Exploiting Redundancy to Conserve Energy in Storage Systems *

Eduardo Pinheiro
Rutgers University
edpin@cs.rutgers.edu

Ricardo Bianchini
Rutgers University
ricardob@cs.rutgers.edu

Cezary Dubnicki
NEC Labs America
dubnicki@nec-labs.com

ABSTRACT

This paper makes two main contributions. First, it introduces Diverted Accesses, a technique that leverages the redundancy in storage systems to conserve disk energy. Second, it evaluates the previous (redundancy-oblivious) energy conservation techniques, along with Diverted Accesses, as a function of the amount and type of redundancy in the system. The evaluation is based on novel analytic models of the energy consumed by the techniques. Using these energy models and previous models of reliability, availability, and performance, we can determine the best redundancy configuration for new energy-aware storage systems. To study Diverted Accesses for realistic systems and workloads, we simulate a wide-area storage system under two file-access traces. Our modeling results show that Diverted Accesses is more effective and robust than the redundancy-oblivious techniques. Our simulation results show that our technique can conserve 20-61% of the disk energy consumed by the wide-area storage system.

Categories and Subject Descriptors

D.4 [**Operating systems**]: Storage management

General Terms

Design, experimentation

Keywords

Energy management, energy modeling, disk energy

1. INTRODUCTION

Large storage systems, such as those of popular Internet services, outsourced storage services, and wide-area storage utilities, consume significant amounts of energy. For example, one report indicates that the storage subsystem can represent 27% of the energy consumed in a data center [16]. Even worse, this fraction tends to increase as storage requirements are rising by 60% annually [17].

*This research has been supported by NSF under grant #CCR-0238182 (CAREER award).

SIGMetrics/Performance'06, June 26–30, 2006, Saint Malo, France.

Because the energy consumption of storage systems is reflected in their electricity bills, several research groups have been seeking to reduce it [2, 3, 9, 14, 15, 19, 27, 28, 29]. However, only a few of these efforts [14, 15, 27] have explicitly leveraged redundancy and they did so in a limited context. Redundancy is present in all practical storage systems, since it is mostly through redundancy that these systems achieve high reliability, availability, and throughput. Redundancy is typically implemented by replicating the "original data" (as in mirrored disk arrays, cluster-based storage systems [13], or wide-area storage systems [21, 23]) or by storing additional information that can be used to reconstruct the original data in case of disk failures (as in RAID 5 or erasure-code-based wide-area storage systems [8, 10, 12]). We refer to the replicas and the additional information as "redundant data".

Our approach is to leverage this redundancy to conserve disk energy without performance degradation. In particular, we propose a technique called *Diverted Accesses* that segregates original and redundant data on different disks. We refer to these disks as original and redundant disks, respectively. The segregation allows the system to concentrate the requests on the original disks (under light or moderate demand for disk bandwidth), leaving the redundant disks idle. During the idle periods, the disks can be sent to low-power mode. The redundant disks only need to be activated in three cases: (1) when the demand for bandwidth is high; (2) when one or more disks fail; and (3) periodically to reflect changes made to the original data. In this last case, the writes to the original disks need to be logged until the corresponding redundant disks are activated.

Because the benefits of our technique vary with the amount and type of redundancy built into the system, our evaluation analytically quantifies the effect of redundancy on several system characteristics, including disk energy consumption and the potential of different techniques to conserve energy. More specifically, we develop energy models for Diverted Accesses and previous redundancy-oblivious techniques, and couple them with well-known models of reliability, availability, and throughput. Our modeling results show that Diverted Accesses can provide substantial energy savings across a wide range of redundancy, request rate, and write percentage parameters. Other techniques are only useful in small parts of this parameter space.

Designers can use our models to determine the best redundancy configuration for new storage systems. Our approach is to select the configuration that achieves the required throughput, reliability, and availability but consumes the least amount of energy. Our results show that non-intuitive redundancy configurations are often the best ones when all metrics are considered.

Finally, to demonstrate our technique for a system that requires high redundancy, we simulate a wide-area storage system with stable nodes and data replication under two realistic file-access traces.

The goal is to mimic a world-wide corporation that owns and operates its dedicated, distributed storage resources. Our results show that Diverted Accesses can reduce disk energy consumption by 20-61%. These results are close to those predicted by our models.

We conclude that considering redundancy can provide significant disk energy savings beyond those of previous techniques. Furthermore, we conclude that designing a storage system requires quantifying several metrics, which are all affected by the redundancy configuration. Our models are key in this design process.

The remainder of this paper is organized as follows. The next section discusses the related work and our contributions. Section 3 describes Diverted Accesses. Section 4 describes the energy models for the conservation techniques we study. Section 5 overviews previously proposed models for throughput, reliability, and availability, and discusses the selection of the best redundancy configuration. Section 6 presents our modeling results. Section 7 presents our real-trace results. Section 8 concludes the paper.

2. BACKGROUND AND RELATED WORK

2.1 Redundancy

Redundancy is typically implemented in storage systems through replication, parity schemes, or erasure codes. These methods can be defined in terms of their redundancy configurations by (n, m) tuples, where each block of data is striped, replicated, or encoded into n fragments, but only m fragments ($m \leq n$) are needed to reconstruct the data. For instance, a RAID 1 storage system is represented by $(n = 2, m = 1)$, since there are two copies of each block but only one copy is enough to reconstruct the block. A recent wide-area storage system based on erasure codes [10] used $(n = 48, m = 5)$ to resist massive correlated failures.

Several papers have studied these redundancy approaches, e.g. [1, 26]. Replication typically requires more bandwidth and storage space than the parity schemes. However, parity schemes can only tolerate small numbers of concurrent failures. Erasure codes require less bandwidth and storage than replication (for the same levels of reliability and availability), can tolerate more failures than parity schemes, but involve coding and decoding overheads.

Contributions. Our work complements these previous studies as we consider the impact of redundancy configuration on disk energy consumption and conservation.

2.2 Disk Energy Conservation

Several techniques have been proposed for disk energy conservation in storage systems.

Threshold-Based Techniques. The simplest threshold-based technique is *Fixed Threshold (FT)*. In FT, a disk is transitioned to low-power mode after a fixed threshold time has elapsed since the last access. Inspired by competitive policies, the threshold is usually set to the break-even time, i.e. the time a disk would have to be in low-power mode to conserve the same energy consumed by transitioning the disk down and back up to active mode. FT is one of the techniques to which we compare Diverted Accesses.

Data-Movement Techniques. In this category are those techniques that migrate or copy data across disks. The *Massive Array of Idle Disks (MAID)* technique [3] uses extra cache disks to cache recently accessed data. On each access miss in the cache disks, the accessed block is copied to one of the cache disks. If all cache disks are full, one of them evicts its LRU block to make space for the incoming block. The goal is to concentrate the accesses on the cache disks, so that the non-cache disks can remain mostly idle and, thus, be transitioned to low-power mode.

In contrast to the copy-based approach of MAID, *Popular Data Concentration (PDC)* [19] migrates data across disks according to frequency of access or popularity. The goal is to lay data out in such a way that popular and unpopular data are stored on different disks. This layout leaves the disks that store unpopular data mostly idle, so that they can be transitioned to low-power mode.

MAID and PDC use FT for power management.

Redundancy-aware Techniques. Only three works have exploited redundancy to conserve energy in storage systems: EERAID [14], eRAID [15], and RIMAC [27]. (EERAID is actually the only one that pre-dates our technical report on Diverted Accesses [20].) The most closely related work is eRAID, which is similar to Diverted Accesses but only considered RAID 1 storage. EERAID and RIMAC were targeted at RAID 5 organizations and as such are only capable of conserving $1/N$ of the energy of an array with N disks. In contrast with these three systems, we are interested in a broader range of storage systems, including those that are based on erasure codes, in which n may not be equal to $m + 1$ as in RAIDs 1 and 5. EERAID, eRAID, and RIMAC represent only a couple of points in this spectrum.

Other Techniques. Carrera *et al.* [2] and Gurumurthi *et al.* [9] proposed disks with more than one speed and showed that they can provide significant energy savings for different server workloads. Zhu *et al.* [28] exploited intelligent disk speed setting and data migration to conserve energy in arrays comprised of multi-speed disks without degrading response time. Carrera *et al.* also showed that a combination of laptop and SCSI disks can be even more beneficial in terms of energy, but only for over-provisioned servers. Papathanasiou and Scott [18] propose replacing server-class disks with larger arrays of laptop disks. Zhu *et al.* [29] proposed storage cache replacement algorithms that selectively keep blocks of data in memory, so that certain disks can stay in low-power mode for longer periods.

Contributions. Our work introduces Diverted Accesses, a novel and effective technique for leveraging redundancy. Further, our work presents energy models for FT, MAID, PDC, and Diverted Accesses that take redundancy configurations into account, and a case study of the application of Diverted Accesses in the context of a realistic wide-area storage system. Finally, ours is the first study of these techniques as a function of redundancy.

2.3 Storage System Design

Anderson *et al.* [5] proposed Ergastulum, a tool that quickly evaluates the space of possible data layouts and storage system configurations and finds a near-optimal design. Ergastulum uses performance models to determine whether a potential design is acceptable. Minerva [7] is similar but not as efficient as Ergastulum. Hippodrome [6] uses Ergastulum to adjust the design of the storage system as a result of dynamic changes in the workload.

Contributions. We extend the previous work on Ergastulum, Minerva, and Hippodrome by considering the reliability, availability, and energy consumption of different designs. Since we only focus on selecting the redundancy configuration (n, m), our space of possible designs is much more constrained than in these systems. For our purposes, exhaustive search works fine.

3. DIVERTED ACCESSES

Our approach to reducing energy consumption in storage systems leverages their redundancy. The key observation is that the redundant data is only read in two scenarios: (1) during periods of high demand for disk bandwidth, to increase performance; and (2) when disk failures occur, to guarantee reliability and availability.

Given this observation, it is clear that the redundant data need not be readily accessible at all times. For example, storage systems used for data backup exhibit long periods of low read loads during the day in between periods of intense write activity at night. As another example, regular file system activity in engineering environments is high during the day and low at night. Regardless of the type of storage system, if one or more disks fail, the only redundant data needed is that corresponding to the failed disks.

Unfortunately, current storage systems cannot take advantage of these characteristics to conserve disk energy. Our Diverted Accesses technique addresses this limitation by segregating the original data from the redundant data onto different subsets of disks: original and redundant disks, respectively. Even when all fragments contain redundant information, i.e. no fragment is strictly "original", we can still store m fragments of each block on a subset of the disks, which would act as the original disks.

The goal of Diverted Accesses is to keep redundant disks in low-power mode during periods of light and moderate offered disk load. More specifically, the handling of reads and writes depends on the offered load as follows:

- Reads are directed to the original disks only, unless the offered load is high enough that redundant disks need to be activated to help service it.
- Writes are performed on all disks, when the load is high. When the load is light or moderate, writes are directed to the m original disks immediately (they are synchronously performed on the media) but only propagated to the $n - m$ redundant disks periodically.

Writes and Reliability. The handling of writes to redundant data under light and moderate loads requires further discussion. These writes can be buffered in non-volatile memory (NVRAM) at storage appliances, disk-array controllers, or at the disk controllers themselves. While the buffer is being filled, the disk can stay in a low-power mode. When the buffer fills up (or is about to fill up), all redundant writes are performed on disk, completely emptying the buffer.

To implement this buffering at disk or JBOD controllers, hosts should identify these writes as special, "flush-when-full" writes. Storage appliances and RAID controllers can manage the redundant data they generate in the same way. (Original writes – and redundant writes during periods of high disk load – are not special and proceed normally.) This buffering of redundant writes does not affect the level of redundancy, as long as no controller or appliance is responsible for more than one redundant fragment of a block. Fortunately, this is the case in practice even in typical RAID organizations, for which $n = m + 1$.

Note that these changes to controller or appliance firmware are fairly minor, since several products on the market today (e.g., [4, 25]) already have NVRAM buffers for reliably optimizing writes. Unfortunately, these products transfer dirty blocks from the buffer to disk frequently, one/few at a time when the buffer fills up or when the disk is idle, preventing energy savings.

Nevertheless, we can avoid firmware changes by instead using off-the-shelf NVRAM cards or extra buffer disks [11]. This type of buffering does not affect the level of redundancy either, as long as no *host* is responsible for more than one redundant fragment of a block. This property is easily enforceable in typical distributed storage systems.

Thus, the best approach for redundant write buffering depends on the type of system. For single-host storage systems, the storage controller/appliance approach is the best, since it retains high reliability at the cost of simple firmware changes. For distributed storage systems, using off-the-shelf NVRAM is the best option, as it retains high reliability with standard firmware. Our high-level modeling in Section 4 is oblivious to where buffering is done, whereas the distributed storage system we simulate in Section 7 buffers writes in $n - 1$ NVRAMs.

A final reliability issue is the disk spin up/down cycles generated by Diverted Accesses. Manufacturers design their disks to support a certain number of cycles, e.g. 50000 for the IBM 36Z15 Ultrastar server disk. When possible, write buffers should be sized such that this limit is not exceeded during the disks' lifetime (typically three years). To guarantee that the limit is not exceeded, a simple approach is to keep track of the number of cycles imposed on each disk and leave it in high-power mode when the limit is reached. Our modeling in Section 4 does not include the notions of lifetime or cycle limits, whereas the distributed storage system and workloads of Section 7 generate only a fraction of the cycle limit on each disk.

Energy. Regardless of where redundant writes are buffered, this approach produces increased idle times at the redundant disks. However, when the buffer size is small or writes are frequent, the resulting idle times may still be too short to justify power-mode transitions. For this reason, we power-manage the redundant disks (as well as the original disks) with FT, rather than just sending these disks to low-power right after buffer flushes. Interestingly, write buffering as in Diverted Accesses does not increase disk idle times for the other conservation techniques we study, since they do not segregate read and write accesses.

With FT as its primitive power-management technique, Diverted Accesses produces significant energy savings in most scenarios, as we shall demonstrate. However, note that in parity-based storage systems with "small write" workloads, the energy benefits of Diverted Accesses are reduced, since the redundant disk(s) need to be activated before the parity data can be computed. Thus, the designer needs to understand the workload before applying Diverted Accesses in parity-based systems.

Performance. To avoid performance degradation, we define the load as high when the set of original disks is not enough to provide the required disk bandwidth. This approach works by keeping track of the average utilization of the disks and activating the redundant disks when the original disk are almost fully utilized.

During periods of light and moderate load, the write buffering at the redundant disks has no real effect on performance for two reasons: (1) only writes are directed to redundant disks during these periods; and (2) standard controllers and appliances already acknowledge writes right after they are written to memory.

However, Diverted Accesses does pose a problem for parity-based storage systems in the presence of small writes. For example, applying Diverted Accesses to RAID 5 would transform it into RAID 4. Under high disk load, the bandwidth requirements of the parity disk in RAID 4 may degrade performance for small writes. Again, this highlights the importance of understanding the workload before applying Diverted Accesses in parity-based systems.

Implementation. Diverted Accesses can be implemented entirely at the storage system level (i.e., underneath the file system layer), since it is solely concerned with laying data out across the disks, assessing the offered disk load, and directing disk accesses.

In terms of disk technology, Diverted Accesses can operate effectively with standard disks since disk spin ups and downs occur infrequently and typically in the background of read accesses. Thus, our technique has little use for multi-speed disks.

Scope of This Study. In general, Diverted Accesses is applicable whenever there is redundancy: from data-center storage systems using RAID to wide-area storage systems, such as world-wide digital libraries and storage utilities, using replication or erasure codes. A large fraction of our study focuses on wide-area storage systems (owned by a single institution or corporation), as these systems require significant redundancy and, thus, have a more serious need for redundancy-aware energy conservation. In particular, our energy models are general, whereas our modeling results and simulations focus on the high redundancy configurations that are necessary in wide-area storage systems, e.g. [8, 10, 12].

Interestingly, an important aspect of wide-area systems is that they are accessed by a large population of independent clients across the Internet. This characteristic is the reason why we consider throughput rather than response time; in these systems, even large degradations in response time are typically overwhelmed by the latency of multiple network transfers per request. Furthermore, our study focuses mostly on data allocation, data access, and energy issues, to assess the potential energy savings achievable with Diverted Accesses; other aspects of storage systems are beyond the scope of the paper. Finally, we focus solely on disk energy in this paper. However, Diverted Accesses can be applied to entire nodes in a distributed storage system. In this case, we could send memories and processors to low-power mode as well.

4. MODELING ENERGY

We model block-based storage systems comprised of identical disks. The disks may be attached to one or more servers, but our models are independent of such issues. Besides the characteristics of the disks, the models' inputs are the redundancy configuration (n, m) and the percentage of reads and writes in the workload.

4.1 Overview

We define D as the number of disks required to store the data without any redundancy. Given a redundancy configuration, we define N, the number of required disks to store all data (original plus redundant), $N(n, m) = (n/m)D$. For simplicity, we refer to N instead of $N(n, m)$.

Each disk has a power consumption of P_h Watts when powered on and ready to service requests (high power mode) and P_l when in standby mode, not able to service requests (low power mode). (Note that most high-performance, server-class disks – typically SCSI drives – do not offer more than one power-saving mode.) A disk spin up takes time T_u and energy E_u, whereas a disk spin down takes time T_d and energy E_d. A full transition from high-power mode to low-power mode and back to high consumes $T_t = T_d + T_u$ time and $E_t = E_d + E_u$ energy.

Each request to the system is assumed to have size $blockSize$. Internally, disk data is accessed in fragments of size $fragSize$, which is defined as $blockSize/m$. On each access, the disks take time S to seek to the appropriate track and time R to rotate to the desired sector. A fragment of data is transferred at a nominal rate X. We do not model the energy consumed by disk accesses, since previous works have demonstrated that it is a small fraction of the overall disk energy, even in busy systems [9, 22].

As in previous works [9], we only model the request traffic that reaches the storage system (i.e., beyond any main memory caches) and assume that request inter-arrival times are drawn from a Pareto distribution with average $1/requestRate$, even though our models work with any distribution. Block requests can be reads or writes with probabilities $1 - p_w$ and p_w, respectively.

To estimate energy (average power) we do the following: (1) Draw a request inter-arrival time t; (2) For each energy conservation technique, calculate the average idle time per disk, based on the inter-arrival time; (3) For each technique, estimate the average power (as a proxy for energy) for this idle time based on the underlying power management mechanism and any other technique-specific parameters; (4) Repeat three previous steps until enough samples have been drawn to allow the average inter-arrival time to converge to $1/requestRate$; (5) Compute the overall average power of each technique by dividing the sum of the average powers by the number of inter-arrival times drawn.

Symbol	Description
D	Number of required disks without redundancy
N	Number of required disks with redundancy
P_h	Disk power in high-power mode
P_l	Disk power in low-power mode
T_u	Time to transition to high-power mode
T_d	Time to transition to low-power mode
T_t	Time to transition down and up again
E_u	Energy to transition to high-power mode
E_d	Energy to transition to low-power mode
E_t	Energy to transition down and up again
$blockSize$	Size of blocks
$fragSize$	Size of fragments
S	Average disk seek time
R	Average disk rotation time
X	Disk transfer rate
$requestRate$	Arrival rate of block requests
p_w	Probability of block write requests
D_{maid}	Number of cache disks (MAID)
m_{maid}	Miss rate of cache disks (MAID)
β	Disk popularity coefficient (PDC)
c	Storage system coverage (PDC)
$wbSize$	Size of write buffer per redundant disk (DIV)
$batchSize$	Size of write batch per redundant disk (MAID+DIV)

Table 1: Summary of model parameters and their meanings.

Note that each inter-arrival time t we draw in step 1 above does not represent a single disk access. Rather, it represents a period in which requests arrive with average interval t and access disks in round-robin fashion. Although energy conservation techniques may entail widely different energy consumptions depending on the length of each interval, using the average interval as a proxy for each period is appropriate because we assume that the intervals associated with each average interval are within the same range (the ranges are defined by the "cases" of the next subsection). For example, this means that the intervals during a period of high load, i.e. low average t, are shorter than the disk break-even time. The results of Section 7.1 show that our models are accurate for real workloads, i.e. in the absence of these assumptions.

Table 1 summarizes the model parameters.

4.2 Energy Conservation Techniques

We model six different energy conservation techniques: Fixed Threshold (FT), Popular Data Concentration (PDC), Massive Array of Inexpensive Disks (MAID), Diverted Accesses (DIV), the combination of PDC and DIV (PDC+DIV), and the combination of MAID and DIV (MAID+DIV). The latter five techniques use FT as their low-level power-management technique. We compute the energy savings of each technique with respect to an energy-oblivious (EO) system that keeps all disks at P_h at all times.

FT. In FT, disks are transitioned to low-power mode after an idleness threshold T. We set T to the break-even time, i.e. $T = E_t/(P_h - P_l)$. For each inter-arrival time t, we can approximate the idle time per disk I as:

$$\frac{Nt}{np_w + m(1 - p_w)} \tag{1}$$

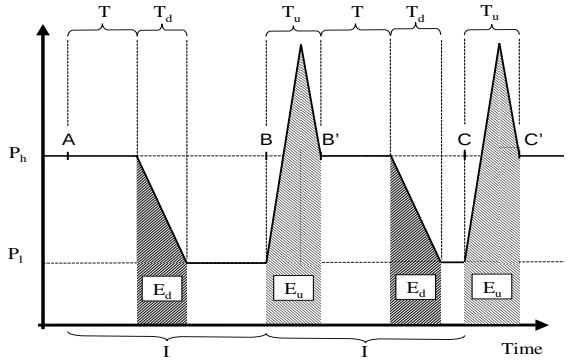

Figure 1: $I \geq T + T_u$**. Accesses arrive at times A, B, and C. Accesses are actually performed at times A, B', and C'.**

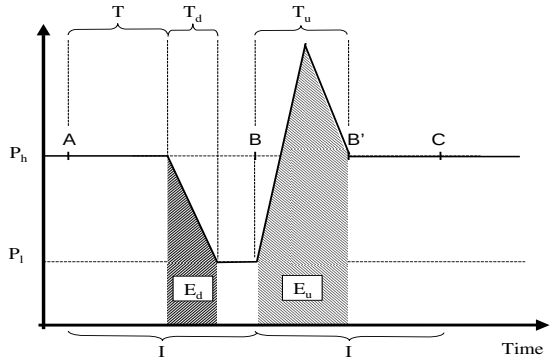

Figure 2: $T \leq I < T + T_u$**. Accesses arrive at times A, B, and C. Accesses are actually performed at times A, B', and C.**

Since the average number of disk accesses per second is $(np_w + m(1-p_w))/t$, the idle time at each disk is simply the inverse of the access rate times the number of disks.

The average power for each idle time is then defined by the three cases below:

$$\begin{array}{ll} P_h N, & case1 \\ (TP_h + (I - T - T_t)P_l + E_t)N/I, & case2 \\ ((I - T_u + T)P_h + (I - T - T_d)P_l + E_t)N/(2I), & case3 \end{array} \quad (2)$$

where $case1$ represents the scenario in which $I < T$, $case2$ represents $I \geq T + T_u$, and $case3$ represents all other cases (i.e., $T \leq I < T + T_u$). (To avoid more cases, our modeling extends idle times that end during a transition to low-power mode until the end of the transition.)

More intuitively, the top equation represents the scenario in which idle times are too short and do not trigger a transition to low-power mode. The energy is simply that of all disks on, so the average power will be $P_h N$. The middle equation represents the scenario in which there is enough time for the disks to transition to low-power mode, perhaps spend some time in low-power mode, and transition back. In this case, all idle times except the first are effectively reduced by the spin up time. Figure 1 illustrates this scenario. The figure shows accesses arriving at times A, B, and C; actual disk accesses occur at times A, B', and C'. Note that, after the first idle time, the behavior between B and C will consistently be repeated while this idle time is in effect. Thus, we use the period between B and C in computing the average power. The last equation represents the scenario in which there is not enough time for the full transition to and from low-power mode. In this case, the next idle time is shortened to a point that no power-mode transitions can happen. Figure 2 illustrates this scenario. The behavior between A and C will be consistently repeated while this idle time is in effect. For this reason, we use the entire period (two idle times) in computing the average power.

MAID. In this strategy, D_{maid} extra disks are used to cache recently-accessed files. Upon each block request, if the block is not yet on one of the cache disks, the corresponding fragments are accessed at the non-cache disks and also copied to one of the cache disks. To avoid high latencies, requests may bypass the cache disks during periods of high load. Thus, the maximum throughput of a MAID system is that of $N + D_{maid}$ disks. In terms of energy, this high load scenario leads to extremely short idle times and an average power of $(N + D_{maid})P_h$. Below, we model MAID under light and moderate loads.

Modeling MAID requires knowledge of the temporal locality of accesses to files. As an approximation, we assume that we know the fragment cache miss ratio m_{maid} of the cache disks. We can then estimate the idle times of the cache disks and the non-cache disks. To conserve energy, we can leverage the cache disks to accumulate the writes to a non-cache disk until a cache disk miss (a read) accesses it. At that point, the accumulated writes can be performed on the non-cache disk. This approach promotes energy conservation at the cost of lower reliability.

We need to calculate two distinct idle times: for the MAID caches (I_{cache}) and for the non-cache disks (I_N).

$$\begin{array}{l} I_{cache} = (D_{maid}t)/((np_w) + (m(1-p_w))) \\ I_N = (Nt)/(m(1-p_w)m_{maid}) \end{array} \quad (3)$$

The average power consumed by the cache disks can be computed as in FT (equations 2), except that we need to replace I_{cache} for I and D_{maid} for N. The average power of the non-cache disks can also be computed using those equations, as long as we replace I_N for I. The overall average power is the sum of the cache and non-cache powers.

Note that our modeling of MAID is simplistic. In the extreme cases in which there are no read accesses ($p_w = 1$) or all read accesses hit the cache disks ($m_{maid} = 0$), the idle time of the non-cache disks is modeled as infinite. In other words, we assume the cache disks to have infinite write buffering capacity. Further, we assume that the energy spent in copying data to the cache disks is negligible. Our goal with these simplifications is to provide an upper bound on the energy conservation potential of MAID.

PDC. Like FT and MAID, the original PDC proposal [19] did not consider redundancy explicitly. However, unlike the other techniques, PDC can hurt reliability significantly if it is applied to all fragments arbitrarily.

To avoid this problem, we modify PDC to migrate data in such a way that the n fragments of each block remain on different disks. Over time, the most popular fragments would then be stored on the "first" set of n disks, the second most popular fragments would be stored on the "second" set of n disks and so on.

In order to model PDC, we need to introduce the notions of disk popularity and file system coverage. Previous research has shown that the popularity distribution of various Web and network workloads follows a power law. In particular, Zipf's power law with coefficient α to describe the popularity of files. Zipf's law states that the probability of a file being accessed is proportional to $1/r^{\alpha}$, where r is the rank (popularity) of the file and α is the degree of skewness of popularity. For example, when $\alpha = 0$, all files are equally likely to be accessed. The larger α is, the more heavy-tailed the distribution is. Based on a similar idea, we use Zipf's power law with coefficient β to describe the popularity of the groups of n disks in steady state, i.e. when all fragments have been migrated to their best locations.

File system coverage represents the percentage of blocks that are actually accessed in a given period of time (a day, a week, or the length of a workload).

To compute the idle times, we need to take the disk popularity and the coverage c into account. We do so using an "idleness weight" w for each set i of n disks:

$$w_i = (Nc/n)\left(1 - \frac{1/i^{\beta}}{\sum_{j=1}^{\lceil Nc/n \rceil} 1/j^{\beta}}\right) \quad (4)$$

The idleness weight of a group of disks ranges from 0 to Nc/n (the number of groups) and is proportional to the fraction of accesses that is not directed to the group (the term inside parentheses on the right of equation 4). In other words, the most popular group will have a weight that tends to 0, whereas the least popular group will have a weight that approaches Nc/n.

These factors can be used to weight the idle time I from equation 1 in computing the average power consumed by the disks for each idle time in PDC:

$$\sum_{i=1}^{\lceil Nc/n \rceil} \begin{cases} w_i P_h n, & c1 \\ (TP_h + (Iw_i - T - T_t)P_l + E_t)n/I, & c2 \\ ((Iw_i - T_u + T)P_h + (Iw_i - T - T_d)P_l + E_t)n/(2I), & c3 \end{cases} \quad (5)$$

where $c1$ represents the scenario in which $Iw_i < T$, $c2$ represents $Iw_i \geq T + T_u$, and $c3$ represents all other cases. Intuitively, this equation sums up the average power consumed by each group of disks for each idle time, noting that each group may actually fall in a different case.

Our modeling of PDC is optimistic for three reasons. First, our coverage parameter assumes that all write accesses are updates of existing data, rather than writes of new data. Second, we assume the energy consumed in data migration to be negligible. Third, due to the complexity of the its data layout, PDC is essentially impractical in the presence of redundancy; it would be very hard to implement in single-node storage systems, and even harder in distributed storage systems. Nevertheless, our goal is to compute an upper bound on the potential benefits of PDC.

DIV. Diverted Accesses stores the original data on D disks and the redundant data on $R = N - D$ disks. The redundant disks can be sent to low-power mode, until they are required for reliability or to provide higher bandwidth under high load. Again, idle times are short under high load, leading to an average power consumption of NP_h. Next, we model DIV under light and moderate loads.

First, we compute the idle times on the original disks, I_D:

$$I_D = \frac{Dt}{m} \quad (6)$$

Note that all (read and write) requests translate into accesses to the D disks. Writes are also buffered, so the expected idle time on the redundant disks (I_R) is the expected time for the write buffer to fill up times R.

$$I_R = \frac{Rt * wbSize}{blockSize * (n-m)p_w}, \quad (7)$$

such that $wbSize \geq blockSize$.

With these idle times, the average power for DIV can be computed as the sum of the power consumed by the original and the redundant disks. These average powers can be computed the same way as in FT (equations 2) with minor differences. For the original disks, the average power is:

$$\begin{cases} P_h D, & c1 \\ (TP_h + (I_D - T - T_t)P_l + E_t)D/I_D, & c2 \\ ((I_D - T_u + T)P_h + (I_D - T - T_d)P_l + E_t)D/(2I_D), & c3 \end{cases} \quad (8)$$

where $c1$ represents the scenario in which $I_D < T$, $c2$ represents $I_D \geq T + T_u$, and $c3$ represents all other cases.

For the redundant disks, the average power is:

$$\begin{cases} P_h R, & c1 \\ (TP_h + (I_R - T - T_t)P_l + E_t)R/I_R, & c2 \\ ((I_R - T_u + T)P_h + (I_R - T - T_d)P_l + E_t)R/(2I_R), & c3 \end{cases} \quad (9)$$

where $c1$ represents the scenario in which $I_R < T$, $c2$ represents $I_R \geq T + T_u$, and $c3$ represents all other cases.

MAID+DIV. MAID can be combined with DIV. In MAID+DIV, the idea is to place a few cache disks in front of DIV-structured disks. D_{maid} extra disks are used to cache recently accessed blocks like in MAID, whereas the $D + R = N$ non-cache disks are organized as in DIV.

Given the extra cache disks, we can accumulate writes aggressively on them, as in MAID. The writes are propagated to the non-cache disks on read misses on the cache disks (original disks) or periodically in a large batch (redundant disks). The resulting idle times, I_{cache}, I_D, and I_R, for MAID+DIV are slight variations of these times in MAID and DIV. Specifically, I_{cache} is defined exactly as in equation 3; I_D is also defined as in equation 3, except that N is replaced by D; and I_R is defined as in equation 7, except that $wbSize$ is replaced by $batchSize$.

The average power consumed by the cache disks in MAID+DIV is computed as in MAID, whereas the average power of the non-cache disks is computed as in DIV. The overall average power is the sum of these components, as the energy used by data copying is assumed negligible.

PDC+DIV. Here, we combine PDC with DIV. The idea is to segregate original and redundant fragments and only migrate the original ones according to popularity. Migration is performed in such a way that the m fragments remain on different disks. Over time, the most popular fragments would then be stored on the "first" set of m original disks, the second most popular fragments would be stored on the "second" set of m original disks and so on.

Due to the concentration of accesses, the computation of the idle times uses idleness weights w for each group i of m original disks, just as in our modeling of PDC (equation 4) except that N is replaced by D. These weighting factors can be used to weight the idle time I_D from equation 6 in computing the average power consumed by the original disks for each idle time in PDC+DIV. This power can be computed as in equation 5, except that n is replaced by m and I is replaced by I_D.

The average power consumed by the redundant disks can be computed exactly as in DIV. The overall average power is the sum of these two original and redundant powers. Again, we assume all writes to be updates to existing data and the energy of data migration to be negligible.

5. DESIGNING REAL SYSTEMS

In this section, we present well-known models for reliability, availability, and performance. Using these models along with our energy models, we can select the best redundancy configuration for new storage systems.

5.1 Reliability and Availability

We define the reliability $r(x)$ of each disk as the probability that the disk does not lose or damage the data it stores (e.g., due to a permanent mechanical drive failure) within time x. To simplify the notation, we refer to $r(x)$ simply as r. We define the availability a of each disk as the probability that the disk is not temporarily inaccessible (e.g., due to a loose cable or a period of offline maintenance) at a given time. We assume disk faults to be independent. Given these assumptions, the combinatorial models that quantify

reliability and availability as a function of the redundancy configuration (n, m) are similar [24]. Equation 10 is the availability model when at least m fragments must be available upon a disk access. A is typically close to 1, so availability is often referred to in terms of the number of nines after the decimal period. The reliability model is the same, except that a is replaced by r.

$$A = \sum_{i=0}^{n-m} \left(\begin{array}{c} n \\ i \end{array} \right) a^{n-i} (1-a)^i \tag{10}$$

Note that, although our independence assumption and simple combinatorial models produce only rough approximations for RAID systems, they are more accurate for distributed storage systems, which are less likely to experience correlated faults.

Diverted Accesses. The reliability and availability definitions above apply to DIV as well, even though it uses NVRAM for temporary storage of recently written redundant data. The reason is that battery-backed RAM can achieve similar reliability to disks in real storage systems, as long as administrators periodically replace batteries. Along the same lines, NVRAM and disks should exhibit similar availabilities, since downtimes are likely to be dominated by the unavailability of their supporting components, such as hosts and cabling. The key observation is that the use of NVRAM in DIV does not reduce the level of redundancy in typical systems, as mentioned in Section 3.

5.2 Performance: Maximum Throughput

Our performance model is the aggregate maximum throughput of the disks in the system, given a fixed configuration. Recall that the number of disks is a function of the redundancy configuration, $N = Dn/m$.

The maximum throughput is defined by the redundancy configuration and the disk bandwidth for the workload:

$$\begin{array}{rcl} fragSize & = & blockSize/m \\ delay & = & S + R + fragSize/X \\ width_r & = & N \times fragSize/delay \\ width_w & = & N \times (fragSize/delay) \times m/n \\ P = totalBW & = & p_w width_w + (1.0 - p_w) width_r \end{array} \tag{11}$$

where $blockSize$ is the maximum between the block size exported by the storage system interface and the weighted mean of a distribution of request sizes (each size being a multiple of the block size); $fragSize$ is the size of each fragment; $width_r$ and $width_w$ represent the *effective* bandwidth for reads and writes, respectively; as previously defined, S, R, and X are the average seek time, the average rotational delay, and the disk transfer rate, respectively; and p_w is the probability of writes.

These equations apply to all energy conservation techniques, even though disks that are in low-power mode limit the maximum throughput of the system. However, all techniques activate all disks when the offered load requires it.

5.3 Putting It All Together

Determining the best redundancy configuration for a storage system involves meeting its storage capacity, reliability, availability, and throughput requirements for the least amount of energy. More specifically, we need to explore the space of potential configurations (n, m) and energy conservation techniques, trying to minimize energy (or average power), subject to the constraints on storage capacity, reliability, availability, and throughput. Given the relatively small search space of $n * m$ possible configurations, for example in $n \in [1..16]$ and $m \in [1..16]$, the problem becomes easy to solve by enumeration (and modeling, of course).

Parameter	Default Value
Request Rate ($requestRate$)	64 reqs/sec
Write Ratio (p_w)	33%
Disk Popularity (β, PDC)	1.0
File System Coverage (c, PDC)	70%
# MAID Cache Disks (D_{maid})	$0.1N$
MAID Fragment Miss Ratio (m_{maid})	40%
MAID+DIV Batch Size ($batchSize$)	1 GB
DIV Write Buffer Size ($wbSize$)	4 MB
Block size ($blockSize$)	8 KB
# Disks without Redundancy (D)	64
# Disks with Redundancy (N)	Varies
Disk reliability (r)	0.999
Disk availability (a)	0.99
High Power (P_h)	10.2 W
Low Power (P_l)	2.5 W
Avg. Seek Time (S)	3.4 ms
Avg. Rot. Time (R)	2.0 ms
Transfer Rate (X)	55.0 MB/sec
Idleness Threshold (T)	19.2 secs
Spin up Time (T_u)	10.9 secs
Spin down Time (T_d)	1.5 secs
Energy transition down+up (E_t)	148.0 J

Table 2: Configurable parameters and their default values.

6. MODELING RESULTS

In this section, we analyze the tradeoffs between different (n, m) configurations, in terms of energy, reliability, availability, and performance. Because the parameter space has at least 6 dimensions – n, m, energy, reliability, availability, and performance –, it is impossible to visualize it all at the same time. Thus, we plot 2-D graphs showing the interesting parts of the space.

We computed the energy results in this section using a synthetic workload generated as follows. We draw 10,000 inter-arrival times from a Pareto distribution with a default average of 64 requests per second and an infinite variance. Requests are for 8-KB blocks. The disk parameters are based on the IBM 36Z15 Ultrastar model. The disk reliability assumes an exponential distribution of faults during one year and an MTTF of 2 million hours. We assume a low disk availability of 0.99 to encompass not only the availability of the disk itself, but also that of its supporting components, such as the power supply, controllers, and cabling. The modeling of DIV assumes the same reliability and availability values for disks and NVRAM. Table 2 summarizes the default parameter values we used. Note that the values for P_h, P_l, T_u, T_d, E_t were actually measured, rather than taken from our disk's datasheet.

Although we study a wide range of parameter values, we carefully selected the default values for the workload-related parameters. In particular, the defaults for $requestRate$ and p_w lie within the ranges created by our two realistic access traces (see Section 7) for these parameters. Further, we selected the default for D_{maid} based on simulations of the cache disk miss rate under our traces; $0.1N$ cache disks leads to the best tradeoff between miss rate and number of disks for our traces. The default value for m_{maid} lies in the middle of the range created by our two traces for the default number of cache disks. Our traces do not include information about coverage, so we arbitrarily chose 70% as its default value but quantify the effect of changes in this parameter explicitly. Finally, we do not have definitive information about β either; we set it to 1.0, but have found that this parameter has a negligible impact on the energy gains achieved by PDC and PDC+DIV.

6.1 Energy

We now evaluate the effectiveness of the energy conservation techniques, as a function of the redundancy configuration. We start

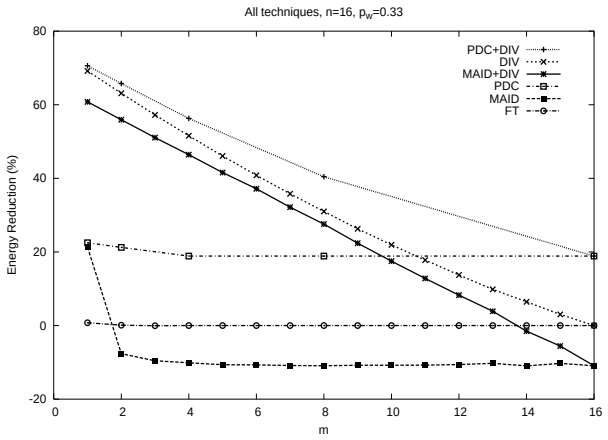

Figure 3: Energy savings for $n = 16$.

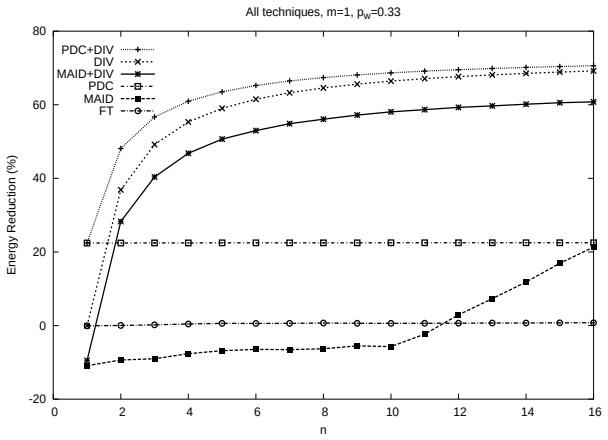

Figure 4: Energy savings for $m = 1$.

by assuming the number of disks N increases (decreases) as the amount of redundancy – n/m – in the system increases (decreases). When $n = m$, $N = D = 64$.

Figures 3 and 4 assess the impact of varying m (for $n = 16$) and n (for $m = 1$), respectively, on the energy savings produced by each technique. We compute the energy savings with respect to an energy-oblivious (EO) storage system. Note though that the percentages of savings represent different absolute energy consumptions, as the number of disks is not fixed. For example, EO consumes 10445 W average power for $m = 1$ and 653 W for $m = 16$. Note also that some configurations are infeasible under PDC and are thus not included in the graphs. These configurations lead to fractional numbers of disk sets (see Section 4).

From figure 3, we can see that FT provides no energy savings across the space. Under the default request rate, there is not enough idle time for energy conservation in FT. In contrast, MAID conserves energy for small m but degrades as we increase this parameter. In fact, MAID conserves a substantial amount of energy when $m = 1$, since the idle time on the non-cache disks is high enough under the default request rate and MAID fragment miss rate. However, when $m \geq 2$, MAID consumes more energy than EO because the cache disks do not filter enough accesses, and thus do not justify their energy overhead. PDC produces roughly the same energy savings, regardless of m. The reason is that these savings result from the constant coverage we assume.

DIV behaves very well until m starts approaching n; larger m means that there is less redundancy, so fewer disks can be in low-power mode during read operations. Comparing the techniques, we find that DIV conserves more energy than MAID regardless of m. In comparison to PDC, DIV conserves more energy when the amount of redundancy is high, i.e. $n >> m$.

Combining MAID or PDC with DIV improves these techniques substantially. MAID+DIV behaves similarly to DIV and better than MAID for highly redundant configurations. With little redundancy, MAID+DIV behaves worse than DIV, consuming more energy than EO. PDC+DIV conserves more energy than PDC for highly redundant systems. In contrast, PDC+DIV conserves more energy than DIV when redundancy is limited. Overall, PDC+DIV behaves best, as it combines the benefits of DIV and PDC under high and low redundancy, respectively.

From figure 4, we see that all techniques and combinations benefit from increases in redundancy, except for PDC and FT. MAID consumes more energy than EO for small n, but produces savings when the system becomes highly redundant. These savings come from the increase in the number of cache disks that results from increasing n (recall that we assume D_{maid} to be a fixed percentage of N). DIV and its combinations produce the most significant and consistent energy savings, since DIV allows an increasing number of disks to be sent to low-power mode with increasing redundancy. PDC+DIV performs best as explained above.

Impact of Workload Characteristics. We now study the impact of different workload characteristics on energy conservation, starting with the request rate. Figure 5 shows the effect of the request rate on a configuration with $n = 8, m = 1$. We may see such a highly redundant configuration in wide-area storage systems, in which many types of events can cause parts of the system to become inaccessible, including network partitions, external attacks, and correlated failures [10]. Recall that our previous results assumed the default request rate of 64 reqs/s.

The figure shows that FT and MAID only conserve energy for very low request rates (less than 32 and 64 reqs/s, respectively). For higher rates, more sophisticated organizations are necessary to increase idle times. Both PDC and DIV provide savings, but DIV conserves between two and three times more energy than PDC in this range of request rates. Adding DIV to MAID or PDC improves their behavior significantly, especially in the case of MAID. In fact, MAID+DIV degrades very slowly with the increase in request rate, due to our optimistic assumptions regarding the write accesses (infinite write buffering at cache disks and large write batch size). Under these assumptions, the cache disks and the original disks remain active, but the redundant disks can be in low-power mode most of the time. The overall trends we see continue until the request rate becomes so high that all disks are needed (not shown).

We now turn to the percentage of writes in the workload. Figure 6 shows a request-rate graph for the techniques that differentiate reads and writes, MAID and the DIV variants, with $n = 8, m = 1$. We omit PDC+DIV because its results follow the exact same trends as those of DIV, but with slightly larger savings. The figure plots results for three p_w settings: 0.33, 0.66, and 0.99. Recall that our other results assume $p_w = 0.33$.

Interestingly, MAID and DIV behave very differently. MAID conserves the most energy for write-mostly workloads, due to our optimistic assumptions for this technique; when reads are a significant fraction of the accesses, MAID actually consumes more energy than EO. In contrast, DIV does best for read-mostly workloads, as the redundant disks can be kept in low-power mode most of the time. Nevertheless, DIV also does well for write-mostly workloads, since the redundant disks are only activated when the write buffers fill up. MAID+DIV behaves well for all write rates by combining the best characteristics of MAID and DIV.

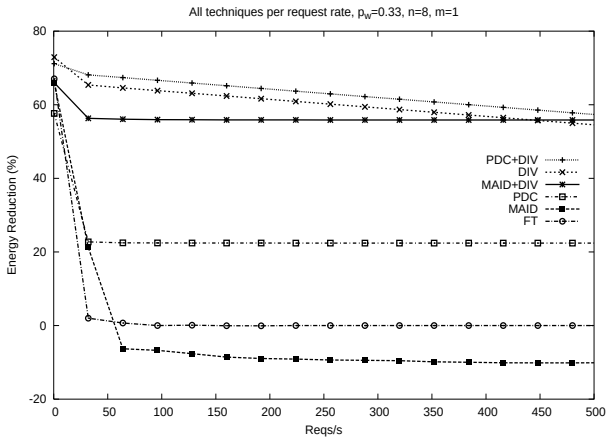

Figure 5: Savings per request rate for $n = 8, m = 1$.

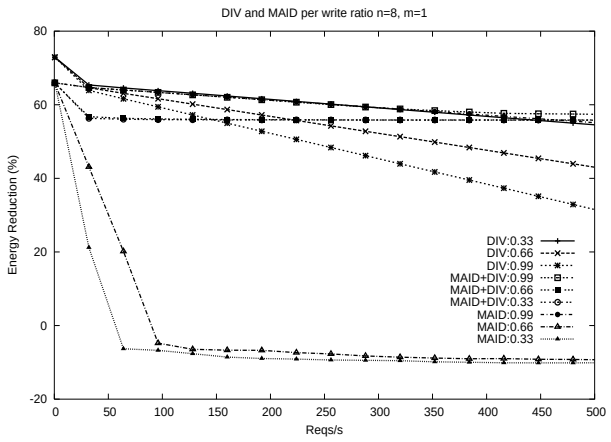

Figure 6: Savings per write percent for $n = 8, m = 1$.

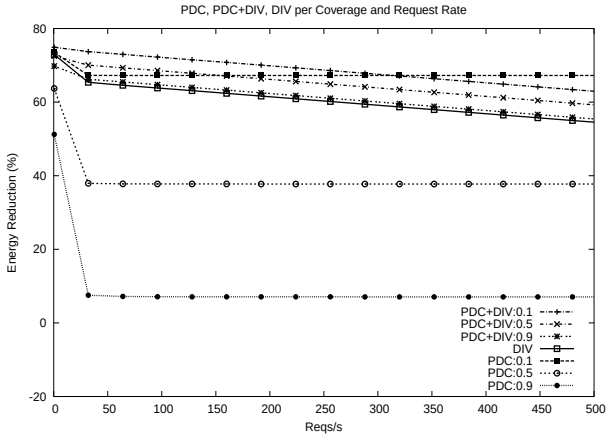

Figure 7: PDC and PDC+DIV savings as a function of coverage (10%, 50%, 90%) for $n = 8, m = 1$.

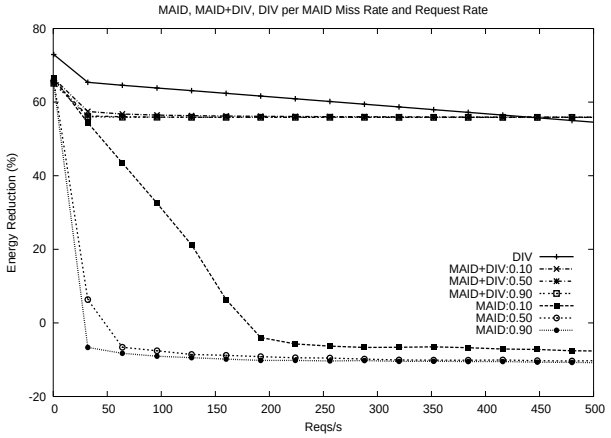

Figure 8: MAID and MAID+DIV savings as a function of cache miss rate (10%, 50%, 90%) for $n = 8, m = 1$.

Figure 7 shows the behavior of PDC and PDC+DIV, as a function of coverage and request rate. Recall that coverage is the percentage of blocks referenced (read or written) over all stored blocks. By default, we have been assuming a coverage of 70% and a request rate of 64 reqs/s. PDC benefits significantly from smaller coverages. In fact, PDC becomes the best technique for very low coverage and high request rate. PDC+DIV also benefits from smaller coverages, but the improvements over DIV are meager when compared with the benefits a smaller coverage has on PDC alone. Note also that the savings potential of data migration is small, since the savings of PDC+DIV with 90% coverage are very similar to those of DIV.

Figure 8 shows the behavior of MAID and MAID+DIV, as a function of cache miss and request rates. The miss rate depends on the workload and on the number of MAID caches used. Here, we study how the miss rate affects energy conservation, without concern for how it is achieved, i.e. without changing other parameters. The figure shows that the miss rate has a significant effect on MAID but not on MAID+DIV. MAID improves with lower miss rate, since lower rates increase the idle time at the non-cache disks, allowing them to be kept in low-power mode longer. MAID+DIV only benefits from extremely low miss rates (much lower than 10%), at which point the original (non-cache) disks can be sent to low-power mode for a long enough period. Again, this shows that MAID+DIV behaves well due to DIV and our favorable modeling assumptions.

Impact of Write Buffer and Write Batch Sizes. The size of the write buffer is very important for DIV when workloads exhibit a non-trivial fraction of writes. For these workloads, DIV may not be able to conserve energy in the absence of a write buffer. As we increase the size of the write buffer, energy savings increase quickly at first but later taper off, as the redundant disks remain idle most of the time. Under our default parameters, the energy savings start to taper off with a 2-MB write buffer. We study the impact of buffer sizes in more detail in Section 7.

The behavior of MAID+DIV is significantly affected by a similar parameter, the write batch size. The batch size can be much larger than the buffer size in DIV, since writes are buffered on the cache disks in the former technique. When the batch size is the same as the buffer size, MAID+DIV conserves less energy than DIV, due to the energy overhead of the cache disks.

Impact of Fixed Number of Disks. We just examined the effect of the redundancy configuration for a variable number of disks. Here, we evaluate the impact of a fixed number of disks, $N = 64$. Under this assumption, all but one configuration ($n = 16, m = 1$) will have unused storage space.

Fixing $N = 64$ leads to similar trends as in figures 3 and 4. Increases in n tend to increase the DIV savings initially, since larger n increases the number of redundant disks. At some point however, write accesses start limiting the idle time of a large number of disks. Increasing m reduces the DIV savings consistently, since this effectively increases the number of original disks.

(n, m)	N	P (MB/s)	A	Tech	E (W)
$(8, 5)$	32	8.1	0.999999	EO	326
$(8, 5)$	32	8.1	0.999999	FT	326
$(8, 5)$	32	9.1	0.999999	MAID	367
$(8, 5)$	32	8.1	0.999999	PDC	265
$(8, 5)$	32	8.1	0.999999	DIV	267
$(8, 5)$	32	9.1	0.999999	MAID+DIV	275
$(8, 5)$	32	8.1	0.999999	PDC+DIV	228
$(3, 1)$	60	66.0	0.999999	EO	612
$(3, 1)$	60	66.0	0.999999	FT	612
$(3, 1)$	60	72.6	0.999999	MAID	674
$(3, 1)$	60	66.0	0.999999	PDC	473
$(3, 1)$	60	66.0	0.999999	DIV	326
$(3, 1)$	60	72.6	0.999999	MAID+DIV	365
$(3, 1)$	60	66.0	0.999999	PDC+DIV	280
$(8, 1)$	160	160.4	1.000000	EO	1632
$(8, 1)$	160	160.4	1.000000	FT	1633
$(8, 1)$	160	176.5	1.000000	MAID	1774
$(8, 1)$	160	160.4	1.000000	PDC	1262
$(8, 1)$	160	160.4	1.000000	DIV	631
$(8, 1)$	160	176.5	1.000000	MAID+DIV	717
$(8, 1)$	160	160.4	1.000000	PDC+DIV	585

Table 3: Sample candidate solutions for simple example.

6.2 Defining a Redundancy Configuration

We illustrate the use of our models in the design of a redundancy configuration with a simple example. Suppose you need to design a system that requires: at least 20 disks to store all the data, at least 5 MB/s of throughput, at least 6 nines of reliability, and at least 5 nines of availability.

We evaluated all combinations of $n, m \in [1..16]$ for this example and our default model parameters. Table 3 summarizes some of the candidate combinations. From left to right, the table lists the redundancy configuration, the number of disks, its throughput, its availability, the energy conservation technique, and the average power consumption. We do not list reliabilities, as all configurations that meet the other requirements easily meet the reliability requirement.

In the table, the first group of rows shows the optimal configuration, $(8, 5)$ with PDC+DIV for energy conservation. In this configuration, the two best techniques, DIV and PDC+DIV, consume 18% and 30% less energy than EO, respectively. It is interesting to note that the optimal configuration is somewhat unintuitive; the intuitive ones are either invalid or sub-optimal. For example, the simplest redundant configuration ($(2, 1)$, not shown) uses 40 disks, delivers enough throughput, but provides insufficient availability. The second group of rows shows results under another simple and intuitive mirrored configuration, $(3, 1)$. For this configuration, DIV and PDC+DIV conserve 47% and 54% of the energy consumed by EO, respectively. The last group shows results for yet another intuitive configuration, $(8, 1)$. In this scenario, DIV and PDC+DIV consume 61% and 64% less energy than EO, respectively. Note that the maximum throughput of $(8, 1)$ is substantially higher than that of $(8, 5)$, because the former configuration uses more disks and larger fragments than the latter.

6.3 Summary

From these results, it is clear that DIV (independently or in combination with other techniques) is an effective technique. In most of the parameter space, the DIV energy savings are large and consistent. DIV is particularly effective for high n, low m, and read-mostly workloads. Wide-area storage utility, digital library, and file sharing systems, for example, exhibit these ideal properties.

DIV is the very reason why MAID+DIV and PDC+DIV behave well; MAID and PDC independently are neither robust nor energy-efficient in most cases. MAID+DIV behaves better than DIV in part of the space, mostly due to our highly favorable modeling of write accesses in MAID+DIV (and MAID). However, in other parts, the cache disks contribute little besides energy overhead; in those scenarios, MAID+DIV consumes more energy than EO.

PDC+DIV conserves more energy than DIV when redundancy is limited. However, we also modeled PDC+DIV (and PDC) under favorable assumptions: perfect popularity categorization and no migration costs. Furthermore, PDC+DIV has one major drawback: it is very complex to implement in practice, especially in the context of a distributed storage system. In fact, determining the best data layout for energy and bandwidth is clearly NP-hard.

Based on these observations, we argue that DIV is the only effective, robust, and practical redundancy-aware energy conservation technique. As we mentioned before, the other redundancy-aware techniques, EERAID, eRAID, and RIMAC, provide more limited savings than DIV as they only apply to RAID systems.

It is also clear from our results that the task of a storage system designer is not simple. Choosing the right redundancy configuration requires making informed decisions based on all the system requirements. Our simple example showed that non-intuitive redundancy configurations may actually lead to the best results.

7. CASE STUDY: WIDE-AREA STORAGE

We now evaluate DIV in the context of a realistic system under both real and synthetic workloads. In particular, we study a simple wide-area storage system with stable nodes and data replication using simulation. The idea is to mimic a storage system owned and operated by a single world-wide institution, enterprise, or data utility on dedicated machines.

We simulate a storage system comprised by geographically distributed nodes. Each file stored in the system is broken down into 8-KB blocks, each of them replicated at k randomly selected nodes. A block-read request is routed directly to a randomly chosen replica. Block writes are routed to all replicas. To model this simple storage system, we describe it using $n = k, m = 1$, since k copies are always available and only one is required to retrieve the original data. Even though we could consider power-managing entire storage nodes, we continue focusing on disks (one disk per node, for simplicity).

The simulator is trace-driven and selects a random node to receive each client request in the trace. The request is then routed to the destination node and the reply is routed back. The network latency is assumed fixed at 50 ms. (We do not experiment with variable network latency to avoid adding sources of noise to the energy computation, which would make it hard to isolate where benefits come from.) We simulate DIV, FT, and EO. The DIV simulation keeps our technique active at all times (a real system would include a separate mechanism to turn DIV on/off). We simulate the same IBM disks we have been studying.

7.1 Comparing Modeling and Simulation

In this section, we compare the results of our most important energy models (FT and DIV) against those of our simulator using synthetic workloads. Although technically not a "validation" of the models, this comparison is intended to build confidence in our main modeling results, as the simulator eliminates several of the modeling assumptions. Note however that our modeling of MAID, PDC, and their combinations with DIV is an upper bound on their energy conservation potential, so we do not consider them here.

Our synthetic workload generator takes the request rate and percentage of writes as input and produces a trace with 10,000 re-

k	N	ReqRate (reqs/s)	p_w	Buffer (MB)	Savings (%) Sim	Savings (%) Model	Error (%)
3	6	10	0.50	0	0.0	0.0	0.0
3	6	10	0.50	8	45.7	46.1	0.6
3	6	10	0.50	∞	49.3	50.3	2.0
3	15	100	0.75	0	0.0	0.0	0.0
3	15	100	0.75	8	24.5	24.8	-0.3
3	15	100	0.75	∞	49.6	50.1	1.0
5	20	100	0.75	0	0.0	0.0	0.0
5	20	100	0.75	8	21.7	22.2	0.8
5	20	100	0.75	∞	59.8	60.1	0.8
3	3	0.1	0.0	0	13.0	12.8	-0.3
3	15	0.1	0.0	0	57.8	58.1	0.7
3	3	10	0.0	0	0.0	0.0	0.0
3	15	10	0.0	0	0.0	0.0	0.0

Table 4: Sample DIV (top) and FT (bottom) comparison results.

Buffer Size (MB)	Energy (MJ)	Spin Downs	Idle Time (s)	Reqs Delayed (%)	Energy Savings (%)
0	120.3	5068	0.3	0.0	2.5
1	99.0	168540	27.9	0.8	19.7
2	78.1	110760	60.2	0.5	36.7
8	55.9	29120	265.9	0.2	54.7
32	50.3	8000	1088.5	0.1	59.3
∞	48.4	1168	34551.4	0.0	60.8
0	499.7	2134	0.9	0.0	0.0
1	307.6	220746	89.9	0.5	38.4
2	278.2	111846	186.7	0.3	44.3
8	255.8	28254	766.8	0.1	48.8
32	250.1	7194	3070.7	0.0	49.9
∞	248.3	390	80632.6	0.0	50.3

Table 5: DIV results: AT&T (top) and IRCache (bottom) traces.

quest arrivals drawn from a Pareto distribution with infinite variance. Each request is directed to a different disk (in round-robin fashion) and accesses a block of 8 KB. Note that the generation of our synthetic traces differs markedly from our modeling approach. In particular, each request arrival corresponds to a single disk access in our synthetic traces.

We executed a large number of simulations varying six system and workload parameters: $p_w \in [0, 0.25, 0.5, 0.75, 1]$, $k \in [3, 5]$, $req\_rate \in [0.01, 0.1, 10, 100, 1000]$, energy conservation technique $\in [FT, DIV]$, $N \in [3, 6, 9, 10, 15, 20]$, and $wbSize \in [0, 1, 8, \infty]$. Parameter combinations that do not make sense (e.g., $k > N$) or require more bandwidth than that of N disks were discarded. We then compared the energy results of the remaining 795 simulations with the corresponding modeling results.

Table 4 shows a fraction of our validation results. The last column shows the percentage difference between the energy consumption predicted by model and simulator. Our modeling results match the simulation results closely; the average error is 1.3%, the standard deviation is 2.8%, and the maximum error is 18%. If requests are directed to disks randomly (rather than in round-robin fashion), these values become 3.4%, 8.1%, and 28%, respectively.

The simulation and modeling trends match very closely. Again, DIV is most effective for high redundancy, read-mostly workloads, and larger write buffers. Also, FT is again only effective for very low request rates. These results build confidence in our parameter space study (Section 6).

7.2 Real Workload Results

We also wanted to simulate our system for real traces. Unfortunately, the real file-system traces available publicly are not appropriate to evaluate wide-area storage systems such as the one we simulate; these traces are typically for local-area systems, which are amenable to small data and meta-data accesses. To approximate the characteristics of the accesses to wide-area systems, we used two proxy traces from AT&T and the IRCache project. Proxy traces log the file accesses of a large set of clients to a large content base; the same characteristics of our wide-area system.

The AT&T trace was collected between 01/16/99 and 01/22/99, whereas the IRCache trace was collected at three locations from 09/29/04 to 10/05/04. To mimic a system in which files are stored on disk and later retrieved, we pre-processed the traces to transform all file accesses into file read operations. Since the traces do not include information about file creation, we also introduced a write access for each unique file at a random time before the first access to it. After pre-processing, the AT&T trace exhibits 21,150,244 block requests, a 34% write percentage, an average request rate of 35 reqs/s, and a peak rate of 2266 reqs/s. The IRCache trace includes 42,976,431 block requests, a 34% write percentage, an average request rate of 71 reqs/s, and a peak rate of 10635 reqs/s.

Before simulating, we need to define the ideal redundancy configuration for these workloads. First, we set the maximum throughput requirements as their peak request rates. Second, we set the target availability for the system at two different levels: 6 nines (IRCache) and 9 nines (AT&T). Third, we set the target reliabilities two nines higher than the target availabilities.

Assuming these constraints and parameters, our optimization procedure finds $N = 20$ disks and $n = k = 5$ (AT&T) and $N = 81$ disks and $n = k = 3$ (IRCache) as the best configurations. We assess the DIV behavior on these realistic traces by simulating the system with this configuration and different write buffer sizes. We list these results in table 5. From left to right, the table lists the write buffer sizes, the amount of energy consumed during the trace, the number of disk spin downs, the average idle times, the percentage of requests that were delayed (due to contention or disk spin ups), and the DIV energy savings with respect to EO. The table shows that DIV conserves between 20% and 61% of the energy, depending on the size of the write buffers. Small buffers (e.g., 8 MB) can achieve most of the energy savings, but when we cripple DIV by eliminating its write buffers, it conserves little if any energy. Because the periods of high load are short, simulating DIV all the time only leads to serious performance degradation when writes are delayed by disk spin ups. However, as the table also shows, these delays are extremely infrequent. Although we do not include this information in the tables, our DIV model matches the simulations closely; the average errors are 3.4% (AT&T) and 1.8% (IRCache), whereas the maximum errors are 10% (AT&T) and 8% (IRCache).

8. CONCLUSIONS

In this paper, we introduced Diverted Accesses, a novel and effective energy conservation technique designed to leverage the redundancy in storage systems. We also introduced models that predict the disk energy consumption of Diverted Accesses and the previously proposed techniques, as a function of the system's redundancy configuration. Our evaluation coupled a wide parameter space exploration with simulations of a real storage system under two realistic workloads. This study was the first to consider the previous techniques in the presence and as a function of redundancy. Our modeling and simulation results showed that Diverted Accesses is very effective and robust throughout most of the parameter space; other techniques are either not robust or impractical. Furthermore, we found non-intuitive redundancy configurations to be ideal in a simple example, showing that designing a storage system requires quantifying and trading off several metrics;

our energy models are key in this design process. For our realistic system and workloads, and ideal configuration, Diverted Accesses was able to conserve 20-61% of the disk energy consumed by an energy-oblivious system.

We conclude that Diverted Accesses should be extremely useful for large-scale storage systems, such as outsourced storage services or wide-area storage utility, digital library, and file sharing systems. In fact, we believe that our technique would be even more beneficial (in absolute energy consumption terms) if applied to entire nodes rather than just their disks.

Acknowledgements

We would like to thank our shepherd Arif Merchant, as well as Enrique V. Carrera, Uli Kremer, Athanasios Papathanasiou, Anand Sivasubramaniam, and Yuanyuan Zhou for comments that helped us significantly improve the paper. We would also like to thank Prof. Arthur Goldberg from New York University for giving us the AT&T trace.

9. REFERENCES

[1] R. Bhagwan, D. Moore, S. Savage, and G. M. Voelker. Replication Strategies for Highly Available Peer-to-Peer Storage. In *Proceedings of International Workshop on Future Directions in Distributed Computing*, May 2002.

[2] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving Disk Energy in Network Servers. In *Proceedings of the 17th International Conference on Supercomputing*, June 2003.

[3] D. Colarelli and D. Grunwald. Massive Arrays of Idle Disks For Storage Archives. In *Proceedings of the 15th High Performance Networking and Computing Conference*, November 2002.

[4] Data Domain. Data Domain DD400 Enterprise Series. http://www.datadomain.com/, 2005.

[5] E. Anderson et al. Ergastulum: Quickly Finding Near-Optimal Storage System Designs. Technical Report HPL-SSP-2001-05, HP Laboratories SSP, June 2002.

[6] E. Anderson et al. Hippodrome: Running Circles Around Storage Administration. In *Proceedings of the International Conference on File and Storage Technology*, pages 175–188, January 2002.

[7] G. A. Alvarez et al. Minerva: An Automated Resource Provisioning Tool for Large-Scale Storage Systems. *ACM Transactions on Computer Systems*, 19(4):483–518, November 2001.

[8] A. Goldberg and P. N. Yianilos. Towards an Archival Intermemory. In *Proceedings of IEEE Advances in Digital Libraries, ADL 98*, 1998.

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

[10] A. Haeberlen, A. Mislove, and P. Druschel. Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, May 2005.

[11] Y. Hu and Q. Yang. DCD – Disk Caching Disk: A New Approach for Boosting I/O Performance. In *Proceedings of the 23rd International Symposium on Computer Architecture*, June 1995.

[12] J. Kubiatowicz et al. OceanStore: An Architecture for Global-scale Persistent Storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, November 2000.

[13] E. K. Lee and C. A. Thekkath. Petal: Distributed Virtual Disks. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, 1996.

[14] D. Li and J. Wang. EERAID: Energy-Efficient Redundant and Inexpensive Disk Array. In *Proceedings of the 11th ACM SIGOPS European Workshop*, Sept 2004.

[15] D. Li and J. Wang. Conserving Energy in RAID Systems with Conventional Disks. In *Proceedings of the International Workshop on Storage Network Architecture and Parallel I/Os*, Sept 2005.

[16] Maximum Throughput, Inc. Power, Heat, and Sledgehammer, April 2002.

[17] Fred Moore. More Power Needed, November 2002. Energy User News.

[18] A. Papathanasiou and M. Scott. Power-efficient Server-class Performance from Arrays of Laptop Disks. Technical Report 837, Department of Computer Science, University of Rochester, May 2004.

[19] E. Pinheiro and R. Bianchini. Energy Conservation Techniques for Disk Array-Based Servers. In *Proceedings of the 18th International Conference on Supercomputing (ICS'04)*, June 2004.

[20] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting Redundancy to Conserve Energy in Storage Systems. Technical Report DCS-TR-570, Rutgers University, March 2005.

[21] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility. In *Proceedings of the International Symposium on Operating Systems Principles*, 2001.

[22] S. Gurumurthi et al. Interplay of Energy and Performance for Disk Arrays Running Transaction Processing Workloads. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software*, March 2003.

[23] Y. Saito, C. Karamonolis, M. Karlsson, and M. Mahalingam. Taming Aggressive Replication in the Pangaea Wide-Area File System. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, Dec 2002.

[24] D. Siewiorek and R. Swarz. *Reliable Computer Systems Design and Evaluation*. A K Peters, third edition, 1998.

[25] Sun Microsystems. Sun StorEdge 3320. http://www.sun.com/storage/, 2005.

[26] H. Weatherspoon and J. Kubiatowicz. Erasure Coding vs. Replication: A Quantitative Comparison. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, March 2002.

[27] X. Yao and J. Wang. RIMAC: A Redundancy-based, Hierarchical I/O Architecture for Energy-Efficient Storage Systems. In *Proceedings of the 1st ACM EuroSys Conference*, Apr 2006.

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

[29] Q. Zhu and Y. Zhou. Power-Aware Storage Cache Management. *IEEE Transactions on Computers*, 54(5), 2005.