PB-LRU: A Self-Tuning Power Aware Storage Cache Replacement Algorithm for Conserving Disk Energy *

Qingbo Zhu, Asim Shankar and Yuanyuan Zhou
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbaba, IL 61801
{qzhu1, shankar, yyzhou}@cs.uiuc.edu

ABSTRACT

Energy consumption is an important concern at data centers, where storage systems consume a significant fraction of the total energy. A recent study proposed power-aware storage cache management to provide more opportunities for the underlying disk power management scheme to save energy. However, the on-line algorithm proposed in that study requires cumbersome parameter tuning for each workload and is therefore difficult to apply to real systems.

This paper presents a new power-aware on-line algorithm called PB-LRU (Partition-Based LRU) that requires little parameter tuning. Our results with both real system and synthetic workloads show that PB-LRU without any parameter tuning provides similar or even better performance and energy savings than the previous power-aware algorithm with the best parameter setting for each workload.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Power Management, Disk Storage, Cache Management

1. INTRODUCTION

Internet-based services such as web-hosting, application services and out-sourced storage are increasingly being co-hosted in large data centers. The steady growth of such infrastructure has brought with it a major concern of *energy consumption*. Energy User News [33] predicts that power requirements at data centers will increase from 150-200W/ft^2 for a typical service provider today to 200-300W/ft^2 in the near future. Therefore the energy cost, increasing as much as 25% annually, is a growing consideration in the total cost of ownership of a data center [34].

Among various components of a data center, storage is one of the biggest energy consumers, consuming almost 27% of the total [1]. To make matters worse - the demands of increasing performance have led to disks with higher power requirements; moreover, storage demands are continuously growing (by 60% annually according to an industry report [34]).

To reduce energy consumption, modern disks provide multiple power modes including active, idle, standby and possibly other intermediate modes. At low power modes, disks consume less energy than at active mode. However, accessing disks at low power modes leads to a significant delay and energy penalty (e.g., 10.9s and 135J with an IBM Ultrastar 36Z15 disk) when spinning disks up to full power (active) mode. Therefore, only when the idle period is long enough to justify the spin-up costs, it is energy-beneficial to spin-down a disk.

Even though a lot of research has been carried out in disk power management, most of it has focused on single disks in mobile devices. So far only a few recent studies [5, 16, 15, 3, 37] have looked at power management for multiple-disk storage systems in data centers. The high volume of activity in such systems restricts the potential for energy savings as the average idle periods of the disks are too small to justify the cost of spinning the disks up and down. To overcome this, multi-speed disk architectures have been proposed by Gurumurthi et al. [15] and Carrera et al. [3]. The introduction of intermediate power modes in these architectures reduces the spin-up and spin-down costs and makes energy savings possible even when idle periods are small.

Since most of commercial storage servers use a large memory cache (storage cache) to speed up I/O, not every application I/O request results in a disk access. We recently investigated the impact of storage cache management on disk energy consumption [45] and showed that the performance-optimal algorithm which gives the best cache hit ratio is not energy-optimal. The study presented an on-line power-aware cache replacement algorithm called PA-LRU to provide more opportunity for underlying disk energy management schemes to save energy. The main idea was to selectively increase the idle periods for some disks to allow them to stay in low-power modes longer and thus save energy. That study also proposed power-aware write policies to avoid writes spinning up disks at low power modes.

While our previous work [45] on power-aware storage cache management makes a first step in a promising direction for storage energy management, it has a major limitation: the power-aware

*This research is partially supported by the NSF CCR-0313286 grant and NSF CCR-0305854. Our equipment was provided by the IBM SUR grant and the NSF EIA 02-24453 grant.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICS'04, June 26–July 1, 2004, Malo, France.
Copyright 2004 ACM 1-58113-839-3/04/0006 ...$5.00.

cache algorithm PA-LRU has at least four parameters and requires cumbersome parameter tuning for each workload as there is no single set of parameters that work well for all workloads. In addition, many of the parameters do not have direct relationship to disk energy consumption and response time, and are therefore difficult to achieve self-tuning using some simple extensions to the algorithm. As a result, it is difficult to use this algorithm in real systems.

This paper addresses this limitation in power-aware cache management. We propose a more elegant new power-aware on-line algorithm called PB-LRU (Partition-based LRU) that requires little parameter tuning. This algorithm dynamically divides the storage cache into separate partitions, one for each disk. Moreover, we also present a technique to estimate, at run time, the correlation between a disk's energy consumption and its corresponding storage cache partition size. This information is used to guide an energy-optimal cache partitioning scheme by formalizing it to a Multiple-Choice Knapsack Problem (MKCP), which is then solved using dynamic programming. Our experimental results with four workloads show that PB-LRU without any parameter tuning can provide similar or even better energy saving and I/O response time than the previous algorithm (PA-LRU) with the best parameter setting for each workload. In addition, we also present an energy-optimal off-line cache replacement algorithm in the Appendix.

This rest of the paper is organized as follows. The next section gives an overview of disk power models and power management schemes. Section 3 presents the partition-based power-aware cache replacement algorithm. The evaluation methodology is described in Section 4, followed by simulation results in Section 5. Section 6 summarizes related work and Section 7 concludes.

2. BACKGROUND

2.1 Disk Power Model and Power Management Schemes

The traditional disk power model includes two modes - active and standby. When the disk is "active", the platters rotate at full speed; when the disk is in "standby" mode, the platters do not rotate. The disk can only service requests in "active" mode. Standby mode consumes less energy than active mode. However, when the disk is in standby, a significant time and energy cost is incurred as the disk spins up to active mode before servicing a request. The minimum length of an idle period to justify the spin-up/spin-down costs is called the *break-even time*. Irani et al. describe a scheme to calculate the break-even times for multiple power modes [23].

Recent studies [16, 15, 3, 37] have shown that the traditional power model cannot save energy under server workloads as the idle periods between disk accesses in such workloads are too short to justify the expensive spin-up/spin-down costs of server disks. To address this problem, multi-speed disk models have been proposed by Gurumurthi et al. [15] and Carrera et al. [3]. Intermediate rotational speeds consume less energy than full speed and the time and energy costs associated with transitions from/to different speeds are less than those between active and standby mode. Although multi-speed disks are not in production yet, we use multi-speed disks in our simulations because of their potential to save energy under data-center workloads. Multi-speed disks can be designed to service requests either at all rotational speeds or only at full speed. We use the latter model since it is a simple extension of traditional 2-mode disk model.

To save energy without adversely affecting performance, disk power management schemes (DPM) must decide when to switch a disk to a lower power mode. There are two schemes, the Oracle and Practical DPMs, that have been commonly used in disk energy management studies. The Oracle DPM [29] knows how long an upcoming idle period is. It then decides which power mode the disk should be in based on the break-even times. It also spins up the disk just before the next request arrives. Thus, the Oracle DPM never incurs extra delay in servicing I/O requests.

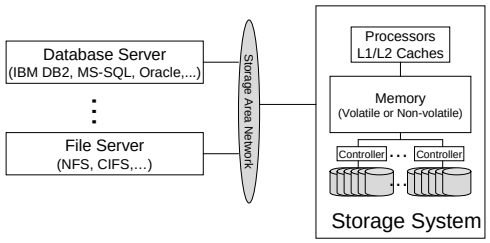


Figure 1: Modern storage architecture

Real systems use a Practical DPM based on thresholds [6, 8, 28, 13, 7, 26, 19, 42, 3, 16, 15]. In such schemes, if the disk remains idle for a certain threshold time, it is switched to the next lower power mode. The disk is spun up upon the arrival of the next request. Irani et al. also showed that by setting the thresholds to be the break-even times, the Practical scheme is 2-competitive to the Oracle scheme [23]. [45] describes the details of these schemes. This paper also uses thresholds calculated using the same methods.

Besides disk energy consumption, I/O response time is also a concern. If underlying disks use the Practical DPM, requests that arrive when the disk is in a low power mode will be significantly delayed (a few seconds) as the disk spins up to active mode.

2.2 Power Aware Cache Management

Not all accesses to a storage system go to disks. A typical architecture for a modern storage system is shown in Figure 1. Such systems use a large storage cache to reduce the number of disk accesses and improve 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 [10]. The IBM ESS system can have up-to 64 GB of storage cache [22]. Different from the 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 storage systems [44, 41, 32, 4].

The storage cache management policy influences the sequence of disk accesses. Different cache management policies generate different disk access sequences, which directly affects disk energy consumption. In other words, by changing the cache management scheme, it may be possible to increase the average idle time between disk accesses, thus providing more opportunities for the disk power management scheme to save energy.

Based on this observation, a previous study [45] took the first step towards investigating the role of storage cache management algorithms in reducing disk energy consumption. It formulated the energy-optimal problem and showed by example that the performance-optimal algorithm (which gives the best cache hit ratio) is not energy-optimal, i.e. the underlying disk power management scheme may lead to higher energy consumption than it would have with a different storage cache replacement algorithm. It also studied the effects of cache write policies and proposed two power-aware policies to avoid disk spin-ups due to writes.

The previous work [45] presented an on-line power-aware cache replacement algorithm called PA-LRU (Power-aware LRU). The main idea of PA-LRU is to selectively keep blocks from "inactive"

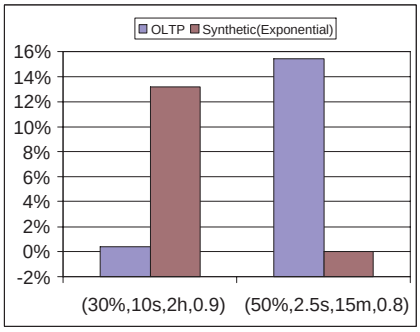


Figure 2: Energy savings (compared to LRU) with different parameter settings (α, β, th, p) in PA-LRU

disks in the storage cache longer and thus extend the idle period lengths of those disks. This can reduce the energy consumed by these disks as they can stay in lower power modes longer and spin up and down fewer times. The cost of this is a larger number of misses to "active" disks, but the energy penalty for this sacrifice is small since active disks stay in the high-power active mode most of time and accessing an active disk is relatively cheap in terms of energy (no spin-ups). Therefore, the overall storage energy consumption can be reduced. In PA-LRU, the measurement of disk activeness is adjusted periodically at the beginning of every "epoch".

However, PA-LRU requires cumbersome parameter tuning for each workload and no parameter setting works well for all workloads. To classify a disk as inactive, four parameters are required: (1) α - the maximum cold miss ratio for a disk; (2) β - the minimum average idle period length for a disk; (3) p - the probability of having this average idle period length and (4) th - the epoch length.

Figure 2 shows the energy savings made by PA-LRU with two different parameter settings on two workloads (The details of these workloads are described in Section 4). As shown in the figure, the parameter settings that lead to the most energy savings in one workload lead to virtually no energy savings in the second workload and vice-versa. Since there are four parameters in PA-LRU, finding the best setting for each workload can be extremely arduous. Moreover, for dynamic workloads, it is hard to find a setting that works all the time. Therefore, PA-LRU may not be practical for real systems.

Furthermore, it is difficult to use simple extensions to dynamically tune these parameters at run time as there is no direct correlation between several parameters and the disk energy consumption or I/O response time. For example, it is unclear how to dynamically determine values of α and β which will lower the disk energy consumption for the current workload.

To tackle this problem, this paper presents a fundamentally different power-aware storage cache replacement algorithm called PB-LRU that requires little parameter tuning. PB-LRU with virtually no parameter tuning provides similar or even better energy savings and I/O response times than PA-LRU with the best parameter setting for a workload. Moreover, we also present an energy-optimal storage cache replacement algorithm in the Appendix

Since the power-aware write policies proposed in the previous work [45] already avoid spinning up disks due to writes, in this paper, we focus on read requests (though writes still affect the storage cache content). Similar to the previous work, we assume that the storage cache is always active as it is critical to storage system performance.

3. PARTITION-BASED ALGORITHM

This section presents our partition-based on-line storage cache replacement algorithm. The goal of this algorithm is to be practical, i.e. to dynamically adapt to workload changes without tedious parameter tuning.

3.1 Main Idea

The main idea of PB-LRU (Partition-Based LRU) is to divide the entire cache into separate partitions, one for each disk, and manage each partition separately using a basic replacement algorithm such as LRU. The partitioning is such that the total energy consumption of all disks is minimized. This algorithm is motivated by the observation that storage system workloads are not equally distributed amongst the disks. The cache miss sequence going to each disk is thus easier to control if the disk caches are treated separately.

In order to find an energy-optimal partitioning, we first estimate, for each disk, the energy that *would be* consumed with different partition sizes. Symbolically, if we have n disks $\{1 \ldots n\}$, we estimate the energy, $E(i, s)$, that would be consumed by disk i if it had a partition of size s. These estimates are then used to find a partitioning that will minimize the total energy consumption of all disks. Of course, the sum of each partition size cannot exceed the total cache size S.

Let us first formalize the problem. Suppose there are m possible partition sizes: $0 < p_1 < p_2 < \cdots < p_m \leq S$. Let x_{ij} indicate whether disk i has a partition of size j (1 means "yes" and 0 means "no"). Obviously, each disk can only have one partition size, so we have $\sum_{j=1}^{m} x_{ij} = 1$. For disk i, its partition size S_i would be $\sum_{j=1}^{m} p_j x_{ij}$. Therefore, we have the following:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} E(i, S_i) \\ \text{subject to} \quad & \sum_{i=1}^{n} S_i \leq S, S_i = \sum_{j=1}^{m} p_j x_{ij} \\ & \sum_{j=1}^{m} x_{ij} = 1, x_{ij} = 0 \text{ or } 1 \end{aligned}$$

This problem is a form of the Multiple Choice Knapsack Problem (MCKP) [30], a variant of the famous 0-1 knapsack problem.

To solve this, PB-LRU needs to address two issues: (1) accurate estimation of the energy, $E(i, s)$, that would be consumed by disk i if it had a partition size s and (2) solving the MCKP which has been proved to be NP-hard [30].

3.2 Run-time Energy Estimation for Different Partition Sizes

In this subsection we describe a technique to dynamically determine the energy that each disk would consume with various possible partition sizes. A disk's energy consumption depends on the sequence of cache misses and the time of each miss. Since we must estimate energy for various possible partition sizes at run time, it is infeasible to conduct real measurement on the energy consumption with each different partition size using real organizations.

One simple method to address this problem would be to have multiple disk simulators, one for each partition size, to estimate the energy consumption for various partition sizes. This method would require too many ($numCacheSizes \times numDisks$) disk simulators running in the background and thus has time and space overheads too large to be practical.

Instead, PB-LRU uses a much more elegant technique to do the

estimation. Essentially, we expect to obtain a curve showing energy consumption as a function of partition size, for each disk, at run time. This is similar to the miss ratio vs. cache size curve, which has been dynamically obtained in several previous studies [25, 36] using the Mattson's Stack algorithm. To our knowledge, our study is the first to dynamically estimate how energy consumption varies with cache (partition) size.

Mattson's Stack algorithm was initially proposed by Mattson et al. in 1970 [31] to reduce trace-driven processor cache simulation time. It can determine the hit ratio of all processor cache sizes with a single pass through the trace file. It was later extended by Hill et al. [21, 20] and Wang et al. [39] to provide efficient trace simulations for set-associative caches. The main idea of this algorithm is to take advantage of the *inclusion property* in many cache/memory replacement algorithms [31] including the commonly used Least Recently Used (LRU) algorithm, the Least Frequently Used (LFU) algorithm and the offline Belady's algorithm. The inclusion property states that at any given time, the contents of a cache of k blocks is a subset of the contents of a cache of $k+1$ blocks for the same sequence of accesses. Therefore, if we maintain a "stack" (e.g. an LRU stack), an access to a block at depth i in the stack would lead to a miss in caches with size smaller than i, and a hit in others. Since the stack records only addresses of blocks and not their data, the space overhead of the stack is small.

Unfortunately, the Mattson's stack algorithm only gives us the correlation between cache size and cache miss ratio. Our goal is to minimize total disk energy consumption, not the cache miss ratio. Even though the miss ratio for two partition sizes may be significantly different, the resulting disk energy consumption can still be similar because extra misses may not cause any disk spin-ups.

We extend Mattson's Stack algorithm to dynamically track the variation of energy-consumption with possible partition sizes for each disk. PB-LRU first uses the Mattson's Stack algorithm to determine whether a request would result in a cache hit or miss for different partition sizes. If a request is a miss in a partition of size p (and all smaller sizes), the request will access the corresponding disk. If we know the last access time to this disk (with partition size p), we can estimate the energy consumption from the last access to the current one based on the underlying power management scheme. For example, if Practical DPM is used, we can decide what the current disk power mode is and thus calculate how much *idle energy* is consumed during this idle period (including the spin-up energy). The idle period is obtained from the current and previous disk access times. To get the *active energy*, we first measured the average disk access time on an IBM Ultrastar36Z15 disk and used this value (10ms) in our simulation. As shown in Section 5, our energy estimation is very accurate with an error of at most 1.8%.

Therefore, for each disk and each possible partition size, PB-LRU maintains the last access time to the disk (i.e. previous cache miss time, prev_miss_time) and its energy consumption. In our experiments, we set the basic allocation unit to be 1MB. Thus, if the total cache size is 128MB, for each disk we maintain 128 energy estimates and last access times corresponding to partition sizes of 1,2,3,...,128MB. At each access, besides changing the real cache to service this access based on the cache replacement algorithm, the following 3 steps take place to update the stack:

1. Search the requested block number in the stack of the appropriate disk. If it is found to be the i^{th} element from the top of the stack, its depth is i. If it is not found, its depth is ∞.

2. For all partition sizes less than the depth, increment the energy estimates. Update the previous miss time to the current access time.

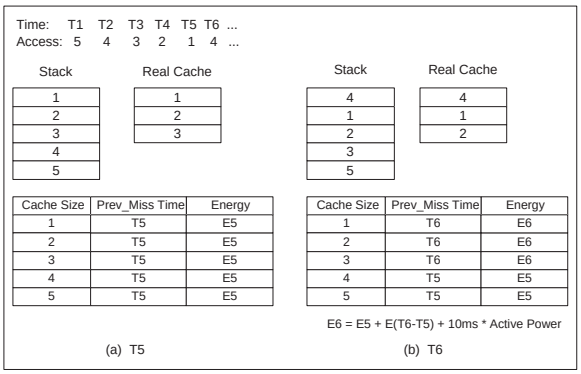


Figure 3: An example of energy estimate in PB-LRU for a disk

3. Update the stack using the same replacement policy as the real cache, e.g. PB-LRU brings the requested block number to the top of the stack.

Figure 3 illustrates the process of energy estimation for a disk with PB-LRU. Since the first five accesses are cold misses, at time $T5$, the prev_miss_time is $T5$ and the energy consumed is $E5$, for all partition sizes. At time $T6$ block 4 is accessed, which has a depth of 4 in the stack. For the partition sizes less than 4, a miss would occur. So the prev_miss_time for those sizes is set to $T6$ and the total energy consumption is incremented by the sum of idle energy consumed during the last idle period and the active energy, which is $E(T6-T5)+10ms*ActivePower$ (calculated based on the underlying DPM). However, if the partition for this disk has size of 4 or 5 blocks, there would have been a hit and no change is needed. Finally, both the stack and the real cache are updated based on LRU.

Although in our study we use LRU as the basic replacement algorithm, the methodology of partitioning described above is applicable to all policies that exhibit the property of inclusion such as LFU, 2Q [24] and MQ [44].

3.3 Solving the MCKP Problem

The Multiple-Choice Knapsack Problem has been proved to be NP-hard [30]. However, it can be solved in pseudo-polynomial time by dynamic programming as described below. The time complexity of the solution is $O(nm^2)$, where n is the number of disks and m is the number of potential partition sizes. Let $K(i,s)$ ($s \in \{p_1, p_2, \ldots, p_m\}$, $0 < p_1 < \ldots < p_m \leq S$) be the energy consumed by i disks when a total cache size of s is partitioned among those disks.

$$K(0,s) = \begin{cases} 0 & \text{if } s = 0 \\ \infty & \text{otherwise} \end{cases}$$

$$K(i,s) = \begin{cases} \min_{\{j|0<p_j\leq s\}}\{K(i-1, s-p_j) + E(i,p_j)\} \\ \qquad \text{if } s > 0 \\ \infty \quad \text{otherwise} \end{cases}$$

$$K^* = min_{\{s|0\leq s\leq S\}}\{K(n,s)\}$$

where K^* gives the minimum total energy consumption when a storage cache of size S is partitioned among all n disks.

As results will demonstrate, this technique has a tendency to increase the size assigned to relatively inactive disks and give only small sizes to disks which remain active. This is because the energy penalty incurred by reducing the partition size of an active

disk is small, whereas the energy saved by increasing the partition size of a relatively inactive disk is large.

To adapt to workload changes, we recalculate the energy-optimal partitioning solution periodically, at the beginning of an "epoch" (time interval). The epoch length is the only parameter in PB-LRU. However, our experiments show that PB-LRU is relatively insensitive to this parameter (See section 5.5).

4. EVALUATION METHODOLOGY

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

We simulate a complete storage system to evaluate our power-aware cache management schemes. We enhanced the widely used DiskSim simulator [11] by adding a disk power model. The power model used is similar to that proposed by Gurumurthi et al. [15] for multi-speed disks. We also developed a storage cache simulator, CacheSim and use it with DiskSim.

Accesses to the simulated disks first go through the 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 seeks, rotations and transfers). Therefore, if a replacement algorithm introduces extra misses, the energy consumed in servicing those extra misses is also included in our simulation results.

The specifications for the disk used in our study are similar to that of the IBM Ultrastar 36Z15. The parameters are similar to those used in [3], some of which are shown in Table 1. We added four lower-speed modes of 12k, 9k, 6k and 3k RPM to the disk and call the corresponding power modes NAP1, NAP2, NAP3 and NAP4 respectively. To calculate the parameters for each NAP mode we use the linear time and power models as suggested in Gurumurthi et al. [15]. For power management, we use the 2-competitive thresholds as stated in Section 2.

For our experiments, we use two real system traces (OLTP and Cello96) and two synthetic traces. The OLTP trace is collected from a VI-attached database storage system connected to a Microsoft SQL Server via a storage area network [43]. A Microsoft SQL Server client is connected to the SQL Server via Ethernet and runs the TPC-C benchmark [27]. This trace includes all I/O access from the SQL Server to the storage system, but excludes writes to the log disk. A detailed description of this trace can be found in [44, 4]. The Cello96 trace was collected from the Cello File Server and obtained from HP. OLTP has 21 disks (two 10-disk RAID0 plus an additional single disk). Cello96 has 19 disks. We use 128MB of total storage cache for OLTP and 32MB for Cello96. This is because the working set size of the Cello96 trace is significantly smaller than that of OLTP. The write ratios for OLTP and Cello96 are 22% and 38%, respectively.

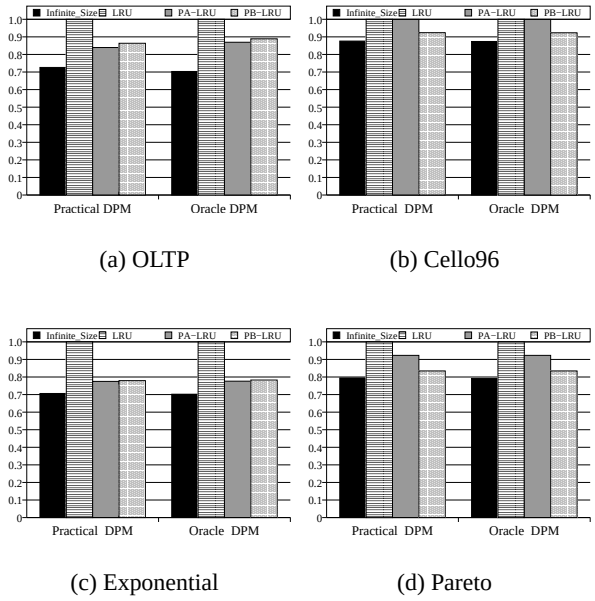
The two synthetic traces are generated based on storage system workloads observed in previous studies [44, 4]. For example, most workloads have an uneven distribution among disks and also among blocks. To simulate these characteristics, we use zipf distribution to distribute requests among 24 different disks and also amongst blocks in each disk. Moreover, as observed by previous studies [44], requests to storage systems have poorer temporal locality than those to first-level buffer caches and the reuse distances are distributed in a "hill" shape, or theoretically speaking, are log-normally distributed. Based on these characteristics, we use a log normal distribution with mean 32,000 references to reflect temporal locality. For inter-request arrival distribution, similar to [15], we also use two distributions - exponential and Pareto with 100ms as the average time between requests. We call these two synthetic traces Exponential and Pareto in the rest of this paper.

5. SIMULATION RESULTS

5.1 Overall Results

We evaluate the energy saving on each of the four traces made by four different algorithms - Infinite cache, LRU, PA-LRU (with the best parameter settings) and PB-LRU - using both Practical and Oracle DPM schemes. Oracle DPM combined with an infinite cache size serves as a lower bound on the total energy consumption for a workload. The relative energy consumption of the four algorithms is shown in Figure 4 and the relative average response time is compared in Figure 5. All results are normalized to the performance on an LRU managed cache. Since the Oracle DPM always spins disks from low power to active mode just in time for a request, it does not slowdown the average response time. Therefore, we only show the response time with the Practical DPM. All PB-LRU results are achieved with the epoch length as 16,000 requests. Section 5.5 will show that PB-LRU results are insensitive to the epoch length as long as it is large enough for the cache to "warm-up" after the re-partitioning.

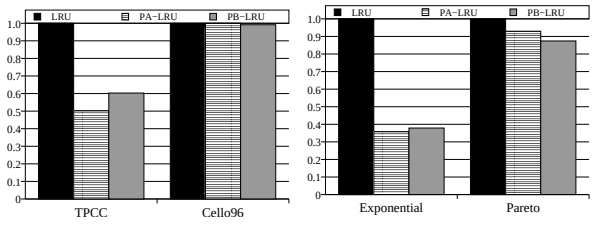
As shown in the Figures 4 and 5, PB-LRU with no parameter tuning can achieve energy savings and performance similar to, or in some workloads even better than, the PA-LRU with the best pa-



(a) OLTP (b) Cello96

(c) Exponential (d) Pareto

Figure 4: Energy Consumption



(a) OLTP and Cello96 (b) Synthetic Traces

Figure 5: Average Response Time

rameter settings. For example, for the Pareto trace (Figure 4(d)), PB-LRU saves 9% more energy and provides 5% better response time than PA-LRU. The results indicate that PB-LRU would be a better alternative to be used in real systems.

PB-LRU performs similar to PA-LRU for both the OLTP trace and the Exponential trace. With the OLTP trace (Figure 4(a)), both PB-LRU and PA-LRU consume 15% less energy and have 40-50% faster response time than LRU. With the Exponential trace (Figure 4(c)), again PB-LRU performs as well as PA-LRU, saving 22% in energy and reducing response time by nearly 65%.

PB-LRU outperforms PA-LRU in the other two traces. Besides Pareto, with the Cello96 PB-LRU saves 7-8% more energy than PA-LRU. However, the saving with PB-LRU for the Cello96 is still limited. This is due to the high percentage of cold misses (64% of all accesses), i.e., at least 64% of all accesses go to disk irrespective of the cache replacement policy. Moreover, the inter-arrival time of this cold miss sequence is very small, making energy saving even harder. Response time improvement is thus also limited, only 0.86% by PB-LRU while PA-LRU shows no improvement.

5.2 Accuracy of Energy Estimation

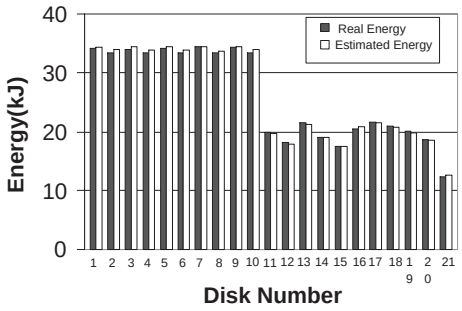


Figure 6: Real Energy vs. Estimated Energy: the difference is within 1.8%, which demonstrates the accuracy of energy estimation.

Figure 6 shows the difference between the disk energy actually consumed with the real cache and that estimated by PB-LRU for the same size, for each of the 21 disks in a randomly-selected epoch (using OLTP with Practical DPM). The largest deviation of estimated energy from real energy is 1.8%, suggesting that the energy estimated is quite accurate. The same is the case for the other traces we used.

5.3 Cache Partition Sizes

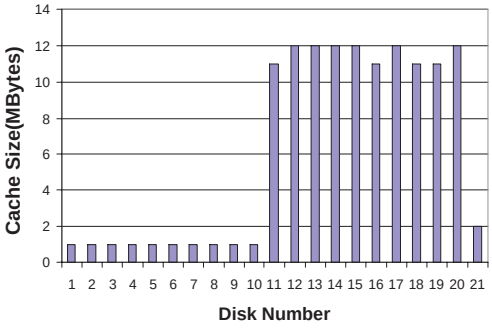


Figure 7: The cache partition sizes assigned by MCKP solver in a random epoch for OLTP(Practical DPM)

Figure 7 shows, in a random epoch, the partition sizes which were assigned by the MCKP solver based on the estimated energies of the previous epoch (using OLTP with Practical DPM). For the first 10 disks, the MCKP solver only assigns 1MB to their partitions while 11-12MB is given to the next 10 disks. The OLTP workload is such that the first 10 disks are active while the next 10 are relatively inactive. The MCKP solver has a tendency to increase the size assigned to relatively inactive disks and give only small sizes to disks which remain active. This is because the energy penalty incurred by reducing the partition size of a disk that remains active is small while large gains are made by increasing the partition size of a relatively inactive disk, as doing so allows it to remain in lower-power modes longer. In this way, overall energy savings can be made.

Cache partition size also affects the response time. Since the first 10 disks are in active mode, accesses to those disks do not need to wait several seconds for the disk to spin-up before requests are serviced. Because of the greater partition space given to the next 10 disks, the number of misses and consequently the number of expensive spin-ups from low-power to active mode is reduced and thus response time improves.

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 PB-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 PB-LRU over LRU. Between 67.5J and 270J, the energy-savings of PB-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 power-aware algorithms as PB-LRU. At the other extreme, if the spin-up cost decreases a lot and spin-up becomes very cheap, the energy savings of PB-LRU decreases because in this case, even with LRU, disks are already in low-power modes most of the time.

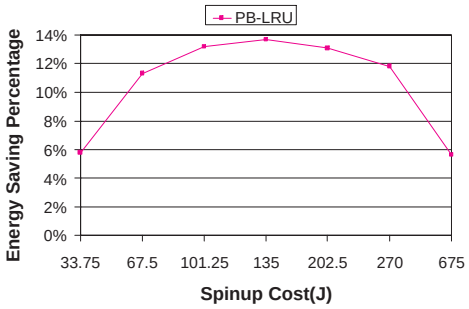
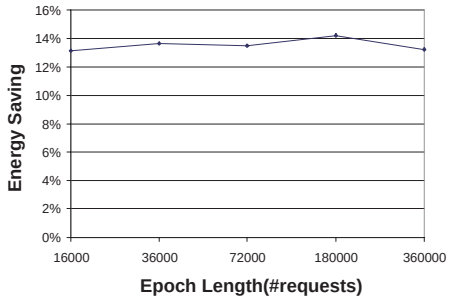
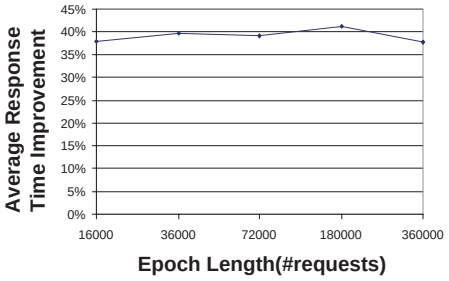


Figure 8: Percentage energy-savings for PB-LRU over LRU versus spin-up cost (energy needed for transitioning from standby mode to active mode)

5.5 Sensitivity Analysis on Epoch Length



(a)Energy Consumption



(b)Average Response Time

Figure 9: The energy saving and average response time improvement for OLTP trace change with the increasing of epoch length

PB-LRU only has one parameter, the epoch length. Obviously, the epoch length cannot be too small or infinitely large. But, fortunately, our results indicate that once it is large enough to accommodate the "warm-up" period after re-partitioning, the results of PB-LRU are relatively insensitive to the epoch length within a very large range, as shown in Figure 9. The results for the other traces are similar. In real systems, it is not difficult to pick a large enough epoch length, especially since most data center workloads are continuously running for days or even months.

6. RELATED WORK

6.1 Modeling Disk Power Consumption

Modeling disk energy consumption is important to design effective power management schemes. Greenawalt, for example, uses a purely analytical model which assumes requests arrive according to a Poisson distribution [14]. Helmbold et al. model disk power consumption in terms of seconds of activity instead of using Joules [19]. A recent study conducted by Zedlewski et al. [42] presented a disk simulator called Dempsey which can accurately model energy consumption of a single disk for mobile devices.

6.2 Power Management for Mobile Devices

Most previous research has focused on power management for a single disks in mobile devices such as laptops. These studies can be roughly divided into two categories. The first one does not change the timing for a given request sequence, but tries to dynamically adapt thresholds used to switch to low power modes based on these workloads. Many adaptive schemes have been proposed to vary thresholds [13, 7, 26, 19]. [28, 23] calculate the threshold analytically. A recent study [12] uses program counter based techniques to predict the length of upcoming idle periods.

The second category investigates ways to reorganize idle periods in I/O request sequences by delaying or pre-fetching requests. Weissel et al. proposed a scheme called Cooperative I/O to allow applications to specify deferrable or abortable I/O operations which provide more flexibility for underlying disk power management [40]. T. heath et al. considered application transformation to increase idle periods and inform the operating system about the length of each upcoming idle period [18]. Papathanasiou suggested delaying asynchronous requests when the disk is at low-power mode and pre-fetching some requests when the disk is at full power mode [35].

Our study also tries to reorganize idle periods in I/O request sequences, however, it differs from other studies in two aspects. First, our work uses the storage cache to reorganize idle periods and does not require any modifications to applications. Second, our work focuses on multiple disks with data center workloads instead of a single disk with interactive I/O workloads.

6.3 Power Management for Data Centers

Some recent studies [5, 15, 16, 3, 37] have looked into energy management for high-end storage systems. A number of them [3, 15, 16] have shown that the average idle period between requests in server workloads is small compared to the spin-up and spin-down cost of server disks, which limits the possible energy saving. Gurumurthi et al. [15] and Carrera et al. [3] have both proposed multi-speed disks to address this problem. A recent work [37] proposes a technique called PDC to migrate frequently accessed data to a subset of disks so that other disks are able to stay in low-power modes due to light loads. These studies share the same goal as our study, but they investigate other aspects of disk power management.

Our previous study [45] began to look at power-aware storage cache management algorithms to minimize disk energy consumption. However, compared to PA-LRU, PB-LRU presented in this paper is more theoretically elegant, achieving similar energy saving and response time improvement with significantly less cumbersome parameter tuning.

Our paper is also related to a couple of other studies. Corella and Grunwald [5] proposed a disk backup organization called MAID that uses a small number of disks as "cache drives" to save energy by reducing spin-ups of data drives. Several recent studies [38, 17] 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. A few studies [2, 9] have investigated the processor energy consumption of front-end servers using dynamic voltage scaling. Our research focuses on high-end storage systems in data centers.

7. CONCLUSION

In this paper we presented a simple, robust on-line storage cache replacement algorithm, called PB-LRU, which partitions the total system cache amongst the individual disks. Key to this algorithm was the dynamic estimation of the energy that would be consumed by a disk had the partition size assigned to it been different. Once these estimates were available, the optimal partitioning was obtained by reducing the problem to the Multiple Choice Knapsack Problem (MCKP). PB-LRU is fundamentally different from the previous power-aware algorithm, PA-LRU. Our results with four workloads show that PB-LRU with no parameter tuning provides performance and energy-savings similar to or better than PA-LRU with the best possible parameter settings.

Investigation of the role of cache management in energy conservation is far from complete. First, we plan to take pre-fetching into account wherein we must be able to optimally divide the total cache between the cache and pre-fetching buffers. Secondly, similar to many previous studies [19, 15, 42], we use a simulator in our evaluation. We are currently in the process of adding a disk power model to the V3 storage system [43], in a way similar to Carrera and Bianchini's work [3, 37], to allow us to measure disk energy consumption in real storage systems.

Acknowledgements

We would like to thank the anonymous reviewers for their invaluable feedback. We also thank Anand Sivasubramaniamm from PSU for the multi-speed disk power model and Greg Ganger from CMU for providing us Disksim.

8. REFERENCES

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

[2] 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.

[3] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving disk energy in network servers. In *Proceedings of the 17th International Conference on Supercomputing*, June 2003.

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

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

[6] 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.

[7] 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.

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

[9] 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.

[10] EMC Corporation. Symmetrix 3000 and 5000 Enterprise Storage Systems product description guide. http://www.emc.com/ products/product pdfs/pdg/symm_3_5_pdg.pdf, 1999.

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

[12] C. Gniady, Y. C. Hu, and Y.-H. Lu. Program counter based techniques for dynamic power management. In *10th International Symposium on High Performance Computer Architecture*, pages 24–35, Feb. 2004.

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

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

[15] 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*, pages 169–179, June 2003.

[16] 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 *Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 123–132, Mar. 2003.

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

[18] 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*, Sept 2002.

[19] 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.

[20] M. D. Hill. *Aspects of Cache Memory and Instruction Buffer Performance*. PhD thesis, Unversity of Berkeley, 1987.

[21] M. D. Hill and A. J. Smith. Evaluating associativity in CPU caches. *IEEE Transactions on Computers*, 38(12), 1989.

[22] IBM. IBM Enterprise Storage Server. www.storage.ibm.com/hardsoft/products/ess/ess.htm IBM Corporation, 1999.

[23] 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.

[24] T. Johnson and D. Shasha. 2Q: A low overhead high performance buffer management replacement algorithm. In J. Bocca, M. Jarke, and C. Zaniolo, editors, *VLDB*, pages 439–450, Los Altos, CA 94022, USA, 1995. Morgan Kaufmann Publishers.

[25] J. Kim, J. Choi, J. Kim, S. Noh, S. Min, Y. Cho, and C. Kim. A low-overhead high-performance unified buffer

management scheme that exploits sequential and looping references. OSDI, 2000.

[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*, 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] S. Martello and P. Toth. Knapsack problems: Algorithms and computer implementations. John Wiley and Sons, Ltd., 1990.

[31] 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.

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

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

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

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

[36] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka. Informed prefetching and caching. In *the 15th ACM Symposium on Operating System Principles*, 1995.

[37] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *the 18th International Conference on Supercomputing*, June 2004.

[38] 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.

[39] W. H. Wang and J. L. Baer. Efficient trace-driven simulation method for cache performance analysis. In *SIGMETRICS*, 1990.

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

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

[42] 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.

[43] 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.

[44] 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.

[45] 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. In *10th International Symposium on High Performance Computer Architecture*, 2004.

APPENDIX

A. ENERGY-OPTIMAL CACHE REPLACEMENT ALGORITHM

It is important to understand optimal cache replacement algorithms since they can help further understand the problem of cache replacement and serve as a baseline for other algorithms. Our previous work [45] showed that the performance-optimal algorithm which minimizes the number of misses is not energy-optimal, but did not give an energy optimal algorithm. In this appendix, we present an energy-optimal storage cache replacement algorithm using dynamic programming.

For simplicity, we consider a single disk with b blocks and only two power modes. Let the sequence, S, of disk references be $a_0, a_1, ...a_{n-1}$, thus n is the input size. Let k be the number of blocks in the cache. When a block is read from the disk, it is stored in the cache. If the cache is full, a cache block is replaced. The disk spends 1 unit of energy per reference when it is switched on and does not consume energy when in standby mode. For simplicity, we also assume that the disk goes back to standby after m units of time and I/O requests arrive uniformly at the rate of one per time unit. Other arrival distributions can be handled by inserting fake requests that will always be present in the cache (for example, repeat the last I/O request) at each idle time unit. Due to space constraints we do not go into the details of this extension. The objective of an energy-optimal cache replacement algorithm is to determine the sequence of cache replacement decisions that minimizes the energy consumption of the disk.

We can represent a cache replacement algorithm using a directed acyclic graph (DAG). The state of the cache can be represented by a tuple (C, t, i) which means that the cache contains the blocks in set C after the first $i+1$ references - $a_0, a_1, ...a_i$ - and the last t consecutive references were cache hits. If the next reference a_{i+1} is found in the cache ($a_{i+1} \in C$), the next state is $(C, t+1, i+1)$. If $a_{i+1} \notin C$, the next state could be one of several possible states of the form $(C', 0, i+1)$, where C' is one of k possible sets that result from replacing one block in C by a_{i+1}. Since the disk is accessed due to a cache miss, t goes to 0.

In this model, minimal energy consumption corresponds to the maximum time that the disk can spend in the standby mode. We define $A(C, t, i)$ as the maximum time that the disk spends in standby mode until some appropriate sequence of cache replacements results in state (C, t, i) being reached. $A(C, t, i)$ can be obtained in a recursive manner as follows:

$$A(C,t,i) = \begin{cases} max_{t',C'} A(C', t', i-1) | \\ \quad C \in repl(C', a_i) \wedge a_i \notin C' & \text{if } a_i \in C \wedge t = 0 \\ A(C, t-1, i-1) + 1 & \text{if } a_i \in C \wedge t \neq 0 \wedge t > m \\ A(C, t-1, i-1) & \text{if } a_i \in C \wedge t \neq 0 \wedge t \leq m \\ \infty & \text{if } a_i \notin C \end{cases} \quad (1)$$

where $repl(C, a_i)$ is the set of possible resulting caches after replacing some block in C with a_i. The above equation can be explained as follows. If $t \neq 0$, then from the transition diagram it is clear that state (C, t, i) can be reached only from $(C, t-1, i-1)$ where $a_i \in C$, hence the second and third cases. The second case adds 1 since if the disk is not accessed for $t > m$ references, the disk goes to standby mode. The fourth case states that (C, t, i) can never occur if $a_i \notin C$ since every page reference has to be cached. However $(C, 0, i)$ can be reached from any of the states $(C', t', i-1)$ such that $a_i \notin C'$ and C can be obtained by replacing one block in C' with a_i. The first case just maximizes over those possibilities. By computing $A(C, t, i)$ in the lexical ordering of the pair (i, t) such that $t \leq i$, we can ultimately arrive at $max_{C,t}(A(C, t, n-1))$

which is the maximum time for which the disk can stay in standby mode over the entire input sequence.

The algorithm is shown below.

1: **for** $i = 0$ to $n - 1$ **do**
2: **for** $t = 0$ to i **do**
3: **for all** C such that C is a set of b distinct disk blocks **do**
4: **if** $a_i \in C \wedge t = 0$ **then**
5: $A(C, t, i) \leftarrow max_{t',C'}(A(C', t', i-1) | C \in repl(C', a_i) \wedge a_i \notin C')$
6: **else if** $a_i \in C \wedge t \neq 0 \wedge t > m$ **then**
7: $A(C, t, i) \leftarrow A(C, t-1, i-1) + 1$
8: **else if** $a_i \in C \wedge t \neq 0 \wedge t \leq m$ **then**
9: $A(C, t, i) \leftarrow A(C, t-1, i-1)$
10: **else**
11: $A(C, t, i) \leftarrow \infty$
12: **end if**
13: **end for**
14: **end for**
15: Maximum Standby Time $\leftarrow max_{C,t}(A(C, t, n-1))$
16: **end for**

We now analyze the worst-case time complexity of the above algorithm. Let us first find the number of steps required to compute $A(C, t, i)$ assuming that $A(C', t', i')$ is known for all C', t' and $i' < i$. If $t \neq 0$, then by the second and third cases in Equation 1, we need $O(1)$ time to find $A(C, t, i)$. If $t = 0$, then we will have to maximize over all possible cache configurations C' such that $C \in repl(C', a_i)$ and $t' \leq i - 1$. The number of different C''s is b, the number of possible different blocks in the cache that a_i can replace. The number of different t' values is i. Thus, the total number of choices to be considered when computing $A(C, 0, i)$ is $i \times b$. Now t can vary from 0 to i. Therefore time taken to compute $A(C, t, i)$ for all possible $t \leq i$ is $i \times b + i$. This has to be done for b^k possible cache configurations. When summed over all $i < n$, we obtain the time complexity as $\Sigma_i b^k i \times (b+1)$ which is $O(b^{k+1} n^2)$.

The above algorithm can be extended to work with multiple disks with multiple power modes as well. After representing the state as $(C, t_1, t_2, ..., t_r, i)$, a recursive formula can be written for $A()$ which would be the maximum time that is spent in standby mode of all the disks put together. Space constraints prevent us from going into details.