Power-Saving Approaches and Tradeoffs for Storage Systems

BEN BUZBEE, Florida State University
WEISU WANG, Florida State University
AN-I ANDY WANG, Florida State University

Power is becoming a major concern when designing storage systems for platforms ranging from mobile devices to data centers. While many solutions exist, different solutions make very different tradeoffs between energy savings and storage performance, capacity, reliability, cost of ownership, etc. This survey walks through layers of the legacy storage stack, exploring tradeoffs made by a representative set of energy-efficient storage approaches. The survey also points out architectural implications of implementing energy-efficient solutions at various storage layers.

Categories and Subject Descriptors: **D.4.2 [Operating Systems]**: Storage Management; **D.4.8 [Operating Systems]**: Performance

General Terms: Design, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Power savings, energy efficiency, file systems, RAID, hard drives, solid-state disks, flash

1. INTRODUCTION

The U.S. Energy Information Administration [2013] reports that the average price of electricity has risen by 36% (from 7.29 to 9.9 cents per KWh) from 2001 to 2011, with no signs of breaking this trend. The rising cost of electricity, along with limited power availability (due to a fixed power infrastructure, limited battery density, etc.), has created a need for more energy-efficient system designs. While significant advances have been made to improve the energy efficiency for various system components (most notably CPUs), storage devices are lagging behind. In laptops, 4-7% of power consumption on an idle system can be attributed to storage. This number can increase beyond 10% during storage-intensive operations [Mahesri and Vardhan 2004; Kothuru et al. 2010; Somavat et al. 2010]. In data centers, storage systems can account for up to 40% of the energy consumption [Pinheiro et al. 2006; Schulz 2007].

Power consumption can also have an amplifying effect on cost. As more energy is consumed, more heat is generated due to inefficiency. This additional heat raises cooling requirements, which already account for 30-40% of a data center's total energy consumption [Fredslund et al. 2009]. Increased heat also means density of hardware must be lowered, translating into higher space requirements and the associated costs.

Even if the power-related costs were not an issue, the power itself may be in limited supply. Mobile users, for example, are constantly demanding both better performance and longer battery life (two goals that often conflict with each other). However, with the energy density of batteries already approaching that of a hand-grenade [Perry 2008], increasing the power supply to these devices is becoming a less viable option. Even for stationary machines plugged into a direct power source, power availability can still be a limiting factor. Often, for metropolitan areas, the energy-providing infrastructure is fixed (e.g., for a given building) and cannot be revamped without incurring major costs.

To address these concerns surrounding high energy consumption, the computing industry has begun a collective push toward greener hardware and software techniques. In spite of this shift, improvements to storage systems are lagging behind that of other components. For example, modern processors have been particularly aggressive when it comes to implementing power-saving techniques. In 2005, processors like the Intel® Pentium® D 840 consumed around 130 W for a

single core [Intel 2006]. Fast-forward to 2011, and processors such as the Xeon® E7-4870 are consuming 10 W per core and near zero when idle [Intel 2011]. Compare this to hard drives like the 900GB Seagate Savvio® 10K.6, which consumes 7.9 W on average during random reads and writes and 3.9 W while idle [Seagate 2012]. Compared to processors, the energy consumption of this drive has improved relatively little over the 13 W active (and 9.3 W idle) consumption of a 750 GB Seagate Barracuda 7200.10 disk earlier [Seagate 2007].

The on-going exponential trend of increased storage demand also paints a bleak future unless more energy-efficient solutions are introduced. In 2011, 1.8 zettabytes (10^{21} bytes) of data were created or replicated, and some projections show that this number is likely to continue to double every two years [Gantz and Reinsel 2011]. The growing popularity of cloud storage and social media further increases the storage demands on data centers.

Some may expect the emergence of flash storage devices could help to alleviate the ever-rising concern of energy consumption in storage media. Devices such as the Samsung SM825 use only about 1.8 W while active and 1.3 W while idle [Samsung 2011]. Unfortunately, the cost-per-gigabyte ratio of flash exists at an order of magnitude higher than that of hard disks. The aforementioned SM825 sells for $499 for the 100GB edition [Samsung 2012] (about $5 per gigabyte), whereas the 900GB Savvio can be found for $390 ($0.43 per gigabyte). This price disparity causes the adoption of flash devices to be cost-prohibitive in many situations. To illustrate, the social networking site Facebook recently reported that it was storing 100 petabytes (10^{15} bytes) of data [Chen 2012]. Using flash instead of hard disks would cost Facebook an additional $360 million. Also, since it takes more flash units to host the same storage capacity, the total power required for flash is actually more than using hard disks. However, due to their shock-resistant and energy saving properties, flash devices have become the preferred storage medium for mobile devices such as smart phones, tablets, and ultra-books. While the storage capacity of these solutions is somewhat limiting, it is often enough for the average mobile device consumer. Thus, flash devices are a reminder that whether an energy-efficient storage solution is acceptable for mainstream use depends on many factors such as the total cost of ownership, reliability, performance, capacity, and more.

Complementary to existing surveys [Llopis et al. 2013; Bostoen et al. 2013] that focus on data-center solutions, this survey will walk through a representative set of energy-efficient storage approaches (not limited to data centers) organized by where they are applied at various layers in the legacy storage stack. This organization highlights that even similar techniques (e.g., replication) applied at different layers (e.g., within a disk, across disks, or across machines) have very different architectural challenges. This survey also shows the potential compatibility and conflicts of tradeoffs for various approaches.

The rest of this survey is arranged as follows. Section 2 contains a brief overview of the legacy storage stack and the basic functionality of each layer. Sections 3-8 introduce each layer of the storage stack in more detail and survey a set of energy-saving techniques whose applications mostly affect that layer. Section 9 introduces large-scale power-saving techniques that span a multitude of devices. Section 10 summarizes the overall trends, tradeoffs, and compatibility of various techniques.

2. THE LEGACY STORAGE STACK

The legacy storage stack (Figure 2.1) refers to the layering of abstract storage components that enables the swift evolution of existing layers and makes it easy to insert additional ones. However, the choice of layer to provide energy-efficiency

features can have wide-ranging implications on aspects such as portability and applicability. This Section only briefly reviews the roles of each layer, and the implications of devising solutions at each layer will be discussed in the corresponding sections.

At the lowest level of the storage stack is the *hardware layer*, which is the physical media (disks or flash devices) used to store data. On top of the hardware layer is the *device-driver software* that handles hardware-specific communications and optimizations for the underlying storage media.

Above the device drivers there may be a software *device mapper* which maps logical addresses to a different set of physical addresses. This can include multi-device drivers (e.g. software RAID), which are designed to coordinate multiple physical devices to behave as one logical device to the above layers. For flash devices, a Flash Translation Layer (FTL) system may also exist at this layer. The FTL allows flash storage devices to be accessed like block devices (such as hard drives) for legacy compatibility.

One step higher on the storage stack is the *file system layer*, which is responsible for providing namespaces to organize the data blocks on the logical devices exposed by the multi-device layer. The file system provides a namespace for files and handles data structures stored on the storage hardware.

Finally, on the top of the storage stack, a virtual file system (VFS) provides a uniform interface for accessing a variety of file systems. This allows multiple file systems, each optimized for different workloads (multimedia, smaller files, etc.), to be added below the VFS layer without changing the way user-space applications access data. Common file system functionalities, such as caching and permission handling, are also provided by this layer.

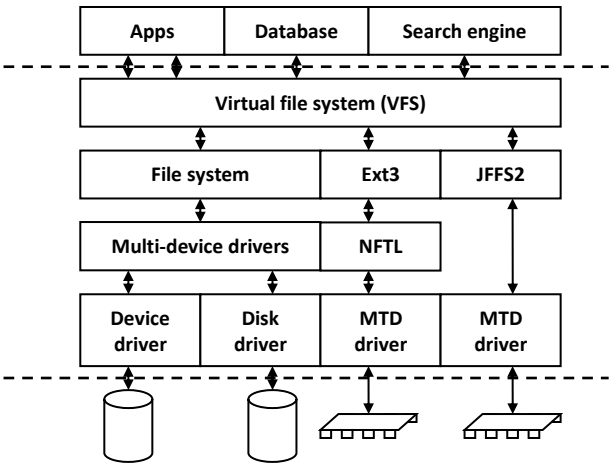

Fig. 2.1: The legacy storage stack.

3. STORAGE-HARDWARE-LEVEL POWER-SAVING TECHNIQUES

The hardware level contains the physical media (e.g., hard disks or flash devices) used for persistent storage. Improvements targeted to this level are transparent to the layers above and thus highly portable. On the other hand, due to interface limitations, higher layers may not be able to have direct and full control of energy-savings features at this layer (e.g., disabling energy-savings features at times to preserve performance characteristics).

3.1 Hard Disk Storage Background

The hard disk is the most common storage medium in use today for laptops, desktops, and data centers. To illustrate its energy tradeoffs, we briefly review its mechanics. A hard disk consists of one or more rotating magnetic platters. Each platter is divided into concentric *tracks* and sliced into arc-shaped *sectors*, containing fixed-size chunks of bytes (e.g., 512 bytes, 2KB, 4KB, etc.). A sector is accessed when the *head* (mounted on an actuator arm) is positioned over it. The time it takes to access data on a hard disk is referred to as the *disk access time* and consists of the *seek time* (the time it takes for the arm to move to the correct track), the *rotational delay* (the time it takes for the platter to rotate so that the target sector is under the disk head) and the *transfer time* (the time it takes to transfer the sectors).

Each disk has multiple *power modes* ("on" and "off," at a minimum), each of which represents a level of tradeoff between power and access time. Table 3.1.1 describes the different power modes of the Hitachi Deskstar 7K1000 [Hitachi 2007]. A hard disk also has a limited number of *power cycles* (the number of times a disk can park its heads, sometimes referred to as load/unload cycles). For example, the Hitachi Deskstar 7K1000 is rated at 50,000 power cycles and therefore may only reliably park its head 50,000 times (27 times a day for 5 years). To postpone reaching this limit, changes between these power modes must be done conservatively.

Table 3.1.1: Common hard disk power modes of Hitachi Deskstar 7K1000 [Hitachi 2007].

Mode	Description	Avg. Wattage
Active	Platters spinning, head working	11.5 W
Idle	Platters spinning, head idle	7.6 W
Spun Down	Disk spun down, head idle	3.5 W

Laptop hard drives are designed to undergo additional power cycles. Their smaller platters and lower rotational speed (measured in rotations per minutes, or RPMs) translate into a lower energy cost but also slower latency and smaller capacity. Table 3.1.2 compares a Hitachi 7K1000 1 TB desktop drive [Hitachi 2007] with a laptop hard drive, the HGST (formerly Hitachi Global Storage Technologies) Z5K500 500GB [HGST 2012].

Table 3.1.2: Comparison of laptop and desktop hard drives [Hitachi 2007; HGST 2012; ServerSupply 2013].

Model	Capacity	Price	Seek Time	RPM	Transfer Rate	Power Cycles	Power (Watts)			
							Start	Active	Idle	Low
Hitachi 7K1000	1,000 GB	$80	9 ms	7,200	37.5 MB/s	50K	24	12.3	8.4	3.7 Low RPM
Hitachi Z5K500	500 GB	$65	13 ms	5,400	125.5 MB/s	600K	4.5	1.6	1.5	0.1 Sleep

While the energy-efficiency characteristics are crucial to extend the battery life of laptops, replacing desktop drives with laptop drives for the same purpose may be a losing proposition. In the case of these two drives, to achieve the storage capacity of one 7K1000 would require two laptop disks costing $50 more in total [ServerSupply 2013]. The exchange would save about 9.1 W per hour, or about $47 of electricity (at $0.10 per KWh) over 5 years. Considering other factors, such as the increased rotational delay of slower RPM drives, other energy-saving approaches may be more suitable for desktop or server applications than replacement by laptop disks.

3.2 Flash Storage Background

Flash storage is becoming popular as an alternative to hard disk drives, especially in mobile devices. Some analysts predict that 50% of laptops will use flash-based solid-state drives by as soon as late 2013 [Kerekes 2012], and most mobile phones and tablets already utilize this technology for primary storage.

Flash stores data bits in memory cells, each cell made from floating-gate transistors. The number of electrons trapped in the floating gate can control the resulting charge level of a memory cell which is then used to represent discrete bits of data. Unlike disks, flash memory is solid state, thus making it very shock resistant and more power efficient. Both of these traits make it an attractive choice for the mobile market.

There are two primary types of flash storage, *NOR flash* and *NAND flash*. NOR flash, similar to RAM, is byte-addressable and so supports direct execution of programs without a need to first load the program into a byte-addressable medium (e.g., RAM). Compared to NAND flash, however, the popularity of NOR flash has lessened in recent years, likely due to its higher cost-per-gigabyte ratio. By contrast, NAND flash is page-addressable and must be accessed in this larger "page" granularity (rather than per-byte). As a result, programs cannot be run directly from NAND flash, but must first be loaded onto a byte-addressable medium such as RAM. Additionally, NAND flash is erased in *erase blocks*, which consist of many contiguous *flash pages*. Therefore, a single erase of any size on NAND flash will erase many pages.

Although NAND flash can achieve faster read and write access times (0.29µs for reads and 100µs for writes [Samsung 2012]) compared to disks, one major consideration of flash is that it can only write to empty pages. This means that in order to over-write a flash page (or even a single byte), the entire erase block which contains the page must first be erased. Erasing in flash can thus be orders of magnitude slower than simply reading [Inoue and Wong 2003], and if the erase block contains still-valid pages, those pages must be migrated elsewhere before erasure. To avoid the overhead of an erasure, writes to a non-empty page will often be remapped to an empty one. The original write destination is then marked invalid. The hope is that by the time a flash block must be erased for space, most of the pages it contains are either (1) empty or (2) invalid and do not need to be migrated.

Additionally, each memory cell (for both NAND and NOR flash) may only be erased a finite number of times [Inoue and Wong 2003]. The Samsung SM825, for example, can erase each memory cell approximately 17,500 times in its lifetime [Samsung 2012]. In order to increase the lifetime of a flash device, a technique known as *wear leveling* is often employed, which attempts to spread erase requests evenly over the erase blocks on a flash device by remapping a write request to a fresher block.

Techniques such as the delaying of an erase request and wear leveling need to be considered when designing energy-saving solutions. In particular, solutions that involve migrating data to create opportunities to shut down devices may lead to a significantly reduced lifespan of flash by incurring additional erases. If flash memory is employed for caching, writes should be conserved as much as possible.

The applications for which flash is well suited are still being discovered, with factors such as limited erase cycles and an order of magnitude higher cost-per-gigabyte ratio (over hard disks) slowing the adoption rate. In order to alleviate the storage cost, many techniques attempt to increase the capacity of flash devices.

These techniques include (1) increasing the density of memory cells and (2) employing multi-level cell technology.

In Single-Level Cell (SLC) flash, a bit is encoded by designating two voltage levels (e.g., one level representing the bit 0, and another representing the bit 1). In Multi-Level Cell (MLC) flash, two bits are stored in four voltage levels. These levels must be chosen to be sufficiently far apart to allow for imprecise programming and natural fluctuations in voltage. A linear increase in bit density requires an exponential increase in the number of voltage levels (i.e., n bits require 2^n voltages), which limits the potential of this approach. Another constraint is that the distance between the maximum and minimum voltage of the cells must be kept large enough to provide sufficient padding between each voltage level. Raising the maximum voltage would allow for a higher bit density with sufficient padding, however it would also result in higher power consumption.

3.3 Consolidate Drives

One simple way to decrease power consumption of the storage devices at the hardware level is to consolidate data from many small drives onto fewer, bigger hard disks. Table 3.3.1 illustrates the power consumption of a 3TB system constructed using two models of the Deskstar 7K1000 [Hitachi 2007] with varying capacities.

Table 3.3.1: Capacity comparison for the Deskstar 7K1000 [Hitachi 2007].

Capacity	Active Power	Peak Bandwidth	Active Power for a 3TB System	Aggregate Peak Bandwidth a 3TB system
750GB	11.5 W	37.5 MB/s	46 W	150 MB/s
1TB	12.3 W	37.5 MB/s	36.9 W	112.5 MB/s

Consolidating data from four 750GB disks to three 1TB disks saves about 20% of the power while all the disks are active. However, having fewer disks means fewer opportunities for parallelism, which is reflected in the 25% reduction of peak aggregate bandwidth.

3.4 Smaller Platters and Slower RPM

Having smaller platters results in a lower capacity for the disk, and a lower RPM will cause an increase in rotational delay, but each of these modifications can lead to reduced power requirements for spinning the disk at active RPM. Smaller disks still require the same hardware complexity, but it is amortized over fewer bytes, leading to a higher price-per-byte. An example is the 4GB Hitachi Microdrive [HGST 2003] (Table 3.4.1), which measures only 42.8 mm by 36.4 mm and has a maximum RPM of 3,600, which is 4.7 times as power efficient as the 7K1000 while active. However, the Microdrive has an access latency of 20ms (2.2 times as slow as the Hitachi 7K1000).

Table 3.4.1: Power consumption of a 4GB Hitachi Microdrive [HGST 2003; eSaiTech 2013; ServerSupply 2013].

Model	Capacity	Price	Seek Time	RPM	Transfer Rate	Power Cycles	Power (Watts)		
							Active	Idle	Low
Hitachi 7K1000	1,000GB	$80	9 ms	7,200	37.5 MB/s	50K	12.3	8.4	3.7 Low RPM
Hitachi Microdrive 3K4	4GB	$50	12 ms	3,600	7.2 MB/s	300K	2.6	0.14	-

Flash tends to be the more attractive option for small and power-efficient storage applications due to factors such as a lower price, greater shock resistance and improved access latency [SanDisk 2003].

3.5 Variable RPM

The traditional power mode model has several shortcomings, particularly in systems with unpredictable workloads. Once a disk has been spun down, it can take some time to bring it back up to a speed capable of servicing new requests. Spinning the platters faster than necessary also wastes power, since a slower RPM may be sufficient to meet some workload demands.

The variable RPM model attempts to make up for some of these shortcomings by offering more levels of rotational speed, often controlled by the value of a register on the disk. This allows the disk to be spun up or down to a speed appropriate for the current workload instead of simply being "idle" or "active" [Gurumurthi et al. 2003a]. The Western Digital Caviar Green disk drive [Western Digital 2012] implements a variable RPM system and uses only 30% of the power that the 1TB Hitachi 7K1000 uses on average during random reads and writes. Note that the Caviar Green numbers may also reflect additional proprietary energy-saving techniques.

Table 3.5.1: Western Digital Caviar Green energy specifications [Western Digital 2012].

Capacity	Average Active Power	Average Idle Power	Load/Unload Cycles
3TB	6.0 W	5.5 W	300,000
2TB	5.3 W	3.3 W	300,000
1TB	3.7 W	2.1 W	300,000

Variable RPM disks typically price higher than other disks of the same capacity and may result in increased latency in systems with unpredictable workloads, since the disk may choose a lower RPM setting and then immediately have to spin back up to serve a new request. Due to physical and cost constraints, these drives have not been able to penetrate the market [Guerra et al. 2010].

3.6 Hybrid Drives

Hybrid disks are traditional hard disks that contain a small amount of flash memory for caching and storing read-mostly data items. These drives attempt to take advantage of the speed and energy-efficiency of flash while still utilizing the cheap storage capacity of traditional hard disks. The true effectiveness of hybrid drives depends largely on the locality of data and the caching policies implemented by the disk controller. The Seagate Momentus® XT 750GB hard drive contains 8GB of SLC NAND flash for caching and uses a constant 7200 RPM spindle speed. Table 3.6.1 shows the average power requirements [Seagate 2011].

Table 3.6.1: Seagate Momentus® XT 750 GB energy specifications [Seagate 2011].

Startup	Read	Write	Idle (active)
6 W	3.3 W	3.2 W	1.2 W

Hybrid drives can be priced higher than other drives of the same capacity; however, the difference is often small. Poor locality or a poor caching policy could also result in a (negligible) decrease in performance and energy efficiency over other drives. Additionally, the introduction of flash means worrying about its erase cycle limits.

4. DEVICE DRIVER-LEVEL POWER-SAVING TECHNIQUES

A device driver is a piece of software tailored for a specific hardware device while providing a common interface to the upper layers of the legacy storage stack. Drivers often contain optimizations, such as built-in request reordering, to achieve higher device utilization. Device drivers can also control the disk drives directly and command them to switch power modes (or RPM in the case of variable RPM drives). Improvements at this level are highly-portable and can exert a strong level of control over the workings of the disk.

4.1 Predicting Idle Periods for Disks

One approach for power savings at the driver level is to predict periods of disk idleness to promote energy savings. The basic strategy is to spin the disk down when the predicted idle period is long enough to save more energy than that required to spin the disk back up (Figure 4.1.1). Practically, however, available power cycles and performance constraints limit how often this can be done. Disks can often take a long time to spin back up to serviceable speeds (8 seconds for the Seagate Constellation ES2 [Seagate 2010]).

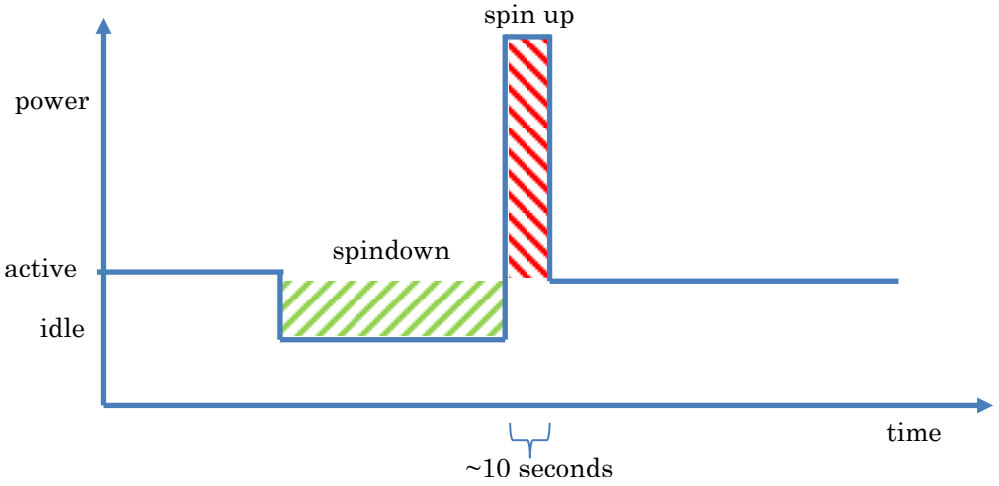

Fig. 4.1.1: Mode-switching energy tradeoff.

The most generally applicable technique is to predict an idle period and to spin down the disk, or reduce its RPM, when there have been no requests for X sequential seconds. This is the traditional threshold-based model, and can improve energy savings significantly. Some early works have achieved energy savings of nearly 50% with a one-second threshold [Douglis et al. 1994]. This approach makes no attempt to analyze load patterns and can perform very differently under different workloads.

One prediction technique [Kirshnan et al. 1995] maps the power-mode switching problem to the ski rental problem: one must decide whether to continue paying a cost for something (renting), or to pay a larger one-time cost (buying) with a goal of paying less in the end for some finite usage. For this purpose, discrete time chunks are used. At the beginning of each time slot, an algorithm must decide whether to keep the disk spinning (renting) or to spin it down (buying). The costs of these operations are defined in terms of power consumption and additional latency that

will be introduced. Through simulation and traces from a workstation, this approach can outperform the standard 5-second threshold-based prediction from 6% to 42%.

Gniady et al. [2004] propose a system named Program Counter-based Access Predictor (PCAP), which aims to better predict upcoming idle periods by examining the relationship between idle periods and associated program counters. PCAP is inspired from success of program-counter-based branch predictors used in CPU architecture. The tested implementation uses path-based prediction, which keeps track of the series of program-counter values that proceed idle periods long enough to justify a change to a lower power mode. Evaluated with a trace-based simulation of normal user workloads (media players and web browsers were among the applications traced), PCAP could save 76% of energy on average, which is 2% from matching that saved by a perfect oracle predictor (which always powers the disk down when idle time will exceed the time needed for spin-up power to equal power saved).

Prediction techniques can incur extra overhead, such as utilizing memory to store histories or CPU time to calculate predictions, but this overhead is typically insignificant in comparison to the time it takes to perform disk I/O operations. Powering the disk down, as always, consumes power cycles.

4.2 Using Flash for Caching

FlashCache [Marsh et al. 1994] uses flash storage as a second-level cache (below DRAM) and, in doing so, attempts to prevent I/O operations from reaching the disk. This lengthens the disk idle periods and thus provides more power-saving opportunities.

In testing, a small cache buffer was allocated in DRAM that, once full, would be flushed to a small flash memory card using an LRU caching policy. Today, a solid-state disk might be a more attractive cache choice due to its relatively large capacity. Since flash storage is persistent, the cached data does not need to be flushed for consistency. In fact, the flash device is not flushed until it becomes full, or a read miss occurs at both the DRAM and flash levels of cache.

Trace-based evaluations of FlashCache showed a decrease in energy usage of 20-70% on a typical workstation workload, while reducing overall response times by 20-70%. The simulated disks were spun down after 1 second of idle time. In some cases, read response time was lengthened by 20-50% due to read misses that incurred a spinup.

Caching solutions may not be suitable for systems that have many reads with poor locality. If a flash device is used as a cache, its erase cycle limit must also be carefully managed. For loads that do not have many writes or random reads, flash-based caching presents an attractive solution for increasing the number and the duration of disk idle periods and reducing read times.

4.3 Deduplication

Techniques to improve performance by avoiding storage access may also yield energy savings as a side effect. Besides caching, deduplication is another example. Basically, data blocks of the same content need not to be stored redundantly, leading to fewer storage accesses. However, the tradeoff is the computational cost of hashing and bookkeeping to identify identical blocks. Also, consolidating blocks with identical content may no longer preserve the spatial locality of original blocks belonging to the same files, and increase the probability of disks performing more random seeks.

Costa et al. [2011] performed an empirical study of writing system checkpoint images with different levels of similarities via a MosaStore deduplication storage layer. They measured the power of the entire system and reported that energy savings can be achieved once the level of content similarity is above 18%. To break even with the deduplication computational overhead, the similarity level needs to achieve 40%.

5. MULTI-DEVICE -LEVEL POWER-SAVING TECHNIQUES

The multi-device layer of the legacy storage stack coordinates multiple physical storage devices and presents a unified logical structure to the above layer. Multi-device solutions are widely used by data centers, so energy-saving techniques targeting this layer can have wide-ranging applications for server-class disks. However, this layer often has reliability mechanisms to protect against single-device failures, and the solutions devised need to be backward compatible in terms of the reliability semantics. Another issue of building solutions at this layer is the lack of access to information governed by upper layers (e.g., which blocks are allocated/free). Thus, certain solutions need to circumvent this constraint.

5.1 RAID Background

The most well-known implementation at this layer is software RAID (Redundant Array of Independent Disks), which coordinates multiple devices to provide parallelism. However, increasing the number of devices in a system also increases the chance a single device will fail. Thus, RAID provides different *levels* of redundancy to safeguard against failed devices. RAID levels 0, 1, and 5 are the most widely used.

RAID level 0 provides no redundancy and exploits parallelism by splitting a piece of data into *blocks* of equal length and storing them to the disks simultaneously (i.e., block 1 is written to disk 1; block 2, to disk 2; block 3, to disk 3; and so on). The corresponding blocks from different disks form a *stripe*. When accessed in parallel, RAID-0 will achieve a higher aggregate bandwidth. However, if one disk fails, data blocks located on the failed disk are also lost.

RAID level 1 implements redundancy by mirroring data. Any data written to one of the disks in the RAID volume is also written to a second disk. As long as one of the mirrored disks survives failure, no data will be lost. This level speeds up reads, since a read request can be fulfilled by the disk whose head is closest to the data, but may slow down writes, since a write request is not completed until both of the mirrored disks have completed the write.

RAID level 5 achieves parallelism in both reads and writes while also surviving a single disk failure. In RAID 5, data blocks are striped across disks (like RAID 0) along with an additional *parity* block per stripe. The parity blocks from different stripes are distributed across the disks to balance the parity update loads (writes to update a stripe's parity block). If any single disk fails, the content of lost block for each stripe can be recomputed using the remaining blocks within the same stripe.

Many other RAID levels exist, such as RAID 10 (1+0), which involves striping across mirrored pairs, and RAID 01 (0+1) in which 2 RAID 0 arrays are mirrored. The design of energy-efficient solutions needs to be backward compatible to existing RAID levels to be practical for deployment.

5.2 Spinning Down Individual Drives Based on Simple Thresholds

Similar to single-disk spin-down techniques, individual disks within a RAID volume can use a simple idle period threshold to switch power-saving modes. This approach

requires no special hardware. However, under server environments, both the spin-up latency and the lack of idle time may render this approach less effective. [Gurumurthi et al. 2003b].

5.3 Caching

Given that the key to saving power for disks is to create opportunities for them to spin down, it is possible to sacrifice the energy efficiency of a few disks in order to decrease overall energy usage. By diverting most of the I/O operations to a few disks, for example, we allow the remaining disks to stay idle. One way to accomplish this is to use cache disks to store frequently accessed items. These cache disks could also be exchanged for flash devices for even further improved energy efficiency. The primary tradeoffs include (1) the potential bottle-neck introduced by directing all of the most frequent requests to a small subset of the available devices, (2) migration of cached data sets to and from the cache devices, and (3) eventual propagation of updates from the cached devices to the storage devices. Having to acquire disks solely for caching may also increase cost.

Massive Arrays of Idle Disks (MAIDs) [Colarelli and Grunwald 2002] are well suited to meet large archival storage demands where random reads and writes are relatively uncommon. In a MAID, most of the disks remain powered down until needed except for a few always-on disks that cache recently accessed data or MAID metadata. MAID was evaluated via simulation with respect to an always-on striped RAID configuration. Block-level trace replays show that Colarelli and Grunwald's MAID has the potential to use 15% of the power of an always-on RAID system with negligible performance degradation. This configuration would fare poorly in systems where data is accessed frequently, and is best suited for large and rarely-accessed archives of data. Additionally, a MAID is not designed to maximize parallelism among the disks in the volume.

5.4 Content Regeneration

Content regeneration techniques take advantage of the ability of some multi-device techniques (e.g., RAID 5) to regenerate missing data. RIMAC [Yao and Wang 2006] defines the concept of a Transformable Read in Cache (TRC) for read requests to a RAID 5 volume. A TRC occurs when there are at least *N-1* blocks in a stripe available in cache. If this occurs, the remaining block in the stripe is regenerated. This condition would allow one disk in the RAID 5 to remain offline even if a block from the disk is requested and not cached. If a TRC hit cannot be made, a Transformable Read on Disk (TRD) policy is applied in which the minimum number of disk reads occur so as to recover at enough blocks to perform regeneration. TRD allows a single disk to remain in standby by utilizing the other (at most *N-1*) disks in the RAID, as well as the cache, in order to recover thought blocks necessary to regenerate the N^{th} block.

For writes, RIMAC adopts a write-back policy with a preference to write to active disks. To decide how to write updates, RIMAC proposes four schemes to update the parity for a stripe group. The goal is to choose the scheme which incurs the smallest number of disk accesses to update the parity, with priority always given to active disks.

RIMAC was evaluated using a trace-driven simulation and compared to a base RAID 5 system. It showed 29% improved performance under an IO-intensive load (Cello99 trace) and 2-6% for transaction processing (TPC-D) loads. RIMAC saved 14-15% energy with the Cello99 trace, and 33-34% energy for the transaction processing

trace. RIMAC best outperformed the standard RAID 5 when the number of disks was fewer than seven. RIMAC implementations require modifications to the RAID controller, as well as the use of non-volatile RAM.

RIMAC can save power for at most one disk since a RAID-5 requires $N - 1$ disks available to reconstruct missing data. However, through the use of erasure encoding, it is possible to generalize the concept of content regeneration to reconstruct the content of N disks by using any $M < N$ disks. This paradigm can be denoted as (N, M) erasure-encoded RAID. For example, RAID-5 would implement a (N, N-1) erasure encoding. Some erasure-encoding based methods have even achieved as high as (N = 48, M = 5) [Haeberlen et al. 2005].

Diverted Access [Pinheiro et al. 2006] exploits this (N, M) encoding to conserve power, where the N - M disks are read when there is a large demand for disk bandwidth, or when one of the M disks fails. Additionally, if a disk fails, not all disks need to be utilized to recover its data. Diverted access defines "original" (N) and "redundant" (M) disks in a volume, where redundant disks are only needed in a failure, and attempts to direct most I/O operations to the N original disks. Much like other approaches, writes to inactive disks are buffered (in non-volatile RAM) and only flushed when space must be reclaimed.

Diverted Access was designed for wide-area storage systems and primarily evaluated using simulated models for this workload. However, modified proxy traces were replayed for a real-workload comparison and achieved a 20-61% energy savings. These numbers are less (though comparable) to savings predicted by their models. The savings of Diverted Access largely depend on the size of the write buffer, without which "little if any" energy savings occur. Diverted Access requires modifications at the RAID controller level (for buffering) as well as non-volatile RAM, but can be applied to many levels of redundancy.

Content-regeneration-based solutions seek to exploit the inherent regeneration ability of RAID 5 and its erasure encoding to provide good energy savings. However, the performance overhead may be severe for some general erasure encoding schemes. Also, the need for non-volatile RAM to buffer writes can introduce new failure modes not anticipated by the RAID levels and erasure encoding.

5.5 Variable RPM

Although we have already discussed the technique of varying the rotational speed of disk drives at the per-device level, the multi-device level opens up new design space to this technique. By using an array of disks, a high-level software system can vary the RPM of disks in a volume with consideration to the ability of other member disks to compensate for any loss in performance.

Hibernator [Zhu et al. 2005] divides time into discrete chunks called *epochs*. At the beginning of each epoch, each member of a volume is assigned a speed configuration using a target average response time as well as its predicted workload for the next epoch based on the previous one.

In order to conserve energy, Hibernator proposes a customized multi-disk layout (Figure 5.4.1) where disks (D_0 to D_4 in the figure) in a volume are divided into *tiers* in which all disks run at the same speed. Tier 1 contains disks (D_0 to D_2) running at their maximum RPM, and each consecutive tier runs at a lower speed. During power-saving periods, since each stripe (e.g., blocks 0, 1, 100, and 200 and their corresponding parity block P) is stored across disks in different tiers, the layout needs to ensure that the parity blocks are stored on disks in tier 1 in order to avoid bottlenecks introduced by lots of parity updates. Under peak loads, all disks, regardless of their tiers, will be spinning at the maximum RPM. Hibernator needs to

shuffle blocks prior to serving peak loads so that the parities become more evenly distributed.

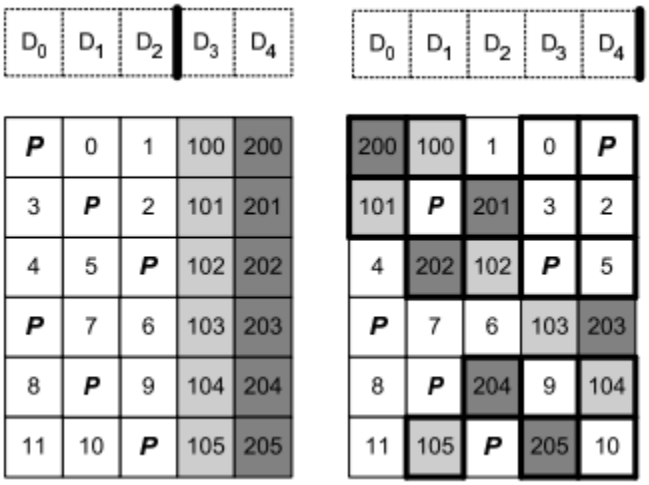

Fig. 5.4.1: Hibernator data layout. The left shows Hibernator in the power-saving mode, where only the disks (D_0 to D_2) in the first tier are powered. The parity blocks (P) are located within the first tier. The right shows Hibernator in the full power mode, where the parity blocks are distributed across all disks.

Hibernator was evaluated with simulation and trace replays from an OLTP (online transaction processing) workload. The length of the first epoch was varied between 60 and 240 minutes. Changes in the epoch length showed little improvement when compared to RAID 5. Using their *tier*-based data layout, they saved up to 63% power over a standard RAID 5 with no energy-saving techniques. The overhead of the tier-based disk layout is the need to reshuffle the data layout at the start of each epoch. The reshuffling process can last for 27 minutes, and slightly slow down the average response time of foreground requests from 6.3ms to 8.8ms.

5.6 Data Replication

In modern hard disks, storage tends to be inexpensive and abundant (~4-5 cents per gigabyte) [Pricewatch 2013]. Data replication techniques attempt to take advantage of free space in RAIDs in order to provide more spin-down opportunities.

PARAID (Power-Aware RAID) [Weddle et al. 2007] implements data replication to promote energy savings and is designed to adjust its aggressiveness to meet fluctuations in storage demand. PARAID sits below the RAID controller and is transparent to legacy RAID levels. It defines the concept of *gears* (Figure 5.6.1). When PARAID is running in the highest gear it utilizes all of the disks in the array with no changes to the original RAID layout (a PARAID-5 in the highest gear is essentially equivalent to a standard RAID-5). Each lower gear uses fewer disks than the preceding one. Before shifting to a lower gear (*down-shifting*) PARAID replicates data blocks from to-be-powered-down disks to the free space of other disks. PARAID switches gears based on load thresholds and leverages cyclic patterns of daily loads to reduce the number of power cycles.

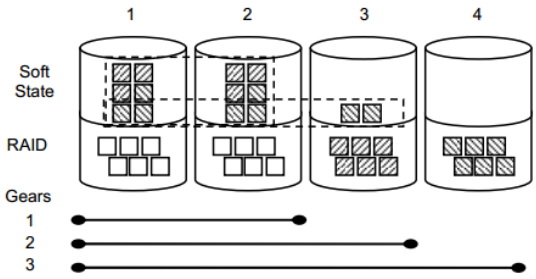

Fig. 5.6.1: PARAID gear organization.

In order to retain the same level of redundancy as the original RAID (for example, RAID-5's ability to survive a single disk failure), the replicated blocks are written as if they were a member with the same RAID level. For example, if a down-shift occurs in a RAID-5 that causes one disk to power down, the data blocks from that disk are striped across the remaining (N-1) disks as if those disks formed their own RAID-5 volume.

PARAID was implemented and tested with the Cello99 trace and a web trace. For the web trace, they saved 10-34% energy when compared with a standard RAID-0. For the Cello99 trace, they saved 3.5-13% when compared with a standard RAID-5. Performance was tested with PostMark, and showed little to no performance overhead in most cases, except when up shifting due to short bursts of requests which can affect performance as much as 13%. However, the number of power transitions occurs infrequently and is throttled to preserve the number of disk power cycles. More organic algorithms for determining when to up shift might alleviate this cost. PARAID may not be suited for write-intensive environments due to update overhead, or for systems with little free space.

SRCMap [Verma et al. 2010] reduces the overhead by replicating only the active data sets (e.g., emails for the past 20 days), or working sets (typically small) to one or more disks. However, to achieve replication at the multi-device layer, SRCMap needs to know whether a physical location is allocated or free, and this information resides at the file system layer. The solution is to implement SRCMap at a storage virtualization layer, which maps multiple physical devices into logical devices, where file systems can decide which logical locations are allocated, but the virtualization layer has full access and control of which physical locations are allocated.

During light loads, the system selects a minimum subset of disks to serve requests using replicated working sets. As for synchronization of updates, the replica to become active is synchronized immediately, while other replicas are synchronized in the background. A read miss can trigger updates to the replicated working set and can cause a spun down disk hosting the entire data set to spin up. However, the updated working set replica can help the disk hosting the entire data set power down again. The workload projection, replication, working set determination, and mapping between replicas and disks are reconfigured periodically (e.g., 2 hours).

Replica placement policy needs to consider a working set's size, popularity, and stability (estimated by number of read misses) as well as the corresponding disk's power consumption level and performance characteristics (potentially heterogeneous). Power hungry disks hosting lightly loaded replicas are good candidates for being powered down to save energy.

SRCMap is evaluated empirically and through simulation on 8 disks using webserver-server and home-directory-server workloads to eight data volumes. The

results show that SRCMap can achieve up to 36%. For a 10TB system, the memory overhead for mapping virtual to physical locations is about 3.2GB.

5.7 Transposed Mirroring

DiskGroup [Lu et al. 2007] is an alternative to RAID-01 which presents data mirroring in a novel and energy-focused way. A DiskGroup volume consists of two groups of multiple disks, dubbed the "primary" and "secondary" groups. The primary disks are mirrored across the secondary disks as in RAID-1, except the data from each disk in the primary group is striped across the secondary group as in RAID-0 (Figure 5.7.1). DiskGroup with one disk in each group would have precisely the same disk layout as RAID-1.

Primary Group				Secondary Group			
Disk 0	Disk 1	Disk 2	Disk 3	Disk 0	Disk 1	Disk 2	Disk 3
A1	B1	C1	D1	A1	A2	A3	A4
A2	B2	C2	D2	B1	B2	B3	B4
A3	B3	C3	D3	C1	C2	C3	C4
A4	B4	C4	D4	D1	D2	D3	D4

Fig. 5.7.1: DiskGroup layout of 8 disks divided into 1 pair of disk groups [Lu et al. 2007].

When the load is light, the primary group services all requests and the secondary group remains offline. In RAID-01, an overloaded stripe of data can result in all of the disks being online even when each disk is barely overloaded. With DiskGroup however, it is possible to offload data from many overloaded disks in the primary group to just a few active secondary disks, since every disk in the secondary group has at least some data from each disk and stripe in the primary group. DiskGroup stores writes in non-volatile RAM until the target secondary disk is powered up.

DiskGroup was evaluated through simulation by generating random block requests uniformly distributed across blocks on the disk. A DiskGroup of 16 disks with 16,000 data blocks each was used. These evaluations found that DiskGroup only needed to spin up 12-50% of the secondary disks (instead of 100% like RAID-01) on average, and energy savings varied with workload. Energy savings varied from 7% to 63% when compared with RAID-1; the best savings were seen when just a few primary disks were overloaded.

One tradeoff of DiskGroup is that it no longer adheres to the reliability semantics of RAID-01. For a conventional RAID-01, if one disk fails in the first RAID-0, a second corresponding mirrored disk needs to fail in the second RAID-0 to lose data. However, for the example disk layout in Figure 5.7.1, if one disk fails in the first RAID-0, any disk failure in the second RAID-0 will lead to data loss.

6. FILE SYSTEM-LEVEL POWER-SAVING TECHNIQUES

A file system provides naming for files and directories in addition to defining how they should be laid out on the disk. Examples of popular file systems include ext4 (popular on Linux), NTFS (primary file system for Windows), and FAT32 (popular on USB flash drives). One issue of building solutions at this layer is the lack of complete control of the physical locations of individual files, which are governed by storage layers below. Thus, certain solutions need additional coordination mechanisms across layers.

6.1 Popular-based Data Consolidation

In addition to cache disks (Section 5.3), one way to shift the load of a system onto fewer disks is migrating commonly-referenced files to target *hot* disks. This method would alleviate the load on the original disk and allow it to have more idle time.

Pinherio and Bianchini [2004] refer to this technique as Popular Data Concentration (PDC) and evaluate its effectiveness by creating a user-level file server known as *NomadFS*. The idea of PDC is simplified to the context of a large array of disks in which the most popular data will always reside on the first disk in the array, the next most popular data which does not fit on the first disk will reside on the second, and so on. Metadata for the entire array is always stored on the most popular (first) disk.

To account for the changing popularity of data, periodical migration occurs to account for new popularity measurements. NomadFS handles this migration and will explicitly spin down disks that have been idle for too long. For two-speed disks, the disks manage their own speed settings, and data is migrated off a disk only when disks are running at full speed. Unlike MAID, PDC implements a frequency-based migration whereas MAID caches based on how recently data was accessed.

Evaluations were performed via simulator using proxy server traces and synthetic traces generated to follow Zipf's law. The test bed consisted of a 9-10 disk array and the period between migrations was varied. During light workloads of 0-200 requests per second, PDC demonstrated up to 40% savings over an energy-oblivious disk. Because the most popular files are concentrated on one disk, PDC does not exploit parallelism.

6.2 Remote RAM Caching

BlueFS [Nightingale and Flinn 2004] is designed for mobile storage devices (e.g., laptop drives, USB thumb drives). While its primary goal is to reduce energy consumption to accommodate the limited battery power of mobile, BlueFS tries not to compromise its response time. It does this by replicating data at a file server as well as each mobile device which it manages: it exploits the now ubiquitous network access capabilities of most mobile devices. BlueFS decides which replica to access by analyzing the power states of the various devices as well as the power consumption and access time of different access strategies. Updates (writes) are stored on the server as a primary replica and pushed to mobile storage devices to act as persistent caches. Furthermore, BlueFS masks the overhead of device state transitions by accessing data from alternative sources (e.g., mask the disk spin-up latency by fetching some data from the remote replica first).

BlueFS changes device power states proactively by disclosing hints about the devices that it intends to access. However, simple strategies may not work well if the system makes decisions based on processing individual storage requests one at a time, since a single request may not justify the overhead of spinning up the local disk. BlueFS solves this problem through a mechanism known as *ghost hints*, which computes the opportunity cost of having a series of operations carried out together with a changed device state.

For writes, BlueFS maintains an asynchronous write queue for each available storage device, and write operation can be aggregated according to the power state of each device. The write aggregation could amortize the energy cost of spinning up the disk and powering on network interfaces. For data coherence, BlueFS adopts optimistic consistency semantics, so that replicas can temporarily become out of sync when the network is not available, and most of the conflicting updates can be later resolved using version stamps and automated conflict resolvers. In terms of its

performance, BlueFS has shown up to a 55% energy savings and 76% reduction in access latency.

6.3 Data Replication

FS2 [Huang et al. 2005] replicates frequently accessed data blocks to hot disk regions, so that it can reduce the average seek time and associated energy consumption. Unlike other schemes that migrate data, FS2 maintains one or more replicas for selected data blocks. Additionally, instead of reserving fixed amounts of disk space for periodic layout modifications, FS2 utilizes free space to accommodate replicas dynamically. While accessing data, FS2 can select from available replicas in order to reduce seek and rotation overhead as well as energy consumption.

To manage replicas, at the device-driver layer, FS2 divides a disk into regions, and identifies hot regions based on access frequency. For each hot region, it creates replicas of those resident data blocks in other regions that seem to have a read relationship with blocks in the hot region. Replicas are placed in free space closest to current disk head positions in hot regions. However, to know which blocks are free, the device layer needs to consult with the file system layer, which keeps track of the allocation status and file membership of blocks. Similarly, the file-system layer needs to inform the device-driver layer if original data blocks are deleted from a file, and the device-driver layer is then responsible for invaliding its replicas.

When handling read requests, FS2 chooses the replica closest to the current disk head's position, among other criteria. For writes, FS2 simply invalidates all related replicas and then updates the original. Invalidation to less frequently accessed replicas also occurs if the amount of free space on the disk drops below the low watermark. FS2 uses a hash structure to track locations and usage information of data blocks and replicas. The consistency between the structure and blocks is maintained by periodically flushing modified entries in the structure to disk and at shutdown time. In the case of system crashes, the consistency needs to be restored through a tool. By increasing data availability and exploiting free spaces on the disk, FS2 could reduce user-perceived access time by 34%. The computed disk power consumption for per disk access also shows a 71% reduction.

6.4 Local Cache for Remote Data

GreenFS [Joukov and Sipek 2008] reduces the local disk power consumption by using remote storage and local flash whenever possible. GreenFS reverses the role of remote backup and local copy. When network connectivity is available, updates are sent to the remote storage, and the local flash is used to cache reads. When network connectivity is not available, GreenFS uses flash to buffer updates, and the local disk spins up at least once a day to synchronize with flash. Otherwise, the local disk only spins up and reads/writes/synchronizes before the local system shuts down, when the flash is near full, or when disk is needed to improve the performance. The number of disk power cycles is tracked and capped.

GreenFS saves energy under a variety of workloads: idle, small reads/writes, and large reads/writes. When idle, only remote storage is powered. For small reads and writes, the local flash cache significantly reduces the local disk traffic and saves energy. Even with cache misses, data is read from remote storage through the network, which still consumes less power than spinning up and reading the local disk. Power for data transfer through network interface may exceed power to spin up disk and transfer data with large reads or writes. As a performance benchmark result, GreenFS saved 12% power for servers, 14% for desktops, and 2-3% for mobile devices.

7. VIRTUAL FILE SYSTEM-LEVEL POWER-SAVING TECHNIQUES

The virtual file system is the top layer of the storage stack and serves to provide a common interface allowing applications to be agnostic of the file system type. Energy savings achieved at this layer can be enjoyed by all file system types.

7.1 Encourage IO Bursts

A disk cannot spin down, even if this load is light. It is possible, however, to redistribute requests to an under-loaded disk (Figure 7.1.1) in such a way that it instead experiences shorter periods of maximum load as well as longer periods of idleness (Figure 7.1.2). This access pattern is called burstiness [Papathanasiou and Scott 2003].

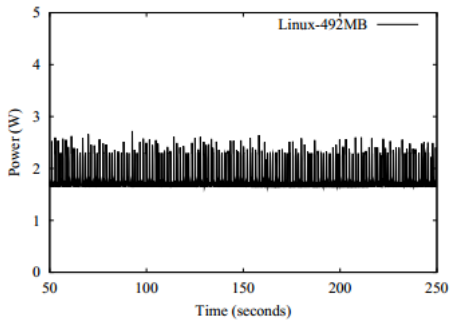

Fig. 7.1.1: A disk under constant low-load with very short idle periods [Papathanasiou and Scott 2003].

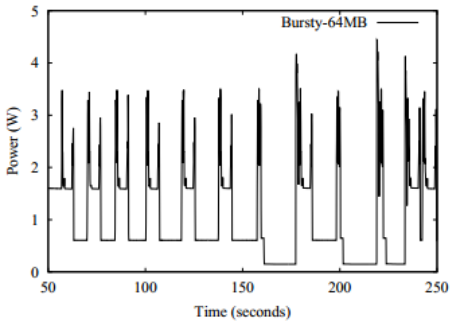

Fig. 7.1.2: A disk with bursts and longer idle periods [Papathanasiou and Scott 2003].

Papathanasiou and Scott developed a system to increase the burstiness of disk accesses in order to allow for more periods of idleness. To accomplish this, modifications were made to the file system and Linux kernel memory management. These modifications attempt to pre-fetch data from the disk in much larger chunks than existing pre-fetching algorithms. The goal is to pre-fetch most of the data an application will need in a few IO-intensive operations (bursts). Aggressive policies such as pre-fetching all of the files an application references, or all of the files in its working directory, are utilized in order to ensure the application does not need to make many small requests.

Additionally, a pre-fetch daemon is added to the kernel in order to provide multi-process coordination for reads, similar to the update daemon's existing role for write requests. Without a pre-fetch daemon, each application would independently issue read requests that would likely result in out of sync bursts. Many applications with

this policy this would result in the lack of long idle periods from the perspective of the disk. File updates are flushed once per minute. Depending on applications (e.g., compiled .o files, file copying), certain file updates can be postponed to the file-close time to address the data loss concerns due to power outage.

The system was evaluated by measuring the current consumption of a Hitachi disk under a read-intensive load (media playback) and a read/write-intensive load (media playback and encoding). Energy savings were computed by comparing the current consumption of their system with that of an unmodified Linux kernel. For the read-intensive workload, they found energy savings of 40% with 64MB of RAM used for caching, and 79% for 492MB of RAM. For the combined workload, energy savings ranged from a small loss at 64MB to just over 60% savings with 492MB. Energy savings can be significant, but at the cost of RAM and an IO-intensive pre-fetch stage. Additionally, depending on the workloads, too much RAM devoted for prefetching may cause more cache-misses and subsequently negate the effects of energy savings.

7.2 Cache Cold Disks

Traditional caching techniques target items based on frequency and recency of use, thereby avoiding the high cost of disk accesses. However, in the context of caching multiple disks to conserve power, caching can target items from less heavily used disks, so that idle periods for those disks can be lengthened to save more power. Zhu et al. [2004] presents a set of cache management algorithms designed with this observation to save energy. In particular, a scheme referred to as "PA" is introduced. For each disk, PA tracks important cache-related workload characteristics such as the percentage of cold misses (cache misses caused by a first-time access) and the cumulative distribution of interval lengths (time between requests). PA uses the profiles it builds to decide which disks are "cold", where cold disks are the better candidates for cache space due to their already relatively low access frequency.

A cold disk in PA is expected to exhibit few cold cache misses and large interval lengths. All disks which are not classified as cold are regular disks. To account for changes in workloads, this classification is recalculated periodically at the start of a time chunk known as an epoch. This system can then be applied to most cache-replacement algorithms by modifying them such that they prefer to give cache space to cold disks over regular disks. This means we can make any caching policy power aware. For example, a modified LRU (PA-LRU) will always evict the least-recently-used block belonging to a regular disk, even if there are older blocks in cache which belong to a cold disk.

PA-LRU was evaluated through simulation of the cache management policies and multi-speed disks. Replaying an online transaction processing trace with 128MB of cache space, PA-LRU saved 16% on energy and improved the average response time by 50% when compared with a standard LRU caching policy. However, PA-LRU was only able to save 2-3% when replaying the Cello96 trace with 32MB of caching space. The decrease in savings is attributed to a high percentage (64%) of cold cache misses in the trace. The lower memory size was chosen to fit the smaller size of the Cello96 trace.

PA is an attractive solution to energy-saving concerns since it may be applied to existing cache-replacement policies and complements other energy-saving techniques. For random and write-intensive workloads, however, PA-LRU is less effective.

8. APPLICATION-LEVEL POWER-SAVING TECHNIQUES

Application developers can implement strategies to help save energy on storage. These strategies can utilize high-level knowledge about the application-specific workload.

8.1 Energy-efficient Encoding

Multi-level cell SSD technology allows for encoding four values (two bits) in each memory cell. These values, however, do not always require the same energy to encode. In fact, encoding the middle two values (10 and 01) can require as much as six times the energy and time as encoding the edge values (00 and 11) [Joo et al. 2007]. While most modern compression techniques focus on encoding a bit string into a more compact one to save space, the resulting bit string may not be energy-efficient to write to flash. If we instead focus on energy efficiency, it is possible to encode into bit strings, which are more energy efficient to write even when they are less space efficient.

Table 8.1.1: Required programming time and energy for a single SSD [Joo et al. 2007].

Operation	Time (μs)	Energy (μJ)
Program 00	110.00	2.37
Program 01	644.23	14.77
Program 10	684.57	15.60
Program 11	24.93	0.38

Joo et al. [2007] presents an approach for energy-efficient encoding, which allows cost (in terms of energy) to be assigned to individual bit patterns to reflect the characteristics of the NOR flash storage medium. In particular, they use 32-bit words as the standard data unit size, and define a cost function to predict the energy-cost to write a data unit. Due to the parallel way in which NOR flash programs memory cells, it is hard to deterministically predict the energy cost of a data unit, so a probabilistic algorithm, which determines the expected cost of each symbol is used. Finally, integer linear programming is utilized to find an encoding which maximizes energy savings.

This method was able to achieve a 35% energy savings by utilizing 50% more space when encoding JPEG images. No changes are required to the storage stack, and this method can be implemented in any application that will use NOR flash for storage. However, storage overhead can be significant, and it is unclear whether this encoding scheme incurs additional computational and associated energy overhead.

9. LARGE-SCALE POWER-SAVING TECHNIQUES

Some power saving solutions can be applied on a scale larger than a single machine. These large-scale power-saving techniques take advantage of multiple machines (such as the layout of a data-center) to find ways to save power in the big picture.

9.1 Spin-up Tokens

For archival workloads, reads are fairly rare and performance is often not an important concern. Energy can be saved in these systems by limiting the number of disks which are active without seeing an unacceptable change in performance. One way to approach archival systems is to require disks to possess a limited resource (e.g., a token) in order to spin up. This would ensure that only a limited number can be active at any one time. When disks spin down, they release this resource which allows another disk to spin up and take its place.

Pergamum [Storer et al. 2008] is a large-scale archival workload solution employing this mechanism. The Pergamum system divides a set of hardware into *tomes* (a grouping of a hard disk, flash memory, a low-power CPU and a network interface). Each tome is responsible for running its own copy of software, which performs operations such as (1) consistency checking, (2) power management and (3) inter-tome operations. Tomes share data blocks for inter-tome redundancy.

Pergamum makes use of non-volatile RAM at each tome in order to store consistency information and metadata for the underlying storage. This cache allows Pergamum to verify the integrity and existence of both local and distributed data without incurring the energy cost of spinning up the underlying disk. When a tome becomes unavailable, its data can be rebuilt by the remaining tomes in the network.

During normal operation, Pergamum attempts to keep at least 95% of the disks in the network idle. To limit total spin-ups, a Pergamum network has a limited number of *spin-up tokens* available: these are passed between tomes via the Pergamum software and a tome must possess one in order for it to spin up its disks. When a tome wants to spin up, it must first request a token from a tome which holds one. When that tome receives multiple requests, it will calculate which tome needs it the most based on a variety of factors such as time since last spin-up, number of requests, etc.

Pergamum introduces a storage overhead of about 30%. If we assume that Pergamum meets its goal of keeping an average disk inactive 95% of the time, we can roughly calculate the potential energy usage for the average disk as $1.3 * (95\% \, idlePower + 5\% \, activePower)$. For the Seagate Savvio® 10K.6 [Seagate 2012], this yields an average of 5.3 W, or about 67% of the normal active power. Since Pergamum is designed for archival workloads, this number cannot be directly compared to storage systems with more active workloads.

9.2 Write Offloading

Narayanan et al. [2008] introduced a *write offloading* technique for enterprise workloads based on two observations: (1) idle periods for server-class storage can be exploited for power saving, and (2) for enterprise workloads, read operations can be largely handled by caching, leading to periods of write-dominant traffic toward the backend storage. By temporarily offloading writes from spun down disks to active disks, we can extend the idle periods of spun down disks to achieve greater energy efficiency. When the spun down disks wake up, the offloaded content can be migrated to its final destined disks. This approach also succeeds in buffering writes so that many writes can be consolidated into one update propagation instead of incurring multiple ones.

To facilitate this system, each volume in the network is assigned a *manager* who determines when to change the power modes of the associated disks, when and where to off-load writes to those disks while they are idle, and when and how to reclaim the data to maintain consistency. When a block is written to an idle volume, the volume's manager chooses some active remote storage (called a *logger*). The block is temporarily *offloaded* to that remote storage. When the volume becomes active again, the manager is responsible for reclaiming the latest version of every block which was offloaded, as well as invalidating older versions. Different blocks, as well as different versions of the same block, may be offloaded to any number of remote loggers.

This write-offloading system was evaluated by replaying traces (which were 70% reads) on a rack-based testbed and comparing the numbers with those of a baseline system, which does not spin down disks or offload writes. Blocks were offloaded to an

extra partition on remote hard disks. A system that offloads to volumes on other machines sharing the same rack uses about 40% of the energy the baseline system used on the idlest day of the trace, and 80% on the most active day. For comparison, a system that spins down after sixty seconds without reads and ten seconds without writes used 64% (idlest) and 87% (most active) of the baseline. Similar, though slightly reduced, savings can be achieved when offloading only to volumes on the same machine (rather than the same rack).

Write offloading can save energy without requiring specialized hardware or changes to applications or drivers. On the other hand, since this approach focuses on writes, short intervals between read misses can still consume disk power cycles [Verma et al. 2010].

9.3 Graph Coverage

In a read-mostly system that employs simple disk mirroring to provide data replication, it is more energy-efficient to keep the replica disks spun down than active. For d primary disks with r replicas (r=1 means no replication), this system would require dr disks in total, and at least $\frac{1}{r}$ of them must be active for full availability of data. If we utilize free space on other disks instead of adding a new one for each replica, we can achieve even better energy efficiency without sacrificing availability or redundancy. This model can be thought of as a bipartite graph in which one part contains storage media and the other contains data. An edge between the parts represents where data resides. In order to maintain availability, each data node must have an edge to an active storage node (Figure 9.2.1).

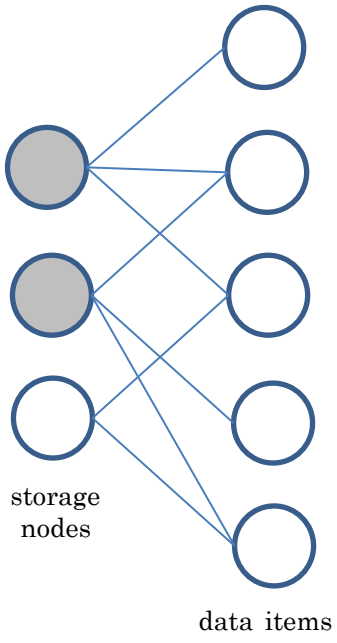

Fig. 9.2.1: A bipartite storage-node graph with full coverage. The shaded storage nodes are powered on. Replication factor is two. Full coverage is achieved.

Harnik et al. [2009] proposes an energy-saving model, which emphasizes data availability for systems that require replication. For this model, a data center's storage facilities are divided into *storage nodes* (the left part of the graph), ranging from single disks to entire portions of the data center, which can be independently turned on and off. All stored data is divided into *redundancy units* (the right part), which can represent things like blocks, files, etc. Each redundancy unit belongs to one or more storage nodes (more nodes per unit mean more redundancy) and is assigned to them based on a *placement function*. Finally, time periods in which the data center can expect low utilization are defined.

This model can be applied to many replication systems by defining specific node sizes or placement functions. Typically, a good placement function tries to minimize the number of nodes that must be active to achieve full coverage (where each data node has an edge to an active storage node), and does so in a way that does not sacrifice performance during high utilization periods. By varying the placement function and number of replications (replications are defined as the number of storage nodes a single replication unit is copied to), different full coverage solutions can be achieved. For better energy savings and slightly lower performance, some rarely accessed data nodes may be left uncovered by an active storage node (particularly if most other data nodes it services are already covered).

When only two replications are needed, this system performs similar to simply mirroring the data and keeping the mirrored set powered down (thus achieving full coverage by keeping the primary set powered up). However, when more than two replications are needed, clever placement functions can lead to a much smaller set of necessary disks than simple mirroring.

This model can be applied easily to multi-disk or multi-machine environments, and its modularity allows many specialized possibilities to be achieved by varying the placement function or number of replications. Energy savings with a good placement function can be significant, with one evaluation showing savings of over 45% at the cost of 5% additional storage (with three replications). Writes are delayed and written to a log for later propagation.

9.4 Energy Proportionality

Energy proportionality was proposed by Barroso and Holzle [2007]. The basic observation is that the energy efficiency tends to be low when a system is lightly loaded. Ideally, the amount of energy expended should be proportional to the amount of computation completed. This paper highlights the inefficiencies in memory and the storage subsystems. A subsequent paper [Guerra et al. 2010] also summarizes how existing power-saving techniques can be applied toward realizing energy proportionality.

Amur et al. [2010] proposed a distributed file system called Rabbit, which strives to realize this energy proportionality. That is: it aims to keep the energy efficiency of the system at low load the same as the efficiency at the peak load. To achieve this, each replica of data is distributed across more drives than the previous replica (Figure 9.4.1). The first replica, which is distributed among the smallest number of drives, is the only replica that must be online to guarantee the availability of data. When more performance is needed than this *primary replica* can supply, additional replicas are spun up as necessary to meet increased demand. Under light loads, writes are handled by performing write offloading (see Section 9.2).

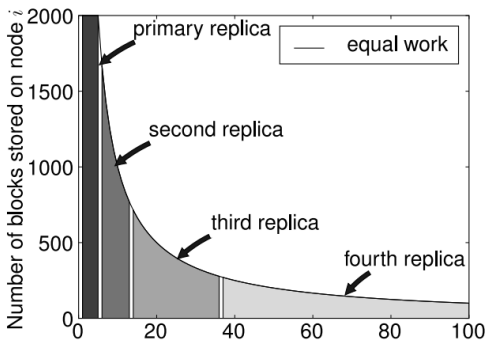

Fig. 9.4.1: Rabbit target data layout [Amur et al. 2010].

An additional goal of Rabbit is the ability to maintain a near ideal energy-proportional curve in a failure state (i.e., after a disk has failed and during rebuilding of a replacement disk). To mitigate the effects of a failure, *each* disk is assigned to two distinct groups. These groups should have no intersecting elements except for the disk in question. When data is written to the disk, the primary replica is distributed across one group and the second replica is distributed across the other. In this way, a failure in the group containing the primary replica should not affect the availability of data necessary for recovery since it is stored on separate disks.

Evaluations through both empirical measurement and simulation with 14 storage nodes and two datasets showed an average power-proportionality for both tests and datasets held above 80%.

9.5 Hot and Cold Zones

Based on 2600 Yahoo servers, Kaushik and Bhandarkar [2010] observed that 89% of files are actively referenced for about 10 days after their creation, and 60% of files become inactive after 20 days. These observations led to the design of GreenHDFS, which classifies files into hot, popular and frequently accessed files, and cold files, which have poor locality and low access frequency. Hot files are stored on high performance servers, with files striped to exploit parallelism, and with no power-saving features enabled. Cold files are compressed, not striped, and stored on storage-intensive servers that aggressively power down system components. As hot files cool down in their reference patterns, they are moved from hot-zone machines to cold-zone machines. As cold files become popular, they are moved from the cold zone to the hot zone. During peak utilization, cold zone servers can still contribute to the overall performance.

GreenHDFS is evaluated using a trace-driven simulator. Over a three-month period, GreenHDFS showed a 26% energy savings when being applied to 2,600 servers. Disks in the cold zone made only about 70 power transitions within the three-month period.

10. CONCLUSION

This survey has presented a representative set of techniques to reduce power consumption in data storage. Table 10.1 summarizes their key characteristics, and interestingly, similar approaches (e.g., content regeneration) at the same layer can offer very different tradeoffs. However, some trends emerge. The tradeoffs are most diverse near the hardware level. For large-scale solutions, the main tradeoffs are extra space for replication and associated costs for data migration and

synchronization. Under interactive and archival workloads, performance can be a good tradeoff to achieve energy savings. Solutions that exploit additional hardware may not be able to retain the original reliability semantics, if the hardware introduces new failure modes (e.g., failure of non-volatile RAM buffer).

The table also shows the potential compatibility and conflicts for approaches based on their tradeoffs. For example, approaches from different layers would not be compatible if one trades energy with storage capacity; the other, performance. On the other hand, of both approaches share similar tradeoffs (e.g., BlueFS and FS2), both approaches are more likely to be combined to achieve greater energy savings. However, it is less clear that the net energy savings will compound.

Overall, depending on the requirements, storage architects can craft their new design by choosing the tradeoffs they are willing to compromise, and energy savings do not come without a cost.

Table 10.1: Summary of Characteristics for Power-saving Techniques.

Techniques	Preserving peak performance	No need for extra storage	Preserving the reliability semantics	No extra hardware (e.g., non-volatile RAM for write buffering, multi-speed disks, extra disks)	No data migration overhead	Little extra hardware cost
Hardware solutions						
Slower RPM		✓	✓	✓	✓	✓
Variable RPM disks	✓	✓	?		✓	?
Smaller disk platters	✓		✓	✓	✓	
Higher capacity disks		✓	?	✓	✓	✓
Hybrid drives	✓	✓	?		✓	✓
Device driver-level solutions						
Spin down disks	✓	✓		✓	✓	✓
Use flash for buffering	✓	✓	?		✓	?
Deduplication	?	✓	?	✓	✓	✓
Multi-device solutions						
Spin down individual disks	✓	✓		✓	✓	✓
Use cache disks (MAID)		✓	✓			
Content regeneration (RIMAC)	✓	✓	?			✓
Erasure encoding			?		?	✓
Variable RPM (Hibernator)	✓	✓	?			✓
Erasure encoding			?	✓	?	✓
Replicate data (PARAID, SRCMap)	✓		✓	✓		✓
Transposed mirroring	✓	✓				✓
File system-level solutions						
Popularity-based Migration (PDC)		✓	✓	✓		✓
Remote RAM Caching (BlueFS)	✓		✓	✓		✓
Replicate data (FS2)	✓		✓	✓		✓
Local cache for remote data (GreenFS)	✓	✓	✓			✓
VFS-level solutions						
Encourage bursty IOs	✓	✓	?	✓	?	✓
Cache cold disks	✓	✓	✓		✓	✓
Application-level solutions						
Energy-efficient encoding			✓	✓	✓	✓
Large-scale solutions						
Spun-up token (Pergamum)			✓		✓	✓
Graph coverage	✓		✓	?		✓
Write offloading	✓		?	✓		✓
Power proportionality (Rabbit)	✓		✓	✓		✓
Hot and cold zones	?	✓	✓	✓		✓

A solution with a certain characteristics is marked by ✓. Certain characteristics that are unclear and are denoted by '?' (e.g., whether variable-RPM drives have the same reliability characteristics as fixed-RPM drives).

ACKNOWLEDGMENTS

This work is sponsored by NSF CNS-0410896. Opinions, findings, and conclusions or recommendations expressed in this document do not necessarily reflect the views of the NSF, FSU, or the U.S. Government.

REFERENCES

Hrishikesh Amur, James Cipar, Varun Gupta, Gregory Ganger, Michael Kozuch, Karsten Schwan. 2010.

Robust and Flexible Power-Proportional Storage. In *Proceedings of the 1st ACM Symposium on Cloud Computing*. ACM, New York, New York, 217-228.

Luiz Andre Barroso and Urs Holzle. 2007. The Case for Energy-Proportional Computing. *Computer*, IEEE Computer Society, 40, 12(2007), 33-37.

Tom Bostoen, Sape Mullendar, and Yolande Berbers. 2013. Power-reduction Techniques for Data-center Storage Systems. *ACM Computing Surveys*, 4, 3(2013).

Brian Chen. 2012. Facebook's Filing: The Highlights. *New York Times*, Retrieved June 15, 2013 from http://bits.blogs.nytimes.com/2012/02/01/facebooks-filing-the-highlights.

Dennis Colarelli and Dirk Grunwald. 2002. Massive Arrays of Idle Disks for Storage Archives. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*. IEEE Computer Society Press, Los Alamitos, California, 1-11.

Lauro Beltrao Costa, Samer Al-Kiswany, Raquel Vigolvino Lopes, and Matei Ripeanu. 2011. Assessing Data Deduplication Tradeoffs from an Energy and Performance Perspective. In *Proceedings of Green Computing Conference and Workshops*. IEEE Computing Society, 1-6.

Fred Douglis, P. Krishnan, Brian Marsh. 1994. Thwarting the Power-Hungry Disk. In *Proceedings of the USENIX Winter 1994 Technical Conference on USENIX*, USENIX Association, Berkeley, California, 293-306.

eSaiTech. 2013. Retrieved June 22, 2013 from http://www.esaitech.com/hitachi-microdrive-3k4-hms360404d5cf00-08k2532-4gb-3600rpm-128kb-removable-hard-drive.html?pk_campaign=ga-pla&gclid=CN_yu4_A97cCFZNj7AodKR8ATQ.

P. Krishnan, Philip Long, and Jeffrey Scott Vitter. 1995. Adaptive Disk Spindown via Optimal Rent-to-Buy in Probabilistic Environments. In *Proceedings of the 12th International Conference on Machine Learning*, NAACL.

Kristian Fredslud, Fridrik Rafn Isleifsson, Julie Juel Andersen, Martina Zamboni, and Oxana Pantchenko. 2009. Green Data Center Cooling. Retrieved June 15, 2013 from https://localrenew.soe.ucsc.edu/sites/default/files/Green_Data_Center_Cooling_report.pdf.

Fujitsu. 2002. MAP3147NC/NP, MAP3735NC/NP, MAP3367NC/NP Disk Drives Product/Maintenance Manual. Retrieved June 15, 2013 from http://www.andovercg.com/datasheets/fujitsu-map-10k-rpm_prod-manual.pdf.

John Gantz and David Reinsel. 2011. Extracting Value from Chaos. Excerpts from the IDC iView Extracting Value from Chaos. Retrieved June 15, 2013 from http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf.

Chris Gniady, Charlie Hu, and Yung-Hsiang Lu. 2004. Program Counter Based Techniques for Dynamic Power Management. In *Proceedings of the 10th International Symposium on High-Performance Computer Architecture*.

Jorge Guerra, Wendy Belluomini, Joseph Glider, Karen Gupta, and Himabindu Pucha. 2010. Energy Proportionality for Storage: Impact and Feasibility. *ACM SIGOPS Operating Systems Review*, ACM New York, New York, 44, 1(2010), 35-39.

Sudhanva Gurumurthi, Anand Sivasubramaniam, Mahmut Kandemir, and Hubertus Franke. 2003a. DRPM: Dynamic Speed Control for Power Management in Server Class Disks. In *Proceedings of the 13th International Symposium on Computer Architecture*, ACM New York, New York, 169-181.

Sudhanva Gurumurthi, Jianyong Zhang, Anand Sivasubramaniam, Mahmut Kandemir, Hubertus Franke, N. Vijaykrishnan, Mary Jane Irwin. 2003b. Interplay of Energy and Performance for Disk Arrays Running Transaction Processing Workloads. In *Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE Computer Society, Washington DC, 123-132.

Andreas Haeberlen, Alan Mislove, and Peter Druschel. 2005. Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures. In Proceedings of the 2nd Symposium on Networked Systems Design and Implementation. USENIX Association, Berkeley, California, 143-158.

Danny Harnik, Dalit Naor, and Itai Segall. 2009. Low Power Mode in Cloud Storage Systems. In *Proceedings of the 2009 IEEE International Parallel & Distributed Processing Symposium*. IEEE Computer Society, 1-8.

HGST. 2003. Hitachi Microdrive 3K4 Digital Media. HGST. Retrieved June 22, 2013 from http://www.hgst.com/tech/techlib.nsf/techdocs/FF5400DF7594877E86256D830072F4CF/$file/HGST4GB_Microdrive9.10.03.PDF.

HGST. 2012. Travelstar Z5K500 2.5-Inch Mobile 5400 7mm Hard Disk Drives. HGST. Retrieved June 15, 2013 from http://www.hgst.com/tech/techlib.nsf/techdocs/A50EBE5C3021DB33882577FA000ACBC2/$file/TSZ5K500_ds.pdf.

Hitachi. 2007. Deskstar 7K1000 3.5-Inch Hard Disk Drives. Hitachi. Retrieved June 15, 2013 from http://www.hgst.com/tech/techlib.nsf/techdocs/67A68C59B27368FC862572570080FC70/$file/Deskstar7K1000_010307_final.pdf.

Hai Huang, Wanda Hung, and Kang Shin. 2005. FS2: Dynamic Data Replication in Free Disk Space for

Improving Disk Performance and Energy Consumption. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles*. ACM, New York, New York, 263-276.

Intel. 2006. Intel® Pentium® D Processor 800 Sequence Datasheet. Intel. Retrieved June 15, 2013 from http://ark.intel.com/products/27514/Intel-Pentium-D-Processor-840-2M-Cache-3_20-GHz-800-MHz-FSB.

Intel. 2010. Intel® Xeon® Processor X7560 Datasheet. Intel. Retrieved June 15, 2013 from http://ark.intel.com/products/46499.

Intel. 2011. Intel® Xeon® Processor E7-4870 Datasheet. Intel. Retrieved June 16, 2013 from http://ark.intel.com/products/53579.

Atsushi Inoue and Doug Wong. 2003. NAND Flash Applications Design Guide. Toshiba. Retrieved June 15, 2013 from http://measure.feld.cvut.cz/groups/edu/x38nrp/Materialy/NandDesignGuide.pdf.pdf.

Yongsoo Joo, Youngjin Cho, Donghwa Shin, and Naehyuck Chang. 2007. Energy-aware Data Compression for Multi-level Cell (MLC) Flash Memory. In *Proceedings of the 44th Annual Design Automation Conference*. ACM, New York, New York, 716-719.

Nikolai Joukov and Josef Sipek. 2008. GreenFS: Making Enterprise Computers Greener by Protecting Them Better. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems*. ACM, New York, New York, 69-80.

Rini Kaushik and Milind Bhandarkar. GreenHDFS: 2010. Towards An Energy-conserving, Storage-efficient, Hybrid Hadoop Compute Cluster. In *Proceedings of the 2010 International Conference on Power-aware Computing and Systems*. USENIX Association, Berkeley, California, 1-9.

Zsolt Kerekes. 2012. Charting the Rise of the Solid State Disk Market. Retrieved June 15, 2013 from http://www.storagesearch.com/chartingtheriseofssds.html.

Hemanth Kothuru, Girish Solur Virupakshaiah, Shraddha Jadhav. 2010. Component-wise Energy Breakdown of Laptop. In Proceedings of the 6th Annual Symposium: Graduate Research and Scholarly Projects. Wichita State University.

Dong Li, Peng Gu, Hailong Cai, and Jun Wang. 2004. EERAID: Energy Efficient Redundant and Inexpensive Disk Array. In *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*. ACM, New York, New York, Article No. 29.

Pablo Llopis, Javier Garcia Blas, Florin Isaila, and Jesus Carretero. 2013. Survey of Energy-efficient and Power-proportional Storage Systems. *The Computer Journal*. The British Computer Society.

Lanyue Lu, Peter Varman, and Jun Wang. 2007. DiskGroup: Energy Efficient Disk Layout for RAID1 Systems. In *Proceedings of the International Conference on Networking, Architecture and Storage (NAS)*. IEEE Computer Society, 233-242.

Aqeel Mahesri and Vibhore Vardhan. 2004, Power Consumption Breakdown on a Modern Laptop. In *Proceedings of the 4th International Conference on Power-Aware Computer Systems*. Springer-Verlag Berlin, Heidelberg, 165-180.

B. Marsh, F. Douglis, and P. Krishnan. 1994. Flash Memory File Caching for Mobile Computers. In *Proceedings of the 27th Hawaii International Conference on System Sciences*.

Micron. 2013. Choosing the Right NAND. Micron. Retrieved June 15, 2013 from http://www.micron.com/products/nand-flash/choosing-the-right-nand.

Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write Off-Loading: Practical Power Management for Enterprise Servers. *ACM Transactions on Storage*, ACM, New York, New York, 4, 3(2008), Article No. 10.

Edmund Nightingale and Jason Flinn. 2004. Energy Efficiency and Storage Flexibility in the Blue File System. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*. USENIX Association, Berkeley, California, 363-378.

Athanasios Papathanasiou and Michael Scott. 2003. Energy Efficiency through Burstiness. In *Proceedings of the 5th IEEE Workshop on Mobile Computing Systems and Applications*. IEEE, 44-53.

Rakuten. 2013. Retrieved June 22, 2013, http://www.rakuten.com/prod/hgst-microdrive-3k6-hms360603d5cf00-3-gb-hard-drive-plug-in-module/201934858.html?listingId=270861696.

Tekla Perry. 2008. The Lady and the Li-ion. *IEEE Spectrum* (2008). Retrieved June 15, 2013 from http://spectrum.ieee.org/energy/renewables/the-lady-and-the-liion.

Eduardo Pinheiro and Ricardo Bianchini. 2004. Energy Conservation Techniques for Disk Array-Based Servers. In *Proceedings of the 18th Annual International Conference on Supercomputing*. ACM, New York, New York, 68-78.

Eduardo Pinheiro, Ricardo Bianchini, Cezary Dubnicki. 2006. Exploiting Redundancy to Conserve Energy in Storage Systems. In *Proceedings of the 2006 Joint International Conference on Measurement and Modeling of Computer Systems*, ACM, New York, New York, 15-16.

Pricewatch. 2013. Hard/Removable Drives. Pricewatch. Retrieved June 16, 2013 from http://www.pricewatch.com/hard_removable_drives.

Samsung. 2011. Samsung SSD SM825 2.5" SATA 3.0Gb/s E-MLC Data Center Edition. Samsung. Retrieved June 15, 2013 from http://www.samsung.com/us/business/oem-solutions/pdfs/SM825_Product%20Overview.pdf.

Samsung. 2012. SM825 FAQs. Samsung. Retrieved June 15, 2013 from

http://www.samsung.com/us/business/oem-solutions/pdfs/SSD-FAQs.pdf.

SanDisk, 2003. SanDisk SD Card Product Manual, Version 1.9. SanDisk document 80-13-00169. SanDisk. Retrieved June 16, 2013 from http://www.circlemud.org/jelson/sdcard/SDCardStandardv1.9.pdf.

Greg Schulz. 2007. Storage Industry Trends and IT Infrastructure Resource Management. Storage IO Group. Retrieved June 15, 2013 from http://www.storageio.com/DownloadItems/CMG/MSP_CMG_May03_2007.pdf.

Seagate. 2007. Barracuda 7200.10 Experience the Industry's Proven Flagship Perpendicular 3.5-inch Hard Drive. Retrieved June 16, 2013 from http://www.seagate.com/docs/pdf/datasheet/disc/ds_barracuda_7200_10.pdf.

Seagate. 2010. Constellation ES 2TB: Editor's Choice. Seagate. Retrieved June 16, 2013 from http://enterprise.media.seagate.com/2010/11/inside-it-storage/constellation-es-2tb-editors-choice.

Seagate. 2011. Momentus XT Product Manual, Revision D. Seagate. Retrieved June 16, 2013. http://www.seagate.com/files/staticfiles/support/docs/manual/100665905d.pdf

Seagate. 2012. Savvio® 10K.6 Enterprise Performance 10K HDD—The Perfect Performance Upgrade. Retrieved June 16, 2013 from http://www.seagate.com/files/www-content/product-content/savvio-fam/savvio-10k/savvio-10k-6/en-us/docs/savvio-10k6-data-sheet-ds1768-1-1210us.pdf.

ServerSupply. 2013. Retrieved Juen 22, 2013 from http://www.serversupply.com/products/part_search/request.asp?q=0A38016.

Pavel Somavat, Shraddha Jadhav, and Vinod Namboodiri. 2010. Accounting for the Energy Consumption of Personal Computing Including Portable Devices. In *Proceedings of the 1st International Energy-efficient Computing and Networking*. ACM, New York, New York, 141-149.

Mark Storer, Kevin Greenan, Ethan Miller, and Kaladhar Voruganti. 2008. Pergamum: Replacing Tape with Energy Efficient, Reliable, Disk-based Archival Storage. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies*. USENIX Association, Berkeley, California.

U. S. Energy Information Administration. 2013 Table 2.4 Average Retail Price of Electricity to Ultimate Customers by End-Use Sectors 2001 through 2011 (Cents per Kilowatthour), Retrieved June 15, 2013 from http://www.eia.gov/electricity/annual/html/epa_02_04.html.

Akshat Verma, Ricardo Koller, Luis Useche, and Raju Rangaswami. 2010. SRCMap: Energy Proportional Storage using Dynamic Consolidation. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies*. USENIX Association, Berkeley, California.

Charles Weddle, Mathew Oldham, Jin Qian, An-I Andy Wang, Peter Reiher, and Geoff Kuenning. 2007. PARAID: A Gear-Shifting Power-Aware RAID. *ACM Transactions on Storage* (TOS), 3, 3(2007), Article no 13.

Western Digital. 2012 WD Caviar® Green™ Desktop Hard Drives. Western Digital. Retrieved June 16, 2013 from http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-701229.pdf.

Xiaoyu Yao and Jun Wang. 2006. RIMAC: A Novel Redundancy-based Hierarchical Cache Architecture for Energy Efficient, High Performance Storage Systems. In *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*. ACM, New York, New York, 249-262.

Qingbo Zhu, Francis David, Christo Devaraj, Zhenmin Li, Yuanyuan Zhou, and Pei Cao. 2004. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In *Proceedings of the 10th International Symposium on High Performance Computer Architecture*. IEEE Computer Society,

Qingbo Zhu, Zhifeng Chen, Lin Tan, Yuanyuan Zhou, Kimberly Keeton, and John Wilkes. 2005. Hibernator: Helping Disk Arrays Sleep Through the Winter. In *Proceedings of the 20th Symposium on Operating Systems Principles*. ACM, New York, New York, 177-190.