Power Reduction of Multiple Disks Using Dynamic Cache Resizing and Speed Control *

Le Cai
School of Electrical and Computer Engineering
Purdue University
lc@purdue.edu

Yung-Hsiang Lu
School of Electrical and Computer Engineering
Purdue University
yunglu@purdue.edu

ABSTRACT

This paper presents an energy-conservation method for multiple disks and their cache memory. Our method periodically resizes the cache memory and controls the rotation speeds under performance constraints. The cache memory stores the data from the disks for reuse. Enlarging the cache memory reduces disk accesses and disk utilization. This allows the disks to reduce their speeds and conserve energy because the disks' power consumption is quadratic to their speeds. However, the cache memory itself consumes power to retain data. Shrinking cache memory can save memory power while increasing disk accesses and degrading performance. Choosing proper cache sizes and rotation speeds can reduce the energy consumption of both memory and disks with satisfactory performance. We model cache resizing and speed setting as an optimization problem with minimizing the power consumption as objective and limiting disk utilization as constraints. We compare our method with the methods resizing cache based on request rates. The simulation results show that our method achieves better energy savings while limiting disk access latency.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance attributes

General Terms

Design, performance

Keywords

Disk cache, disk rotation speed, power management

*This work is supported in part by National Science Foundation CNS-0347466, and Purdue Research Foundation. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

ISLPED'06, October 4–6, 2006, Tegernsee, Germany.

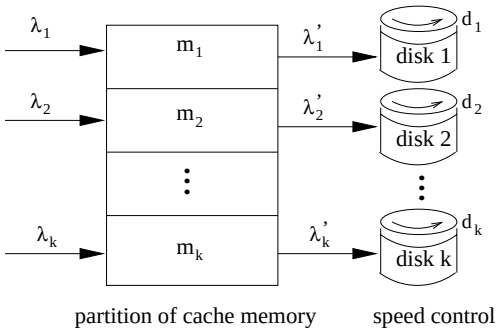


Figure 1: Caching based speed management (CBSM) for multiple disks.

1. INTRODUCTION

Energy conservation in storage system increasingly receives research attention [6, 8, 9, 14] because storage system, including cache, memory, and disks, consumes significant amounts of power. Most storage devices support low-power modes to reduce power consumption. For example, hard disks can switch to the standby or sleep mode when no disk accesses occur; memory banks can stay in the low-power modes after accesses. In addition to low-power modes, hard disks can reduce their power consumption by decreasing rotation speeds [4]. Adjusting disk rotation speeds achieves similar energy savings from the disks as frequency scaling from processors. When the rotation speeds drop, the power consumption of disks quadratically decreases while the disks' service time increases almost linearly [4, 13]. Therefore, energy savings can be achieved by lowering the rotation speeds.

Main memory is used as the cache of disks. More data cached in memory can eliminate more accesses to the disks so that they can be slowed down to save power. However, the memory itself consumes power to retain data. Using more memory consumes more power. Choosing proper cache memory sizes and disk rotation speeds can minimize the power consumption of both memory and disks. Allocating more memory to one disk reduces the cache memory used by other disks. Such cache memory resizing can save energy by allocating more cache memory to the disks which can achieve larger energy savings.

This paper presents a method to periodically adjust the cache sizes and the rotation speeds of multiple disks to achieve better energy savings while limiting disk utilization. We call the method caching based speed management (CBSM). The cache memory is distinguished from the rest of the main memory because the cache memory stores the

method	disk PM	memory PM	cache size/partition	feature	performance constraint
PB-LRU[16]	sleep	no	fixed/yes	caching	no
DRPM[4]	speed control	no	fixed/no	speed control	no
PDC[10]	speed control	no	fixed/no	file migration+speed control	no
Hibernator[15]	speed control	no	fixed/no	file migration+speed control	yes
CR[7]	sleep	no	fixed/no	code restructuring	no
CBSM	speed control	yes	variable/yes	caching+speed control	yes

Table 1: Various power management of disk arrays. PM: power management. No: not used. Yes: used. Sleep: transition disks to sleep or standby mode.

disk data for reuse. We study server systems because their data are often reused and the cache memory can considerably reduce disk accesses. Figure 1 shows the concept of CBSM. The cache size and the request rate of the i^{th} disk are represented by m_i and λ_i, respectively. After caching, the request rate decreases to $\lambda_i^{'}$. The i^{th} disk runs at a rotation speed of d_i. The goal of CBSM is to determine m_i and d_i to reduce the total power consumption under performance constraints. The performance constraints are to limit the average disk utilization because high utilization causes longer latency [5]. The utilization is defined as the ratio between disk request rate and disk bandwidth. We model CBSM as an optimization problem. The power consumption and the disk utilization are expressed as the functions of m_i and d_i based on three relationships: (a) the cache size and the disk request rate, (b) the rotation speed and the disk's service time, and (c) the rotation speed and the disk's power consumption. We simulate the disk cache and the power manager to compare CBSM with other methods of resizing cache memory and controlling rotation speeds for multiple disks. The results show that CBSM consistently achieves energy savings with lower latency of disk accesses.

2. RELATED WORK

Researchers present many methods to reduce the power consumption of disk arrays. We compare these methods in Table 1. Zhu et al. [16] present a partition based LRU (PB-LRU) replacement to dynamically partition the cache memory for multiple disks. Their method enlarges the disks' idleness and saves power by switching them to low-power modes. However, in server systems, the disks experience short idleness even when the workload is light. Gurumurthi et al. [4] exploit multiple rotation speeds for power savings, called dynamic rotations per minute (DRPM). File migration can help DRPM to concentrate the workload on a small set of disks. When disks support only a few speeds, workload concentration makes a few disks running at high speeds while other disks can slow down and save power. Pinheiro et al. [10] propose popular data concentration (PDC) to move data across disks so that the first disk contains the most popular data, the second disk contains the second most popular data, and so on. Due to the limited bandwidth of a single disk, their method has to tradeoff power savings with performance degradation. Zhu et al. [15] improves PDC by dividing a disk array into groups and each group contains the same popular data. Their method, called Hibernator, models the relationship between the disk request latency and the rotation speed to limit performance degradation caused by speed control. Kandemir et al. [7] propose code restructuring (CR) to concentrate disk accesses on a small set of disks so that the rest disks obtain longer idleness and more power-saving opportunities. Compared with these methods, our method has two advantages: (a) Using caching to manage workload avoids the complicated file migration or source codes revision. (b) Considering the energy consumed by the cache memory provides more energy-saving opportunities.

symbol	description	unit
d	rotation speed	RPM
m	cache size of one disk	MB
p	memory static power	W/MB
e	memory dynamic energy	J/MB
r/R	min/max rotation speed	RPM
f	transfer rate at R	MB/s
h	rotation delay at R	second
λ	request rate	request/s
N	data set size	MB
α	Zipf parameter	
s	seek time	second
q	request size	MB
l	$h + \frac{q}{f}$	second
u	disk utilization	

Table 2: Symbols and their meaning. Symbols with subscript i: related to the i^{th} disk.

3. CACHING BASED SPEED MANAGEMENT

We model the power consumption of memory and disks to obtain proper cache sizes and disk speeds. We also estimate how they affect disk utilization. Table 2 lists the symbols used in our modeling.

3.1 Power and Performance Model

The disk's power consumption includes: static power when no accesses occur and dynamic power for accesses. We use the quadratic power model [4] to estimate the power consumption at different speeds. This is the latest model we can find for the power consumption of multiple-speed disks. The static power is $ad_i^2 + bd_i + c$, where d_i is within $[r, R]$. The values of r and R are 3600 and 12000 RPM. The coefficients a, b, and c are disk-specific constants. The dynamic power is proportional to the disk speed. The disk's idle power and peak power at R are 22.3W and 39W [4]. The dynamic power at different speeds is estimated using $u_i g d_i$, where u_i is the i^{th} disk utilization. The value of g is $\frac{39-22.3}{12000} = 1.4 \times 10^{-3}$.

The memory's power consumption also includes static and dynamic parts. We use a power model of Rambus dynamic

random-access memory (RDRAM) [11]. We consider that the RDRAM stays in the nap mode after memory accesses as proposed in [6, 8]. A 16MB memory chip consumes 10.5mW in the nap mode [11]. The static power per MB is computed: $p = \frac{10.5}{16} = 0.67$mW/MB. The dynamic power is estimated using the product of the data rate and the energy consumed to access one unit of data. When the RDRAM chip reaches its peak bandwidth 1.6 GB/second, its power consumption is 1325mW. The dynamic energy consumed to access one MB data (e) is estimated as $e = \frac{1325-10.5}{1.6\times1024} = 0.81$mJ/MB.

The disk utilization is defined as the ratio between request rate and disk bandwidth. The former is affected by the cache sizes while the latter varies with the rotation speeds. The disk's bandwidth is estimated using the reciprocal of the average disk's service time for one request. The service time includes three main parts: seek time (s_i), rotation delay, and transfer time. The rotation speed affects the latter two parts. At the peak speed R, the transfer rate is f_i and the rotation delay is h_i. We assume that the average transfer rate is proportional to the rotation speed and the rotation delay is inversely proportional to the rotation speed. Hence, at speed d_i, the service time is calculated as following.

$$s_i + \frac{R}{d_i}h_i + \frac{R}{d_i}\frac{q_i}{f_i} = s_i + \frac{R}{d_i}l_i, \quad l_i = h_i + \frac{q_i}{f_i} \tag{1}$$

3.2 Problem Formulation

In Figure 1, the request rates change from λ_i to $\lambda_i^{'}$ after the cache memory. We model the relationship between m_i and $\lambda_i^{'}$ assuming Zipf-like distribution of data popularity. Previous studies show that the distribution of data requests follows Zipf-like distributions for both web servers and multimedia servers [1, 3]. The probability of the j^{th} most popular MB of data is proportional to $\frac{1}{j^{\alpha_i}}$. Here, α_i is the Zipf exponent. Its value is smaller than one ($0 < \alpha_i < 1$) [1, 3]. Larger α_i indicates that the accesses concentrate on a smaller portion of data. We use $v(j)$ to represent the probability of accessing the j^{th} most popular MB of data. The value of $v(j)$ can be expressed as $v(j) = \frac{\Omega}{j^{\alpha_i}}$, where $\Omega = (\sum_{j=1}^{N_i}\frac{1}{j^{\alpha_i}})^{-1}$ and N_i is the data set size of the i^{th} disk. We model the behaviors of the LRU replacement by assuming the cache always stores the most popular data. This simple model is used because we focus on the relationship between memory sizes and disk speeds. More accurate models can be found in [12]. The cache hit ratio is calculated as $\sum_{j=1}^{m_i} v(j)$. The request rate after caching is:

$$\lambda_i^{'} = \lambda_i[1 - \textstyle\sum_{j=1}^{m_i} v(j)] \simeq \lambda_i[1 - (\frac{m_i}{N_i})^{1-\alpha_i}] \tag{2}$$

The value of $\sum_{j=1}^{m_i} v(j)$ can be estimated using $(\frac{m_i}{N_i})^{1-\alpha_i}$ because an integral can approximate summation: $\sum_{j=1}^{N_i}\frac{1}{j^{\alpha_i}} \simeq \int_0^{N_i}\frac{1}{x^{\alpha_i}}dx = \frac{1}{1-\alpha_i}N_i^{(1-\alpha_i)}$. Hence, $\sum_{j=1}^{m_i} v(j) = \frac{\sum_{j=1}^{m_i}\frac{1}{j^{\alpha_i}}}{\sum_{j=1}^{N_i}\frac{1}{j^{\alpha_i}}} = (\frac{m_i}{N_i})^{(1-\alpha_i)}$. When m_i increases, $[1 - \sum_{j=1}^{m_i} v(j)]$ decreases so the disk request rate $\lambda_i^{'}$ decreases. The value of m_i is not larger than N_i because the disk cache cannot be larger than the disk's capacity. Hence, $\frac{m_i}{N_i} \le 1$. As α_i grows, $[1-(\frac{m_i}{N_i})^{1-\alpha_i}]$ and $\lambda_i^{'}$ decrease. Larger α_i means that the requests concentrate on a smaller amount of data. More requests are served from the cache memory and the number of disk requests decreases.

Based on the above models, we express the problem of cache resizing and speed control for k disks as an optimization problem. The objective function is to minimize the power consumption of disks and their cache memory. Increasing request rates or reducing disk speeds enlarges the latency of each request and disk utilization. The constraints are (a) The sum of cache memory for each disk cannot exceed the installed memory size (M). (b) All disks' average utilization is smaller than the allowed utilization (U). The variables in the optimization problem are d_i and m_i. Other parameters are either constants or can be obtained at runtime.

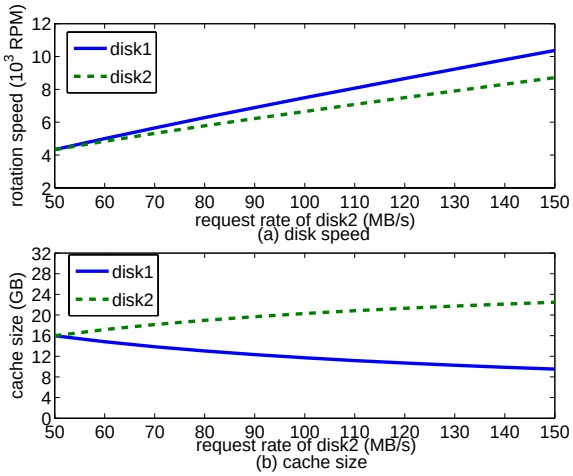


Figure 2: Request rates. The first disk's request rate is constant.

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{k}(ad_i^2 + bd_i + c) + u_i g d_i + pm_i + eq\lambda_i \\ & \textstyle\sum_{i=1}^{k} m_i \le M \\ & u_i = \lambda_i[1 - (\frac{m_i}{N_i})^{1-\alpha_i}](s_i + \frac{R}{d_i}l_i) \le U \end{aligned} \tag{3}$$

In the objective function, $(ad_i^2 + bd_i + c)$ and $u_i g d_i$ are the static power and the dynamic power of the i^{th} disk, respectively. The cache's static and dynamic power are represented by pm_i and $e\lambda_i$. The value of u_i is estimated using the product of disk request rate and disk service time. Their values are computed from (2) and (1). Enlarging m_i reduces u_i because the disk request rate decreases. Therefore, the disks' dynamic power decreases. When the decrease of the disks' dynamic power exceeds the increase of the cache memory's power, enlarging cache memory can save power.

3.3 Cache Sizes and Rotation Speeds

The formula (3) includes non-linear expressions. Although no explicit solutions are derived from (3), we conduct numeric analysis on proper cache sizes and disk speeds with different parameters. A two-disk array is used as an example to show the effects of request rate (λ) and disk utilization (U). Each disk stores a data set of 32GB with the Zipf exponent of 0.9. We use 32GB data sets to fit into a 33GB disk model [4]. Our method can be applied to larger data sets and the disks with larger capacity. Two disks have the same average seek time: $s_1 = s_2 = 3.4$ms. Their average rotation and transfer delays are same: $l_1 = l_2 = 3$ms. These numbers are obtained from practical workloads. We fix U as 50%. The first disk receives requests at 50MB/s while the second disk's request rate varies from 50MB/s to 150MB/s. Based on (3), we compute the proper disk speeds and cache

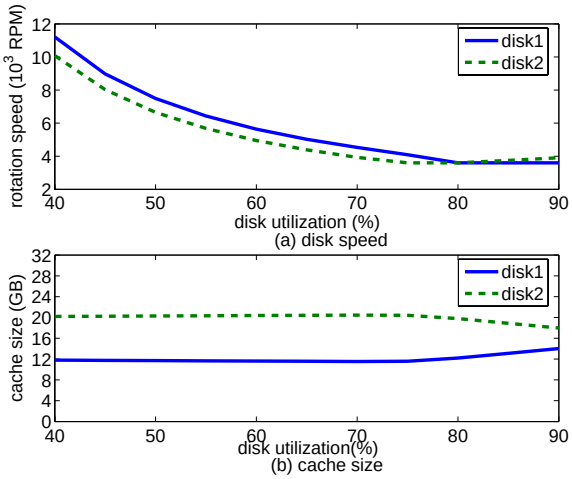


Figure 3: Cache size and rotation speed for different disk utilization. The first disk's request rate is constant.

sizes using *fmincon* in Matlab. The results are shown in Figure 2. The x-axes represent the request rate of the second disk. The y-axes are the disk rotation speeds and the cache sizes of two disks. When the request rate is 50MB/s, both disks use the same speed and cache size. As the second disk's request rate grows, the disk uses more cache memory to reduce disk accesses. As a result, the second disk runs at lower speeds than the first disk to save power. This result indicates that it is not most power-saving to make $\lambda_1^{'} = \lambda_2^{'}$. Figure 2 (b) shows that the ratio between the cache sizes is not equal to the ratio of the request rates. For example, at 150MB/s, the rate ratio is $\frac{50}{150} = 0.33$ while the ratio of the cache sizes is $\frac{9.5}{22.5} = 0.42$. Allocating cache based on the request rates cannot achieve best power savings.

The allowed disk utilization (U) also affects proper disk speeds and cache sizes. In Figure 3, we fix the request rates as 50MB/s and 100MB/s. The value of U varies from 40% to 90%. Lower utilization cannot be achieved due to the request rates. In Figure 3, the disk speeds drop rapidly with the increase of U while the cache sizes remain almost constant. The ratio of cache sizes remains $\frac{12}{20} = 0.6$ because the request rates are fixed. Increasing utilization allows the disks to use lower speeds and smaller caches. However, reducing disk speeds save more energy. When U becomes 80%, the disk speeds reach minimum. In Figure 3 (b), further enlarging U changes the ratio of the cache sizes. More memory is allocated to the first disk because its dynamic power decrease more rapidly as the cache memory increases.

Our method adjusts the cache memory and the rotation speed of each disk every 10 minutes. We choose this period length to achieve fast adaption and small optimization overhead. When each period starts, the proper cache sizes and speeds are computed by solving the optimization problem (3). The solver is from Lindo system Inc. written in C. The values of the parameters in current period are predicted using the information obtained during previous period. By recording the accesses to the cache memory and disks, we obtain the request rates, the request sizes, and the sizes of working sets. We also count the access number of each file to estimate the Zipf exponent of data popularity. The disk seek time and the rotation delay are estimated using approximation because they are invisible to the power manager. We estimate the disk seek time using the product of the average single-track seek time and the seek track number. The track number is computed by multiplying the block distance between two consecutive disk requests and the average block number per disk track. We use the time to rotate half circle as the approximate for the average rotation delay.

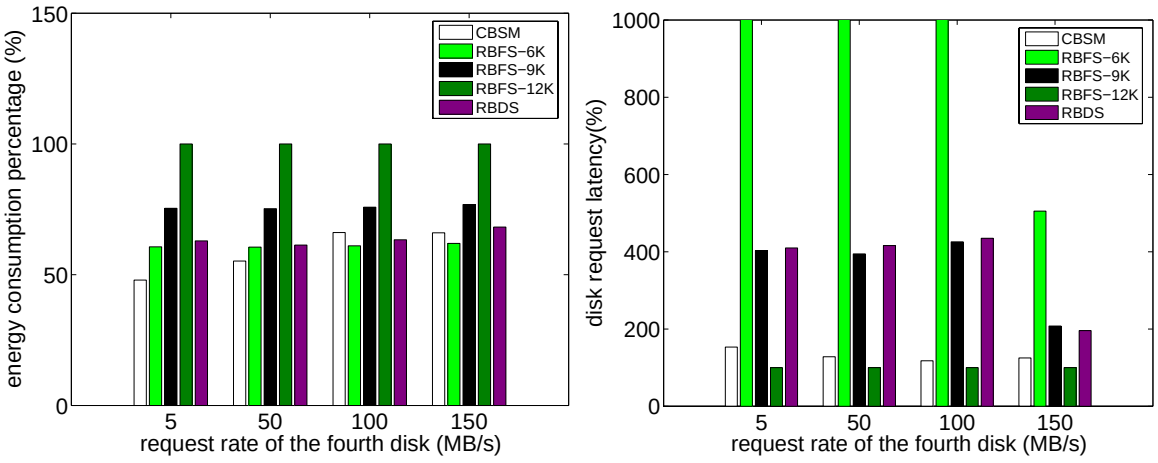
4. SIMULATION AND RESULTS

We simulate the power manager and the disk cache to evaluate the energy savings and the performance impact of our method and the methods in comparison. SPECWeb99 is used to create web server workloads on a real machine. We use SPECWeb99 because web servers are one of most important server systems and SPECWeb99 is the state-of-art generator of web requests. We revise Linux kernel in the web server to record the access traces to the cache memory. The traces are adjusted for different data rates. The traces are processed by the simulator of the disk cache with LRU replacement. Its output is the traces of the disk accesses used by the disk simulator Disksim [2] and the simulator of the power manager. The former obtains the information of disk performance, such as disk access latency. The latter manages both memory and disks for power savings.

We compare our method with four power management methods. Each method includes power management schemes for memory and disks. For memory management, we use rate-based (RB) resizing: the cache sizes are proportional to the request rates. The memory size step is 16MB because different types of DRAM can support this granularity. For disk management, one of the four methods uses dynamic speed (DS) control based on the same performance model used in CBSM. The speed step is 1200 RPM as in [4]. The other three methods use fixed disk speeds (FS): 6000 RPM, 9000 RPM, and 12000 RPM. The four methods are named using the combination of cache resizing methods and disk speed control methods: RBDS, RBFS-6K, RBFS-9K, RBFS-12K. For CBSM and RBDS, the disk speeds are chosen to limit the disk utilization to 20%.

The workload varies in many characteristics, such as request rate and working set. CBSM can adapt cache sizes and disk speeds to the workload. In contrast, the methods using fixed speeds or cache sizes need many times of simulation to determine the proper cache sizes and disk speeds for different workloads. The methods resizing cache only based on one workload characteristics may not choose the proper cache sizes and disk speeds. In the simulation, we change the request rate of one disk to show how our method can save energy with low latency by adjusting cache sizes and disk speeds. We use a four-disk array with 64GB memory. Each disk stores a 32GB data set. The first three disks receive requests at constant rates: 5MB/s, 50MB/s, and 100MB/s, respectively. The fourth disk's request rate varies from 5MB/s to 150MB/s. Every ten minutes, the methods adjust cache sizes and disk rotation speeds. The disks' transfer rate (f_i) is 80MB/s at R.

Figure 4 shows the energy consumption and the disk request latency of the methods in comparison. The energy consumption includes that of cache memory and four disks. The x-axes represent the request rate of the fourth disk while the y-axes denote the energy consumption percentage and the percentage of disk request latency. The percentage is based on RBFS-12K because this method always



(a) energy consumption

(b) average disk request latency

Figure 4: Energy consumption and disk request latency of a four-disk array. The first three disks' request rates are constant: 5, 50, and 100MB/s. The latency percentage over 1000% is not shown.

keeps the highest rotation speeds to achieve better performance. Higher bars indicate more energy consumption or longer request latency. In Figure 4 (a), compared to RBFS-12K, CBSM saves 34% to 53% energy when the fourth disk receives different workloads. The energy savings increase as the request rate decreases because the fourth disk can use less cache memory and lower rotation speeds. CBSM consumes less energy than RBFS-9K since using fixed and high speeds cannot save power from disks when they receive different workloads. Although RBFS-6K save more energy than CBSM at 100MB/s and 150MB/s, RBFS-6K causes longer delay to disk requests. In Figure 4 (b), CBSM only increases the average latency by 17% to 53% while the average latency of RBFS-6K exceeds 1000% of RBFS-12K. CBSM's latency is also much shorter than RBFS-9K because CBSM adjusts disk speeds according to different rates of the disks. At most request rates, CBSM consumes less power and causes shorter latency than RBDS due to the joint management of cache memory and disk speeds. Overall, CBSM consistently achieves both energy savings and low request latency across different workloads.

5. CONCLUSION

This paper presents the joint power management of memory and multiple disks by adjusting the cache size and the rotation speed of each disk. The method is formulated as an optimization problem with constraints of limiting disk utilization. The method solves the optimization problem at runtime to obtain the proper cache sizes and rotation speeds. The simulation results show that our method can adapt to the workload variation and achieves both energy savings and low disk access latency for multiple disks. This paper can be extended to consider latency constraints and other cache replacement algorithms.

6. REFERENCES

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *INFOCOM*, pages 126–134, 1999.

[2] J. S. Bucy and G. R. Ganger. The disksim simulation environment version 3.0 reference manual. http://www.pdl.cmu.edu/DiskSim/, 2003.

[3] C. Griwodz, M. Bar, and L. C. Wolf. Long-term movie popularity models in video-on-demand systems or the life of an on-demand movie. In *ACM International Conference on Multimedia*, pages 349–357, 1997.

[4] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. DRPM: Dynamic Speed Control for Power Management in Server Class Disks. In *International Symposium on Computer Architecture*, pages 169–181, 2003.

[5] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 3 edition, 2003.

[6] H. Huang, P. Pillai, and K. G. Shin. Design and Implementation of Power-Aware Virtual Memory. In *USENIX Annual Technical Conference*, pages 57–70, 2003.

[7] M. Kandemir, S. W. Son, and G. Chen. An evaluation of code and data optimizations in the context of disk power reduction. In *ISLPED*, pages 209–214, 2005.

[8] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis. Power Aware Page Allocation. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 105–116, 2000.

[9] Y.-H. Lu, E.-Y. Chung, T. Simunic, L. Benini, and G. D. Micheli. Quantitative Comparison of Power Management Algorithms. In *Design Automation and Test in Europe*, pages 20–26, 2000.

[10] E. Pinheiro and R. Bianchini. Energy Conservation Techniques for Disk Array-based Servers. In *International Conference on Supercomputing*, pages 68–78, 2004.

[11] Rambus Company. 128Mb RDRAM Split Bank Architecture Advanced Information, March 2003.

[12] G. S. Rao. Performance analysis of cache memories. *Journal of the ACM*, 25(3):278–395, 1978.

[13] E. Shriver. *Performance Modeling for Realistic Storage Devices*. PhD thesis, New York University, May 1997.

[14] T. Simunic, L. Benini, P. Glynn, and G. D. Micheli. Dynamic Power Management for Portable Systems. In *International Conference on Mobile Computing and Networking*, pages 11–19, 2000.

[15] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: Helping disk arrays sleep through the winter. In *ACM Symposium on Operating Systems Principles*, October 2005.

[16] Q. Zhu, A. Shankar, and Y. Zhou. PB-LRU: A Self-tuning Power Aware Storage Cache Replacement Algorithm for Conserving Disk Energy. In *International Conference on Supercomputing*, pages 79–88, 2004.