a flexible trade-off between power consumption, performance, capacity, and dependability. The authors hope, with this survey, to set off a new wave of synthesis-oriented research in the domain of power-aware storage systems in support of a sustainable growth of cloud storage.

ACKNOWLEDGMENTS

The authors would like to thank Fabio Pianese, Davy Preuveneers, and the anonymous reviewers for making numerous valuable suggestions for improvement.

REFERENCES

ALLALOUF, M., ARBITMAN, Y., FACTOR, M., KAT, R. I., METH, K., AND NAOR, D. 2009. Storage modeling for power estimation. In *Proceedings of the 2nd Israeli Experimental Systems Conference (SYSTOR'09)*. 3:1–3:10.

AMUR, H., CIPAR, J., GUPTA, V., GANGER, G. R., KOZUCH, M. A., AND SCHWAN, K. 2010. Robust and flexible power-proportional storage. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC'10)*. 217–228.

APPLE. 2004. Hfs plus volume format. Tech. rep. TN1150, Apple, Cupertino, CA, March. `http://developer.apple.com/technotes/tn/tn1150.html`.

BARROSO, L. A. AND HÖLZLE, U. 2007. The case for energy-proportional computing. *Computer 40*, 33–37.

BENINI, L. AND MICHELI, G. D. 2000. System-level power optimization: Techniques and tools. *ACM Trans. Des. Autom. Electron. Syst. 5*, 115–192.

Bisson, T., Brandt, S., and Long, D. 2007. A hybrid disk-aware spin-down algorithm with i/o subsystem support. In *Proceedings of the 26th IEEE International Performance Computing and Communications Conference (IPCCC'07)*. IEEE, Los Alamitos, CA, 236–245.

Bisson, T., Brandt, S. A., and Long, D. D. E. 2006. Nvcache: Increasing the effectiveness of disk spin-down algorithms with caching. In *Proceedings of the 14th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'06)*. IEEE Computer Society, Washington, DC, 422–432.

Carrera, E. V., Pinheiro, E., and Bianchini, R. 2003. Conserving disk energy in network servers. In *Proceedings of the 17th Annual International Conference on Supercomputing (ICS'03)*. 86–97.

Caulfield, A. M., Grupp, L. M., and Swanson, S. 2009. Gordon: Using flash memory to build fast, power-efficient clusters for data-intensive applications. *SIGPLAN Not. 44*, 217–228.

Chrobak, M. 2010. Sigact news online algorithms column 17. *SIGACT News 41*, 114–121.

Colarelli, D. and Grunwald, D. 2002. Massive arrays of idle disks for storage archives. In *Proceedings of the ACM/IEEE Conference on Supercomputing (Supercomputing'02)*. 1–11.

Deng, Y. 2011. What is the future of disk drives, death or rebirth? *ACM Comput. Surv. 43*, 23:1–23:27.

Douglis, F., Krishnan, P., and Bershad, B. N. 1995. Adaptive disk spin-down policies for mobile computers. In *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*. 121–137.

Douglis, F., Krishnan, P., and Marsh, B. 1994. Thwarting the power-hungry disk. In *Proceedings of the USENIX Winter Technical Conference*. 23–23.

Essary, D. and Amer, A. 2008. Predictive data grouping: Defining the bounds of energy and latency reduction through predictive data grouping and replication. *Trans. Storage 4*, 2:1–2:23.

Freitas, R. F. and Wilcke, W. W. 2008. Storage-class memory: The next storage system technology. *IBM J. Res. Dev. 52*, 439–447.

Ganesh, L., Weatherspoon, H., Balakrishnan, M., and Birman, K. 2007. Optimizing power consumption in large scale storage systems. In *Proceedings of the 11th USENIX Workshop on Hot Topics in Operating Systems*. 9:1–9:6.

Golding, R., Bosch, P., Staelin, C., Sullivan, T., and Wilkes, J. 1995. Idleness is not sloth. In *Proceedings of the USENIX Technical Conference (TCON'95)*.

Greenan, K. M., Long, D. D. E., Miller, E. L., Schwarz, S. J., T. J. E., and Wylie, J. J. 2008. A spin-up saved is energy earned: achieving power-efficient, erasure-coded storage. In *Proceedings of the 4th conference on Hot topics in system dependability*. HotDep'08. 4–4.

Grupp, L. M., Caulfield, A. M., Coburn, J., Swanson, S., Yaakobi, E., Siegel, P. H., and Wolf, J. K. 2009. Characterizing flash memory: Anomalies, observations, and applications. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'42)*. ACM, New York, NY, 24–33.

Guerra, J., Belluomini, W., Glider, J., Gupta, K., and Pucha, H. 2010. Energy proportionality for storage: Impact and feasibility. *SIGOPS Oper. Syst. Rev. 44*, 35–39.

Guerra, J., Pucha, H., Glider, J., Belluomini, W., and Rangaswami, R. 2011. Cost effective storage using extent based dynamic tiering. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST'11)*. USENIX Association, Berkeley, CA. 20.

Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., and Franke, H. 2003a. Drpm: Dynamic speed control for power management in server class disks. *SIGARCH Comput. Archit. News 31*, 169–181.

Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., and Franke, H. 2003b. Reducing disk power consumption in servers with drpm. *Computer 36*, 59–66.

Gurumurthi, S., Sivasubramaniam, A., and Natarajan, V. K. 2005. Disk drive roadmap from the thermal perspective: A case for dynamic thermal management. *SIGARCH Comput. Archit. News 33*, 38–49.

Gurumurthi, S., Zhang, J., Sivasubramaniam, A., Kandemir, M., Franke, H., Vijaykrishnan, N., and Irwin, M. J. 2003c. Interplay of energy and performance for disk arrays running transaction processing workloads. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*. 123–132.

Harnik, D., Naor, D., and Segall, I. 2009. Low power mode in cloud storage systems. In *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing (IPDPS'09)*. 1–8.

Hong, B., Plantenberg, D., Long, D. D. E., and Sivan-Zimet, M. 2004. Duplicate data elimination in a san file system. In *Proceedings of the 21st IEEE Conference on Mass Storage Systems and Technologies (MSST'04)*. IEEE Computer Society, Los Alamitos, CA, 301–314.

Huang, H., Hung, W., and Shin, K. G. 2005. Fs2: Dynamic data replication in free disk space for improving disk performance and energy consumption. *SIGOPS Oper. Syst. Rev. 39*, 263–276.

Irani, S., Singh, G., Shukla, S. K., and Gupta, R. K. 2005. An overview of the competitive and adversarial approaches to designing dynamic power management strategies. *IEEE Trans. Very Large Scale Integr. Syst. 13*, 1349–1361.

Joukov, N. and Sipek, J. 2008. Greenfs: Making enterprise computers greener by protecting them better. *SIGOPS Oper. Syst. Rev. 42*, 69–80.

Kaushik, R., Abdelzaher, T., Egashira, R., and Nahrstedt, K. 2011. Predictive data and energy management in greenhdfs. In *Proceedings of the 2nd International Green Computing Conference (IGCC'11)*. IEEE Computer Society, Washington, DC, 1–9.

Kaushik, R. T. and Bhandarkar, M. 2010. Greenhdfs: Towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster. In *Proceedings of the International Conference on Power Aware Computing and Systems (HotPower'10)*. 1–9.

Kaushik, R. T., Cherkasova, L., Campbell, R., and Nahrstedt, K. 2010. Lightning: Self-adaptive, energy-conserving, multi-zoned, commodity green cloud storage system. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC'10)*. 332–335.

Kim, J. and Rotem, D. 2011. Energy proportionality for disk storage using replication. In *Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT'11)*. 81–92.

Kothiyal, R., Tarasov, V., Sehgal, P., and Zadok, E. 2009. Energy and performance evaluation of lossless file data compression on server systems. In *Proceedings of the 2nd Israeli Experimental Systems Conference (SYSTOR'09)*. 4:1–4:12.

Kulkarni, P., Douglis, F., LaVoie, J., and Tracey, J. M. 2004. Redundancy elimination within large collections of files. In *Proceedings of the USENIX Annual Technical Conference (ATEC'04)*. 5.

Lang, W., Patel, J. M., and Naughton, J. F. 2010. On energy management, load balancing and replication. *SIGMOD Rec. 38*, 35–42.

Lee, H. J., Lee, K. H., and Noh, S. H. 2008. Augmenting raid with an ssd for energy relief. In *Proceedings of the Conference on Power Aware Computing and Systems (HotPower'08)*. 12.

Lee, S. and Kim, J. 2010. Using dynamic voltage scaling for energy-efficient flash-based storage devices. In *Proceedings of the International SoC Design Conference (ISOCC'10)*. IEEE Computer Society, Washington, DC, 63–66.

Leverich, J. and Kozyrakis, C. 2010. On the energy (in)efficiency of hadoop clusters. *SIGOPS Oper. Syst. Rev. 44*, 61–65.

Li, D. and Wang, J. 2004a. Eeraid: Energy efficient redundant and inexpensive disk array. In *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop (EW'11)*.

Li, D. and Wang, J. 2004b. A performance-oriented energy efficient file system. In *Proceedings of the International Workshop on Storage Network Architecture and Parallel I/Os*. 58–65.

Liao, X.-L., Bai, S., Wang, Y.-P., and Hu, S.-M. 2011. Isra-based grouping: A disk reorganization approach for disk energy conservation and disk performance enhancement. *IEEE Tran. Computers 60,* 2, 292–304.

Lu, L., Varman, P., and Wang, J. 2007. Diskgroup: Energy efficient disk layout for raid1 systems. In *Proceedings of the International Conference on Networking, Architecture, and Storage (NAS'07)*. IEEE Computer Society, Los Alamitos, CA, 233–242.

Lu, Y.-H., Chung, E.-Y., Šimunić, T., Benini, L., and De Micheli, G. 2000. Quantitative comparison of power management algorithms. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'00)*. 20–26.

Manzanares, A., Bellam, K., and Qin, X. 2008. A prefetching scheme for energy conservation in parallel disk systems. In *Proceedings of the International Symposium on Parallel and Distributed Processing (IPDPS'08)*. IEEE Computer Society, Washington, DC, 1–5.

Mohan, V., Gurumurthi, S., and Stan, M. 2010. Flashpower: A detailed power model for nand flash memory. In *Proceedings of the Conference on Design, Automation, and Test in Europe (DATE'10)*. IEEE Computer Society, Washington, DC, 502–507.

Narayanan, D., Donnelly, A., and Rowstron, A. 2008. Write off-loading: Practical power management for enterprise storage. *Trans. Storage 4*, 10:1–10:23.

Narayanan, D., Thereska, E., Donnelly, A., Elnikety, S., and Rowstron, A. 2009. Migrating server storage to ssds: Analysis of tradeoffs. In *Proceedings of the 4th ACM European Conference on Computer Systems (EuroSys'09)*. 145–158.

Okada, K., Kojima, N., and Yamashita, K. 2000. A novel drive architecture of hdd: "Multimode hard disc drive". In *Proceedings of the 18th International Conference on Consumer Electronics (ICCE'00)*. IEEE, Los Alamitos, CA, 92–93.

Otoo, E., Rotem, D., and Tsao, S. C. 2009. Analysis of trade-off between power saving and response time in disk storage systems. In *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing*. 1–8.

PAPATHANASIOU, A. E. AND SCOTT, M. L. 2004a. Energy efficient prefetching and caching. In *Proceedings of the USENIX Annual Technical Conference (ATEC'04)*. 22.

PAPATHANASIOU, A. E. AND SCOTT, M. L. 2004b. Power-efficient server-class performance from arrays of laptop disks. Tech. rep. 837, University of Rochester, Rochester, NY. May.

PARK, J., YOO, S., LEE, S., AND PARK, C. 2009. Power modeling of solid state disk for dynamic power management policy design in embedded systems. In *Proceedings of the 7th IFIP WG 10.2 International Workshop on Software Technologies for Embedded and Ubiquitous Systems (SEUS'09)*. Springer-Verlag, Berlin, Heidelberg, 24–35.

PINHEIRO, E. AND BIANCHINI, R. 2004. Energy conservation techniques for disk array-based servers. In *Proceedings of the 18th Annual International Conference on Supercomputing (ICS'04)*. 68–78.

PINHEIRO, E., BIANCHINI, R., AND DUBNICKI, C. 2006. Exploiting redundancy to conserve energy in storage systems. *SIGMETRICS Perform. Eval. Rev. 34*, 15–26.

QUINLAN, S. AND DORWARD, S. 2002. Venti: A new approach to archival data storage. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST'02)*.

SANKAR, S., GURUMURTHI, S., AND STAN, M. R. 2008. Intra-disk parallelism: An idea whose time has come. *SIGARCH Comput. Archit. News 36*, 303–314.

SEAGATE 2000. Seagates sound barrier technology (sbt)—the art of quiet disc drives. `http://www.seagate.com/docs/pdf/whitepaper/sound\_barrier.pdf`.

SEAGATE 2010. Seagate powerchoice technology provides unprecedented hard drive power savings and flexibility. `http://www.seagate.com/docs/pdf/en-GB/whitepaper/tp608\_powerchoice\_tech\_provides.pdf`.

SEHGAL, P., TARASOV, V., AND ZADOK, E. 2010. Optimizing energy and performance for server-class file system workloads. *Trans. Storage 6*, 10:1–10:31.

SEO, E., PARK, S. Y., AND URGAONKAR, B. 2008. Empirical analysis on energy efficiency of flash-based ssds. In *Proceedings of the conference on Power Aware Computing and Systems (HotPower'08)*. USENIX Association, Berkeley, CA, 17–17.

SHVACHKO, K., KUANG, H., RADIA, S., AND CHANSLER, R. 2010. The hadoop distributed file system. In *Proceedings of the 26th IEEE Symposium on Massive Storage Systems and Technologies (MSST'10)*. 1–10.

STAELIN, C. AND GARCIA-MOLINA, H. 1991. Smart filesystems. In *Proceedings of the USENIX Winter Technical Conference (USENIX Winter'91)*. USENIX Association, Berkeley, CA, 45–52.

STORER, M. W., GREENAN, K. M., MILLER, E. L., AND VORUGANTI, K. 2008. Pergamum: replacing tape with energy efficient, reliable, disk-based archival storage. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08)*. 1:1–1:16.

THERESKA, E., DONNELLY, A., AND NARAYANAN, D. 2011. Sierra: Practical power-proportionality for data center storage. In *Proceedings of the 6th Conference on Computer Systems (EuroSys'11)*. 169–182.

VENKATACHALAM, V. AND FRANZ, M. 2005. Power reduction techniques for microprocessor systems. *ACM Comput. Surv. 37*, 195–237.

VERMA, A., KOLLER, R., USECHE, L., AND RANGASWAMI, R. 2010. Srcmap: Energy proportional storage using dynamic consolidation. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST'10)*.

WANG, J., ZHU, H., AND LI, D. 2008. eraid: Conserving energy in conventional disk-based raid system. *IEEE Trans. Comput. 57*, 359–374.

WEDDLE, C., OLDHAM, M., QIAN, J., WANG, A.-I. A., REIHER, P., AND KUENNING, G. 2007. Paraid: A gear-shifting power-aware raid. *Trans. Storage 3*.

WEISSEL, A., BEUTEL, B., AND BELLOSA, F. 2002. Cooperative i/o: A novel i/o semantics for energy-aware applications. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*. 117–129.

WILDANI, A. AND MILLER, E. 2010. Semantic data placement for power management in archival storage. In *Proceedings of the 5th Petascale Data Storage Workshop (PDSW'10)*. IEEE Computer Society, Los Alamitos, CA, 1–5.

XIE, T. 2008. Sea: A striping-based energy-aware strategy for data placement in raid-structured storage systems. *IEEE Trans. Comput. 57*, 748–761.

YAO, X. AND WANG, J. 2006. Rimac: A novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems. *SIGOPS Oper. Syst. Rev. 40*, 249–262.

YOO, B., WON, Y., CHOI, J., YOON, S., CHO, S., AND KANG, S. 2011. Ssd characterization: From energy consumption's perspective. In *Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'11)*. USENIX Association, Berkeley, CA, 3.

ZHU, Q., CHEN, Z., TAN, L., ZHOU, Y., KEETON, K., AND WILKES, J. 2005. Hibernator: Helping disk arrays sleep through the winter. *SIGOPS Oper. Syst. Rev. 39*, 177–190.

ZHU, Q., DAVID, F. M., DEVARAJ, C. F., LI, Z., ZHOU, Y., AND CAO, P. 2004. Reducing energy consumption of disk storage using power-aware cache management. In *Proceedings of the 10th International Symposium on High Performance Computer Architecture (HPCA'04)*. 118–129.

ZHU, Q. AND ZHOU, Y. 2005. Power-aware storage cache management. *IEEE Trans. Comput. 54*, 587–602.

ZONG, Z., BRIGGS, M., O'CONNOR, N., AND QIN, X. 2007. An energy-efficient framework for large-scale parallel storage systems. In *Proceedings of the International Symposium on Parallel and Distributed Processing (IPDPS'07)*. IEEE Computer Society, Washington, DC, 1–7.

Received September 2011; revised December 2011; accepted February 2012