Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management

Qingbo Zhu, Francis M. David, Christo F. Devaraj, Zhenmin Li, Yuanyuan Zhou and Pei Cao*
Department of Computer Science
University of Illinois at Urbana Champaign, Urbana, IL 61801
{qzhu1, fdavid, devaraj, zli4, yyzhou}@uiuc.edu
*Cisco Systems Inc., San Jose, CA 95134
cao@cisco.com

Abstract

Reducing energy consumption is an important issue for data centers. Among the various components of a data center, storage is one of the biggest consumers of energy. Previous studies have shown that the average idle period for a server disk in a data center is very small compared to the time taken to spin down and spin up. This significantly limits the effectiveness of disk power management schemes.

This paper proposes several power-aware storage cache management algorithms that provide more opportunities for the underlying disk power management schemes to save energy. More specifically, we present an off-line power-aware greedy algorithm that is more energy-efficient than Belady's off-line algorithm (which minimizes cache misses only). We also propose an online power-aware cache replacement algorithm. Our trace-driven simulations show that, compared to LRU, our algorithm saves 16% more disk energy and provides 50% better average response time for OLTP I/O workloads. We have also investigated the effects of four storage cache write policies on disk energy consumption.

1 Introduction

Trends in Internet infrastructure are driving a shift toward service-based computing. Data centers will play a key role in this new architecture since they are commonly used to provide a wide variety of services including web hosting, application services, outsourced storage, electronic markets, and other network services. A data center usually consists of thousands of components including processors, memory chips, disks and network hardware. Storage is one of the biggest components in data centers. Storage demand is also growing by 60% annually [33]. By 2008, data centers will manage 10 times as much data as they do today.

The steady growth of data centers introduces a significant problem: *energy consumption*. Data centers typically have very high power requirements. According to EUN (Energy User News) [32], today's data centers have power requirements that range from 75 W/ft^2 for small- to medium-sized enterprises to 150-200 W/ft^2 for typical service providers. In the future, this is predicted to increase to 200-300 W/ft^2. These increasing power requirements are driving energy costs up as much as 25% annually and making it a growing consideration in the TCO (total cost of ownership) for a data center [33]. High energy consumption also prevents easy expansion and has negative environmental implications.

Among various components of a data center, storage is one of the biggest consumers of energy. A recent industry report [1] shows that storage devices account for almost 27% of the total energy consumed by a data center. This problem is exacerbated by the availability of faster disks with higher power needs as well as the increasing shift from tape backups to disk backups for better performance.

Even though a lot of research has been carried out in disk power management, most of it has focused on a single disk for mobile devices. So far only a few studies [8, 19, 18, 5] have addressed the energy problem for storage systems at the scale of data centers. These studies have shown that the average idle time between requests for server disks is small compared to the time taken to spin down and spin up. This significantly limits the energy that can be saved since the disks have to be spinning and ready almost all the time. To address this problem, Gurumurthi et al. have proposed the use of multi-speed disks with smaller spin-up and spin-down times to reduce disk energy consumption for server workloads [18]. They have proposed a power model for this scheme and have shown promising results using synthetic workloads. In another study, Carrera and Bianchini at Rutgers have studied using combinations of laptop disks and server disks and also suggested using multiple rotational speed disks to save energy [5].

Not all accesses to a storage system go to disks. A typical architecture for a modern storage system is shown in Figure 1. Many modern storage systems use a large storage cache to reduce the number of disk accesses and improve

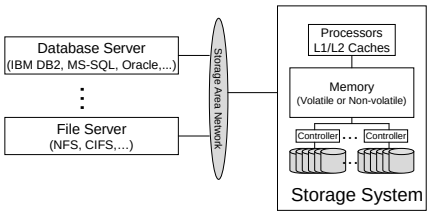


Figure 1: Modern storage architecture

performance. For example, the EMC Symmetrix storage system with a capacity of 10-50 TBytes can be configured with up-to 128 GB of non-volatile memory as the storage cache [13]. The IBM ESS system can also have up-to 64 GB of storage cache [23]. Different from those small (usually 1-4 MB) buffers on a SCSI disk, which are mainly used for read-ahead purposes, these large caches are used to cache blocks for future accesses. Therefore, the cache replacement algorithm plays a very important role in a storage system [42, 37, 31, 7].

The storage cache management policy influences the sequence of requests that access disks. Different cache management policies may generate different disk request sequences, which directly affects disk energy consumption. In other words, by changing the cache management scheme, it is possible to change the average idle time between disk requests, thus providing more opportunities for the disk power management scheme to save energy. For example, if the cache replacement algorithm can selectively keep some blocks from a particular disk in the cache (without significantly increasing the number of misses to other disks), that disk can stay in a low power mode longer. Since storage caches are very critical to storage system performance, our study assumes that storage caches are active all the time.

Besides disk energy consumption, I/O response time is another concern. If underlying disks use power management schemes, some requests can be significantly delayed because it takes a long time (a few seconds) for a disk to spin up from a low power mode to the active mode. Consequently, if a cache replacement algorithm is only designed to minimize the number of cache misses and ignores the underlying disk power management schemes, it can result in very high I/O response time. Therefore, it is important to design cache management schemes that are power-aware (aware of the underlying disk energy consumption and power management schemes).

This paper studies the effects of storage cache management schemes on disk energy consumption and proposes *power-aware* cache management schemes. We look into both cache management for read accesses and for write accesses. Specifically:

- For read accesses, we present an off-line power-aware greedy algorithm. Our trace-driven simulations show that the greedy algorithm is more energy-efficient than Belady's off-line algorithm (which minimizes cache misses only) while still providing acceptable average response time.
- Based on the insights from our analysis of offline algorithms, we propose an online power-aware LRU cache replacement algorithm called PA-LRU. Simulation results show that PA-LRU can reduce disk energy consumption by 16% compared to LRU and also provide 50% better average response time for on-line transaction processing (OLTP) workloads.
- For write accesses, we study the effects of storage cache write policies on disk energy consumption. Our results show that write-back can save up-to 20% more energy compared to write-through. Write-back with eager updates further reduces disk energy consumption of write-back up-to 45%. We also propose a policy called write-through with deferred updates that reduce energy consumption up-to 55% while still providing persistency semantics comparable to write-through.

The paper is organized as follows: The next section briefly describes the background. Section 3 discusses the off-line power-aware greedy algorithm. Section 4 presents the on-line power-aware algorithm, followed by simulation results for power-aware replacement algorithms in Section 5. The sections above deal with read accesses. Section 6 discusses the effects of four storage cache write policies on energy consumption. Section 7 summarizes related work. Finally, Section 8 concludes the paper.

2 Background

2.1 Disk Power Model

To reduce energy consumption, modern disks use multiple power modes that include active, idle, standby and other intermediate modes. In active mode, the platters are spinning and the head is seeking or actively reading or writing. In idle mode, a disk is spinning at its full speed but no disk activity is taking place. Therefore, staying in the idle mode when there is no disk request provides the best-possible access time since the disk can immediately service requests, but it consumes the most energy. To simplify discussion, we do not differentiate between active mode and idle mode since in both modes the disk is operating at full power. In standby mode, the disk consumes much less energy, but in order to service a request, the disk has to incur significant energy and time overheads to spin up to active mode.

Recently, Gurumuthi et al. have proposed multi-speed disks to increase the amount of energy saved with data center workloads [18]. Lower rotational speed modes consume less energy compared to higher speed modes, and the energy and time costs to shift between different rotational speeds

are relatively small compared to the costs for shifting from standby to active. Such multi-speed disks are still only a design on paper and there are no real products yet. We however simulate and use multi-speed disks in our experiments because of their potential to save more energy.

A multi-speed disk can be designed to either serve requests at all rotational speeds or serve requests only after a transition to the highest speed. Carrera and Bianchini [5] use the first option. We choose the second option since it is a direct extension to the simple 2-mode power model. We use the specifications for the IBM Ultrastar 36Z15 disk as listed in Table 1, with extensions to support 4 extra intermediate power modes (lower-speed modes).

2.2 Disk Power Management

The goal of disk power management is to try and save energy by switching disks to lower power modes whenever possible without adversely affecting performance [8, 5, 19, 18]. If the entire disk request sequence is known in advance, the power management scheme can make perfect decisions. This is called Oracle disk power management (Oracle DPM) [29] which gives us an upper-bound on the energy that can be saved for a given request sequence, assuming that a request cannot be prefetched or delayed. As soon as a request completes, Oracle DPM decides if the disk should stay in idle mode, or go down to a lower power mode. The idle period needed to justify the cost of spinning the disk up and down is called the *break-even time*. Oracle DPM examines the interval length t between the current request and the next request. If t is greater than the break-even time, it is more beneficial to spin down the disk to standby mode. Therefore, the disk is spun down immediately after the current request is serviced and spun up to active mode just in time for the next request. Otherwise it is better to stay in the idle mode after the current request completes.

This scheme can easily be extended to disk models with multiple power modes. Let us assume that $P_i, (0 \leq i \leq m)$ is the power consumed in mode i and that P_i is greater than P_j for all $i < j$. Once an idle interval starts, Oracle DPM has to switch the disk to one of the m modes that minimizes energy consumption. The disk must also be back in mode 0 when the next request arrives.

To get the minimum energy consumption, we plot lines $E_i(t) = P_i(t - T_i) + C_i$ as in Figure 2 for each power mode i, where P_i is the power dissipation in mode i, and T_i and C_i are the time and energy required to spin-down and spin-up from power mode i to 0 (T_0 and C_0 is 0). $E_i(t)$ is the energy consumed if the disk spends the entire interval of length t in mode i. Let us call the lower envelope of all of these lines $LE(t) = min_i\{P_i(t - T_i) + C_i\}$. This gives us the minimum energy consumption possible for an interval of length T. If the next request is time T away from the current request, Oracle DPM can use $LE(t)$ to determine which

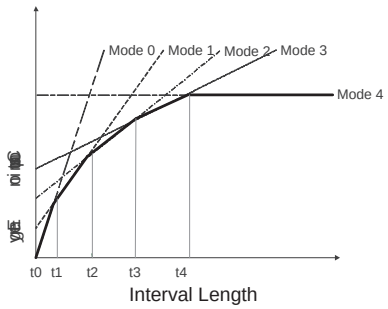


Figure 2: Energy consumption for each mode in a 5-mode disk power model and the lower envelope $LE(t)$ function for minimum energy consumption

power mode to switch the disk to. If the disk is switched to the power mode j where $LE(T) = E_j(T)$, the energy consumption during this interval is minimized.

Practical disk power management (Practical DPM) schemes use thresholds to determine when to spin down disks [9, 11, 28, 16, 10, 26, 21, 40, 5, 19, 18]. These threshold-based schemes are on-line. In such schemes, after the disk remains idle at a power mode for a certain threshold time, it is switched into the next lower power mode. Irani et al. have shown if the threshold values are determined by the intersection points of the lines in Figure 2, the power management scheme is 2-competitive to the Oracle scheme in terms of energy consumption [24]. This scheme transitions the disk from mode i to mode $i + 1$ after time t_{i+1}, where t_{i+1} is the time corresponding to the intersection point of lines $E_i(t)$ and $E_{i+1}(t)$ as shown in Figure 2. We use thresholds obtained by this method in our study.

3 Power-Aware Off-line Algorithms

Off-line algorithms have knowledge about the future. Such algorithms are usually studied because they provide upper and lower bounds for all on-line algorithms. For example, Belady's off-line algorithm [2, 30], which replaces the block with the longest future reference distance, is used to derive a lower bound on the cache miss rate. Since the study of off-line algorithms is important, we first investigate such algorithms for power aware cache management.

3.1 Energy-optimal Problem

The goal of a power-aware cache replacement algorithm is to take a given request sequence as input and generate a miss sequence for which the disks consume the least energy. If we use S to denote an I/O request sequence from storage applications, a replacement algorithm A is a function that maps S and a cache with k blocks into a miss request sequence S', i.e. $A : (S, k) \rightarrow S'$ or $A(S, k) = S'$. Given a disk power management scheme P and a disk request se-

quence X, let $P(X)$ be the total energy consumed by the disks. Therefore, we have the following formalization and definition of an energy-optimal replacement algorithm:

Remark: Given an I/O request sequence S, a cache replacement algorithm A, a cache with k blocks, and a disk power management scheme P, the total disk energy consumption is $P(A(S,k))$.

Definition: A storage cache replacement algorithm A is energy-optimal iff for any other algorithm B, $P(A(S,k)) \leq P(B(S,k))$ for any I/O request sequence S and any storage cache size k.

The number of misses resulting from a storage cache replacement algorithm obviously affects disk energy consumption. One would expect that if there are few cache misses, the disks would consume little energy. However, the energy consumption is also affected by the arrival patterns of the cache misses. If misses are clustered together leaving long idle periods, it would allow disks to stay in the low power mode for longer periods of time. On the other hand, if the misses arrive uniformly spaced, most idle periods may be too small for a disk to save energy by going to the low power mode, or the disk may spin up and down frequently, wasting a lot of energy in transitions. Furthermore, when there are multiple disks, it is better if misses are directed to a cluster of disks rather than uniformly distributed over all the disks. This allows the other disks to be in standby mode more often and thus save energy.

There are two reasons why Belady's algorithm is sub-optimal in terms of disk energy consumption. Firstly, it only minimizes the number of misses and pays no attention to the arrival patterns of the cache misses or how they are clustered. In other words, it ignores all information about time. Below we give an example of this case. Secondly, it does not take into account the number and characteristics of disks in a multiple disk scenario.

Figure 3 gives an example to show why Belady's cache replacement algorithm is not energy-optimal. In this example, the storage cache has only four entries and the power model is the simple 2-mode model. For simplicity, we assume that the disk can spin up and spin down instantaneously. We also assume that disk spins down after 10 units of idle time. This disk power management scheme is a threshold-based scheme (described in Section 2.2). The area of the shaded region in the figure represents the energy consumed. In this example, using Belady's algorithm results in more disk energy consumption than the alternative algorithm, even though the alternative algorithm has 2 more misses than Belady's algorithm.

We have developed an energy-optimal cache replacement algorithm which runs in polynomial time by using dynamic programming. Due to space constraints, we do not explain it in detail here. Please refer to our technical report [43] for more details.

Time	Req	Belady		Alternative	
		cache	Hit?	cache	Hit?
0	A	[A,-,-,-]	Miss	[A,-,-,-]	Miss
1	B	[A,B,-,-]	Miss	[A,B,-,-]	Miss
2	C	[A,B,C,-]	Miss	[A,B,C,-]	Miss
3	D	[A,B,C,D]	Miss	[A,B,C,D]	Miss
4	E	[E,B,C,D]	Miss	[A,E,C,D]	Miss
5	B	[E,B,C,D]	Hit	[A,B,C,D]	Miss
6	E	[E,B,C,D]	Hit	[A,E,C,D]	Miss
7	C	[E,B,C,D]	Hit	[A,E,C,D]	Hit
8	D	[E,B,C,D]	Hit	[A,E,C,D]	Hit
...					
16	A	[A,B,C,D]	Miss	[A,E,C,D]	Hit

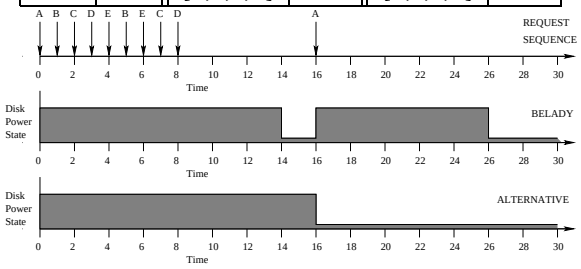


Figure 3: An example showing that Belady's algorithm is not energy-optimal.

3.2 Off-line Power-Aware Greedy Algorithm

Since the energy-optimal algorithm is too complex to implement and evaluate, we propose a heuristic off-line power-aware greedy (OPG) algorithm that consumes less energy than Belady's algorithm for real-system workloads.

The main goal of the OPG algorithm is to minimize energy consumption by taking advantage of information about future bound-to-happen misses based on cache content at some point in time. We will call the bound-to-happen misses *deterministic misses* because they will happen no matter what the replacement algorithm does after that point.

If we know that there is a deterministic miss at a future time t, the disk from which this missed block will be accessed has to be active at t in order to service this request. For convenience of description, for any access a, we call the closest deterministic miss to the same disk but occurring before a as a's *leader*. Similarly, we call the closest deterministic miss to the same disk as a but occurring after a as a's *follower*.

If the disk power management uses the Oracle scheme, the energy consumption for an idle period of length t is $LE(t) = min\{E_i(t)\}$ as described in Section 2.2 (see Figure 2). If the disks use the Practical DPM, the disk energy consumption $OL(t)$ during an idle period of length t can be calculated as follows: $\sum_{i=0}^{l-1}(P_i * (t_{i+1} - t_i)) + P_l * \delta + C_l$, where the disk goes to power mode i at time t_i, $t_l (< t)$ is the cross point closest to t and δ is the distance between t and t_l, i.e. $t = t_l + \delta, 0 \leq \delta < t_{l+1} - t_l$ (see Figure 2).

The cache replacement algorithm uses energy penalties to choose from all the resident blocks $B_1, \cdots, B_i, \cdots, B_k$

when it needs to evict a block, where k is the number of blocks in the cache. For any i, let b_i represent the next access to B_i. Suppose b_i is, respectively, L_i and F_i time apart from its leader and its follower. If the algorithm evicts B_i, it will cause a miss for b_i, whose energy penalty is calculated as follows:

$$\begin{cases} LE(L_i) + LE(F_i) - LE(L_i + F_i) & \text{if Oracle DPM} \\ OL(L_i) + OL(F_i) - OL(L_i + F_i) & \text{if Practical DPM} \end{cases}$$

Intuitively, with the Oracle DPM scheme, the energy cost for the idle period between the *leader* and *follower* is $LE(L_i + F_i)$ if b_i is not a miss (therefore, there is no misses to this disk between *leader* and *follower* based on the definitions of *leader* and *follower*) . The access b_i cuts the original idle period into two chunks, whose aggregate energy cost is $LE(L_i) + LE(F_i)$. Thus, the energy penalty for evicting block B_i is the difference between the two energy costs $LE(L_i) + LE(F_i) - LE(L_i + F_i)$. The energy penalty with the Practical DPM can be calculated in a similar way, replacing $LE()$ by $OL()$ in the formula.

Once the algorithm calculates the energy penalty for evicting every resident block, it evicts the block with the minimum energy penalty. If multiple blocks have the same energy penalty, it evicts the one with the largest forward distance, i.e., whose next access is the furthest in the future.

Initially, the set of deterministic misses, S, only includes all the cold misses. After each replacement, the algorithm updates the set S. Suppose the currently accessed (missed) block is B_1 and the evicted block is B_2. The algorithm deletes B_1 from the set S and adds the first future reference to B_2 into S. Then the algorithm moves on to the next request until all requests are processed. The time complexity of the OPG algorithm for a list of n requests is at most $O(n^2)$ since the newly inserted deterministic miss can become the leader or follower of many block accesses and the energy penalties of those blocks should thus be updated.

This algorithm is a heuristic algorithm because it looks at only the current set of deterministic misses when calculating the energy penalty for evicting a block. Therefore, it may not make the best decision at a replacement. As we discussed in Section 3.1, each cache miss leads to a disk access, which costs additional energy. Hence, higher miss ratios would increase the energy consumption. We use a simple mechanism to consider both miss ratio and energy penalty for a miss. The main idea is not to differentiate among blocks whose energy penalties are smaller than a threshold η. Any energy penalty that is smaller than η is rounded up to η. Obviously, when η is large enough, it is Belady's algorithm; when $\eta = 0$, it is the pure OPG algorithm. This mechanism thus subsumes Belady's algorithm at one extreme, and the pure OPG algorithm at the other.

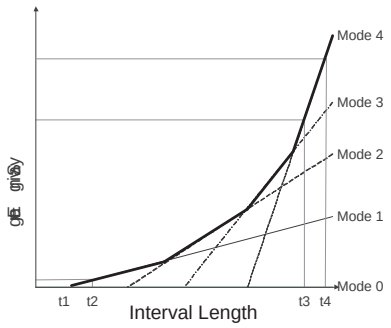


Figure 4: Energy savings (not Energy Consumption) over mode 0 for each mode in a 5-mode disk power model as a function of interval length

4 Power-Aware On-line Algorithm

Section 3.2 shows that an algorithm that minimizes the energy penalty of each eviction could save energy. In practice, we do not have future knowledge and thus cannot use OPG. However, it does give us insights on how to design a power aware online algorithm that saves energy. Such an algorithm should avoid evicting blocks with larger energy penalties.

We first investigate how the length of the intervals affects energy savings. Figure 4 shows the energy savings that can be obtained by switching to lower power modes given the interval length. Similar to Figure 2, we plot lines $ES_i(t) = E_0(t) - E_i(t)$ for each power mode i, where ES_i is the energy saved by going into mode i and E_i is defined in Section 2. Note ES_0 is 0. Let us define the upper envelope of all of these lines $UE(t) = max_i\{ES_i\}$, which gives us the maximum energy saved for an interval of length t.

The super-linear property of $UE(t)$ indicates that even small increases in the interval length of inactive disks can result in significant energy savings. The cache replacement algorithm can reshape the access pattern for each disk. By keeping more blocks from inactive disks in the cache, we can make the average interval length for these disks larger. Then these disks could stay in the low power modes longer. Although the average interval lengths for other active disks may be decreased due to an increased number of misses, the energy penalty we pay for these other disks is much smaller than the energy savings we gain from the inactive disks. As shown in Figure 4, assuming there are two disks, although the average idle period of disk 0 is reduced from t_2 to t_1, we increase the average interval length of disk 1 from t_3 to t_4. Therefore, overall energy saving is achieved.

However, average interval length is not the only factor that affects the amount of energy that can be saved (1) **the percentage of capacity misses** (misses caused by previous evictions) should be reasonably large since a cache replacement algorithm cannot avoid any cold misses(misses due to first-time accesses). If most of the accesses to a disk are cold misses, we cannot do much to avoid expensive disk spin-

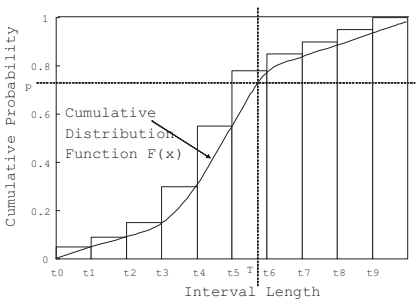


Figure 5: The histogram approximates the cumulative distribution function of interval lengths.

ups or make interval lengths longer. (2) **the distribution of accesses** also affects the opportunities to save energy. For example, for the same average interval length t_2 in Figure 4, disks with larger deviation have more opportunities to save energy than disks with strictly periodic arrivals.

To keep track of the number of cold misses, we use a *Bloom Filter* [3, 14] to identify cold misses. The idea is to allocate a vector v of m bits, all set to 0 initially, and then choose k independent hash functions, $h_1, h_2, \ldots, h_k$, each with range $\{1, \ldots, m\}$. Given an access for block a we check the bits at positions $h_1(a), h_2(a), \ldots, h_k(a)$. If any of them is 0, then a is definitely a cold miss. In this case, the bits at positions $h_1(a), h_2(a), \ldots, h_k(a)$ in v are set to 1. Otherwise, we conjecture that a is already in the set, which means it is not a cold miss, although there is a certain probability that we are wrong due to hash conflicts. For an estimated $1.6M$ blocks, with v set to $2M$ bytes and $k = 7$, this probability is only 0.0082.

To estimate the distribution of accesses for each disk, instead of using mean and standard deviation, we employ a simple but effective epoch-based *histogram* technique [39]. In each epoch, we keep track of the interval length between two consecutive accesses for each disk. We obtain a histogram as shown in Figure 5. Let n_i be the number of intervals of length between $[t_i, t_i + 1)$ and let n be the total number of intervals. The height of each bin in Figure 5 is $\sum_{j=0}^{i} \frac{n_j}{n}$, which approximates the cumulative probability of the interval length being less than t_{i+1}. All the bins together form a histogram, which approximates the cumulative distribution function of interval length for a disk, i.e., $F(x) = P[X \leq x]$, where X is a random variable that represents the interval length for a disk.

We design a power-aware on-line cache management scheme called PA. The main idea is to dynamically keep track of workload characteristics for each disk, including the percentage of cold misses and the cumulative distribution of interval lengths. Based on these characteristics, PA classifies all disks into two categories, *regular* and *priority*. Disks that exhibit (1) small percentage of cold misses, (2) large interval lengths with high probability belong to "priority" class, and others belong to "regular" class. To adapt to workload changes, the classification is *epoch-based*, and is thus adjusted periodically based on the latest workload.

PA can be combined with most existing storage cache replacement algorithms to make them "power aware". This includes several recently proposed algorithms such as ARC [31], LIRS [25],DEMOTE [37], and MQ [42]. In this paper, we use the common LRU algorithm as an example and present a Power-Aware LRU algorithm (PA-LRU).

PA-LRU maintains two LRU stacks, LRU0 which keeps blocks that belong to disks in the "regular" class and LRU1 which keeps blocks that belong to disks in the "priority" class. When choosing a block to evict, PA-LRU always evicts the bottom block of LRU0 if it is not empty. If LRU0 is empty, PA-LRU evicts the bottom block from LRU1.

PA-LRU uses the request sequence's characteristics during the previous epoch for each disk. (1) If the percentage of cold misses is larger than a threshold α, the blocks from this disk go to LRU0 during the current epoch. (2) As shown in Figure 5, given a cumulative probability p, we can easily calculate the corresponding T based on the cumulative distribution function, that is, $P(X \leq T) = p$. If T is less than a threshold β, the blocks from this disk go to LRU0 as well. Otherwise, blocks go to LRU1. α, β, p and the length of the epoch can be specified by the target system.

5 Evaluation of Power-Aware Cache Replacement Algorithms

5.1 Evaluation Methodology

We simulate a complete storage system to evaluate our power-aware cache management schemes. We have enhanced the widely used DiskSim simulator [15] and augmented it with a disk power model. The power model we use is similar to that used by Gurumurthi et al. [18] for multi-speed disks. We have also developed a storage cache simulator, CacheSim and we use it together with DiskSim to simulate a complete storage system. CacheSim implements several cache management policies. Accesses to the

IBM Ultrastar 36Z15	
Standard Interface	SCSI
Individual Disk Capacity	18.4 GB
Maximum Disk Rotation Speed	15000 RPM
Minimum Disk Rotation Speed	3000 RPM
RPM Step-Size	3000 RPM
Active Power(Read/Write)	13.5 W
Seek Power	13.5 W
Idle Power@15000RPM	10.2 W
Standby Power	2.5 W
Spinup Time(Standby to Active)	10.9 secs
Spinup Energy(Standby to Active)	135 J
Spindown Time(Active to Standby)	1.5 secs
Spindown Energy(Active to Standby)	13 J

Table 1: Simulation Parameters

simulated disks first go through a simulated storage cache. The simulator reports the energy consumed by each disk in every power mode and the energy consumed in servicing requests (energy to perform seek, rotation, and transfer). Therefore, if a power-aware replacement algorithm introduces extra misses, the disk energy consumed in servicing those extra misses is also included in the numbers reported.

The specifications for the disk used in our study are similar to that of the IBM Ultrastar 36Z15. The parameters are taken from the disk's data sheet [22, 5]. Some of these parameters are shown in Table 1.

Other than active and standby, we also use four low-speed power modes: 12k RPM, 9k RPM, 6k RPM and 3k RPM. For convenience of description, we call them NAP modes: NAP1, NAP2, NAP3 and NAP4. To calculate the parameters for each NAP mode, we use the linear power and time models proposed in [18]. We use the 2-competitive thresholds described in Section 2 for Practical DPM. For PA-LRU, we use an epoch length of 15 minutes. Other parameters are $\alpha = 50\%$, $p = 80\%$ and $\beta = 5seconds$. α, β, p are described in Section 4. β is set to be the same as the break-even time for NAP1 mode.

Our experiments use two real system traces to evaluate the power-aware cache replacement algorithms. The OLTP trace is an I/O trace collected on our previously built VI-attached database storage system connected to a Microsoft SQL Server via a storage area network. The Microsoft SQL Server client connects to the Microsoft SQL Server via Ethernet and runs the TPC-C benchmark [27] for 2 hours. The OLTP trace includes all I/O accesses from the Microsoft SQL server to the storage system. Writes to log disks are not included in the trace. A more detailed description of this trace can be found in our previous work [42, 7]. The other trace, Cello96, is obtained from HP and was collected from the Cello File Server. The characteristics of these traces are listed in Table 2. In our experiments, we use 128 MBytes as the storage cache size for the OLTP trace, and 32 MBytes for the Cello96 trace because its working set size is smaller than that of the OLTP trace.

5.2 Overall Results

We evaluate four cache replacement algorithms: Belady, OPG, LRU and PA-LRU, using the two real-system traces. We have also measured the disk energy consumption with an infinitely large cache size, in which case only cold misses go to disks. This serves as a lower bound for the energy consumed as a result of any cache replacement algorithm because no cache replacement algorithm with a limited cache size can save more energy if the underlying disks use the Oracle DPM.

	Disks	Writes	Average interarrival time
OLTP	21	22%	99ms
Cello96	19	38%	5.61ms

Table 2: Trace Characteristics

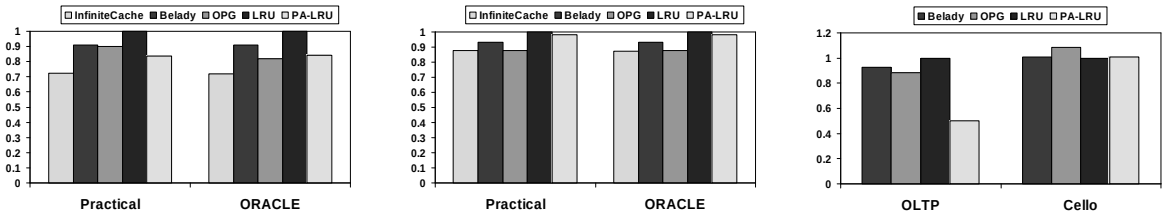
With the Practical DPM, infinite storage cache size may cause more energy consumption than limited cache sizes, so it cannot serve as a theoretical lower bound. To give a counter-example, suppose the inter-arrival time between two consecutive cold misses to the same disk is just slightly larger than the idle threshold value. After the first cold miss, the disk will transition into a low-power mode after remaining idle for a threshold period of time. Then it has to immediately transit back to active in order to service the second cold miss. So it spends extra energy in disk spin-down/spin-up. However, if a replacement algorithm can introduce another miss in between these two cold misses, it is possible to avoid the disk spin-down/spin-up.

Figures 6 (a) and (b) compare the disk energy consumption for all four storage cache replacement algorithms and an infinite cache with both Oracle DPM and Practical DPM. Figure 6 (c) shows the average response time for the four storage cache replacement algorithms for the Practical DPM. Since the Oracle DPM can always spin up disks in time for the next request, the average response time difference among the four schemes is very small.

Comparing the two off-line algorithms, even though Belady's algorithm gives the optimal cache miss ratios, OPG can save 2-9% more energy than Belady's algorithm. With the Cello96 trace, OPG can save 5-7% more energy than Belady's algorithm. With the OLTP trace, OPG can save 9% more energy than Belady's algorithm if disks use the Oracle scheme. With the Practical DPM, OPG's savings over Belady's is smaller, only 2%. For average response time, OPG is 5% better with OLTP but 6.4% worse with Cello96.

In the OLTP I/O trace, PA-LRU can save 16% more energy compared to LRU and it also improves the average response time by 50%. The reason is that PA-LRU can selectively keep blocks from certain disks in the storage cache for a longer time. Those disks could stay in the low power modes longer and we avoid expensive disk spin-ups. Since a disk-spin up from a low power mode takes a few seconds, avoiding this can also improve the average I/O response time. If underlying disks use the Practical DPM, PA-LRU saves more energy than OPG because OPG results in more disk spin-ups.

However, PA-LRU can only save 2-3% energy over LRU for the Cello96 trace. The reason is that in this trace, 64% of accesses are cold misses. In other words, 64% of accesses will go to disks even with an infinite cache size. In addition, the request inter-arrival gap is very small, even for the cold miss sequence, which does not allow PA-LRU to save energy. Even with an infinite cache size, the energy consumption is only 12% lower than LRU with 128 MBytes of cache. The difference in average I/O response time between LRU and PA-LRU is very small for Cello96.



(a) Energy: OLTP (b) Energy: Cello96 (c)Average response time

Figure 6: Effects of power-aware cache replacement (normalized to LRU).

For the OLTP trace, an infinite cache size saves 28% more energy than LRU, and 20% more than Belady's algorithm. Since OPG is power-aware, the differences between an infinite cache size and OPG are smaller, 10% for OLTP and less than 1% for Cello96.

5.3 Performance Analysis

In this section, we analyze the results to understand why PA-LRU can reduce disk energy consumption over LRU for the OLTP trace.

Figure 7 (a) shows the percentage time breakdowns for two representative disks. Each breakdown gives the percentage of time consumed in each power mode and also during spin-up/spin-downs. With PA-LRU, disk 14 spends 59% of time in standby mode , whereas it spends only 16% of time in standby mode with LRU. Moreover, PA-LRU significantly reduces time in performing spin-up/downs from 25% to 13%. Even though PA-LRU increases the percentage of time in active mode for other disks such as disk 4 from 78% to 84%, the amount of increase is very small. PA-LRU also reduces the time that disk 4 spends in spin-up/downs from 16% to 6%. Also, because of the significantly fewer disk spin-up/downs, PA-LRU has 50% lower average I/O response time.

Figure 7 (b) shows the mean request inter-arrival time, i.e., average interval length, for the same two representative disks (disk 4 and disk 14). The mean request inter-arrival time shown here is much larger than the inter-arrival time in the original application I/O sequence because requests are first filtered through a 128 MByte storage cache.

Since PA-LRU keeps blocks from disk 14 in the priority LRU list, there are fewer accesses to disk 14. As a result, the mean request inter-arrival time on disk 14 from a PA-LRU-managed cache is three times as large as that from a LRU-managed cache. With 40 second inter-arrival gaps, disk 14 has a lot of long idle periods to stay in low power modes, and thus saves significant amounts of energy.

To favor disk 14's blocks, disk 4's blocks are more likely to be evicted with PA-LRU than with LRU. Thus, the mean request inter-arrival time on disk 4 with PA-LRU is a factor of 2.4 shorter than that with LRU, which explains why PA-LRU causes disk 4 to stay in the active mode longer. Since the original mean inter-arrival time with LRU is already smaller than the threshold, disk 4 does not have much opportunity to go to the low power modes. Thus, shortening the mean inter-arrival time on disk 4 does not cause disk 4 to spend significantly less time in low power modes.

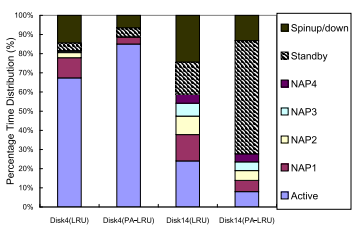
5.4 Effects of Spin-up Cost

In our simulations, we use the spin-up cost of the IBM Ultrastar 36Z15, i.e., 135J from standby to active mode. In this section, we discuss how spin-up cost affects the energy-savings of PA-LRU over LRU using the OLTP trace. We vary spin-up costs as energy needed for transitioning from standby mode to active mode. The spin-up costs from other modes to active mode are still calculated based on the linear power model described earlier.

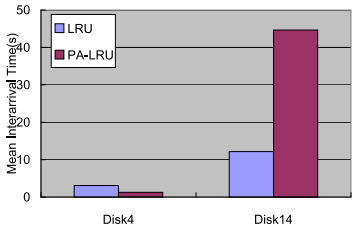
Figure 8 shows the percentage energy-savings for PA-LRU over LRU. Between 67.5J and 270J, the energy-savings of PA-LRU over LRU are fairly stable. The spin-up costs of most current SCSI disks lie in this range. At one extreme, with the increase of spin-up cost, the break-even times increase. Therefore, the thresholds calculated based on the break-even times also increase. In this case, due to lack of long enough intervals, disks have less opportunities to stay in low power modes even using PA-LRU. At the other extreme, if the spin-up cost decreases a lot and spin-up becomes very cheap, the energy savings of PA-LRU decreases because in this case, even with LRU, disks are already in low-power modes most of the time.

6 Effects of Write Policies on Disk Energy Consumption

In this section, we investigate the effects of four storage cache write policies on energy consumption. The first two policies, write-back and write-through, are commonly used in caches. The write-back caching policy only writes a dirty block to disks when the block is evicted from the cache. This policy reduces the number of disk writes and enables fast response to clients, but could compromise data persistency if the cache is on volatile storage. The write-through caching policy always writes dirty blocks to disk immediately and does not tell clients that the writes are successful



(a) Percentage time breakdown



(b) Mean request inter-arrival time

Figure 7: Percentage time breakdown and mean request inter-arrival time for two representative disks in OLTP

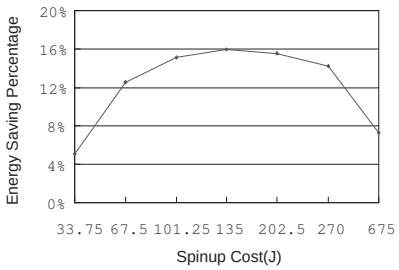


Figure 8: Percentage energy-savings for PA-LRU over LRU versus spin-up cost (energy needed for transitioning from standby mode to active mode)

until the data is committed to disks. The two other policies are variations of the write-back and write-through policies. "Write-back with eager update" (WBEU) does not wait for a dirty block to be evicted from the cache before writing it to the disk. The policy writes back dirty blocks of a disk whenever that disk becomes active. "Write-through with deferred update" (WTDU) temporarily writes dirty blocks to a log instead of writing them to their true destinations, if the destination disks are in low power modes.

To evaluate the effects of write policies, we use synthetic traces with varying write/read ratios. The synthetic traces are also generated with controlled spatial locality and temporal locality. Spatial locality is controlled by the probabilities of sequential accesses, local accesses and random accesses. Temporal locality is controlled by a Zipf distribution of stack distances. The default parameters for the trace generator are listed in Table 3. Similar to [18], we consider two types of distributions for inter-arrival times, Exponential and Pareto. Exponential distribution models a Poisson process, which is almost regular traffic without burstiness while the Pareto distribution introduces burstiness in arrivals. The Pareto distribution is controlled by two parameters, Shape α and Scale β. We use a Pareto distribution with a finite mean and infinite variance. We use LRU as the cache replacement algorithm in our simulation.

Write-back (WB) vs. Write-through (WT): Intuitively, write-back is more energy-efficient than write-through due to a reduced number of writes and potentially longer idle periods. However, few studies have measured the difference in energy consumption quantitatively. Is the difference large enough to justify trading persistency for energy?

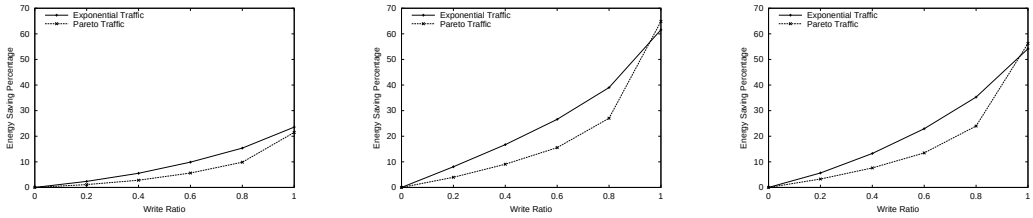
Figure 9(a1) and (a2) show the percentage energy savings of write-back over write-through. Figure 9(a1) shows the results for a mean inter-arrival time of 250ms, and varying write ratios from 0% to 100%. Figure 9(a2) shows the results when the write ratio is 50% and the mean inter-arrival time varies from 10ms to 10,000ms. The results with Oracle DPM are very similar to those with Practical DPM, so we only present results with Practical DPM.

With 100% writes, write-back can save around 20% energy compared to write-through. The difference becomes smaller when the write ratio decreases. When fewer than 40% of the requests are writes, the percentage energy savings of write-back over write-through is less than 5%. As shown in Figure 9 (a2), with a write ratio of 0.5, the benefits of write-back peaks during 100ms to 1000ms mean inter-arrival time, but the benefits are always smaller than 10%. The benefits of write-back over write-through are slightly better in the traces with exponential distributions than in the traces with Pareto distributions because the latter has bursty requests, which can reduce the number of disk spin-ups.

Write-back with Eager Updates (WBEU): The energy consumption with write-back can be further reduced by eagerly flushing dirty blocks to disks when the corresponding disks become active due to a read miss. In the extreme case, if a disk D always stays in a low-power mode, the storage cache will end up with a lot of D's dirty blocks. To avoid this scenario, if the number of such dirty blocks reaches a certain threshold, D is forced to transition into ac-

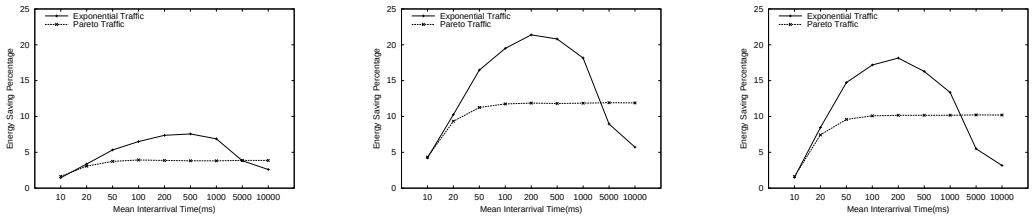
Request Number	1 million
Disk Number	20
Exponential Distribution	$\frac{1}{\lambda} = 10ms$
Pareto Distribution	$1 < \alpha \leq 2, \beta = 5ms$
Hit Ratio	0.3
Write Ratio	0.2
Disk Size	18 GB
Sequential Access Probability	0.1
Local Access Probability	0.2
Random Access Probability	0.7
Maximum Local Distance	100 blocks

Table 3: Default Synthetic Trace Parameters



(a1) WB vs. WT (b1) WBEU vs. WT (c1) WTDU vs. WT

(1) Percentage energy savings for different write ratios (inter-arrival time: 250ms)



(a2) WB vs. WT (b2) WBEU vs. WT (c2) WTDU vs. WT

(2) Percentage energy savings for different mean inter-arrival time (write ratio: 50%)

Figure 9: Effects of write policies on disk energy consumption (All the numbers are percentage energy savings relative to the write-through policy. The underlying disk uses the Practical DRM. Results with the Oracle DPM are similar.)

tive mode and the storage cache flushes D's dirty blocks. This policy is similar to the ones used in [38, 34, 8]. The advantage of this write policy is that writes avoid causing energy-expensive disk spin-ups.

Figure 9 (b1) and (b2) show the percentage energy savings of WBEU over write-through. WBEU can significantly reduce energy consumption. If 100% of the requests are writes, WBEU can save 60-65% more energy than write-through. Therefore, when the percentage of writes is significant in a workload, it is much more energy-efficient to use WBEU if persistency is not a big concern.

For the traces with exponential distributions, WBEU is more sensitive to the mean inter-arrival time. With a write ratio of 0.5, if the mean inter-arrival time is large (10,000ms), the benefits of WBEU are very small. This is because disks are "sleeping" most of the time and requests only cause little energy consumption. If the mean inter-arrival time is small (10ms), WBEU does not provide huge benefits, either. This is because disks are active most of the time due to fast arriving read requests. Mean inter-arrival time has less effect on WBEU's benefits with Pareto traffic because disks have longer idle periods and tend to stay in low power modes.

Write-through with Deferred Update (WTDU): Since write-through causes a lot of energy consumption, WTDU defers updates using a persistent log to avoid spinning up a disk in low power mode. This log can reside in any persistent device such as NVRAM or a log disk that is likely to be always active. In databases, log disks are usually always active because databases rely on its performance for fast transaction commits. With such a log, we can defer energy-expensive updates in write-through.

To ensure persistency, we divide the log space into log regions with one for each disk. The first block of a log region keeps the timestamp for the corresponding disk. This timestamp is also stored together with each block in the log region. The storage cache also keeps a pointer for each disk to remember the next free block in the corresponding log region. When a write request arrives for an inactive disk D, the blocks are first written to the corresponding log region and each block is timestamped with D's timestamp. The cache copies of these blocks are marked as "logged". When D becomes active due to a read miss, all "logged" blocks are flushed from the storage cache into D before servicing any write requests. Then the timestamp stored in the first block of D's log region is incremented by one. Finally, the corresponding free block pointer is reset.

The timestamp is used to ensure consistent recovery when the system crashes. After the system reboots, it first checks each log region to get the timestamps from its first block. Suppose the timestamp for a region A is n. If the timestamps of some blocks in the same region are also n, it means some blocks may not have been written back to the corresponding data disk. Thus, the recovery process will write all blocks with the same timestamp back to the corresponding disk. Otherwise, all blocks are already written back and the recovery process does not need to do anything for this log region and can move on to the next log region.

Figure 9(c1) and (c2) show the percentage energy savings for WTDU over write-through. When we evaluate WTDU, the extra energy consumption for writing to log regions is included in WTDU's results. If all accesses are writes, WTDU can reduce the disk energy consumption by 55% compared to write-through, which indicates WTDU is quite effective. Since this scheme can also provide persistency, it is good for workloads which have a high percentage of writes and require persistency semantics.

7 Related Work

Most previous work focuses on saving energy for a single disk in mobile devices, such as laptops, MP3 players, digital cameras. These studies can be roughly divided into three groups. The first group investigates how to dynamically adapt thresholds used to switch to low power modes. Many adaptive schemes have been proposed to vary thresholds based on workloads [16, 10, 26, 21]. Thresholds can also be calculated analytically [28, 24].

The second group deals with modeling of disk energy consumption. For example, Greenawalt proposes a purely analytical model that assumes requests arrive according to a Poisson distribution [17]. Helmbold et al. model disk power consumption in terms of seconds of activity [21]. A recent study conducted by Zedlewski et al. [40] presents a disk simulator called Dempsey to accurately model energy consumption of a single disk for mobile devices.

The third group of research investigates ways to reorganize idle periods in I/O request sequences by delaying or prefetching requests. Weissel et al. proposed a scheme called Cooperative I/O to allow applications to specify timeouts and abort options for I/O operations in order to provide more flexibility for disk power management [36]. Papathanasiou suggested delaying asynchronous requests if the disk is at low-power mode and prefetching some requests while the disk is at full power mode [34].

Though our study also tries to reorganize idle periods in I/O request sequences, this paper differs from these studies in two aspects. First, our work investigates changing storage cache management to reorganize idle periods, and therefore does not require any modifications to storage applications. Second, other studies focus on a single disk with multimedia workloads while our work focuses on multiple disks with data center workloads.

Recently, a few studies [8, 18, 19, 5] looked into energy management for high-end storage systems. A number of them [5, 18, 19] have shown that idle periods in data center workloads are usually very small compared to the time taken to spin-down and spin-up. Due to the high spin-up energy and time costs of server disks, there is not enough opportunity to save energy. To overcome this problem, Gurumurthi et al. have proposed using multi-speed disks to increase the amount of disk energy saved for data center workloads. Carrera and Bianchini at Rutgers have also suggested using multiple rotational speed disks to save energy [5]. These works motivate our study on power-aware cache management.

In addition to the works mentioned above, our project is also related to a couple of other studies. Corella and Grunwald [8] proposed a disk backup organization called MAID that uses power management of individual drives to achieve a very small power budget. Several recent studies [35, 20] have been conducted to conserve energy for networked servers. Most of them focus on conserving energy by dynamically reconfiguring or shrinking a cluster of networked servers to operate with a few nodes under light load. [6] proposes an architecture to allow services to "bid" for energy and other resources as a function of delivered performance in a hosting center. A few studies [4, 12] have investigated the energy consumption of front-end servers such as web-servers using dynamic voltage scaling of CPUs. Different from these studies, our research focuses on back-end storage in data centers.

8 Conclusions

In this paper, we show that power-aware storage cache management policies can significantly reduce disk energy consumption. We present OPG, a simple power-aware offline greedy algorithm that is more energy-efficient than Belady's cache replacement algorithm. We propose PA-LRU, a power-aware version of LRU, and show that it can use 16% less energy and lead to 50% better average response time than the LRU algorithm. Even though PA-LRU is based on LRU, this technique can also be applied to other replacement algorithms such as ARC [31] or MQ [42].

We also evaluate the effects of different cache write policies on disk energy consumption. Our results show that write-back uses up to 20% less energy than write-through. WBEU further reduces the disk energy consumption by up-to 45%. For systems with strong persistency requirements, we propose the use of a log disk with write through, and show that this policy (WTDU) can reduce the disk energy consumption by up to 55% compared to write-through.

Our study has some limitations. First, the OPG algorithm also works for a single disk. But our power-aware on-line algorithm is only designed for multiple disks since our study focuses on high-end storage systems. It remains our immediate future work to design power-aware on-line algorithms that work for a single disk. Second, we plan to extend our work to consider prefetching as well. Third, similar to many previous studies [36, 34], our experimental measurements do not include storage cache energy consumption because storage caches are usually kept active all the time to provide high performance. Fourth, similar to most previous stud-

ies [21, 18, 40], we use a simulator in our experiments. We are in the process of adding a disk power model to our previously built V3 storage system [41] to emulate disk spin up/spin down in a way similar to Carrera and Bianchini's study [5]. This will allow us to measure energy consumption in a real storage system.

9 Acknowledgments

The authors would like to thank the anonymous reviewers for the invaluable feedback. We are also grateful for the insightful discussion with Lenny Pitt on dynamic programming. This research is partially supported by the NSF CCR-0313286 grant and NSF CCR-0305854. Our experiments are conducted on equipments provided through the IBM SUR grant and the NSF EIA 02-24453 grant.

References

[1] Power, heat, and sledgehammer. White paper, Maximum Institution Inc., http://www.max-t.com/ downloads/ whitepapers/ SledgehammerPowerHeat20411.pdf, 2002.

[2] L. A. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, 5(2):78–101, 1966.

[3] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM*, 13(7):422–426, July 1970.

[4] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The case for power management in web servers. Power Aware Computing, Editors R. Graybill and R. Melhem, Klewer Academic Publishers, 2002.

[5] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving disk energy in network servers. In *ICS*, June 2003.

[6] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centres. In *SOSP*, pages 103–116, 2001.

[7] Z. Chen, Y. Zhou, and K. Li. Eviction-based cache placement for storage caches. In *Usenix Technical Conference*, 2003.

[8] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *SC – 2002*, Nov 2002.

[9] F. Douglis, R. Caceres, M. F. Kaashoek, K. Li, B. Marsh, and J. A. Tauber. Storage alternatives for mobile computers. In *OSDI*, pages 25–37, 1994.

[10] F. Douglis, P. Krishnan, and B. Bershad. Adaptive disk spin-down policies for mobile computers. In *Proc. 2nd USENIX Symp. on Mobile and Location-Independent Computing*, 1995.

[11] F. Douglis, P. Krishnan, and B. Marsh. Thwarting the power-hungry disk. In *USENIX Winter*, pages 292–306, 1994.

[12] E. N. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *the Second Workshop on Power Aware Computing Systems(held in conjunction with HPCA-2002)*, Feb 2002.

[13] EMC Corporation. Symmetrix 3000 and 5000 Enterprise Storage Systems product description guide., 1999.

[14] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.

[15] G. R. Ganger, B. L. Worthington, and Y. N. Patt. The DiskSim simulation environment - version 2.0 reference manual.

[16] R. A. Golding, P. B. II, C. Staelin, T. Sullivan, and J. Wilkes. Idleness is not sloth. In *USENIX Winter*, pages 201–212, 1995.

[17] P. Greenawalt. Modeling power management for hard disks. In *the Conference on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Jan 1994.

[18] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. DRPM: Dynamic speed control for power management in server class disks. In *ISCA*, pages 169–179, June 2003.

[19] S. Gurumurthi, J. Zhang, A. Sivasubramaniam, M. Kandemir, H. Franke, N. Vijaykrishnan, and M. Irwin. Interplay of energy and performance for disk arrays running transaction processing workloads. In *ISPASS*, pages 123–132, Mar. 2003.

[20] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini. Self-configuring heterogeneous server clusters. In *COLP'03*, Sept. 2003.

[21] D. P. Helmbold, D. D. E. Long, T. L. Sconyers, and B. Sherrod. Adaptive disk spin-down for mobile computers. *Mobile Networks and Applications*, 5(4):285–297, 2000.

[22] IBM hard disk drive - Ultrastar 36Z15.

[23] IBM. ESS-the performance leader. IBM Corporation, 1999.

[24] S. Irani, S. Shukla, and R. Gupta. Competitive analysis of dynamic power management strategies for systems with multiple power saving states. Technical report, UCI-ICS, Sept 2001.

[25] S. Jiang and X. Zhang. LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance. In *SIGMETRICS*, pages 31–42. ACM Press, 2002.

[26] P. Krishnan, P. M. Long, and J. S. Vitter. Adaptive disk spindown via optimal rent-to-buy in probabilistic environments. In *12th International Conference on Machine Learning*, 1995.

[27] S. T. Leutenegger and D. Dias. A modeling study of the TPC-C benchmark. *SIGMOD Record*, 22(2):22–31, June 1993.

[28] K. Li, R. Kumpf, P. Horton, and T. E. Anderson. A quantitative analysis of disk drive power management in portable computers. In *USENIX Winter*, pages 279–291, 1994.

[29] Y.-H. Lu and G. D. Micheli. Comparing system-level power management policies. *IEEE Design and Test of Computers*, 18(2):10–19, March 2001.

[30] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger. Evaluation techniques for storage hierarchies. *IBM Systems Journal*, 9(2):78–117, 1970.

[31] N. Megiddo and D. S. Modha. Arc: A self-tuning, low overhead replacement cache. In *FAST'03*, 2003.

[32] B. Moore. Taking the data center power and cooling challenge. Energy User News, August 27th, 2002.

[33] F. Moore. More power needed. Energy User News, Nov 25th, 2002.

[34] A. E. Papathanasiou and M. L. Scott. Increasing disk burstiness for energy efficiency. Technical Report 792, University of Rochester, November 2002.

[35] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. COLP'01, 2001.

[36] A. Weissel, B. Beutel, and F. Bellosa. Cooperative I/O: A novel I/O semantics for energy-aware applications. In *OSDI*, Dec. 2002.

[37] T. Wong and J. Wilkes. My cache or yours? making storage more exclusive. In *USENIX Annual Technical Conference (USENIX)*, 2002.

[38] R. Youssef. RAID for mobile computers. Master's thesis, CMU, 1995.

[39] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *SOSP'03*, Oct. 2003.

[40] J. Zedlewski, S. Sobti, N. Garg, A. Krishnamurthy, and R. Wang. Modeling hard-disk power consumption. In *the 2nd USENIX Conference on File and Storage Technologies*, 2002.

[41] Y. Zhou, A. Bilas, S. Jagannathan, C. Dubnicki, J. F. Philbin, and K. Li. Experiences with VI communication for database storage. In *ISCA'02*, May 2002.

[42] Y. Zhou, J. F. Philbin, and K. Li. The multi-queue replacement algorithm for second level buffer caches. In *Proceedings of the Usenix Technical Conference*, June 2001.

[43] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing energy consumption of disk storage using power-aware cache management. Technical report, UIUC, Nov. 2003.