Energy Conservation Techniques for Disk Array-Based Servers *

Eduardo Pinheiro and Ricardo Bianchini

Department of Computer Science
Rutgers University
Piscataway, NJ 08554

{edpin,ricardob}@cs.rutgers.edu

ABSTRACT

In this paper, we study energy conservation techniques for disk array-based network servers. First, we introduce a new conservation technique, called Popular Data Concentration (PDC), that migrates frequently accessed data to a subset of the disks. The goal is to skew the load towards a few of the disks, so that others can be transitioned to low-power modes. Next, we introduce a user-level file server that takes advantage of PDC. In the context of this server, we compare PDC to the Massive Array of Idle Disks (MAID). Using a validated simulator, we evaluate these techniques for conventional and two-speed disks and a wide range of parameters. Our results for conventional disks show that PDC and MAID can only conserve energy when the load on the server is extremely low. When two-speed disks are used, both PDC and MAID can conserve significant energy with only a small fraction of delayed requests. Overall, we find that PDC achieves more consistent and robust energy savings than MAID.

Categories and Subject Descriptors

D.4 [**Operating systems**]: Storage management; B.4 [**Input/output and data communications**]: Input/output devices

General Terms

Experimentation, measurement

Keywords

Energy conservation, network servers, disk power

1. INTRODUCTION

Energy conservation has been extensively studied in the context of mobile devices. Recently, researchers have realized that energy conservation is also important for servers and server clusters. For these systems, energy conservation is obviously not intended to extend battery life, but nonetheless has serious economical and environmental implications. Economical because the energy consumption of these servers and their cooling systems is reflected in their electricity bills. Environmental because most power-generation technologies are harmful to the environment.

A few efforts [6, 12, 18] have been made to conserve energy in server clusters by tackling the high base power of traditional servers, i.e. the power consumption when the server is on but idle. Other efforts [2, 8, 9] have tackled the energy consumed by the servers' microprocessors. Finally, the energy consumption of disk array-based servers has received some attention as well [5, 7, 10, 11, 20, 21]. The energy consumed by the array can easily surpass that of the rest of the system, depending on the array size [5, 10].

In this paper, we propose a new energy conservation technique for disk array-based network servers, called Popular Data Concentration (PDC). The idea behind PDC is to dynamically migrate the popular disk data (i.e., the most frequently accessed data on disk) to a subset of the disks in the array, so that the load becomes skewed towards a few of the disks and others can be sent to low-power modes. PDC is based on the observation that network server workloads often exhibit files with widely different popularities. For instance, Web server workloads are known to exhibit highly skewed popularity towards a small set of files.

To demonstrate the benefits of PDC, we also propose a user-level file server, called Nomad FS, that takes advantage of our technique. We envision coupling Nomad FS with one or more front-end network servers. Nomad FS uses the MQ algorithm [19] for second-level cache replacement to guide the placement of files in the disk array dynamically. The current prototype of Nomad FS conserves energy by spinning disks down after a period of idleness. Unfortunately, spinning a disk back up to service a request increases the response time for the request significantly.

We also study Nomad FS in the context of disk arrays composed of two-speed disks [5]. These disks conserve energy by automatically reducing their rotational speeds under light load. Thus, Nomad FS does not have to explicitly effect power mode transitions when two-speed disks are used. Furthermore, requests are only delayed during rotational speed transitions, which means that only a small percentage of requests is delayed for stable workloads.

Because our work targets network servers, degrading the response time of a small fraction of requests is acceptable for Nomad FS. In fact, the overhead of multiple message exchanges per request in network servers means that higher server response times can be accommodated. Moreover, network servers typically interact with

*This research was supported in part by NSF grants CCR-0238182 (CAREER award) and EIA-0224428.

ICS'04, June 26–July 1, 2004, Malo, France.

people (or background processes), so even substantial degradations may not be noticeable. For example, a degradation from 10 to 100 milliseconds is usually acceptable for network servers.

To put PDC in context, we compare Nomad FS against a very similar version of it that is based on the Massive Array of Idle Disks (MAID) [7]. Instead of relying on file popularity and migration, MAID relies on temporal locality to place copies of files on a subset of the disks. Each disk in this subset acts as a cache of the files and uses the LRU replacement policy. Non-cache disks then become more lightly loaded, allowing for energy conservation.

The behavior of the systems we consider is affected by several important parameters, such as the type of disks, the disk request rate, the size of the main memory cache, and the percentage of the stored files that is actually accessed. To understand the effect of these parameters and determine the "sweet spot" for each technique, we assess the parameter space extensively using a simulator and synthetic traces. We also simulate two real traces to confirm the observations from the parameter space exploration. To guarantee the accuracy of our results, we validate the simulator against real executions of our prototype implementation of Nomad FS.

Our results for arrays of conventional disks show that PDC and MAID can only conserve energy when the load on the server is extremely low. When arrays of two-speed disks are used, both PDC and MAID can conserve up to 30-40% of the disk energy with only a small fraction of delayed requests. Overall, PDC is more consistent and robust than MAID; the behavior of MAID is highly dependent on the number of cache disks. Furthermore, PDC achieves these properties without the overhead of extra disks. However, the PDC energy savings degrade substantially for long migration intervals. Based on these results, we conclude that the techniques we study will be very useful for disk array-based servers when multi-speed disks become available.

In summary, the contributions of this paper are:

- We propose a file popularity-based energy conservation technique for disk array-based network servers;
- We design and implement an energy-aware file server that takes advantage of the technique; and
- We assess the behavior of two energy conservation techniques for conventional and two-speed disk arrays and a wide range of parameters.

The remainder of this paper is organized as follows. The next section describes PDC, Nomad FS, and MAID. Section 3 describes our evaluation methodology, including our simulator and the parameter space we study. Sections 4 and 5 present our results for synthetic and real traces, respectively. Section 6 discusses the related work. Finally, section 7 concludes the paper.

2. ENERGY CONSERVATION IN ARRAYS

In this section we first describe two of our main contributions: the PDC energy conservation technique and Nomad FS. Next, we move on to describing MAID and the traditional fixed-threshold technique, and qualitatively comparing the approaches.

2.1 Popular Data Concentration

Several types of network servers exhibit workloads with highly skewed file access frequencies. For example, it has been shown that the frequency of file access by a Web server conforms to a Zipf distribution [3] with high coefficient. The same is true of other servers as well, e.g. [13]. More formally, Zipf's law predicts that the frequency of access or popularity τ of a file is proportional to the inverse of its rank r raised to some coefficient α, i.e. $\tau = 1/r^{\alpha}$. When α is high (close to 1), a relatively small number of files is accessed frequently, whereas a large number of files is accessed rarely. Workloads with high α often exhibit skewed popularity in terms of disk accesses as well (albeit with smaller α), even in the presence of large main memory caches [4].

PDC is inspired by such heavily clustered popularities. The idea behind PDC is to concentrate the most popular disk data (i.e., those that most frequently miss in the main memory cache) by migrating it to a subset of the disks. This concentration should skew the disk load towards this subset, while other disks become less loaded. These other disks can then be sent to low-power modes to conserve energy. More specifically, the goal of PDC is to lay data out across the disk array so that the first disk stores the most popular disk data, the second disk stores the next set of most popular disk data, and so on. The least popular disk data and the data that are never accessed will then be stored on the last few disks. In fact, the last few disks will also include the data that most frequently hit in the main memory cache.

To avoid performance degradation, it is important not to overload the disks that store popular data. Thus, the expected access rate for each disk needs to be considered explicitly. This is done by estimating the future load (in MBytes/second) on each disk to be the sum of the recent load directed to the data to be stored on it. PDC should then only migrate data onto a disk until the expected load on the disk is close to its maximum bandwidth for the workload.

Because data popularity can change over time, PDC may have to be applied periodically. In such cases, it might be necessary to free up space on a disk by migrating data out of it, before more popular data can be migrated in.

2.2 Nomad FS

PDC has been implemented in Nomad FS, a prototype energy-aware file server consisting of approximately 13k lines of C++ code. Nomad FS is a user-level, event-driven server that works on top of the local file system. The server associates a helper thread with each disk. This thread is the only part of the server that directly touches the disk, either for read/write operations or migrations. Although the server receives requests for 8-KByte file blocks, entire files are migrated according to PDC. This approach works fine for the workloads we are interested in, such as Web, proxy, ftp, and email server workloads, which access entire files at a time.

To conserve energy in disk arrays composed of conventional disks, PDC is used to idle disks, which are then spun down by the server after a fixed period of idleness (the idleness threshold). The server determines that it is time to spin down a disk by keeping the last access time per disk and periodically testing whether any disk has been idle for longer than the idleness threshold. A spun down disk is reactivated on the next access to it. When we consider disk arrays composed of two-speed disks, Nomad FS does not explicitly effect power mode transitions. Furthermore, Nomad FS does not migrate files out of disks that are already running in low speed to reduce the migration overhead.

The file server exports one single view of the file system, although multiple disks drives can be used. For simplicity of our prototype, each file is permanently stored on one disk only, i.e. no striping or mirroring is used at this time. The user has no control over where (on the disk array) files are stored, since that information can change dynamically. New files are created on any of the disks with enough free space. Metadata information is kept in a well-known location on the disk that stores the most popular files. The metadata for each file contains the file name (at most 128 bytes), the size of the file, the disk on which the file resides, the times of creation and last access, and bookkeeping information (pointers to next and previous files on the same disk, etc). Overall, each metadata entry consumes 168 bytes. The metadata for all files is kept in a large file that is memory mapped to the virtual address space of the server. Only when a file is actually opened is its metadata accessed and physical memory allocated for it.

Nomad FS caches data blocks in memory at the user level. Because the cache is not a first-level cache (it sits behind the clients' and/or network servers' caches), the standard LRU replacement policy would not work well for this cache. Instead, Nomad FS uses the Multi Queue (MQ) algorithm [19] to manage its block cache. MQ has been shown superior to other algorithms, including sophisticated variations of LRU.

The MQ cache works as follows. There are multiple LRU queues numbered Q_0, Q_1, ..., Q_{m-1}; in our prototype $m = 12$, but the system behavior is typically similar for other large numbers of queues. Blocks stay in the LRU queues for a given *lifetime*. This lifetime is defined dynamically by the MQ algorithm to be the maximum temporal distance between two accesses to the same file or the number of cache blocks, whichever is larger. If a block has not been referenced within its lifetime, it is demoted from Q_i to Q_{i-1} or evicted from the cache if it is in Q_0. Each queue also has a maximum access count. If a block in queue Q_i is accessed more than 2^i times, this block is then promoted to Q_{i+1} until it is accessed more than 2^{i+1} times or its lifetime expires. Within a given queue, blocks are ranked by recency of access, according to LRU.

During operation of the server, information about the files referenced in the past are summarized on a list in main memory. The summary for each file stores the file descriptor, the number of disk accesses for the file, a pointer to its metadata, and some other bookkeeping data (such as list pointers), totaling 60 bytes. The amount of memory dedicated to the list of summary information should be large enough to hold at least the number of files in the working set of a given workload.

The ranking of file popularity is done incrementally every time a file access requires a disk access. We do this by reusing the same MQ cache algorithm described above, but instead of using it on cache blocks, we use it on the summary information. More specifically, we keep the summary information on an MQ cache-like list. Every disk access causes the file's summary data to move on the list according to the MQ policy. More disk accesses cause the summary entry to move closer to the head of the list. After a while, popular files (in terms of disk accesses) will be close to the head of the list, while unpopular files will be towards the tail.

Periodically, files are migrated to disks based on this ranked list of disk accesses. The algorithm traverses the queues from Q_{m-1} to Q_0. Initially, files in queue Q_{m-1} are migrated to the first disk (numbered 0) until it is full or the expected load on the disk approaches its maximum bandwidth for the workload. The expected load associated with a file is estimated by dividing its size by its average inter-access time. The next set of files is migrated to the second disk (according to the same conditions) and so on. If the server needs to migrate a file F to disk drive i, there is no space available on i, and a file L on i is less popular than F, then file L is migrated to disk z. Disk z is selected as follows. First, the server tries the disks holding more popular files, starting from disk 0 and checking all disks until disk $i - 1$. If not enough space and bandwidth can be found there, the server checks the disks holding less popular data, starting from disk $i + 1$ and checking all higher numbered disks. This algorithm should always find space for the file, as long as the overall disk space is not excessively tight for the size of the file system. We have not developed an algorithm to deal with this last scenario, because it has not been a problem for us.

2.3 Comparison Against Other Approaches

MAID. The Massive Array of Idle Disks (MAID) [7] has been proposed as a replacement for old tape backup archives with hundreds or thousands of tapes. Because only a small part of the archive would be active at a time, the idea is to copy the required data to a set of "cache disks" and put all the other disks in low-power mode. Accesses to the archive may then find the data on the cache disk(s). Cache disk replacements are implemented using an LRU policy. Replaced data can simply be discarded if it is clean. If it is dirty, it has to be written back to the corresponding non-cache disk.

This same idea can be applied to file servers as well. In fact, we developed a version of Nomad FS that uses MAID to conserve energy. We used as much code from Nomad FS as possible for our MAID-based file server, so that the two servers can be directly compared. The main memory cache management, for instance, is exactly the same for both implementations. On an access to a cached file block, the access is performed on the corresponding cache disk. If the block accessed is not cached, the entire file is copied by the server from its location on one of the non-cache disks to one of the cache disks. If a replacement is required on a cache disk, an entire file that is large enough is replaced according to LRU.

The designers of MAID observed that cache disks can become overloaded [7]. To counter this problem, we optimize MAID by avoiding copying files to the cache disks when the recent load on these disks is approaching their maximum bandwidth. We find that the absence of this optimization always leads to a large percentage of delayed requests, as the cache disks become a bottleneck.

To conserve energy in configurations with conventional disks, the MAID version of our server explicitly spins disks down after a period of idleness. When considering two-speed disks, the server does not control power modes explicitly and does not copy data out of disks that are already at low speed.

In section 4, we quantitatively compare the MAID version of our file server against the PDC version. Here we focus on a qualitative comparison. PDC and MAID have the same objective, namely to increase idle times by moving data around the disk array and spinning disks down. As a result, both techniques sacrifice the access time of certain files in favor of energy conservation. However, instead of relying on file popularity and migration like PDC, MAID relies on temporal locality and copying to conserve energy. This has a few potential disadvantages in terms of energy savings: (1) the cache disk(s) represent energy overhead, since they are additional to the set of disks that are required to store the actual data; (2) the cache disk(s) might not have enough space to store the entire (dynamically changing) working set for a given workload, causing constant accesses to the non-cache disks; and (3) files are randomly spread on the non-cache disks, which may significantly reduce the opportunity for energy conservation if requests miss in the cache disk(s) frequently. Nevertheless, MAID does have a few potential advantages: (1) metadata management is significantly simpler, since it does not need to store and operate on file access information about all files, only those that are on cache disks; and (2) it adapts faster to changes in workload behavior, because file copying (and possibly replacement) take place on a per-file access basis.

FT. When considering disk arrays composed of conventional disks, we also study a fixed-threshold (FT) technique. FT is simple: a disk is spun down after it has been idle for a fixed period. The idleness threshold is the amount of time for which the energy consumed in idle state is the same as that of powering the disk down and later powering the disk back up. The rationale for this definition is similar to the competitive argument about renting or buying skis [15]. A spun down disk is reactivated on the next access to it.

FT does not involve any data movement or re-organization and, thus, is not very effective at conserving energy in busy servers. In contrast, it is very useful when combined with PDC or MAID, as described above. We use an FT-based version of our file server as a basis for comparison in section 4. Again, we re-used most of the Nomad FS code to implement our FT-based file server.

3. METHODOLOGY

Our main goals in this paper are to determine the conditions under which PDC is effective at conserving energy and how it compares to MAID and FT for two disk array types and a wide range of parameters. We explore this parameter space using the simulator that we describe in subsection 3.1.

The reason why we chose simulation rather than real experimentation is two-fold: (1) two-speed disks are not yet available on the market (although Sony already has a disk drive that allows for static speed setting [16]); and (2) using real experiments to perform such a comprehensive parameter space exploration would have taken extremely long. For example, each of our simulations involves 19 million requests at 750 requests/second. A real experiment with these parameters takes 7 hours, whereas a simulation takes less than 40 minutes. Note that accelerating a real execution and then scaling results back would not have worked, since it is not possible to accelerate real disk spin up/spin down operations, file copying, or file migration.

Nevertheless, we strongly believe in realistic simulations, so we carefully validated the simulator against executions of our prototype file server implementations on our own server hardware. We discuss the validation of the simulator in subsection 3.2.

Another advantage of using simulation is that we can easily vary parameters, which is a key component of this paper. In subsection 3.3 we discuss the set of parameters that we have found most relevant in evaluating and comparing the different file servers.

We drive the simulator with both synthetic and real traces. Subsection 3.4 describes our generator of synthetic traces, whereas subsection 3.5 describes our real traces.

3.1 Simulation

We have developed an execution-driven simulator of our prototype file servers. The simulator mimics the implementation of our servers as closely as possible. In fact, several parts of the real servers were reused in the simulator to avoid deviations from the real implementations. The MQ replacement algorithm implementation, for example, was copied straight out of the server code.

At the file system level, files are assumed to be laid out in consecutive blocks of the same disk, whereas i-nodes are assumed to be cached in memory. At a lower level, the simulator models an array of conventional Cheetah disks or an array of two-speed disks. The main characteristics of the conventional disk are shown in table 1. The table also lists the idleness threshold for this disk, which is the sum of the energies to spin the disk up and down divided by the idle power.

The main characteristics of the two-speed disks are shown in table 2. The high speed values are those of the Cheetah disk. These values were scaled down to produce the low speed values. In particular, the power and energy values were scaled down in quadratic fashion [10], assuming that $power = c \times speed^2 + 1.86$ where c is the constant $(5.26 - 1.86)/10K^2$. The disk controller slows the disk down when the offered load becomes lower than 80% of the disk throughput at low speed. Conversely, the controller transitions the disk to high speed when this same threshold is exceeded. The controller can trigger these transitions by observing the actual utilization of the disk. We assume that no accesses can proceed when a disk is transitioning speeds. The speed transition costs are set to half of the corresponding costs for the Cheetah disk. For example, going from low to high speed costs half of the energy and time to spin up the Cheetah disk. These transition cost settings are admittedly somewhat arbitrary. However, note that the transition costs make little difference in our simulations, since the frequency of speed transitions is low for the parameters we consider. This same effect has been observed in previous work [5].

Description	Value
Disk model	Seagate Cheetah ST39205LC
Standard interface	SCSI
Storage capacity	9.17 GBytes
Number of platters	1
Rotational speed	10000 rpm
Avg. seek time	5.4 msecs
Avg. rotation time	3 msecs
Transfer rate	31 MBytes/sec
Idle power	5.26 Watts
Down power	1.86 Watts
Active energy (8-KB read)	61 mJoules
Spin up energy	65.91 Joules
Spin down energy	28.25 Joules
Spin up time	6.12 secs
Spin down time	11.24 secs
Idleness threshold	17.9 secs

Table 1: Main characteristics and measured power, energy, and time statistics of our SCSI disk.

Description	Value
Storage capacity	9.17 GBytes
Avg. seek time	5.4 msecs
High to low speed transition energy	14.13 Joules
High to low speed transition time	5.62 secs
Low to high speed transition energy	32.96 Joules
Low to high speed transition time	3.06 secs
High rotational speed	10000 rpm
Avg. rotation time at high speed	3 msecs
Transfer rate at high speed	31 MBytes/sec
Idle power at high speed	5.26 Watts
Active energy at high speed (8-KB read)	61 mJoules
Low rotational speed	3000 rpm
Avg. rotation time at low speed	10 msecs
Transfer rate at low speed	9.3 MBytes/sec
Idle power at low speed	2.17 Watts
Active energy at low speed (8-KB read)	43 mJoules

Table 2: Main characteristics and simulated power, energy, and time for our two-speed disk. The high speed values are those of our Cheetah disk (table 1). These values were scaled down to produce the low speed values. The speed transition costs are set to half of the corresponding costs in the Cheetah disk.

Finally, the PDC and MAID-based servers involve four major sources of overhead that do not exist in the FT version of our server: migrations, copies, LRU processing, MQ processing. The CPU energy consumed by these operations has not been considered so far. To take this consumption into account, we determined these overheads by running three microbenchmarks. One of them isolates the CPU energy involved in each file block request to be about 37 μJ for both LRU and MQ processing. The other two microbenchmarks isolate the CPU energy consumed by each migration and copy operation, as a function of file size. We have approximated these costs to 1.5 μJ per byte moved. The simulator charges each request, migration, and copy with these CPU energy costs.

Description	Total energy consumed			Files moved		MBytes moved		Spin downs/ups		Delayed requests	
	Sim	Real	Error	Sim	Real	Sim	Real	Sim	Real	Sim	Real
PDC	152023	172843	12.0%	9715	10943	227	283	336	326	1.3%	1.2%
MAID	141349	156335	9.6%	8738	8716	204	204	300	304	1.1%	1.0%
FT	190374	200479	5.0%	n/a	n/a	n/a	n/a	10	13	0.2%	0.2%
EO	190346	200833	5.2%	n/a	n/a	n/a	n/a	n/a	n/a	0.0%	0.0%

Table 4: Summary of simulator validation. Energy values are in Joules.

Description	Value
Request rate	4 file requests/sec
Trace length	9033 secs (2.5 hours)
Number of files	43690 *
File size	20 KBytes *
Total file system size	853 MBytes *
Main memory cache size	64 MBytes *
Number of disks	4
Disk capacity	342 MBytes *
Idleness threshold	17.9 secs
Migration period	every 1.25 hours (PDC only)

Table 3: Values used in the simulator validation: workload characteristics (top), server settings and disk characteristics (bottom). Values marked with "*" are used for fast validation.

The simulator faithfully models the disk powers, energies, and times discussed above to determine disk energy consumption and response times. The CPU energy costs associated with PDC and MAID are added to their disk energy consumption.

3.2 Simulator Validation

It is important to build confidence in our simulator by validating it against real experiments. Unfortunately, we can only validate the simulations that assume conventional disks.

Our server hardware consists of an ASUS A7A133 motherboard with an Intel Pentium-4 1.9-GHz processor, 512 MBytes of memory, 4 Seagate Cheetah 9.1-GByte Ultra SCSI disks, one Adaptec 29160 Ultra 160 SCSI controller, and one Gigabit Ethernet network adaptor. The energy consumed by the disks is monitored by a TDS 3014 oscilloscope. The oscilloscope is connected to a remote computer that collects power consumption data at 1.25 Gsamples/second. These data are averaged every 10 milliseconds. A 1.2-GHz PC connected to the server via a Gigabit Ethernet connection generates load for the server.

We used a synthetic workload to validate the simulator against our prototype file servers running on this server hardware. Our workload generator (explained in detail below) keeps selecting a file to access randomly out of a set of files for a user-specified amount of time. All the blocks of each chosen file are requested in sequence. The timing of the file requests follows an exponential distribution. The main characteristics of the synthetic workload and the server settings are summarized in table 3. Note that some of the workload characteristics and server settings (those marked with "*") are not realistic; they are meant exclusively for fast validation.

We collected results for our file servers, as well as an energy-oblivious server (EO) under which disks are never powered down. The MAID version used one of the 4 disks as a cache disk, so it did not incur any energy overhead due to the cache disk. Table 4 compares the servers in terms of disk energy consumption, files and bytes moved between disks, number of disk spin downs/ups, and the percentage of delayed requests. We consider a file request to be delayed if it takes longer than 200 milliseconds.

The results show that the total energy consumed by each server matches closely (within 5 to 12%) that of the simulated system. The results also match the other metrics closely, especially the number of spin downs/ups, which is a key metric for energy conservation. These results give us confidence that our simulator can be used to illustrate the important trends across the parameter space.

3.3 Parameter Space

Three categories of parameters directly affect the energy conservation techniques and file servers we study: workload characteristics, server characteristics, and disk drive characteristics. Next, we discuss each of these categories in turn.

Workload characteristics. These are characteristics that are inherent to the workload imposed on the file server. Among the large set of parameters that describe a workload, we have identified five key characteristics: the request rate, the file popularity, the coverage of the file system, the percentage of writes, and the temporal correlation of accesses to files.

The *request rate* has to do with the rate at which file requests arrive at the server. We assume that inter-arrival times are exponentially distributed; the request rate is the average of the distribution.

File *popularity* corresponds to the frequency with which each file is accessed. We represent popularity by a value of the α coefficient in Zipf's power law. This coefficient determines how skewed the distribution of accesses to files is. A smaller α (close to 0) means that requests are more evenly distributed among files, whereas a larger α (close to 1) means that requests are more skewed towards a few files.

The file system *coverage* has to do with the percentage of the entire file repository that is actually accessed by the workload. A coverage of 100% means that all files in the file system are accessed at least once.

The *percentage of writes* corresponds to the frequency of file write operations in the workload.

Temporal locality describes how far apart in time consecutive accesses to the same files are. In particular, workloads with high temporal locality exhibit consecutive accesses to the same file that are "close" in time, whereas workloads with low temporal locality exhibit consecutive accesses to the same file that are "far apart" in time. Temporal locality has a direct relationship with popularity, i.e., popular files have a short inter-arrival time due to their popularity. However, temporal locality can also be an effect of *temporal correlation*, as discussed in [14]. Files with high temporal correlation are those for which accesses are close together in time regardless of the absolute popularity of the file. In this study, we are interested in the effect of both aspects of temporal locality on energy conservation.

File server characteristics. The configuration of the file server can impact the energy savings significantly. Important configuration parameters are the main memory cache size, the cache replacement policy, and the number of disks used. We consider the *cache size* and the *number of disks* in this study. We keep the cache replacement policy fixed on the MQ algorithm, which has been shown to be the best policy for second-level caches such as the ones used in our file servers [19].

Disk drive characteristics. Several characteristics of physical disk drives affect energy consumption directly, especially the power consumption in each mode and the spin up/down overheads. Rather than focusing on a detailed exploration of this space however, we concentrate on evaluating the systems we study for disk arrays composed of either *conventional* or *two-speed disks*.

3.4 Synthetic Trace Generator

To understand the impact of the parameters we just discussed, we built a workload generator that can produce request traces with user-specified request rate, popularity, coverage, temporal correlation, number of requests, file system size, and average file size. Currently the generator assumes that all files are of the same size for simplicity.

To generate a request trace, the generator must first compute the total number of files in the file system, by dividing the file system size by the file size. With the total number of files, the generator can determine the number of files that must be requested, according to the desired coverage. Each of the files to be requested then receives a probability weight, based on Zipf's formula $1/i^{\alpha}$, where i is the rank of the file from 1 to the number of files to be requested. Based on these probabilities, the generator randomly selects the actual file requests that will make up the trace. All blocks of a selected file are requested in sequence. Each new request is assigned a timestamp that is equal to the time of the last access plus an exponentially distributed random increment based on the request rate.

The generator continues to select files based on their weights until the number of requests reaches the desired number. If the desired coverage is still missing N files when the trace is missing N requests, a request for each of these N files is appended to the trace sequentially. However, to avoid having a long sequence of previously unreferenced files in the end of the trace, we randomize the trace once all the files have been selected. We have verified that this procedure produces the desired coverage without changing the desired popularity (α) by inspecting several generated workloads.

However, this generation procedure does not consider temporal correlation explicitly. As described, the temporal locality of the workload is the one inherent to the popularity of files. The workloads we consider have this form of temporal locality, except when we explicitly address the effect of temporal correlation. To generate a set of workloads with varying degrees of temporal correlation, the generator does the following. The probability weights described by the Zipf are split into a fixed number of groups with the same number of files per group. Instead of generating requests that span the whole range of files in the coverage, the generator selects files from the first group first, for as many requests as the first group of files should receive. After generating all the requests for the first group, the generator moves to the second group and so on. Grouping files in this way forces files to be selected more closely together in time. For a number of groups larger than 1, the result is a workload with higher temporal correlation. For large numbers of groups, even unpopular files will have much higher temporal correlation. The resulting workload might be unrealistic for more than 1 group, but it does exhibit a controlled temporal correlation behavior, which suffices for our purpose of observing trends.

3.5 Real Traces

We also simulate two real (proxy server) traces, which we call Hummingbird (HUM) and Point of Presence (POP). HUM was collected at AT&T from 01/16/99 to 01/31/99. The total footprint of HUM consists of 76.9 GBytes with an average requested file size of 294 KBytes. POP was collected between 10/15/01 and 10/26/01 at the Federal University of Minas Gerais, Brazil. POP was filtered to eliminate the proxy misses, leading to a total footprint of 39.4 GBytes and an average requested file size of 17 KBytes.

Description	Default Value
Request rate *	750 reqs/s (36 MBytes/sec)
File popularity *	0.85
Coverage *	40%
Percentage of writes *	0%
Temporal locality *	popularity-induced only
Trace length	19,000,000 requests
Main memory cache size *	1 GByte
Number of disks *	8 (PDC), 9 (MAID-1+), 10 (MAID-2+)
File size	48 KBytes
Total file system size	126 GBytes
Migration period *	every 1/2 hour (PDC only)
Cache disks	1 (MAID-1+), 2 (MAID-2+)
Disk characteristics	same as in tables 1 and 2

Table 5: Default parameters used in synthetic trace simulations: workload characteristics (top), server settings (middle), disk characteristics (bottom). We vary the parameters marked with "*".

4. PARAMETER SPACE RESULTS

In this section we present the results of our parameter space exploration. We break the evaluation into two parts: arrays of conventional disks and arrays of two-speed disks. In each part, we start by analyzing workload characteristics and later move on to file server characteristics. To study the effect of each parameter, we assess the energy savings of the PDC, MAID, and FT-based versions of our file server (in comparison to the energy-oblivious version, EO), as a function of different settings for the parameter. To isolate the effect of a single parameter at a time, all other parameters are fixed at their default values. Throughout this section, we refer to the name of each technique and the server that exploits it interchangeably.

The default values for our simulation parameters are listed in table 5. Note that MAID-1+ is the MAID version of our file server that uses one extra disk as cache; MAID-2+ assumes two cache disks. The "+" refers to the optimization we made to MAID to avoid overloading the cache disk(s). The values for our simulated hardware are based on our own real server hardware, except for the main memory cache size. The values for the workload characteristics were chosen because they are "reasonable" values that represent areas of the parameter space where PDC and MAID perform well. In particular, the default request rate was set to half of the peak load we would expect for our real server. Common provisioning practice suggests that a server should be provisioned such that its peak load reaches 80% of its maximum throughput; the 20% extra capacity can handle any unexpected increases in load. We followed these suggestions and determined the maximum throughput of our server, 125 MBytes/sec, when all file requests can be served from the main memory cache. Given that the load peaks and valleys of servers deployed in the field can vary by factors ranging from 3 to 10 [6], our default request rate corresponds to a period of light load, exactly when energy savings are possible. Previous works have made similar assumptions, e.g., [5]. Nevertheless, we do vary the parameters marked with "*" in the table in the next two subsections.

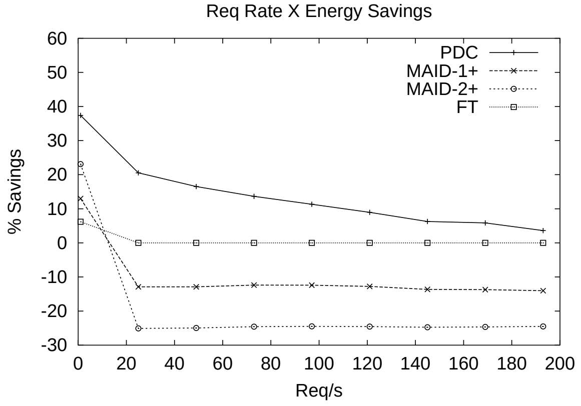

Figure 1: Energy savings per file request rate (conventional disks).

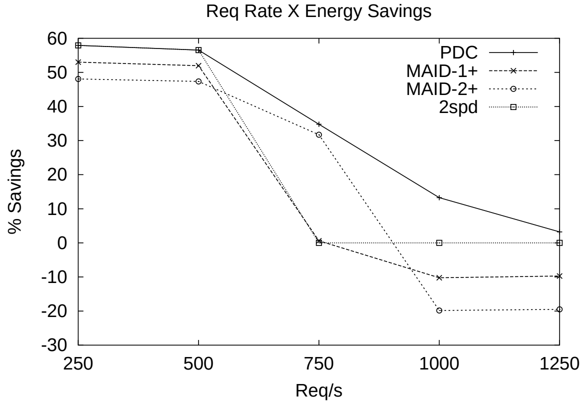

Figure 2: Energy savings per file request rate (two-speed disks).

4.1 Arrays of Conventional Disks

Figure 1 depicts the energy savings that can be accrued by the four versions of our file server, as different average file request rates are imposed on the servers. The energy savings are computed in comparison to EO.

As expected, low request rates show better energy savings. The maximum savings are substantial, reaching almost 40% in the case of PDC. However, note that non-trivial energy savings can only be achieved for very low request rates, less than 100 requests/second. These request rates are less than 10% of our default request rate. A file request rate of 25 request/second, for instance, is equivalent to a throughput of only 1.2 MBytes/second. Even worse, when request rates are very low, a significant percentage of requests is delayed by disk spin up operations. Note also that the MAID systems actually produce negative energy gains, i.e. they consume more energy than EO, across most of the space of request rates. The reason is that the cache disks do not reduce the load on the non-cache disks enough for them to be spun down; they are simply energy overhead.

These request rate and response time results clearly show that conventional disks are not useful when the goal is to conserve energy without noticeable performance degradation. When request rates are high, no energy savings can be accrued. When request rates are low, requests are frequently delayed by disk spin ups. These same negative effects are observed across the whole parameter space of workload and file server characteristics, so we do not present further results for arrays of conventional disks.

4.2 Arrays of Two-Speed Disks

4.2.1 Workload Characteristics

Request rate. Figure 2 depicts the energy savings that can be accrued by the four versions of our file server, as a function of the file request rate. The curve labeled “2spd” represents the version that uses two-speed disks, but no data movement. All energy savings are computed with respect to EO running on an array of conventional (Cheetah) disks.

Again, lower request rates show better energy savings. In fact, the maximum savings in comparison to EO are very substantial, reaching almost 60%, the energy differential between low and high speeds, for PDC and 2spd at 250 and 500 requests/second. In this part of the space, the MAID systems produce lower savings due to the cache disks, which are unnecessary. The energy savings decrease quickly as we increase the file request rate. At 750 requests/second, the 2spd and MAID-1+ systems stop producing energy gains, since all disks remain in high speed virtually all the time. At that request rate, PDC and MAID-2+ still produce gains of 30-40%, as data movement has the effect of reducing the load on several disks. At 1000 requests/second, the gains achievable by both MAID systems become negative, whereas PDC still produces gains of more than 10%. For higher request rates, the PDC gains are reduced and approach 3% at 1250 requests/second.

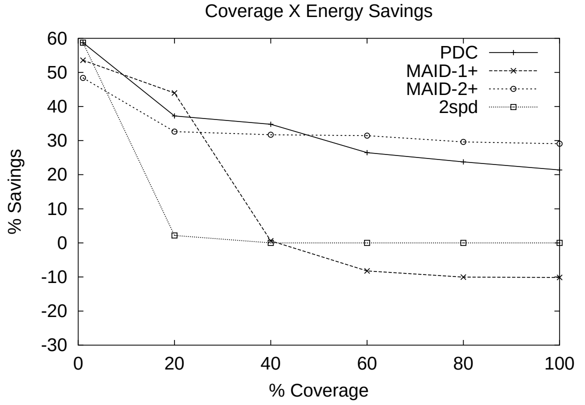

Figure 3: Energy savings per file system coverage.

These results demonstrate that PDC is more consistent and robust than other techniques for the range of request rates that can actually provide energy savings. PDC achieves the highest gains across the range, without ever consuming more energy than EO. The behavior of the MAID systems is heavily influenced by the number of cache disks used. Moreover, the overhead of cache disks is pronounced on both ends of the range. 2spd does not provide gains in most cases.

In terms of response time, we find that all versions of our server exhibit less than 2% delayed requests, regardless of the request rate. In fact, the percentage of delayed requests is consistently very low for all except the most extreme (and clearly unrealistic) of parameters for two-speed disks. For this reason and the fact that network servers can accommodate substantial response time degradations, hereafter we do not discuss response times.

File system coverage. Figure 3 presents the energy savings accrued by the four servers, as a function of file system coverage.

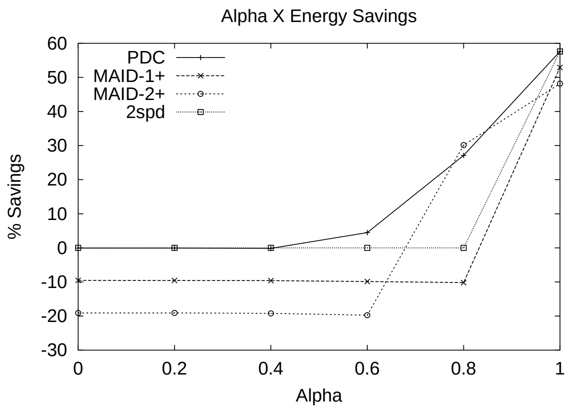

Figure 4: Energy savings per file popularity.

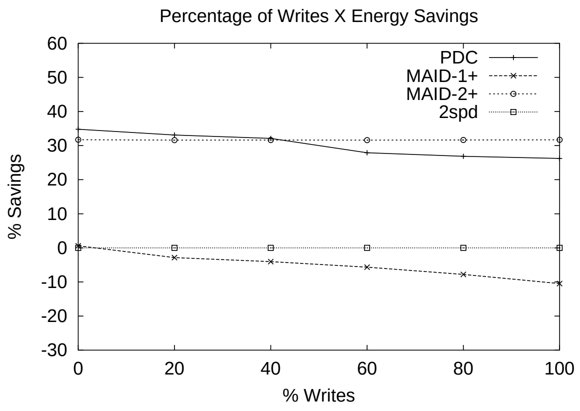

Figure 5: Energy savings per percentage of writes.

Recall that the default coverage we consider in our other analyses is 40%. As expected, higher coverages mean that disk loads increase and therefore the energy savings are reduced. On the other hand, higher coverage also allows the Nomad FS file placement algorithm (MQ) to place a larger number of files more intelligently in the array. As a result, the energy savings accrued by PDC degrade gracefully. MAID-2+ also degrades gracefully, since its two cache disks provide enough extra bandwidth at the default request rate to compensate for increases in coverage. In contrast, MAID-1+ suffers severely with increasing coverage, as a result of the extra pressure on the cache disk. When this disk becomes fully utilized, the non-cache disks also start operating at high speed.

File popularity. Figure 4 depicts the energy gains for different values of the Zipf coefficient (alpha). Alpha represents the file popularity as seen by the server, not the disk subsystem. Recall that the default popularity we assume in our other analyses is 0.85. The figure shows that all systems suffer as we decrease the popularity. The reason is that decreased popularity means lower main memory cache hit rates, which yield higher disk loads. Furthermore, the PDC file migration approach does not work as well when file popularity is decreased. Despite these problems, PDC degrades more gracefully than MAID or 2spd; it provides energy gains for popularities as low as 0.6. MAID does not behave as well as PDC because the files on its cache disks exhibit worse (popularity-induced) temporal locality with decreased popularity. This increases the load on the cache disks. When the cache disks become fully utilized, the traffic directed to the other disks increases further, forcing all disks to operate at high speed. As a result, the MAID systems consume more energy than EO for most of the parameter space.

Percentage of file write operations. Figure 5 shows the effect of the percentage of writes on the energy savings provided by the servers we consider. Recall that we have been considering read-only workloads by default. The figure shows that the energy savings provided by the different techniques decrease slightly (if at all) with increases in the percentage of writes. The reason for the slight decreases is that more writes increase the disk traffic when dirty blocks are evicted from the main memory cache. In fact, in the MAID-based servers, writes can cause even more traffic when dirty disk cache blocks have to be written back to their corresponding non-cache disks.

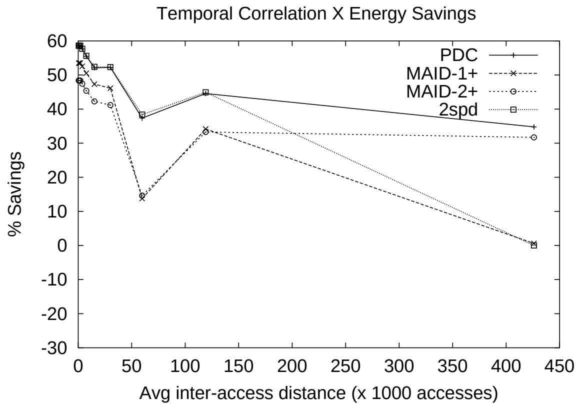

Figure 6: Energy savings per temporal correlation.

Temporal correlation. Figure 6 shows the effect of temporal correlation on the energy savings of the servers we consider. We compute the average number of file accesses between two consecutive accesses to each file and compute the overall average for all files. Recall that we assume no temporal locality besides that induced by file popularity in our other analyses. For each curve in the figure, this means the rightmost point.

We can see that all systems benefit from higher temporal correlation (i.e., smaller average inter-access distance). The higher correlation increases the main memory cache hit rate and reduces the load on the disk subsystem. This effect is highly beneficial for 2spd. MAID has the added benefit that high temporal correlation causes the cache disks to work more effectively. However, the MAID systems have the energy overhead of the cache disks themselves. Again, PDC is the most consistent and robust system across the parameter space.

4.2.2 File Server Characteristics

Number of disks. Figure 7 shows the energy savings of the different versions of our server, as a function of the number of disks. The total storage size is not changed, therefore simulations with more disks have fewer files per disk and do not require additional cache disks for the MAID systems. Recall that the default number of disks we assume in our other analyses is 8 for PDC and 2spd, 9 for MAID-1+, and 10 for MAID-2+.

We can see that increasing the number of disks increases savings quite significantly, as one would expect. The smaller set of accesses directed to each disk allows all disks to operate at low speed. Note

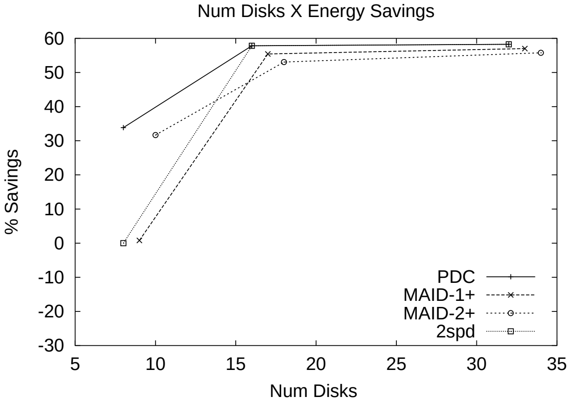

Figure 7: Energy savings per number of disks.

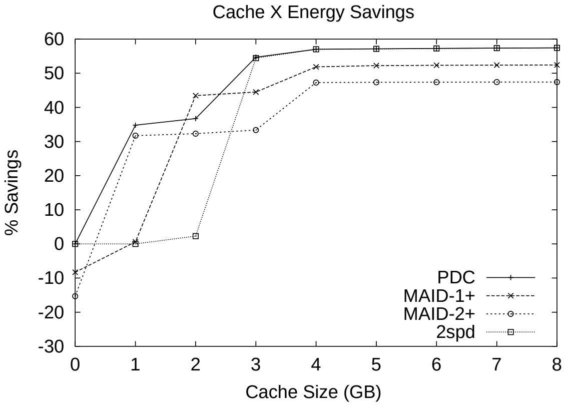

Figure 8: Energy savings per main memory cache size.

also that the energy overhead of cache disks is better amortized in larger array configurations. However, if we had assumed the number of cache disks to increase proportionally to the number of regular disks, we would have seen overheads of around 10% and 20% for the two MAID systems in these large configurations.

Main memory cache size. Figure 8 considers the effect of main memory cache size. Recall that the default size of the memory cache we assume in our other analyses is 1 GByte.

Cache size increases produce significant benefits for all servers. With caches of 5 GBytes or larger, both PDC and 2spd approach the maximum achievable energy savings of 60%, since the cache miss rate becomes so low that all disks start to operate at low speed. The MAID systems also benefit from larger main memory caches, but settle at lower energy savings with very large caches.

Migration frequency. Figure 9 shows the effect of changes in the migration period in Nomad FS. Recall that the default migration period we assume in our other analyses is every 1/2 hour. For very short migration periods, there may not be enough time to complete the migrations until the next round of migrations needs to start. In these cases, we simply stop the current migration process midway and start the next. Since MAID and 2spd do not involve migrations, we plot their energy gains as horizontal lines for comparison.

From the figure we can see that the frequency of migrations has a substantial impact on the gains achieved by PDC. With longer periods, PDC takes longer to adjust to access patterns. It is also interesting to note that short migration periods provide slightly higher energy savings than our default setting. The reason is that the energy cost of file placement changes is a small fraction of the energy consumed in between migrations. However, these shorter periods also come with substantial increases in the percentage of delayed requests, due to the intense file migration traffic.

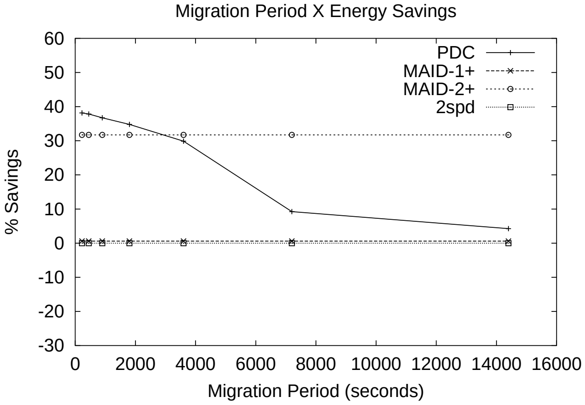

Figure 9: Energy savings per reconfiguration period.

Description	HUM	POP
Avg. request rate	241 reqs/sec	263 reqs/sec
Coverage	100%	100%
File popularity	0.70	0.93
Percentage of writes	35%	0%
Temporal correlation	467280 reqs	94568 reqs
Cache size	1 GByte	64 MBytes
Migration period	672 minutes	168 minutes

Table 6: Parameters of real traces and server settings. Request rate values correspond to the accelerated traces. Migration periods correspond to one week in the non-accelerated traces.

5. REAL TRACE RESULTS

The results so far have assumed synthetic traces and exponential request arrivals with a fixed mean arrival rate. In this section, we study real traces that exhibit pronounced load intensity variations, with load peaks in the afternoon and valleys at night. Note however that we accelerated the traces by factors of 15 (HUM) and 60 (POP), because even the peak loads in the original traces are relatively low for our disk array.

We simulate a tight coupling between the file server and the network server; the file server forms the "bottom half" of the network server on a single machine. We assume a "warm up" period corresponding to one week of the original, non-accelerated traces. Table 6 lists the characteristics of the traces and server settings with respect to the parameters we just explored. All simulations assume two-speed disks and 8 (PDC), 9 (MAID-1+), or 10 (MAID-2+) disks in the array.

Table 7 lists the energy and percentage of requests delayed (by more than 200 milliseconds) for each technique and trace after their warm up periods. EO consumes 2022095 J (HUM) and 256745 J (POP) with virtually no delayed requests.

These results confirm the main observations of the previous section, i.e. under light load, PDC conserves a significant amount of

Technique	HUM	POP
PDC Energy	1472438 J	147332 J
MAID-1+ Energy	1605547 J	170232 J
MAID-2+ Energy	1700585 J	181815 J
2spd Energy	1538228 J	160114 J
PDC Delayed	5.3%	8.0%
MAID-1+ Delayed	14.3%	8.5%
MAID-2+ Delayed	11.2%	8.1%
2spd Delayed	6.0%	7.6%

Table 7: Real trace results: energy and percentage of requests delayed.

energy (27% for HUM and 43% for POP) while the cache disks in MAID reduce energy gains. The one exception is that 2spd conserves more energy than both MAID systems. 2spd performs well because, throughout most of the traces, the disk load is low enough that all disks can be at low speed. This characteristic makes these traces ideal for 2spd. Despite these good results, PDC still conserves more energy than 2spd (14% more for HUM and 13% more for POP) without an increase in the percentage of delayed requests.

Note also that the percentage of delayed requests is higher than in our parameter space exploration. The main reason for these results is that, going from each valley to the next peak, disk loads increase very quickly in the accelerated traces. Under this scenario, disk contention develops until the two-speed disks can transition to high speed. This should not be a problem in practice however, since load variations are substantially slower in real time.

6. RELATED WORK

We originally suggested PDC and our energy-aware file server at the work-in-progress session of SOSP'01 [17]. This paper is the first detailed and complete evaluation of our proposals. The inspiration for this work came from our work on Load Concentration (LC) and dynamic cluster reconfiguration for energy conservation [18]. PDC is an application of a similar concept to LC to storage systems. However, we found that it was much harder to successfully apply PDC than LC, given the information required to make intelligent migration decisions and the large number of important parameters involved.

As far as we know, there have been six other works on disk energy conservation for servers [5, 7, 10, 11, 20, 21]. Carrera *et al.* [5] and Gurumurthi *et al.* [10] considered multi-speed disks for stand-alone servers. These papers showed that significant energy savings can be accrued by adjusting speeds according to the load imposed on the disk. Carrera *et al.* also show that a combination of laptop and SCSI disks can be even more beneficial in terms of energy, but only for over-provisioned servers. Our work is orthogonal to these contributions in that we seek to conserve energy in the context of disk array-based servers. For these systems, exploiting data movement between disks can provide significant additional gains, as we demonstrate in section 4.2.

In terms of disk array-based servers, Gurumurthi *et al.* [11] considered the effect of different RAID parameters, such as RAID level, stripe size, and number of disks, on the performance and energy consumption of stand-alone database servers running transaction processing workloads. They also observed that it is not possible to exploit idleness in this context. However, they did not consider any other energy conservation techniques.

Colarelli and Grunwald [7] proposed MAID and evaluated it for a large scientific workload. Although they mentioned a migration-based version of MAID, they decided not to pursue it in favor of the copy-based version we studied here. For their workload, MAID conserves no more energy than FT. We extend their work by implementing a prototype file server based on MAID, validating our simulator against the prototype, considering a broader range of parameters, and comparing copying against migration.

Finally, Zhu *et al.* [20] recently considered storage cache replacement techniques that selectively keep blocks from certain disks in the main memory cache, so that the disks can stay in low-power mode for a longer period of time. In another paper, Zhu *et al.* [21] study a more elegant approach to the same problem. Their work is similar to ours in that it also attempts to affect the load directed to disks. As another similarity, their work evaluated replacement techniques for disk arrays composed of multi-speed disks. However, they did not consider data movement between disks, which can provide greater benefits.

Bianchini and Rajamony [1] surveyed these and other works on power and energy conservation for server systems.

7. CONCLUSIONS

We have introduced a new energy conservation technique called PDC for disk arrays. We have also designed and implemented an energy-aware file server called Nomad FS that exploits this technique. For comparison, we developed two versions of our server that use other energy conservation techniques, MAID and FT. Using a validated simulator, we performed a comprehensive study of the parameters that affect energy conservation the most, pinpointing the scenarios under which each technique is useful, for disk arrays composed of either conventional or two-speed disks. The study allowed us to map the areas of the parameter space for which no technique can conserve energy. We also studied the behavior of the different techniques for two real traces.

In summary, we found that it is possible to conserve a substantial amount of energy during periods of light load on the server, as long as two-speed disks are used. We have also found that Nomad FS can deal more gracefully with increases in request rate and decreases in file popularity than the other servers. In addition, PDC achieved consistent energy savings for most of the parameter space. However, the PDC gains degrade substantially for long migration intervals. In comparison, the behavior of MAID is heavily dependent on the number of cache disks used. Furthermore, the energy overhead of these disks is pronounced in several parts of the parameter space. For the real traces, PDC achieved the best results, confirming the observations from the synthetic traces. We also showed that using two-speed disks without data movement behaves well when real disk loads remain light most of the time.

Based on these results, we conclude that the techniques we study will be very useful for disk array-based network servers when multi-speed disks become available.

Further research is required on the reliability implications of PDC. In fact, it is clear that PDC negatively affects the reliability of conventional disks due the large number of power mode transitions it causes. However, we have shown that PDC is not useful for conventional disks anyway. For multi-speed disks, we expect PDC to have a negligible impact on reliability. The reason is that real network server workloads (which exhibit load peaks in the afternoon and valleys at night) should induce at most two speed transitions/disk/day in the presence of PDC.

Another interesting research issue is the interaction of PDC with storage system reliability approaches, such as mirroring and RAID. Extending Nomad FS to deal with mirroring should be easy, but combining PDC with data striping can be more challenging.

Finally, our parameter space exploration and experience with real traces suggest that PDC is highly sensitive to the length of the migration period and the estimation of the load to be imposed on disks after migrations. Another research topic is how to make Nomad FS automatically determine the best settings for these parameters.

Acknowledgements

We would like to thank Enrique V. Carrera and the other members of the Rutgers DARK Lab for comments that helped improve this paper. We would also like to thank Prof. Uli Kremer for lending us the power measurement infrastructure of the Energy Efficiency and Low-Power Lab at Rutgers. Finally, we would like to thank Prof. Arthur Goldberg from New York University for giving us the Hummingbird trace.

8. REFERENCES

[1] R. Bianchini and R. Rajamony. Power and Energy Management for Server Systems. Technical Report DCS-TR-528, Department of Computer Science, Rutgers University, June 2003.

[2] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The Case for Power Management in Web Servers. In Graybill and Melhem, editors, *Power-Aware Computing*. Kluwer Academic Publishers, January 2002.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of IEEE InfoCom'99*, March 1999.

[4] E. V. Carrera and R. Bianchini. Improving Disk Throughput in Data-Intensive Servers. In *Proceedings of the 10th International Symposium on High-Performance Computer Architecture*, February 2004.

[5] 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.

[6] J. Chase, D. Anderson, P. Thackar, A. Vahdat, and R. Boyle. Managing Energy and Server Resources in Hosting Centers. In *Proceedings of the 18th Symposium on Operating Systems Principles*, October 2001.

[7] 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.

[8] E. N. Elnozahy, M. Kistler, and R. Rajamony. Energy-Efficient Server Clusters. In *Proceedings of the 2nd Workshop on Power-Aware Computing Systems*, February 2002.

[9] M. Elnozahy, M. Kistler, and R. Rajamony. Energy Conservation Policies for Web Servers. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, March 2003.

[10] 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.

[11] S. Gurumurthi, J. Zhang, A. Sivasubramaniam, M. Kandemir, H. Franke, N. Vijaykrishnan, and M. J. Irwin. 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.

[12] T. Heath, B. Diniz, E. V. Carrera, W. Meira Jr., and R. Bianchini. Self-Configuring Heterogeneous Server Clusters. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power*, September 2003.

[13] S. Jin and A. Bestavros. GISMO: A Generator of Internet Streaming Media Objects and Workloads. *ACM SIGMETRICS Performance Evaluation Review*, 29(3), November 2001.

[14] S. Jin and A. Bestavros. Sources and Characteristics of Web Temporal Locality. In *Proceedings of IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, August 2000.

[15] A. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive Snoopy Caching. *Algorithmica*, 3(1):79–119, 1988.

[16] K. Okada, N. Kojima, and K. Yamashita. A Novel Drive Architecture of HDD: "Multimode Hard Disk Drive". In *Proceedings of the International Conference on Consumer Electronics*, June 2000.

[17] E. Pinheiro and R. Bianchini. A Power-Aware Cluster-Based File System. Work-in-Progress Session, 18th Symposium on Operating Systems Principles. October, 2001.

[18] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath. Dynamic Cluster Reconfiguration for Power and Performance. In L. Benini, M. Kandemir, and J. Ramanujam, editors, *Compilers and Operating Systems for Low Power*. Kluwer Academic Publishers, September 2003. Earlier version published as “Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems” in the Proceedings of the Workshop on Compilers and Operating Systems for Low Power, September 2001.

[19] Y. Zhou, P.M. Chen, , and K. Li. The Multi-Queue Replacement Algorithm for Second-Level Buffer Caches. In *Proceedings of the 2001 USENIX Annual Technical Conference*, June 2001.

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

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