Conserving Disk Energy in Network Servers *

Enrique V. Carrera, Eduardo Pinheiro, and Ricardo Bianchini
Department of Computer Science
Rutgers University
Piscataway, NJ 08854-8019
{vinicio,edpin,ricardob}@cs.rutgers.edu

ABSTRACT

In this paper we study four approaches to conserving disk energy in high-performance network servers. The first approach is to leverage the extensive work on laptop disks and power disks down during periods of idleness. The second approach is to replace high-performance disks with a set of lower power disks that can achieve the same performance and reliability. The third approach is to combine high-performance and laptop disks, such that only one of these two sets of disks is powered on at a time. This approach requires the mirroring (and coherence) of all disk data on the two sets of disks. Finally, the fourth approach is to use multi-speed disks, such that each disk is slowed down for lower energy consumption during periods of light load. We demonstrate that the fourth approach is the only one that can actually provide energy savings for network servers. In fact, our results for Web and proxy servers show that the fourth approach can provide energy savings of up to 23%, in comparison to conventional servers, without any degradation in server performance.

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 always been a key concern for battery-operated computing devices, since the energy consumed by these devices determines their battery life. In the last two years, researchers have realized that energy conservation is also critical for high-performance network servers, even though these systems are connected to the electrical power grid.

The reason for this new focus is that network servers are often replicated to form large clusters, such as those that support data centers, hosting centers, and a multitude of Internet companies. The Google search engine, for example, is supported by 15K servers divided into 4 installations. These large clusters consume a significant amount of energy, which is reflected in high electricity bills. Not surprisingly, data from several sources (e.g. [30, 24]) show that energy represents a significant fraction of the cost of data and hosting centers. Perhaps more importantly however, energy conservation is an important goal for computer scientists, in that most power-generation technologies (such as nuclear and coal-based generation) have a negative impact on the environment.

A few efforts [29, 28, 6] have been made to conserve energy in network servers. These efforts tackled the high power supply losses observed in traditional servers. Thus, they focused on conserving energy by dynamically reconfiguring (or shrinking) a cluster of network servers to operate with fewer nodes under light load. Other efforts [3, 9, 10] tackled the energy consumed by the CPUs of network servers. Their approach was to conserve energy by using dynamic voltage scaling (e.g., [31, 13]) or request batching under light load.

Even though these efforts have made important strides in conserving energy in network servers, there is still much to be done. The disk energy consumption of network servers, for instance, is only starting to be addressed now. Disk energy consumption can be a serious problem for network servers, given that high server performance is paramount and high-performance disks consume significant amounts of energy, even when compared to microprocessors or power supply losses. In fact, this problem is worst for data-intensive network servers, such as proxy, file, and database servers, which require several disks per server. We even expect the problem to worsen in the future, as an increasing number of high-performance disks will be needed to match the performance of microprocessors.

Thus, in this paper we address the disk energy consumption problem by evaluating four approaches to solving it. The first approach, called *Idle*, is to leverage the extensive work on laptop disks and power disks down during periods of idleness. Unfortunately, our simple modeling of this approach demonstrates that network server disks have extremely short idle times, even during periods of light load and with large main memory file caches. Short idle times render this approach inappropriate, due to the high energy and performance overheads of powering disks up and down.

The second approach is based on the observation that current disks exhibit a variety of performance and energy consumption characteristics. For instance, high-performance disks consume sig-

*This research was supported in part by NSF grants EIA-0224428 and EIA-0203922.

ICS'03, June 23–26, 2003, San Francisco, California, USA.

nificantly more energy than comparably-slow laptop disks. Given this disparity, the second approach considers the direct replacement of each high-performance disk for lower performance, lower power disks. We refer to this as the *Replace* approach. Again, this approach does not work well, as we demonstrate using simple modeling, due to the relatively large number of lower power disks required to achieve the same performance and reliability properties of each high-performance disk.

The third and fourth approaches rely on the wide variations in the intensity of the load offered to real network servers. The third approach is to combine each high-performance disk with a laptop disk. We refer to this idea as the *Combined* approach. At first, this approach may sound counter-intuitive, given that more disks can effectively mean higher energy consumption. However, we can conserve energy by powering down the power-hungry disks when their higher performance is not required, i.e. under light load. Thus, the Combined approach requires that we mirror high-performance disks on laptop disks and dynamically switch between these two sets of disks. During the switch, we have to update the set of disks being powered on to guarantee data coherence, before the other set can be powered off.

Finally, the fourth approach is to use multi-speed disks, such that each disk is slowed down for lower energy consumption during periods of light load. We refer to this approach as *Multi-speed*. Multi-speed has two advantages over Combined: (1) in Multi-speed, there is no need to maintain two copies of data; and (2) the cost of Multi-speed should be lower, since there is no need for two storage media and motors. (Despite the more extensive hardware in Combined, its reliability can be made equivalent to that of Multi-speed, as we discuss later.) However, Multi-speed has two disadvantages: (1) it suffers from the performance overhead of changing speeds on the critical path of disk accesses, whereas the equivalent overheads in Combined can be hidden; and (2) the low-performance disk in Combined can be aggressively optimized for low power consumption, whereas Multi-speed does not allow such flexibility.

Because in academia it is difficult to reason about disk manufacturing costs, pricing strategies, and market considerations, *this paper focuses solely on evaluating the different approaches in terms of their energy and performance implications.* Our results for Web and proxy servers show that Combined can only provide non-trivial disk energy savings when servers are excessively over-provisioned, which is probably not a realistic scenario for modern network servers. In contrast, Multi-speed provides consistent benefits for realistic parameters. In particular, the disk energy savings produced by a two-speed disk range from 14%–23%, compared to a similar conventional disk, without any noticeable degradation in server performance. We expect the savings achievable by other servers with similar variations in load to lie in between these extremes.

Based on our results, we conclude that saving disk energy in network servers is not a straightforward task. We have evaluated four different alternatives and only one of them (the Multi-speed approach) is consistently beneficial. The other three approaches depend on several parameters that are not commonly found in current disks and/or network server workloads.

The remainder of this paper is organized as follow. The next section discusses the motivations for our work in detail. Sections 3, 4, 5, and 6 detail the Idle, Replace, Combined, and Multi-speed approaches to disk energy conservation in network servers, respectively. Section 7 describes the methodology and experiments we used to evaluate the Combined and Multi-speed approaches. Section 8 discusses the related works. Finally, section 9 summarizes our findings and concludes the paper.

2. MOTIVATION

Three main observations motivate our tackling of disk energy in network servers and our approaches to conserving this energy: (1) the energy consumed by disks in well-provisioned network servers is a significant fraction of the overall energy consumed by the servers; (2) the wide disparity in performance and power consumption between current off-the-shelf disks; and (3) the wide variations in the load offered to network servers over time. We next discuss each of these observations in turn.

2.1 Disk Energy in Network Servers

Due to the diversity of the network server equipment that we see in practice (and the fact that our own server hardware is clearly disk bandwidth-limited for certain network servers), determining the fraction of the total energy of real network servers that is consumed by the disk subsystem is not a trivial task. The disk energy consumption depends on the intensity of the load directed to the servers, the effectiveness of the main memory caches, and the number and type of disks used.

A modern high-performance network server typically consists of a powerful microprocessor, one or more network interfaces, and several high-performance SCSI disks. Provisioning such a server often involves estimating the maximum CPU throughput for the expected workload and provisioning the I/O capacity accordingly. Although we know of no published study of actual servers in the field, we expect load peaks to reach somewhere between 80% and 90% of the maximum throughput deliverable by the CPU; lower peaks would mean that the server is excessively over-provisioned, whereas higher peaks would mean that the server would not be able to deal with even small unexpected increases in load.

Regardless of the level of over-provisioning, the network interfaces should collectively be capable of transferring data at the CPU's maximum throughput. In terms of main memory size, when the files requested exhibit high temporal locality, a memory cache should generate a low miss rate (no more than 5%, say). For workloads without temporal locality, such as Web proxy workloads, large memories are almost useless; no reasonable cache size can achieve high hit rates. The set of disks should be capable of servicing the cache's miss rate without performance degradation, so that the microprocessor is not consistently under-utilized. And, of course, the set of disks has to provide enough storage space to hold the server's data.

We translate these observations into numbers using our own high-performance server, a 1.9 GHz Pentium 4-based server with a 15K rpm SCSI disk and a 1 GBit/second network interface. Assuming a common average Web request of 8 KBytes, our experiments show that a Web server running on our system can service 4340 requests/second, when files are always found in the main memory cache. Our experiments also show that our state-of-the-art SCSI disk (an IBM Ultrastar 36Z15 15K-rpm disk) can deliver about 1.5 MBytes/second for the workloads we consider.

For a memory cache miss rate of 5%, the disk subsystem of a Web server would need to provide at least 1.7 MBytes/second. This translates into a disk subsystem with two 15K-rpm SCSI disks. In contrast, the disk subsystem of a proxy server would need to provide at least 31.2 MBytes/second, assuming 4340 requests/second and a 10% *hit* rate in the memory cache (which is an optimistic main memory hit rate assumption for proxies). This disk throughput would require 21 15K-rpm disks.

With these configurations and a few power measurements we can compute the disk energy consumption of well-provisioned network servers. According to our measurements, our fast SCSI disk consumes about 14 Watts when fully utilized, whereas the rest of our server consumes about 90 Watts when fully utilized. (Refer to the second column of table 1 for our fast SCSI disk's main characteristics.) This means that disk energy consumption in a well-provisioned Web server would account for 24% (28 Watts out of 118 Watts over time) of the overall energy consumed by the server, whereas the disk energy consumption of the Web proxy would account for 77% (294 Watts out of 384 Watts over time) of the total energy.

These simple calculations suggest that *disk energy is an important issue for Web servers and a dominating concern for proxy servers.* Just as for proxies, disk energy should also be a dominant effect for other data-intensive servers, such as database, file, or storage servers. Finally, as techniques that tackle CPU energy [3, 9] and power supply losses [29, 28, 6] in servers start to be applied in practice, the disk fraction of the total energy consumption will be even more significant.

2.2 Characteristics of Current Disks

Current disks exhibit a wide variety of performance and energy characteristics. Table 1 compares the key parameters of the three IBM disks we study in this paper: the Ultrastar 36Z15 15K-rpm SCSI disk [17], the Ultrastar 73LZX 10K-rpm SCSI disk [18], and the Travelstar 40GNX 5400-rpm IDE laptop disk [19]. Throughout the paper we refer to the first disk on this list as our high-performance SCSI disk.

Our two Ultrastar disks are very similar, except for their rotational speeds and numbers of platters. These two characteristics affect power consumption in different ways. In particular, we believe that the differences in power consumption (in active and idle states) between these two disks are due mostly to their difference in rotational speed. The reason is that the series that these disks belong to both allow a maximum of 6 platters, so we assume that the spindle motors are also similar, except for rotational speed. Furthermore, there are numerous examples of disk series (e.g., Deskstar 60GXP from IBM, and Cheetah 73LP and Barracuda 180 from Seagate) where disks with different numbers of platters are rated at the same idle power. The effect of the number of platters is most significant on the energy and time overheads associated with spinning disks up and down, as suggested by [11].

In contrast, a comparison between the extremes in performance, the Ultrastar 36Z15 and the Travelstar 40GNX, shows that the laptop disk consumes only a fraction of the energy consumed by the Ultrastar disk. The power consumption of the Ultrastar disk in idle and standby states, for instance, is a factor of 10 higher than that of the laptop disk. The time and energy overheads involved in transitioning states are a factor of at least 3 higher for the Ultrastar disk. However, the performance of laptop disks is also much inferior. In terms of internal bandwidth, for instance, the Ultrastar disk is about twice as fast as the laptop disk.

These characteristics suggest that even a few laptop disks still consume less power and energy than a single fast SCSI disk. Thus, it might be appropriate to replace each SCSI disk of a server with a few laptop disks, provided that some redundancy is also implemented to improve reliability. This observation motivates the Replace and Combined approaches to disk energy conservation.

2.3 Network Server Workloads

The intensity of network server loads is known to vary widely over periods of several minutes or hours (e.g., [1, 6]). For example, it is common for load peaks to occur in the late afternoon (in the US) of weekdays and load valleys to occur over night (in the US). It is also common for loads to be less intense during weekends. These trends are documented in several publicly available Web server traces, such as Clarknet [2], a trace of a commercial Internet service provider, and IBM [6], a trace of the main www.ibm.com site. These traces have peak:valley load ratios of 3:1 and 4:1, respectively.

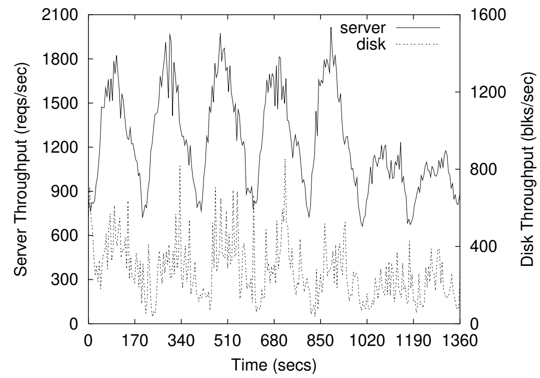

Figure 1: Server and disk throughputs for the Clarknet trace.

These ratios can be even higher for servers of sporting events. For instance, the site of the Nagano Winter Olympic Games had peaks of 1800 requests/second, the 1998 Championships at Wimbledon had peaks of 2400 requests/second, and the 1998 World Cup had peaks around 2800 requests/second. In all this traces, the ratio between the maximum and minimum load is at least 10:1 [20].

We find that these trends in load offered to the server are verified for the disks as well. Figure 1 shows an example of this behavior. The figure shows the server throughput (in requests/second) and the disk throughput (in blocks/second) of a Web server servicing requests according to the Clarknet trace. (We have accelerated the trace in order to reproduce the behavior of busier servers. Further details about our methodology are presented later.) It is interesting to note that the disk throughput follows a similar trend to the overall server throughput. More importantly, the ratio of load peaks and valleys is larger for the disk than for the server as a whole.

Given these trends in disk load and the low consumption of laptop disks, it might be appropriate to switch between sets of high-performance and laptop disks according to the load offered to the disk subsystem. This is the motivation for our proposal and study of the Combined approach to disk energy conservation.

Furthermore, these load trends and the fact that similar disks (such as the Ultrastar disks of table 1) with different speeds consume substantially different power and energy suggests that producing multi-speed disks would provide significant potential to conserve energy. This observation motivates the Multi-speed approach to conservation.

3. IDLE: EXPLOITING IDLENESS

Previous works (e.g., [21, 8, 16, 23, 22, 15, 26]) have proposed several energy management techniques for laptop disks. Most of them are based on powering disks down during periods of idleness, since disk idle times are relatively long for the interactive workloads of laptops. The amount of idle time needed to justify the cost of powering the disk down and up (on the next access) is called the *break-even threshold*. In technical terms, the break-even threshold is defined as the amount of time between two accesses for which the energy consumed in idle state is the same as that of powering the disk down to standby state immediately after the first access and later powering the disk back up for the second access.

The Idle approach tries to leverage this energy conservation technique in the context of network servers. For our server's high-

Parameter	IBM 36Z15 Ultrastar (high perf)	IBM 73LZX Ultrastar (low perf)	IBM 40GNX Travelstar (laptop)
Standard interface	SCSI	SCSI	IDE
Capacity	18 GBytes	18 GBytes	20 GBytes
Number of platters	4	2	2
Rotations per minute	15000	10000	5400
Disk controller cache	4 MBytes	4 MBytes	8 MBytes
Average seek time	3.4 msec	4.9 msec	12 msec
Average rotation time	2 msec	3 msec	5.5 msec
Internal transfer rate	55 MB/sec	53 MB/sec	25 MB/sec
Power (active)	13.5 W	9.5 W	3.0 W
Power (idle)	10.2 W	6.0 W	0.82 W
Power (standby)	2.5 W	1.4 W	0.25 W
Energy (spin down)	13.0 J	10.0 J	0.4 J
Time (spin down)	1.5 sec	1.7 sec	0.5 sec
Energy (spin up)	135.0 J	97.9 J	8.7 J
Time (spin up)	10.9 sec	10.1 sec	3.5 sec

Table 1: Main characteristics of two SCSI disks and an IDE laptop disk, according to IBM's manuals and our own power measurements. Time (and energy) to spin up or down considers the minimum interval between stable power readings before and after the transitions.

performance SCSI disk, the break-even threshold is 15.2 seconds, according to the energies and times defined in table 1. The key question is whether network server disks see this much idle time.

Let us consider this question in the context of our own 1.9 GHz Pentium 4-based system running a Web server, under optimistic assumptions. Again, our server can service 4340 8-KByte Web requests/second when all requests are served from main memory. Assuming that load peaks reach only 50% of this maximum throughput and that load valleys are a factor of 10 less intense than the peaks, this means that a memory cache miss rate that is lower than 0.03% is needed for the average idle time of 15.2 seconds. This is a very small miss rate even for a server with a large memory cache.

Given these negative results, there is no point in repeating this exercise for proxy servers. Proxy server disks are much more highly utilized than Web server disks, due to the inability of memory caches to filter a non-trivial fraction of the requests directed to proxies.

In summary, the Idle approach to disk energy conservation is clearly not appropriate for network servers. Under most reasonable scenarios, the gains (if there are any) will be insignificant, due to the lack of idleness in the disk subsystem. For example, in all our experiments no idle period lasted longer than 15.2 seconds. Thus, we do not pursue this approach further.

4. REPLACE: EXPLOITING LOWER POWER DISKS

Another potential approach to conserving disk energy in network servers is to simply replace each high-performance disk with one or more lower power disks. Such a replacement obviously would have to guarantee the same amount of storage, performance, and reliability as those achieved with the high-performance disks. We next reason about this approach using the disks described in table 1 as an example.

In terms of storage capacity, most current high-performance SCSI disks could potentially be replaced 1-to-1 by lower performance SCSI disks or even laptop disks. In fact, only the largest (and fairly uncommon) of SCSI disks would not have a same-size laptop counterpart. In addition, storage capacity has consistently been a "moving target", given the pace with which new disk generations are produced. If a certain capacity has been reached by SCSI disks but not by laptop disks, all we have to do is wait a few months for the next-generation laptop disk.

In terms of performance, it is not as easy to determine the number of lower power disks required for each high-performance disk. Considering access latency first, no number of lower power disks can provide the same disk latency as a high-performance disk. Nevertheless, latency is not the key performance issue for network servers, throughput (bandwidth) is, since server latencies are often overwhelmed by wide-area network trips.

The average service time (occupancy) of a disk request is defined as $m = avg\_seek\_time + avg\_rotation\_time + req\_size/t$, where t is the internal transfer rate. Thus, we can also define the maximum throughput of a disk as req_size/m. Using these formulas and the values listed in table 1, it becomes clear that at least two lower power SCSI disks would be required to match the throughput of each high-performance disk. At this 1-to-2 ratio, energy gains would be impossible, since our high-performance disk consumes less than twice the energy of the lower power SCSI. Nevertheless, replacing the high-performance disk for laptop disks is still a possibility.

Figure 2 shows the maximum throughput achievable by our high-performance disk ("HP") and several sets of our laptop disks ("LT"), as a function of the average request size. The results for the sets of laptop disks assume perfect load balancing conditions. The figure shows that the disk subsystem would need three laptop disks per high-performance disk, in order to offer almost exactly the same throughput under perfect load balancing.

So far, it seems that three laptop disks should be enough to replace each high-performance disk in terms of both storage capacity and throughput. However, for reliability not to be compromised excessively, some form of redundancy has to be implemented on the laptop disks. Mirroring (RAID 1) or RAID 5, for example, could be used. But, to compensate for the disk bandwidth consumed by duplicated writes or stripe updates, an extra disk (at least) would be required. Thus, for the system to achieve the same storage capacity and throughput of the original server, and a reasonable level of reliability, we would need at least four laptop disks for each high-performance disk.

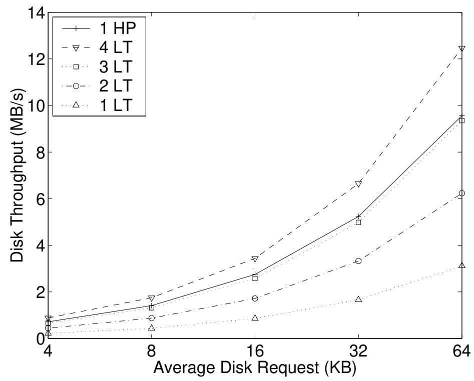

Figure 2: Maximum disk subsystem throughput under perfect load balancing. HP = high performance, LT = laptop.

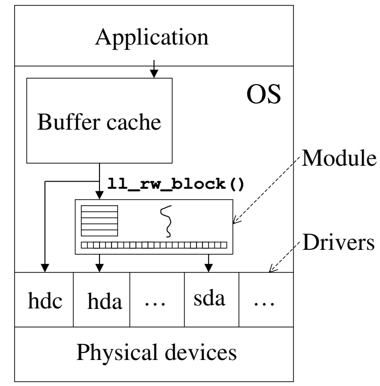

Figure 3: Linux implementation of the Combined approach.

Unfortunately, given the data of table 1, this would not be a profitable replacement in terms of energy consumption, since the ratio of power consumption between high-performance and laptop disks is also approximately four. Although we do not discuss this, we have found that a similar analysis would also be unfavorable to replacing high-performance disks with desktop IDE disks. Thus, we do not pursue this approach further.

5. COMBINED: EXPLOITING OFFERED LOAD VARIATIONS

In Combined, the idea is to associate each high-performance disk with a lower power disk, called a secondary disk. The disks should have the same size and mirror each other. The goal is then to keep only one of the mirrors up at each point in time, according to the load imposed on the disk subsystem; during periods of high load, only the high-performance disks are powered on, whereas during low load only the lower power disks are powered on. For a short period during the switch from one set of disks to the other, both sets of disks remain powered on, so that the set coming up can be made coherent. Coherence actions are only necessary for the updates that occurred while the set of disks coming up was powered off.

Given that in Combined the two disks consume energy all the time (even when in standby state), using a lower power SCSI disk as the secondary disk would leave little margin for energy savings. Thus, we only consider laptop disks as the secondary disks.

Even though these types of disks exhibit significantly lower performance than our high-performance SCSI disk, this is not a problem. The reason is that the secondary disk is only on when the load is low, so high throughput is not necessary. Data reliability is not a major issue either, since we always keep a mirror (old copy plus log of recent updates) of the data set that is currently active. However, the manageability of the disk subsystem does worsen, due to the larger number of disks.

Implementation. To implement the Combined approach, we designed a module for Linux that allows the creation of multiple virtual devices; each virtual device is mapped to a pair of disks. The module has three key components: (1) a translation table per virtual device that specifies which physical disk drive to use on each access; (2) a kernel thread that selects which disk to use depending on the load on the disk subsystem; and (3) a bitmap per disk, specifying all the blocks that have been written since the disks of the corresponding virtual device were last made coherent. For clarity, the rest of this description assumes a single virtual device.

Figure 3 shows the architecture of the Linux kernel after our module has been inserted. Because our module is inserted at a low level, all disk traffic (including metadata accesses) is visible to it. To simplify and optimize our implementation, the module intercepts all calls to the ll_rw_block() kernel routine. This routine is the lowest-level device-independent routine used by the buffer cache to satisfy a disk block miss. For every intercepted call, the module simply changes the mapping from the virtual device number to a physical one, according to the translation table.

The translation table is maintained by the kernel thread. This thread monitors the load on the disks and applies an Exponentially Weighted Moving Average (EWMA) filter to the measure of the load offered to these disks. Our EWMA filter uses an α of 0.875 in order to smooth out the bursty disk load, as suggested in [6]. Load observations occur every 5 seconds. The filter tries to eliminate extraneous load spikes by disregarding a load observation that is more than twice as high/low as the previous observation. After this first disregarded spike, the filter does take new high/low observations into account. Based on the output of the filter, the kernel thread chooses the disk that should service the offered load, and updates the corresponding translation table entry.

A switch between the disks involves updating any blocks that have been written since the disk being brought up was last active. To keep track of which blocks are written, our module implements a bitmap of dirty blocks per disk. Only one bitmap is active at a time, except during coherence maintenance. A bitmap associates a single bit with each 4-KByte block of disk storage (this means a 32K-fold reduction in storage requirements, e.g. a 32-GByte disk only requires 1 MByte for its bitmap). A bit is set in a bitmap when an intercepted ll_rw_block() call produces a disk write. Every time a block is written, the corresponding bit is set in the currently active bitmap. The dirty blocks themselves are not stored separately in main memory. When disks have to be made coherent, the dirty blocks are read from the buffer cache. If they can still be found there, no actual disk reads are performed. Coherence-related disk writes are performed directly to the corresponding disks.

As mentioned above, both bitmaps are active during the coherence procedure. The reason is that we want the coherence procedure to be executed in the background of regular server accesses, so it is important for as many requests as possible to be served by the high-performance disk to avoid performance degradation. During the coherence procedure, disk reads and writes are treated differently. Disk writes use the high-performance bitmap only. Disk reads first check the laptop bitmap. If the corresponding bit is set there, the block is read from the laptop disk. Otherwise, the high-performance disk is read. Because the kernel thread runs concurrently with regular file accesses, access to the bitmaps is synchronized with a lock. After the bitmap of the currently active disk has been cleared, the switch between the disks can happen.

Finally, note that using bitmaps to keep track of dirty blocks has two interesting characteristics: (1) a disk block that has been written by the server more than once is updated only once; and (2) the kernel thread can update the dirty blocks sequentially, thereby decreasing seek and rotational latencies.

6. MULTI-SPEED: EXPLOITING OFFERED LOAD VARIATIONS

The data about our very similar SCSI disks in table 1 suggests that it is possible to conserve energy by changing disk speeds (and power and energy consumptions) according to offered load; the higher the load, the higher the speed. This approach does not require multiple disks, coherence, or special care about reliability or manageability. However, the concern here is that changes in speed should certainly involve performance and energy overheads.

Our study of Multi-speed seeks to assess the potential trade-offs and benefits of these multi-speed disks. In particular, we study a *two-speed* disk that has the performance, power, and energy properties of our SCSI disks. The disk controller maintains information about the disk load (smoothing and filtering it using the same scheme as the kernel thread in Combined) and decides to change speeds based on a comparison between its measure of load and a pre-defined *switching threshold*. We discuss using more sophisticated threshold approaches at the end of section 7.

Emulation. Because multi-speed disks are not available in the market, at least as far as we know, we perform our study using emulation. The emulation keeps our two SCSI disks powered on all the time, but directs read accesses to them according to load; load that is higher than the switching threshold is handled by the 15K-rpm disk, whereas load that is lower is handled by the 10K-rpm disk. All write accesses are immediately directed to both disks, so there is no need for bitmaps or coherence-maintenance periods. The emulation also assigns performance and energy costs to the speed transitions. With independent information about each disk and about the offered load, the emulation determines the performance and energy consumption of our emulated two-speed disk. For example, when the disk speed is supposed to be low, we use the data for the lower performance disk and disregard the data for the fastest disk.

To accomplish this emulation, we use a simplification of the infrastructure used in Combined. More specifically, we again use a kernel module to intercept disk accesses and a translation table to determine which disk to access at each point in time. The speed transition delay is emulated by preventing any accesses to disk for the corresponding period of time. The energy cost of transitions is adjusted after the emulation is over. For each transition, we add the difference between the desired energy cost and the energy already spent during the emulation, i.e. the energy of keeping the high-performance disk idle for the transition delay.

We believe our emulation approach to be more precise than simulation or analytical modeling, because we actually experiment with real network server traces, software, hardware, and perform real power measurements. In fact, note that having a separate cache in the controller of each SCSI disk does *not* affect the accuracy of our emulation of the two-speed disk. The reason is that the caching done at the (very large) application and/or operating system caches eliminates any potential temporal locality in the accesses to the (relatively small) controller cache. We had already observed this effect in [5].

7. EXPERIMENTAL EVALUATION

7.1 Methodology

We use a 1.9 GHz Pentium 4-based machine running Linux (kernel 2.4.18) as our network server hardware. The server also includes 512 MBytes of memory, the SCSI Ultrastar 36Z15 disk, the SCSI Ultrastar 73LZX disk (when evaluating Multi-speed), the laptop Travelstar 40GNX disk (when evaluating Combined), and a Gigabit Ethernet network interface.

For both Combined and Multi-speed, by default our kernel module switches to the lower performance SCSI disk when the offered disk load falls below 80% of the maximum bandwidth of the lower performance disk; when the load increases beyond this same threshold, the module switches back to the high-performance SCSI disk. We also consider a 90% load threshold. When evaluating Multi-speed, we assume each speed change to cost 5 seconds and 68 Joules in performance and energy, respectively, by default. These values correspond to half of the cost of spinning up (all the way from standby state, when the spindle motor is off) our high-performance disk. These are pessimistic estimates of the cost such transitions would have for a real two-speed disk. We study the effect of these two parameters as well.

A 667 MHz Pentium III-based machine generates load for the server according to real workload traces. We run two types of network servers on the server hardware: a Web server called PRESS [4] and a simplified proxy server. The trace for the Web server comes from Clarknet, a commercial Internet service provider, and was collected for a week from 09/04/95 to 09/10/95. The trace involves only 34K different files that occupy approximately 400 MBytes of disk space. The requested files have an average of 9.3 KBytes. The trace is basically comprised of reads, although the update of the filesystem metadata produces 7% of write accesses. Because this trace is small by today's standards, we boot the server with only 160 MBytes of memory to achieve a memory cache miss rate of 3.6%.

The proxy server trace comes from the Hummingbird project at AT&T and was collected on 06/25/98. We have preprocessed the original trace to eliminate non-cacheable and incomplete URLs. The trace has 440K different URLs with a footprint of approximately 4.9 GBytes. The requested files have an average size of 8.3 KBytes and produce a 43% proxy miss rate. Each proxy hit causes a read operation, whereas a miss is reproduced by a write followed by a read. The workload has a large fraction of disk writes (30%). We boot the server machine with the full main memory for the proxy experiments, but the byte hit rate in the memory cache is only 4%.

We accelerate the traces to reach 80% and 90% of the maximum performance of the servers. These are the extremes of the range of utilizations we consider appropriate for network servers. Our Web server is able to service up to 2520 requests/second for the Clarknet trace with the 160 MBytes of main memory. The proxy server can service up to 335 requests/second for the Hummingbird trace.

Note that our server hardware is clearly disk bandwidth-limited compared to the well-provisioned servers discussed in section 2.1. However, our results should be the same for well-provisioned servers as well, assuming the scaling of the server loads, since we determine the achievable savings on a per disk basis.

For power measurements, we use a multimeter that monitors the 5V and 12V lines powering the disks. Periodically, the multimeter sends power information to another computer, which stores it in a log for later use. Due to limitations of our multimeter, the 5V and 12V lines cannot be monitored at the same time. So, each Combined experiment has to be run twice. For each Multi-speed emula-

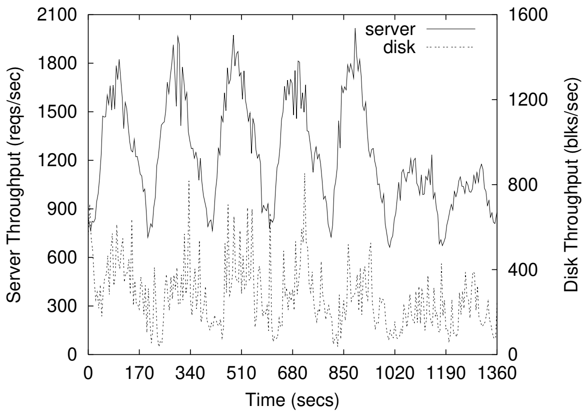

Figure 4: Server and disk throughputs for the Clarknet trace. Peaks of 80% utilization.

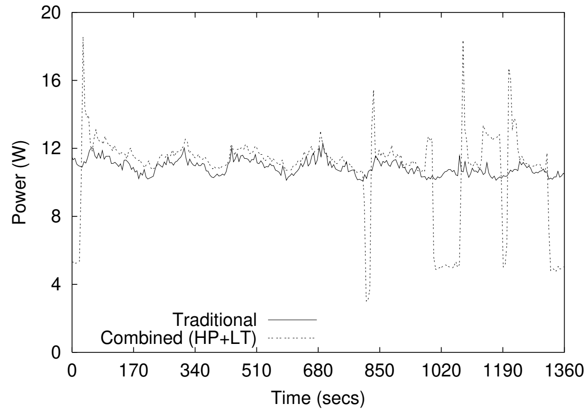

Figure 5: Power consumption of traditional and Combined systems for the Clarknet trace. Peaks of 80% utilization.

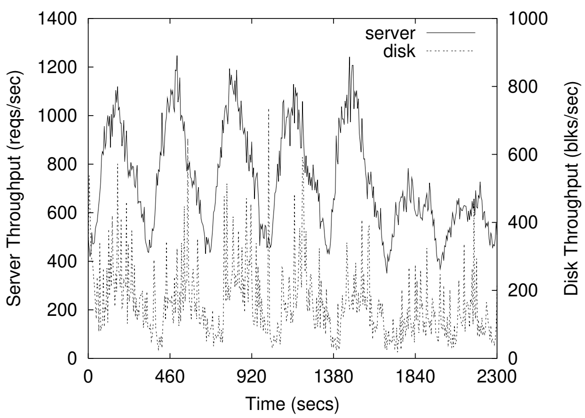

Figure 6: Server and disk throughputs for the Clarknet trace. Peaks of 50% utilization.

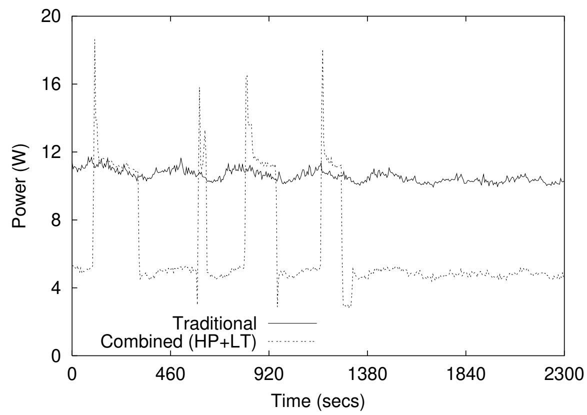

Figure 7: Power consumption of traditional and Combined systems for the Clarknet trace. Peaks of 50% utilization.

tion, two extra runs are required to isolate the consumption of each disk. The logs produced in these runs are later "synchronized".

7.2 Evaluating Combined

Web server. We start our experimental evaluation with the Web server results. Figure 4 shows the server and disk throughputs for the Clarknet trace for server load peaks of 80% of our server's maximum throughput. (Figure 4 is the same as figure 1 but is repeated here for clarity.) The figure shows the expected behavior, namely an alternation of load peaks and valleys with lighter loads on weekends. The disk loads follow the same trend, but are more bursty.

Figure 5 depicts the power consumption of our Web server disks for Clarknet, again when server load peaks reach 80% of the maximum achievable throughput. The figure shows results for a traditional server system with our high-performance disk and a Combined server system with our two disks. The disk energy consumed by the server is equivalent to the area below the two power curves. This figure clearly shows that the Combined system is not capable of conserving much energy under high load. During most of this experiment, the disk load is higher than the bandwidth of the laptop disk, preventing any gains; the high-performance disk was only turned off 4 times. The Combined system is able to conserve only 1% of the disk energy consumed by a traditional server in this experiment. In addition, we find the CPU energy spent in coherence maintenance to be minimal, compared to the overall disk energy.

Proxy server. Our proxy server delivered similarly negative results, so we do not show the corresponding figures. When the trace is accelerated to generate load peaks of 80% of the maximum proxy throughput, the Combined system cannot conserve any energy. The reason is that the load on the disk subsystem is high throughout most of the experiment and the number of disk blocks that are written is quite large. Towards the end of the experiment, when the disk load finally subsides, the laptop disk is brought up-to-date over a small period of time. Overall, we find that the Combined system consumes slightly (2%) more disk energy than the traditional one. The relatively small amount of CPU energy consumed by coherence maintenance makes this result slightly worse.

Discussion. These results for the Combined approach suggest that disk energy savings are not achievable for realistic server loads. We also performed a few experiments to determine how under-utilized the servers would have to be for the Combined approach to succeed. We find that, when servers exhibit load peaks of only 50% of their maximum throughput, the achievable disk energy gains reach 16% for Web proxies and 41% for Web servers. Figures 6 and 7 show the results for the Web server at this level of utilization. Better results can probably be achieved, but only for servers that even more over-provisioned or that exhibit even more significant variations in offered load.

We also considered the effect of using a desktop disk, rather than a laptop disk, as the secondary disk in the Combined approach. Pro-

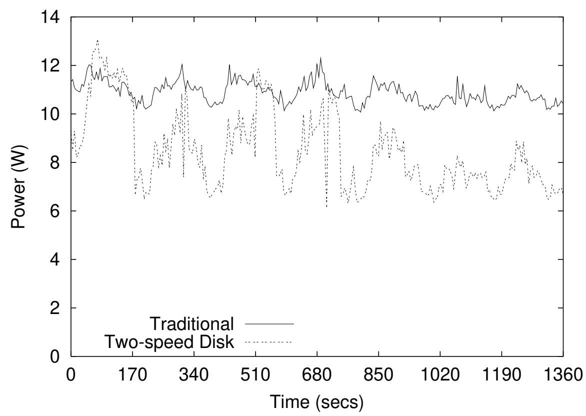

Figure 8: Power consumption of traditional and two-speed systems for the Clarknet trace. Peaks of 80% utilization.

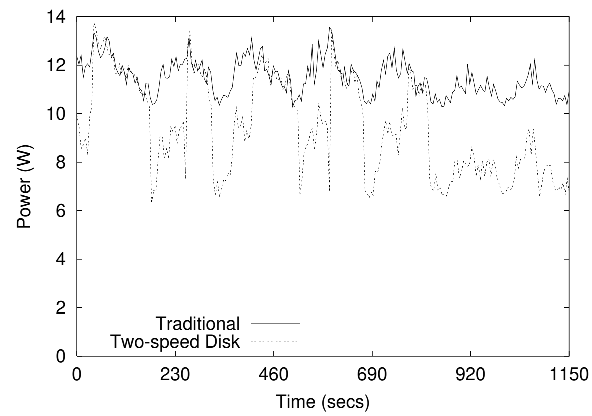

Figure 9: Power consumption of traditional and two-speed systems for the Clarknet trace. Peaks of 90% utilization.

filing the IBM Deskstar 120GXP disk we found that it consumes 7.5, 4.5, and 1 Watts in active, idle, and standby states, respectively. These consumptions are roughly a factor of two lower than that of our high-performance SCSI disk. The performance difference between these disks is also a factor of two. Given these parameters, the desktop disk has the potential to conserve more energy than the laptop disk for intermediate disk loads, i.e. loads that are higher than the laptop disk can handle, but not as high that the high-performance disk would be absolutely necessary. Unfortunately, neither of our traces exhibits long durations of this type of disk load, so the desktop disk always conserves less energy than its laptop counterpart. For example, for the Web server running with load peaks of 50%, the Combined approach saves 26% disk energy when our desktop disk is used as the lower power disk.

In summary, we find that the Combined approach, although interesting, only works well for a range of parameters that we do not consider very realistic for network servers. The main reason for its inability to conserve energy is that realistic disk demands are almost always higher than desktop and laptop disks can efficiently deal with.

7.3 Evaluating Multi-speed

Web server. We start our evaluation of the Multi-speed system by considering the Web server. Figure 8 depicts the power consumption of our Web server disk for Clarknet when server load peaks reach 80% of the maximum throughput. The figure shows results for a traditional server with a high-performance disk and a server with our two-speed disk. In the Multi-speed experiments, we use the default switching threshold of 80% of the disk bandwidth at low speed. We also assume the default 5-second delay.

The results in the figure show that the server with a conventional high-performance disk consumes 14.8 KJ of disk energy on this workload. The Multi-speed results show that the two-speed disk switches to 15K rpm only three times during the whole experiment. After we adjust the energy statistics according to the default 68-J energy cost for each speed transition, we find that our two-speed disk consumes 11.6 KJ of disk energy, leading to a savings of 22%.

Figure 9 depicts the power consumption of our Web server disk, when its load peaks reach 90% of the maximum achievable throughput. All parameters are set to their default values. In this case, the disk switches to high speed five times during the experiment, resulting in a total disk energy savings of 16%.

These energy results are positive. However, we also need to make sure that server performance does not suffer due to speed transitions. Figure 10 shows the profile of the load offered to the server for peaks of 80% utilization, as well as the Multi-speed server throughput for the same workload. The figure assumes the default 5-second delay for transitions.

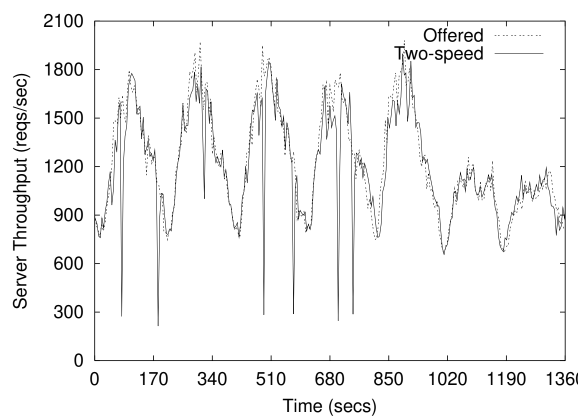

Figure 10: Server throughput for the Clarknet trace with the two-speed disk. Peaks of 80% utilization.

It is interesting to observe that the curves are very similar, though throughput does drop significantly during speed transitions. Nevertheless, performance degradation is minimal. The periods of low throughput reduce the overall number of requests served successfully by only 3%, with respect to the conventional system. This degradation would be even smaller, if we considered non-accelerated traces. In this case, speed transitions would occur at most 2 or 4 times in a day. In contrast, our accelerated Clarknet trace produces 6 speed transitions in just 22 minutes.

To understand the effect of the disk design parameters, we next evaluate Multi-speed with a higher threshold (90% of the disk bandwidth at low speed) for switching between speeds with all other parameters at their default values. For this threshold, the disk energy savings for our Web server are 22 and 21% for load peaks of 80 and 90% of the maximum achievable throughput, respectively. We do not consider switching thresholds lower than 80% of the disk bandwidth at low speed, because 80% is a low enough threshold that no performance degradation is observed.

We have also varied the energy overhead of each transition between 50 J and 86 J, for 80 and 90% load peaks, keeping all other parameters at their default values. The overall disk energy savings

Server	Load Peak	Switching Threshold	Energy Overhead	Energy Traditional	Energy Multi-speed	Savings
Web	80%	80%	68 J	14785 J	11575 J	22%
Web	90%	80%	68 J	13157 J	11088 J	16%
Web	80%	90%	68 J	14785 J	11467 J	22%
Web	90%	90%	68 J	13157 J	10388 J	21%
Web	80%	80%	50 J	14785 J	11467 J	22%
Web	90%	80%	50 J	13157 J	10908 J	17%
Web	80%	90%	50 J	14785 J	11359 J	23%
Web	90%	90%	50 J	13157 J	10280 J	22%
Web	80%	80%	86 J	14785 J	11683 J	21%
Web	90%	80%	86 J	13157 J	11268 J	14%
Web	80%	90%	86 J	14785 J	11575 J	22%
Web	90%	90%	86 J	13157 J	10496 J	20%
Proxy	80%	80%	68 J	62762 J	51866 J	17%
Proxy	90%	80%	68 J	56263 J	47598 J	15%
Proxy	80%	90%	68 J	62762 J	50080 J	20%
Proxy	90%	90%	68 J	56263 J	46272 J	18%
Proxy	80%	80%	50 J	62762 J	51758 J	18%
Proxy	90%	80%	50 J	56263 J	47454 J	16%
Proxy	80%	90%	50 J	62762 J	50008 J	20%
Proxy	90%	90%	50 J	56263 J	46164 J	18%
Proxy	80%	80%	86 J	62762 J	51974 J	17%
Proxy	90%	80%	86 J	56263 J	47742 J	15%
Proxy	80%	90%	86 J	62762 J	50152 J	20%
Proxy	90%	90%	86 J	56263 J	46380 J	18%

Table 2: Disk energy consumed by each server and combination of parameters.

Server	Switching Delay	Throughput Degradation
Web	5 secs	3%
Web	1 sec	1%
Web	10 secs	6%
Proxy	5 secs	1%
Proxy	1 sec	0%
Proxy	10 secs	2%

Table 3: Throughput degradation for each server and switching delay.

remain roughly unaltered throughout this spectrum, as a result of the small number of transitions in our experiments. We do not consider other values for the switching cost because they seem unrealistic. In any case, our results suggest that such energy costs would have to be very high, even higher than our pessimistic assumptions, for the Multi-speed savings to suffer significantly.

Finally, in order to understand the effect of the switching delay on the performance of our Web server, we varied this parameter from 1 to 10 seconds, for 80 and 90% load peaks, keeping all other parameters at their default values. For a delay of 1 second, the overall number of served requests is reduced by only 1% with respect to the system that uses a conventional high-performance disk. If the switching delay is as high as 10 seconds, the overall number of served requests can be reduced by up to 6%. Again, this number should be smaller when non-accelerated traces are considered. Increasing the switching delay further is clearly not reasonable, since it would become higher than the delay to spin the high-performance disk up.

The top part of tables 2 and 3 summarize the energy and performance results of our Web server experiments.

Proxy server. Our proxy server delivers similar results. Figure 11 shows the profile of the server and disk throughputs for the proxy trace for server load peaks of 80% of the server's maximum throughput. Note that the shape of the curves is different than that for Clarknet because we only replay one day (a Thursday) of the Hummingbird trace; running the whole week would have taken excessively long, as we cannot significantly accelerate this trace.

Figure 12 depicts the power consumption of our proxy disk for 80% load peaks. The figure shows results for a traditional server with a high-performance disk and a server system with our two-speed disk. All parameters are set to their default values. The figure shows that the conventional system consumes 62.8 KJ of disk energy on this workload. It also shows that the Multi-speed system only switches to 15K rpm three times. Adjusting the energy of the Multi-speed system, we find that it consumes 51.9 KJ for a disk energy savings of 17%.

Figure 13 depicts the power consumption of our proxy when the server load peaks reach 90% of the maximum achievable throughput. Again, all parameters are set to their default values. The figure shows that the two-speed disk switches to high speed four times during the whole experiment, resulting in a total disk energy savings of 15%.

In terms of server performance, figure 14 shows the profile of the offered load and the Multi-speed server throughput for the proxy trace at 80% load peaks. Due to the 5-second switching delay, the

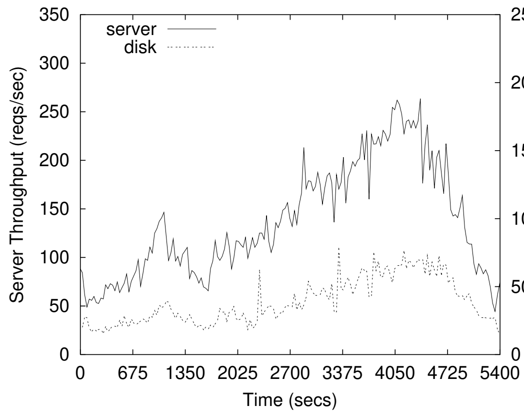

Figure 11: Server and disk throughputs for the proxy trace. Peaks of 80% utilization.

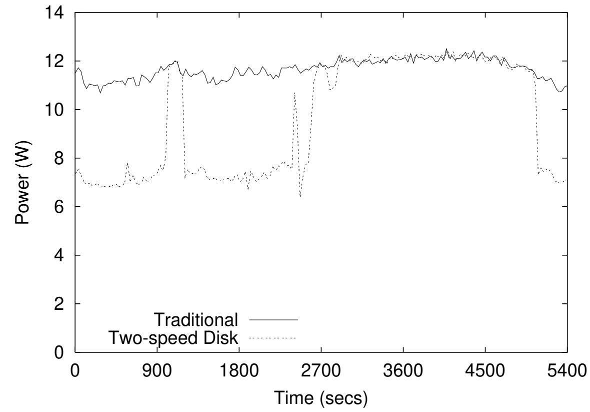

Figure 12: Power consumption of traditional and two-speed systems for the proxy trace. Peaks of 80% utilization.

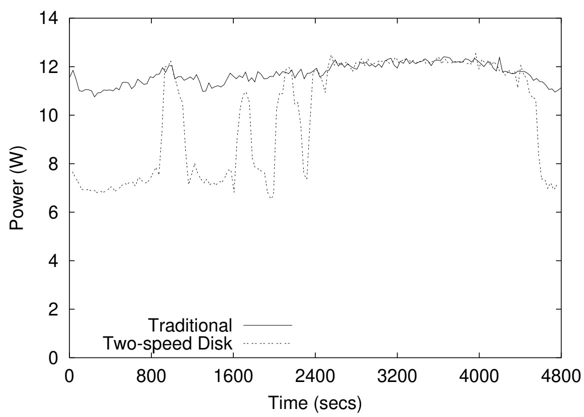

Figure 13: Power consumption of traditional and two-speed systems for the proxy trace. Peaks of 90% utilization.

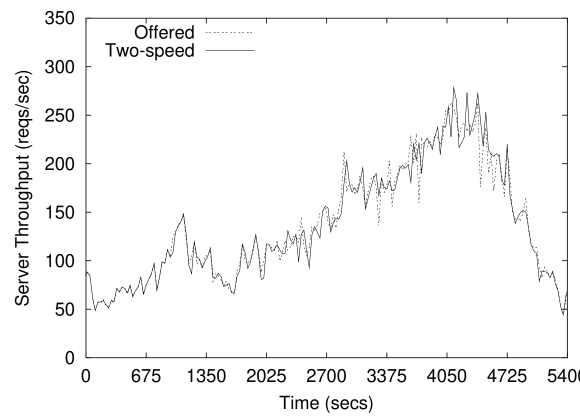

Figure 14: Server throughput for the proxy trace with the two-speed disk. Peaks of 80% utilization.

overall number of served requests is reduced by less than 1% with respect to a system that uses a conventional high performance disk. Note that the periods of low throughput are not visible in this figure because of the coarse grain resolution used for the x-axis.

The effect of the disk design parameters on the proxy results was similar to those of the Web server. When we change the switching threshold to 90% of the disk bandwidth at low speed, the disk energy savings of the Multi-speed system become 20 and 18% for server load peaks of 80 and 90% of the maximum achievable throughput, respectively. The effect of varying the energy cost of transitions from 50 to 86 J is again very small. Finally, the transition delay only affects performance for delays as high as 10 seconds. Still, the throughput degradation in this case is only 2%.

The bottom part of tables 2 and 3 summarize the energy and performance results of our proxy server experiments.

Discussion. These results suggest that disk energy savings around 20% are feasible, even for servers with high load peaks and under pessimistic speed transition overheads. Lower peak loads can improve these gains, reaching 30% for Web servers and 24% for proxy servers when peaks of at most 70% of the maximum achievable throughput are expected. Even better results can probably be achieved, but only for servers that are excessively over-provisioned or that exhibit even more significant variations in offered load. We did not consider these scenarios as they do not seem to represent the common cases out in the field.

In terms of the disk design parameters, the switching threshold is the most important one. Changing this threshold from 80 to 90% of the disk bandwidth at low speed allows for increases in disk energy savings of up to 6% (for our Web server). The effect of the energy cost of speed transitions is very small, given that changes in load trends occur relatively infrequently, even for accelerated workloads. Finally, the delay caused by speed transitions has a limited effect on overall server throughout. We had to set the delay at an almost unreasonable 10 seconds for it to have a non-trivial impact on throughput (for our Web server).

We also considered more sophisticated schemes for the switching threshold. In particular, we considered using two switching thresholds to guarantee stability, i.e. to avoid speed changes that are triggered by slight variations in disk load. We could, for example, switch to low speed when the disk load is below 80% of the disk bandwidth at low speed and switch back to high speed when the load reaches 90% of that bandwidth. We did not implement this scheme for three reasons: (1) we already apply smoothing and filtering of disk load information; (2) our accelerated traces do not cause instability; and (3) given all our other results, it is clear that the energy gains of this more sophisticated scheme would still be around 20%. In any case, a two-threshold scheme would be straightforward to implement.

In summary, our base results and parameter space study suggest that the two-speed disk performs well in a wide range of scenarios.

8. RELATED WORK

We are only aware of four other works on disk energy conservation for servers [12, 11, 7, 27]. Using simulation, Gurumurthi *et al.* [12] considered the effect of different RAID parameters on the performance and energy consumption of database servers running transaction processing workloads. They also observed that it is not possible to exploit idleness in this context. Recently, Gurumurthi *et al.* [11] studied multi-speed disks. Using simulation and synthetic workloads, they showed that multi-speed disks can provide energy savings of up to 60%. Colarelli and Grunwald [7] simulated disk array energy optimizations for scientific workloads. The basic idea is to use "cache disks" to cache active files/blocks, allowing other disks to be spun down. We have studied a different disk array energy optimization for file servers: a popularity-based distribution of file data across the disk array [27]. The idea is to concentrate the most popular data in a subset of the disks, so that the other disks can become idle longer and more often. Using a validated simulator, we compared our strategy with that of Colarelli and Grunwald for a wide range of workload and server parameters.

This paper differs from these studies in that we consider network server systems using real software, hardware (to the extent possible), and power, energy, and performance measurements. In fact, we have previously illustrated the importance of performing real power and energy measurements with disks [14], rather than relying on their often-inaccurate data sheets. Furthermore, our work on multi-speed disks differs from [11] mainly in that: (1) our policy for switching speeds is based on disk throughput, since throughput is usually a more important metric than response time for network servers; (2) we considered busier systems, with average disk request inter-arrival times between 1 and 7 milliseconds per disk, and thus found lower energy gains; and (3) we only considered two-speed disks with simple control, since such disks can probably be manufactured right now without incurring high additional costs.

In addition, the Combined approach has not been addressed before. Perhaps the closest related work to Combined is that of Olsen and Morrow [25], who studied the benefits of offloading simple repetitive tasks from a general-purpose, high-performance microprocessor to a lower performance, lower power microprocessor. Although their idea has a similar flavor, their scheduling of work for the processors was based on pre-determined characteristics of the processes, rather than on system load. As a result, certain issues, such as dynamic load balancing, data sharing and coherence, did not have to be dealt with in practice. In fact, their study did not involve a real implementation, focusing solely on modeling.

9. CONCLUSIONS

In this paper, we showed that two-speed disks can achieve substantial energy savings without performance degradation in network servers. Our results suggest that this technique should be carefully considered by disk manufacturers, as they may want to produce these disks for the high-end server market. The key question is whether manufacturers can produce multi-speed disks with acceptable reliability in practice. The other techniques we studied cannot provide any disk energy savings under realistic network server workload assumptions.

10. ACKNOWLEDGEMENTS

We would like to thank Prof. Arthur Goldberg from New York University for giving us the Hummingbird trace. We are also grateful to Prof. Uli Kremer for lending us the power measurement infrastructure of the Energy Efficiency and Low-Power (EEL) lab at Rutgers.

11. REFERENCES

[1] M. Arlitt and T. Jin. Workload Characterization of the 1998 World Cup Web Site. Technical Report HPL-1999-35R1, HP Laboratories, September 1999.

[2] M. Arlitt and C. Williamson. Web Server Workload Characterization: The Search for Invariants. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems*, May 1996.

[3] 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.

[4] E. V. Carrera and R. Bianchini. Efficiency vs. Portability in Cluster-Based Network Servers. In *Proceedings of the 8th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, June 2001.

[5] E. V. Carrera and R. Bianchini. Improving Disk Throughput in Data-Intensive Servers. Technical Report DCS-TR-500, Department of Computer Science, Rutgers University, September 2002, Revised May 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 SC'2002*, November 2002.

[8] Fred Douglis and P. Krishnan. Adaptive Disk Spin-Down Policies for Mobile Computers. *Computing Systems*, 8(4):381–413, 1995.

[9] 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.

[10] 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.

[11] 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.

[12] S. Gurumurthi, 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.

[13] T. Halfhill. Transmeta Breaks the x86 Low-Power Barrier. In *Microprocessor Report*, February 2000.

[14] T. Heath, E. Pinheiro, and R. Bianchini. Application-Supported Device Management for Energy and Performance. In *Proceedings of the Workshop on Power-Aware Computer Systems*, February 2002.

[15] T. Heath, E. Pinheiro, J. Hom, U. Kremer, and R. Bianchini. Application Transformations for Energy and Performance-Aware Device Management. In *Proceedings of the 11th International Conference on Parallel Architectures and Compilation Techniques*, September 2002. Best student paper award.

[16] David P. Helmbold, Darrell D. E. Long, and Bruce Sherrod. A Dynamic Disk Spin-Down Technique for Mobile Computing. In *Proceedings of the 2nd International Conference on Mobile Computing (MOBICOM96)*, pages 130–142, 1996.

[17] IBM. *IBM Ultrastar 36Z15 Hard Disk Drive*, July 2001. http://www.storage.ibm.com/.

[18] IBM. *IBM Ultrastar 73LZX Hard Disk Drive*, July 2001. http://www.storage.ibm.com/.

[19] IBM. *IBM Travelstar 40GNX 2.5-inch Hard Disk Drive*, July 2002. http://www.storage.ibm.com/.

[20] A. Iyengar, J. Challenger, D. Dias, and P. Dantzig. High-Performance Web Site Design Techniques. *IEEE Internet Computing*, 4(2):17–26, March 2000.

[21] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson. A Quantitative Analysis of Disk Drive Power Management in Portable Computers. In *Proceedings of the 1994 Winter USENIX Conference*, pages 279–291, 1994.

[22] Yung-Hsiang Lu, Eui-Young Chung, Tajana Simunic, Luca Benini, and Giovanni De Micheli. Quantitative Comparison of Power Management Algorithms. In *Proceedings of the Design Automation and Test Europe*, March 2000.

[23] Yung-Hsiang Lu, Tajana Simunic, and Giovanni De Micheli. Software Controlled Power Management. In *Proceedings of the IEEE Hardware/Software Co-Design Workshop*, May 1999.

[24] J. Mitchell-Jackson. Energy Needs in an Internet Economy: A Closer Look at Data Centers. Master of Science Thesis from the Energy and Resources Group, University of California at Berkeley. July, 2001.

[25] C. M. Olsen and L. A. Morrow. Multi-Processor Computer System Having Low Power Consumption. In *Proceedings of the 2nd Workshop on Power Aware Computing Systems*, pages 33–41, February 2002.

[26] A. Papathanasiou and M. Scott. Increasing Disk Burstiness for Energy Efficiency. Technical Report 792, Department of Computer Science, University of Rochester, November 2002.

[27] E. Pinheiro and R. Bianchini. Energy Conservation Techniques for Disk Array-Based Servers. Technical Report DCS-TR-525, Department of Computer Science, Rutgers University, May 2003.

[28] 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, 2003. To appear.

[29] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Proceedings of the International Workshop on Compilers and Operating Systems for Low Power*, September 2001.

[30] The Industry Standard. Down on the Server Farm. February 2001. http://www.thestandard.com/article/display/0,1151,22095,00.html.

[31] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for Reduced CPU Energy. In *Proceedings of the 1st Symposium on Operating System Design and Implementation*, 1994.