Saving Energy in Video Servers by the Use of Multispeed Disks

Manjong Kim, *Student Member, IEEE,* and Minseok Song, *Member, IEEE*

***Abstract*—Energy consumption is an important issue in data centers, and disks use a significant proportion of the total energy. A promising approach to reducing disk energy consumption is to use multispeed disks with lower rotational speeds, and allowing disks to run slowly when workloads are light can reduce their large contribution to the power used by video servers. We formulate the prerequisites for speed reductions, and derive an energy model constrained by retrieval period. We then analyze the relationship between retrieval period, buffer size, and disk speed, and examine the effect of buffer allocation on energy consumption. We then propose a new video data retrieval scheme in which the retrieval period and the speed of each disk is dynamically changed to adjust disk bandwidth utilization, with the aim of allowing disks to run at lower speeds while guaranteeing jitter-free speed transitions. Experimental results show that our scheme achieves appreciable energy savings under all workloads. They also reveal that increasing the number of available speeds reduces energy consumption but the benefits gradually tail off.**

***Index Terms*—Low-power systems, video-on-demand (VoD) systems, video storage servers.**

I. Introduction

THE INCREASING number of Internet-based services which involve web hosting and e-commerce has lead to a proliferation of data centers that house the growing server infrastructure of the Internet. The provision of storage is a major function of data centers and demand for storage is increasing by 60% annually [1], [2]. In particular, the recent growth of video services such as digital libraries, education-on-demand, distance learning, user-created content, and video-on-demand (VoD) has greatly increased the amount of storage provided by data centers greatly [3]. For example, it has been estimated that YouTube serves 40 million videos every day, which amounts to 200TB of data [4].

The energy consumption of servers is a significant problem. Energy User News [5] recently suggested that the power requirements of a typical data center is now 150–200 W/ft^2 and will soon be 200–300 W/ft^2. This represents a significant cost to service providers. For instance, a medium-sized 30 000 ft^2 data center typically requires 15 MW, which corresponds to an annual bill of $13 million [2], [6]. Overall power requirements account for about 19% of a data center's total cost of ownership [7]. The environmental cost is also high [1], [2], and this can also impact the public perception of organizations running and using data centers.

The power consumed in a server naturally ends up as heat, and it has been shown that running at 15 °C above ambient can double the failure rate of a disk drive [2], [6]. But cooling systems designed for high heat densities are expensive and the cooling system itself adds significantly to the power cost of a data center. Thermal constraints also make it difficult to uprate servers.

Due to the combination of high capacity and bandwidth requirements, video servers are typically built as a redundant array of independent disks (RAIDs), which may consist of hundreds of disks. Storage devices are among the components in a server, and a recent report has indicated that they take 27% of the total power [2], [6], [8]. A large disk array may actually use more energy than the rest of a system, depending on the array size [6]. For instance, it has been reported that disk drives use 86% of the energy required by a typical EMC Symmetrix 3000 storage system [9]. This problem is being exacerbated by the arrival of faster disks which need more power.

Disks of modern design make some attempt to save energy by the use of multiple power modes [2], [6], [8]: in active mode the head is reading or writing data, in seek mode the head is seeking, in idle mode the disk spins at full speed but is processing no requests for data, and in standby, a disk stops spinning completely but consumes much less energy than in any other mode. Returning from standby to active mode naturally involves spinning up the disk, and so the energy saved while the disk is in standby mode should be greater than the energy needed to spin it up again; we call the shortest idle period which justifies the energy cost of spinning up again the break-even time.

A commonly used method of energy saving is to transition a disk into standby mode after it has been idle for a while [10]–[12]. If a request arrives while a disk in standby mode, then it immediately transitions to active mode to service the request. But this method of power saving is inapplicable to most servers because the time between consecutive disk requests generated

Manuscript received December 2, 2010; revised May 30, 2011; accepted August 13, 2011. Date of publication October 3, 2011; date of current version April 2, 2012. This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science, and Technology, under Grant 2009-0065248, in part by the IT Research and Development Program of MKE/KEIT, under Grant 10035243 (Component Based Design Theory and Control Kernel for CPS), and in part by the Inha University Research Grant. This paper was recommended by Associate Editor G. G. (Chris) Lee.

The authors are with the School of Computer and Information Engineering, Inha University, Incheon 402-751, Korea (e-mail: kmjlove130@hotmail.com; mssong@inha.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2011.2170112

by a typical server workload is too short [13]. It is particularly unsuitable for VoD systems; video data needs to be read continuously while a movie is playing. Allowing disk to enter standby mode causes playback to be distorted or paused due to violation of the timing constraints of the video data [14]. We refer to this phenomenon as jitter.

Reducing the speed at which a disk spins reduces its power consumption. Gurumurthi *et al.* [8], [15] have suggested that multispeed disks could save a lot of power consumption in servers. A disk running slowly would be able to support a small number of video requests, whereas a single-speed disk servicing the same workload needs to spin at full speed. In addition, disks able to change speed while spinning would require less energy and shorter time to shift between different speeds than a single-speed disk requires to shift from standby to active mode [8], [15]. Multispeed disks are now on the market (e.g., from Hitachi [16] and Sony [17]) and are being used to build energy-efficient storage systems (e.g., Nexsan storage servers [18]). This makes it essential to develop effective power-saving techniques for multispeed disks.

We propose a new energy-aware data retrieval and buffer management scheme for video servers which use multispeed disks. We review work on this topic in Section II, and in Section III, we present our system model. In Section IV, we introduce a resource utilization model which will allow us to examine how the retrieval period and the rotational speed affect disk and buffer utilizations. We analyze the relationship between retrieval period, buffer size, and speed, and then examine how the retrieval period affects energy consumption in Section V. We go on to propose a new data retrieval and buffer management scheme in which the retrieval period and the speed of each disk can be dynamically changed to match disk utilization, with the aim of minimizing overall energy consumption while providing jitter-free speed transitions in Section VI. We assess the proposed scheme through simulations in Section VII. We suggest extensions of our scheme in Section VIII and conclude this paper in Section IX.

II. Related Work

A. Disk Power Management for Data Centers

Several techniques have been proposed to reduce the energy consumption of storage devices in servers, most of which involve extending the periods during which disk can be put into standby mode. Pinheiro *et al.* [19] used load concentration techniques to do this, while Zhu *et al.* [2] introduced a caching scheme that selectively retains data in memory so that some of the disks in an array can stay in standby mode for longer periods.

Redundancy is necessary in all practical storage systems, and this is exploited by several energy conservation schemes. Pinheiro *et al.* [20] redirected data requests to active disks to save energy on standby disks, and Yao *et al.* [21] proposed a redundancy-based hierarchical I/O cache architecture called RIMAC to reduce the energy consumption of disks.

To reduce energy usage when server workloads are intense, Gurumurthi *et al.* [8], [15] have proposed a speed-changing scheme called dynamic rotations per minute (DRPM). There have been other schemes, which involve monitoring the length of the queue of disk requests and then changing disk speeds to keep response times within a predefined range. For example, Zhu *et al.* [6] proposed a flexible disk placement architecture called Hibernator which is designed to reduce disk speeds as often as possible. Son *et al.* [22] proposed a compiler-directed prefetching scheme which allows the disks more opportunity to stay in low-speed mode, and Xie [13] have put forward a data placement technique in which loads are distributed in such a way that some disks can run at low speeds.

All of these techniques try to resolve disk power issues in various ways, but they cannot provide the real-time guarantees essential for video workloads making them unsuitable for video servers. Song [14] has presented a scheme to permit disks to enter standby mode in video servers, but it requires data to be replicated between disks. Its main aim is to extend the length of time for which a proportion of the disks storing this replicated data can stay in standby mode, and the possibility of multispeed disks is not considered. In fact, we are not aware of any previous systematic approach to the design of video servers with multispeed disks.

B. Disk Power Management for Mobile Devices

Disk energy saving techniques for multimedia workloads on mobile devices have received more attention. Cai *et al.* [25] used data buffers to create long idle periods during the playback of MPEG videos. Pettis *et al.* [26] constructed analytical models of disk energy consumption during streaming. Won *et al.* [12] used prefetching schemes to extend the length of time that a disk stays in standby mode. Go *et al.* [43] presented a data prefetching scheme that reads frames into buffer on a just-in-time basis to provide real-time video playback. Kim *et al.* [44] presented a data prefetching scheme to make effective use of a combination of dynamic random access memory (RAM) and Flash memory to store prefetched data. Rao *et al.* [23] investigated the problem of choosing disk speeds, and Liu *et al.* [24] investigated I/O speed setting strategies for multimedia workloads.

To allow a disk to enter standby mode, all of these schemes prefetch data into a buffer so that further requests can be handled from memory without accessing the disk. They were developed for portable media players that use a single disk, whereas our paper focuses on multiple disks with video server workloads. In addition, our scheme uses multispeed disks which have not been seriously considered for mobile devices.

III. Background

A. Retrieval Scheduling

To support the periodic retrieval of video streams efficiently, video servers usually use round-based scheduling: time is divided into equal-sized periods, called rounds, and each client is served once in each round [27]. For example, to serve a stream at 1.5 Mb/s with a round length of 2 s, the server needs to read 3 Mb of data during every round. Since the data transfer rate of a single disk is significantly higher than the playback rate of a single video stream, each disk in a video server is

TABLE I
PERFORMANCE AND POWER CONSUMPTION AT EACH RPM LEVEL

RPM	2880	3960	5040	6120	7200
Transfer rate (MB/s)	24.56	33.77	42.98	52.19	61.4
Rotational delay (ms)	5	3.6	2.86	2.35	2
Idle power (W)	4.08	5.11	6.48	8.17	10.2
Active/seek power (W)	7.38	8.41	9.78	11.47	13.5

able to provide the data for multiple streams. Disk bandwidth utilization can then be defined as the ratio of the total service time spent retrieving all the streams during a round to the round length, and this utilization must be less than or equal to 1 [27].

Disk scheduling determines the order in which blocks belonging to different video streams are retrieved from a disk during a round. There are three categories of scheduling algorithms: round-robin scheduling, scan scheduling, and the grouped sweeping scheme (GSS) [28], [29]. In round-robin scheduling, the order in which clients are served does not vary from one round to another; this results in excessive seek time and poor disk utilization. In scan scheduling, the disk heads scan steadily back and forth across the platters and retrieve a required blocks as they pass over them; this minimizes the total seek time. GSS partitions a round into groups and scan scheduling is then used in each group. We use scan scheduling because it has the best performance in terms of disk throughput and widely used [28], [29].

B. System Model

The disk blocks belonging to a file may be scattered across a disk or stored contiguously. Reading scattered blocks requires a lot of seeks, so files for video applications are usually stored contiguously [27], [28]. However, a video file may be partitioned into chunks and striped over multiple disks for load balancing [30]. But striping reduces scalability, throughput, and reliability, and introduces delays [31], [32], and we will only consider striping as a possible extension of our scheme in Section VIII.

Several components of a disk use power. These include the spindle motor which spins the platters, the actuator that makes the head move, the electrical components involved in data transfer, as well as other circuitry [15], [23]. The power required by the spindle motor typically has a linear or quadratic relationship with speed [6], [15], [23]. Even in idle mode, this motor typically consumes about 80% of the total power used by the disk.

The DRPM scheme requires a disk capable of running at an essentially arbitrary speed [15], [23]. But we will consider disks that can run at a discrete set of speeds, because this is what is presently offered by commercial multispeed disks [6]. However, whereas current commercial multispeed disks have only two speeds [16], [17], we are going to consider disks with a larger number of speeds, because it appears likely that such disks will soon be available [2], [6]. Our power model is based on the Hitachi 7K400 disk [16] and we use a quadratic curve to model the power required by the spindle motor [15], as shown in Table I.

C. Facilitating Reduced Disk Speeds

We have already described how a single-speed disk supplying video data that is not replicated has no opportunity to enter standby mode. But a multispeed disk can service some level of requests at reduced speeds without transitioning to full speed, thus saving energy. To reduce disk power consumption, it is desirable to reduce disk speeds as far as possible, but disk bandwidth utilization cannot exceed 1. To satisfy these two constraints simultaneously, we first calculate disk bandwidth utilization at each speed and then determine the lowest speed that keeps the disk bandwidth utilization below 1; finally, we determine the power consumption at this lowest speed.

It may take a significant amount of time to transition from one speed to another (e.g., 7 s for the Hitachi drive [16]). During the transition period requests for data cannot be handled, leading to violation of the timing constraints of the video data termed jitter, and it is therefore essential to ensure that speed transitions in video server disks do not cause jitter.

IV. BASIC CONCEPT AND RESOURCE UTILIZATION MODEL

A. Basic Concept

On receiving a playback request from a client, a video server reads the required data from a disk, and then temporarily stores that data in a buffer until it is served. In round-based scheduling, the round length plays an important role in determining what disk bandwidth and buffer size are required [28], [29]. Increasing the round length reduces the disk bandwidth utilization because the seek time is amortized over more data, but this increases buffer requirements because a lot of buffer space is needed to store the data retrieved during a round [29].

Because the bandwidth utilization of a disk effectively determines its speed level, our scheme adjusts the round length dynamically with the aim of minimizing the overall energy consumption. Extending the round length reduces disk bandwidth utilization, leading to a reduction in disk speeds. However, a longer round means that a larger buffer is needed to store the data retrieved during each round. So judicious buffer allocation methods are needed because buffer space is limited and must be shared between disks. We will now describe how round length and speed affect disk bandwidth and buffer utilization.

B. Disk Bandwidth and Buffer Utilization

We will consider a disk array with N_d disks. (Table II summarizes the important symbols used in this paper.) Suppose that client i requests a video stream that has a playback rate of b_i bits/s. We can partition the clients into N_d groups: $G_1, \ldots, G_{N_d}$, where the clients in group G_k receive video streams from disk k. We define a list of N_r round lengths, $S = \{R_j | R_j = sr + u(j-1), (j = 1, \ldots, N_r)\}$, where sr is the length of the shortest round, u is the increment of round length, j is the round length index, and lr is length of the longest round length, so $lr = sr + u(N_r - 1)$. This list is sorted into ascending order.

We use a typical seek time model in which a constant seek time T_s is required for one read of contiguous data [27]–[29]. Although the disk speed does not affect the seek time, it does change the rotational delay and the data transfer rate [6], [15]. If a disk is running at speed level m, ($m = 1, \ldots, N_s$), where N_s is the number of levels available, we will write the rotational delay as $T_d(m)$ and the data transfer rate as $r(m)$. Obviously, increasing the speed level m decreases $T_d(m)$ but increases $r(m)$.

Let us now see how buffer and disk bandwidth requirements depend on the speed level m and the round length index j. To provide real-time video playback, the total time spent reading all video streams during a round must not exceed the round duration [27]–[29], and the amount of data that must be read during a round R_j for client i is $R_j b_i$ in order to keep up with the playback rate. Reading the data needed for client i during this round incurs a seek and rotational delay overhead of $T_s + T_d(m)$ and a reading time of $\frac{R_j b_i}{r(m)}$: thus the total time taken to serve client i is $T_s + T_d(m) + \frac{R_j b_i}{r(m)}$ seconds.

Recall that the disk bandwidth is the ratio of total service time to round length; thus we can express the disk bandwidth utilization $U_k^d(j, m)$ of disk k running at speed level m, when the round length index is j, as follows:

$$U_k^d(j, m) = \sum_{i \in G_k} \frac{T_s + T_d(m) + \frac{R_j b_i}{r(m)}}{R_j}. \quad (1)$$

Let B be the total buffer size. To achieve synchronization, double-buffering is used for scan scheduling [27], [33], and buffer utilization is not dependent on disk speed. Therefore, serving client i increases the buffer utilization by $2\frac{R_j b_i}{B}$, where B is the size of the buffer, and we can go on to obtain the buffer utilization $U_k^b(j)$ for disk k when the round length index is j, as follows:

$$U_k^b(j) = 2 \sum_{i \in G_k} \frac{R_j b_i}{B}. \quad (2)$$

C. Resource Requirements for Jitter-Free Speed Transitions

To achieve a jitter-free speed transition, we must prefetch the data that will be consumed during the speed transition period, which requires contingent disk bandwidth and buffer space [34]. Let T_t be the transition time between two different speed levels, and let cr_k be the current round length of disk k. The server not only needs to prefetch enough data for the transition but the data that will be consumed during the round immediately after the transition period also needs to be prefetched, if jitter is to be avoided. Since it is impossible to know the new round length after a speed change, we reserve enough disk bandwidth for the shortest round length sr, because this requires the highest bandwidth as we can see from (1). We assume that prefetching starts N_p rounds before the beginning of the actual transition as shown in Fig. 1.

A speed change may require a total of $T_t + sr$ seconds worth of data to be prefetched for each client. Prefetching starts N_p rounds before the speed transition, so the amount of data needed for client i, which is $(T_t + sr)b_i$, can be prefetched across N_p rounds, as shown in Fig. 1. We, therefore, reserve

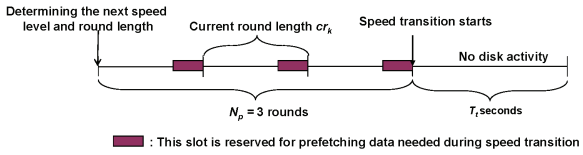

Fig. 1. Example of prefetching operation.

the contingent disk bandwidth for disk k, C_k^d, which can be expressed as follows:

$$C_k^d = \sum_{i \in G_k} \frac{T_s + T_d(m) + \frac{(T_t + sr)b_i}{r(m)}}{N_p cr_k}. \quad (3)$$

Buffer space must be available to store the prefetched data. Because we cannot predict the round length after a speed change as we have already noted, we reserve enough buffer space for the longest round length. This means that buffer space of $(T_t + lr)b_i$ is needed for client i, so the additional buffer space, C_k^b that must be reserved for disk k can be expressed as follows:

$$C_k^b = \sum_{i \in G_k} (T_t + lr)b_i. \quad (4)$$

To allow for all possible patterns of speed changes across a disk array, we need to reserve the full contingent disk bandwidth for every disk, but this bandwidth is only needed for a speed transition. It is, however, wasteful to reserve the full amount of contingent buffer space for every disk, because all the disks are very unlikely to change their speeds simultaneously. We thus limit the number of simultaneous speed changes.

V. Pattern of Energy Consumption

We will now examine how energy consumption depends on the round length index j and the speed level m. We have already noted that the contingent disk bandwidth C_k^d is only needed to prefetch data when speed change occurs. Prefetching data increases the energy consumption before the speed change, but these are the data that would otherwise have been read during the speed transition, so prefetching has no overall impact on the total energy consumption. If a disk is rotating at speed level m, then $P_s(m)$ is the power required during the seek phase, $P_a(m)$ is the power required while the disk is reading or writing, and $P_i(m)$ is the power consumed when the disk is rotating. We can now formulate some energy properties for a disk k.

1) The total seek time during R_j is $\sum_{n \in G_k} T_s$. When the speed level is m, the energy required to perform seeks during R_j, denoted by $E_k^s(j, m)$, is $\sum_{i \in G_k} T_s P_s(m)$.
2) The total data transfer time during R_j is $\sum_{i \in G_k} \frac{R_j b_i}{r(m)}$, so the energy required for reading data during R_j, denoted by $E_k^a(j, m)$, is $\sum_{i \in G_k} \frac{R_j b_i}{r(m)} P_a(m)$.
3) If no disk activity is taking place, the disk is rotating, and if the disk is waiting for data to arrive underneath the head to start reading operation, it is rotating without reading or seeking. In both cases, the disk requires a

TABLE II
IMPORTANT SYMBOLS USED IN THIS PAPER

Symbols	Meaning
N_d	Number of disks in a disk array.
N_s	Number of speed levels.
N_r	Number of round lengths.
N_l	Maximum number of simultaneous speed changes allowed.
N_p	Number of rounds during which prefetching is allowed.
sr	Shortest round length.
lr	Longest round length.
b_i	Bit-rate of video that client i requests.
B	Total buffer size.
T_s	Typical seek time.
T_t	Speed transition time.
$T_d(m)$, $r(m)$	Rotational delay and transfer rate when speed level m is selected, respectively.
$P_s(m)$, $P_a(m)$, $P_i(m)$	Seek, active, and idle power for speed level m, respectively.
$E_k^s(j,m)$, $E_k^a(j,m)$, $E_k^i(j,m)$	Seek, active, and idle energy of disk k during R_j when the round length index is j and the speed level is m, respectively.
S	Set of round lengths R_js; $S = \{R_j \mid R_j = sr + u(j-1), (j = 1, \ldots, N_r)\}$, where j represents the round length index.
$U_k^d(j,m)$	Disk bandwidth utilization for disk k when j is selected as the round length index, and m as the speed level.
$U_k^b(j)$	Buffer utilization for disk k when j is selected as the round length index.
$P_k(j,m)$	Power requirement for round length index j and speed level m.
C_k^d, C_k^b	Contingent disk bandwidth and buffer utilizations imposed on disk k for jitter-free speed transition, respectively.
cr_k	Current round length for disk k.
F_k	A list of feasible round length index and the speed level pairs for disk k.
Z_k	$Z_k = \{(j,m) \mid R_j \in S$ and $U_k^d(j,m)) \le 1 - C_k^d$, and m is the current speed level$\}$.
O_k	$O_k = \{(j,m) \mid R_j \in S$ and $U_k^d(j,m) \le 1 - C_k^d$, and m is not the current speed level$\}$.
L_k	$Z_k \bigcup O_k$.
H_k	A subset of L_k, where elements requiring more buffer space and power than any other elements in L_k are excluded.
I_k	Parameter indicating whether the m^{th} element in L_k is in H_k or not.
N_k^e	Number of elements in the list L_k.
pf	The amount of prefetching buffer space for N_l most heavily loaded disks.
$P_{k,m}^s$	The saving in power when the m^{th} element in L_k is selected.
$B_{k,m}^r$	The buffer requirement when the m^{th} element in L_k is selected.
$Y_{k,m}$	If the speed level of the m^{th} element in L_k is the same as the current speed level, then $Y_{k,m} = 0$; otherwise, $Y_{k,m} = 1$.
$X_{k,m}$	Parameter indicating whether the m^{th} element in L_k is selected or not.

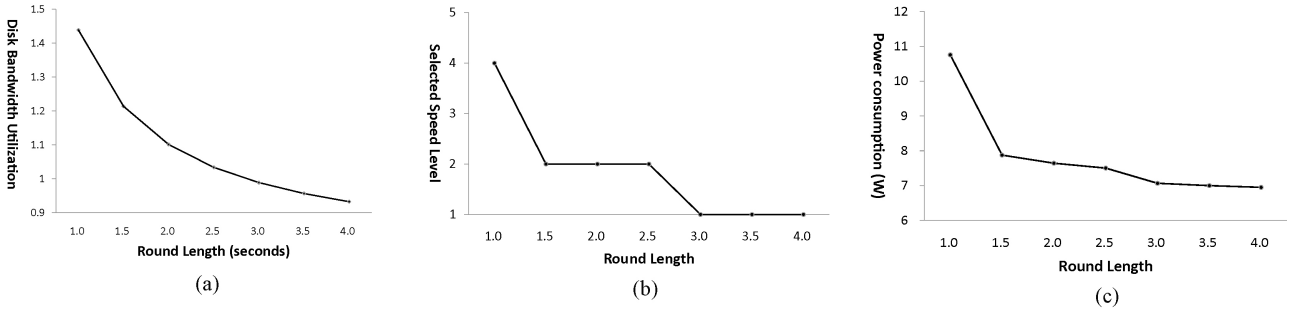

Fig. 2. Disk bandwidth utilization, selected speed level, and power consumption against round length when 50 clients request 3 Mb/s video streams. (a) Disk bandwidth utilization $U_k^d(j, 1)$. (b) Selected speed level. (c) Power consumption.

power of $P_i(m)$. We calculate the total idle time during a round of length R_j by subtracting the seeking and reading times from R_j. We denote the energy consumed during R_j in this idle mode, $E_k^i(j,m)$, which can be expressed as follows:

$$E_k^i(j,m) = (\overbrace{\underbrace{(R_j}_{\text{round length}} - \underbrace{\sum_{i \in G_k} (T_s + \frac{R_j b_i}{r(m)})}_{\text{total read and seek time}}}^{\text{total idle time}}) P_i(m).$$

We can then express the total power requirement, $P_k(j, m)$, in terms of the round length index j and the speed level m as follows:

$$P_k(j, m) = \frac{E_k^s(j,m) + E_k^a(j,m) + E_k^i(j,m)}{R_j}. \quad (5)$$

From (1), we observe that increasing the round length reduces the disk bandwidth utilization at every speed. This reduction may be sufficient to allow the disk to operate at a lower speed, saving energy. To illustrate this possibility, we examine how the disk bandwidth utilization at the lowest speed level, the selected speed level, and energy consumption depend on the round length when 50 clients request a video stream of 3 Mb/s using the disks in Table I. The results, shown in Fig. 2, demonstrate that increasing round length reduces the disk bandwidth utilization, which allows the speed level to be reduced, saving power.

VI. Adaptive Adjustment of Round Length and Speed

A. Problem Formulation

We have just seen that a longer round allows a power reduction. But (2) tells us that the buffer space requirement increases in proportion to the round length. This means that judicious round length selection is important to balance power consumption and buffer space. To meet this end, we now propose an adaptive round length adjustment scheme in which the round length and the disk speed are chosen with the aim of minimizing the total power consumption.

We have seen that a change of speed requires data prefetching, which involves a lot of buffer space. But, because the buffer is shared between disks, we can reduce the need for buffer space by limiting the number of simultaneous speed changes. Let N_l be the maximum number of disks which can change speed simultaneously. Because it is hard to know which disks will change their speeds, we reserve an amount of prefetching buffer, pf, which is enough for the N_l most heavily loaded disks to change speed. To determine pf, we sort all k disks into nonincreasing order on the values of C_k^b, where d_m is the index of the disk with the mth highest value of C_k^b. Thus

$$pf = \sum_{m=1}^{N_l} C_{d_m}^b.$$

Our goal is to find one pair of round and speed levels for each disk that minimizes the total disk power consumption. Recall that a disk's bandwidth utilization cannot exceed 1, and that a speed change requires a contingent disk bandwidth C_k^d to be reserved for each disk k. Thus, we can define a list F_k of feasible round length index and the speed level pairs for disk k as follows:

$$F_k = \{(j,m)|\ R_j \in S \text{ and } U_k^d(j,m) \le 1 - C_k^d\}.$$

To reduce the time needed to select one element in each list, we can eliminate some pairs that will be never selected. Any pair that corresponds to both a larger buffer requirement and a greater power consumption than any other pair in F_k can be removed, if selecting this pair changes the disk speed. For example, consider a list of round length index and the speed level pairs, $F_k = \{(3,2),(4,3),(5,4)\}$, where the current speed level is 4. Here, $(4,3)$ can be removed because it requires a larger buffer requirement and more power than $(3,2)$. But $(5,4)$ remains even though it has the largest buffer and power requirement, because selecting this pair does not change the disk speed. We achieve this removal of hopeless pairs by dividing F_k into two lists Z_k and O_k, defined as follows: $Z_k = \{(j,m)|\ R_j \in S$ and $U_k^d(j,m) \le 1 - C_k^d$, and m is the current speed level$\}$; and $O_k = \{(j,m)|\ R_j \in S$ and $U_k^d(j,m) \le 1 - C_k^d$, and m is not the current speed level$\}$. Using these lists, we can prune the list of the candidate pairs in the following steps.

1) Remove pairs from list O_k that correspond to larger buffer and power requirements than any other pairs in O_k.
2) Remove pairs from list Z_k that correspond to larger buffer and power requirements than any other pairs in Z_k.
3) Establish a new list L_k for disk k, defined as follows: $L_k = Z_k \bigcup O_k$, containing N_k^e pairs. L_k is sorted in nondecreasing order of power requirement.

We must select one pair consisting of a round length index and a speed level from L_k. To represent this selection, we define the binary variables $X_{k,m}$, ($k = 1,\ldots,N_d$ and $m = 1,\ldots,N_k^e$) as follows: if the mth element in L_k is selected, then $X_{k,m} = 1$, otherwise, $X_{k,m} = 0$. To reduce the buffer space needed for prefetching, we limit the number of disks that can change speed to N_l. To express this limitation, we introduce the binary variables $Y_{k,m}$, ($k = 1,\ldots,N_d$ and $m = 1,\ldots,N_k^e$) as follows: if the speed level of mth element in L_k is the same as the current speed level, then $Y_{k,m} = 0$, otherwise, $Y_{k,m} = 1$. Let $B_{k,m}^r$ be the buffer requirement when the mth element in L_k is selected. Then we have the following constraints.

1) $\sum_{m=1}^{N_k^e} X_{k,m} = 1$ and $X_{k,m} \in 0,1$, ($k = 1,\ldots,N_d$ and $m = 1,\ldots,N_k^e$): For each disk, only one pair consisting of a round length index and a speed level, can be selected.
2) $\sum_{k=1}^{N_d}\sum_{m=1}^{N_k^e} X_{k,m}Y_{k,m} \le N_l$: This ensures there are no more than N_l simultaneous speed changes.
3) $\sum_{k=1}^{N_d}\sum_{m=1}^{N_k^e} X_{k,m}B_{k,m}^r \le 1 - \frac{pf}{B}$: The total space, including the space needed for prefetching, must not exceed the total buffer size B.

Let $P_{k,m}^s$ be the energy saved by selecting the mth element in L_k instead of the element in $L_1 \bigsqcup \ldots \bigsqcup L_{N_d}$ that consumes the most energy. Our goal is now to find the element in L_k, ($k = 1,\ldots,N_d$) that saves the most energy without causing the buffer to overflow, while limiting the number of disks that change speeds to N_l. We can formally describe this round length and speed selection problem ($\mathcal{RSSP}$) as follows:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{k=1}^{N_d}\sum_{m=1}^{N_k^e} X_{k,m}P_{k,m}^s \\ \text{subject to} \quad & \textstyle\sum_{k=1}^{N_d}\sum_{m=1}^{N_k^e} X_{k,m}Y_{k,m} \le N_l \\ & \textstyle\sum_{k=1}^{N_d}\sum_{m=1}^{N_k^e} X_{k,m}B_{k,m}^r \le 1 - \frac{pf}{B} \\ & \textstyle\sum_{m=1}^{N_k^e} X_{k,m} = 1, (k = 1,\ldots,N_d) \\ & X_{k,m} \in 0,1, (k = 1,\ldots,N_d \text{ and } m = 1,\ldots,N_k^e). \end{aligned}$$

$\mathcal{RSSP}$ is a variant of the multidimensional multiple-choice knapsack problem (MMKP) in which there is a set of objects and a knapsack with a capacity vector. Each object consists of a set of items, each of which has a weight vector and a profit. The problem is to select exactly one item from each object so as to maximize the total profit while satisfying the capacity constraints [35]. We can relate the $\mathcal{RSSP}$ to MMKP by considering each disk in an array as an object, and regarding each pair of a round length and a speed as items in that object. This is a variant of the multiple-choice knapsack problem (MCKP) which has only one constraint.

Since MMKP is NP-hard, $\mathcal{RSSP}$ is also an NP-hard problem. We have effectively formulated $\mathcal{RSSP}$ as an integer linear programming (ILP), so we can obtain the optimal solution using an ILP solver such as the lp_solve program [36]. But this may require a lot of computation, which may not be acceptable in our case. For this reason, we have developed

a heuristic algorithm for $\mathcal{RSSP}$, which runs in polynomial time.

B. Heuristic Algorithm

Our heuristic algorithm is divided into two phases. In the first phase, it tries to find the best configuration that satisfies the buffer requirement alone. If the number of speed changes exceeds N_l, then it enters the second phase to reduce the number of speed changes. We will outline each phase as follows.

1) By considering the buffer constraint alone, the problem is reduced to be a variant of the MCKP. This is NP-hard [35], but greedy algorithms generally show good performance on MCKP [35], [46], so we use a greedy approach. We maintain parameters representing the ratio of power saving to buffer space and choose pairs with the best ratios.
2) To reduce the number of speed changes, we favor the selection of pairs for which $Y_{k,m} = 0$, meaning that no speed changes are required. We also use a greedy method to find the pair in Z_k with the lowest power consumption, which also meets the buffer constraint.

We will look at our algorithm in more detail. We first define the variables Q_k, $(k = 1, \ldots, N_d)$ which signify that the Q_kth pair in the list of L_k is selected for disk k. Q_k is initialized to N_k^e, which corresponds to the lowest power consumption but the largest buffer requirement. The value of Q_k may then be reduced to meet the buffer constraint

$$\sum_{k=1}^{N_d} \sum_{m=1}^{N_k^e} X_{k,m} B_{k,m}^r \leq 1 - \frac{pf}{B}.$$

In the first phase, we try to find the configuration that leads to the lowest power consumption, subject to the buffer constraint but irrespective of the number of speed changes. Note that several pairs with the current speed level may be included in L_k, even though they require more buffer space and more power than other pairs in L_k. These pairs cannot be selected during the first phase [35], so we exclude them from L_k to create a new set H_k for disk k. We then introduce a new indicator variable $I_{k,m}$ as follows: if the mth element in L_k is in H_k, then $I_{k,m} = 1$, otherwise $I_{k,m} = 0$.

We apply a well-known greedy method [46] to the first phase, and define the parameters $W_{k,n}$ for disk k, as follows:

$$W_{k,n} = \frac{P_{k,N_k^e}^s - P_{k,n}^s}{B_{k,N_k^e}^r - B_{k,n}^r}$$

where $I_{k,n} = 1$ and $n \neq N_k^e$. The numerator in the expression is the increment in power requirement, while the denominator is the decrease in buffer requirement. A greedy algorithm will choose the most profitable change first. It selects the lowest value of $W_{v,l}$ and reduces Q_v to l to keep the buffer requirement as low as possible while minimizing the power requirement. These operations are repeated until the buffer requirement can be satisfied.

If the number of speed changes produced by the first phase exceeds N_l after the first phase, then the algorithm enters its second phase to reduce the number of speed changes.

Algorithm 1 Two-Phase Heuristic Algorithm

```
Set of all values of W_{k,n}, where I_{k,n} = 1: A_f;
Temporary variables: U, C and Q_k, (k = 1, ..., N_d);
Boolean variables: flag1 and flag2;
X_{k,m} initialized to 0 (k = 1, ..., N_d and m = 1, ..., N_k^e);
Q_k initialized to N_k^e, (k = 1, ..., N_d);
U ← Σ_{k=1}^{N_d} B^r_{k,N_k^e};
flag1 ← TRUE;
flag2 ← TRUE;      // Initialization
while U > 1 − pf/B and flag1 = TRUE do
    Find the lowest value of W_{v,l} ∈ A_f and remove that W_{v,l} from A_f;
    if l < Q_v then
        Q_v ← l;
        U ← U − B^r_{v,Q_v} + B^r_{v,l};
        if A_f = φ then
            flag1 ← FALSE;      // Buffer requirement cannot be met
        end if
    end if
end while          // First phase ends.
C ← Σ_{k=1}^{N_d} Y_{k,Q_k};
while flag1 = TRUE and C > N_l do
    A_s = {P^s_{k,n} | Y_{k,n} = 0 and n ≠ Q_k};
    while flag2 = TRUE and C > N_l do
        Find the highest value of P^s_{v,h} ∈ A_s and remove that P^s_{v,h} from A_s;
        if U − B^r_{v,Q_v} + B^r_{v,h} ≤ 1 − pf/B then
            Q_v ← h;
            U ← U − B^r_{v,Q_v} + B^r_{v,h};
            C ← Σ_{k=1}^{N_d} Y_{k,Q_k};
        end if
        if A_s = φ then
            flag2 = FALSE;
        end if
    end while
    if C > N_l then
        if A_f = φ then
            flag1 = FALSE;
        else
            Repeat finding the lowest value of W_{v,l} ∈ A_f and removing that W_{v,l} from
            A_f until an l less than Q_v can be found;
            if Such an element l can be found then
                Q_v ← l;
                U ← U − B^r_{v,Q_v} + B^r_{v,l};
            else
                flag1 = FALSE;
            end if
        end if
    end if
end while          // Second phase ends.
if flag1 = TRUE then
    for k = 1 to N_d do
        X_{k,Q_k} ← 1;
    end for
end if     // If flag1 = FALSE, then there exists no feasible solution.
```

Now we are trying to select pairs which require no speed change, and we maintain an array A_s of $P_{k,n}^s$ values, where $A_s = \{P_{k,n}^s | \ Y_{k,n} = 0 \text{ and } n \neq Q_k\}$. It is clearly profitable in terms of energy to choose the pair with the highest value of $P_{k,n}^s$ in A_s. We select this element, written $P_{v,h}^s$, and change Q_k to h if this can be done without a buffer overflow; this saves the most power without changing the disk speed. These operations are repeated until the number of speed changes is less than or equal to N_l, and the buffer requirement is met. Pseudocode for this two-phase heuristic algorithm (TPHA) is given as Algorithm 1.

C. Algorithm Execution

If changing speed requires the spindle to stop, then frequent changes of speed must be avoided. Basically our scheme limits the number of simultaneous speed changes to N_l. In addition, now describe another way to reduce the number of spindle stops. User access to videos follows a daily pattern and does

TABLE III
COMPARISON BETWEEN TPHA AND OPTIMAL RESULTS

(10, 1 GB)	(15, 1.5 GB)	(20, 2 GB)	(25, 2.5 GB)	(30, 3 GB)	(35, 3.5 GB),	(40, 4 GB)	(45, 4.5 GB)
0.03%	0.08%	0.04%	0.03%	0.06%	0.01%	0.05%	0.04%

TABLE IV
AVERAGE TIMES TAKEN TO SOLVE $\mathcal{RSSP}$

(10, 1 GB)		(15, 1.5 GB)		(20, 2 GB)		(25, 2.5 GB)	
TPHA	lp_solve	TPHA	lp_solve	TPHA	lp_solve	TPHA	lp_solve
2 ms	37 ms	1 ms	71 ms	1 ms	273 ms	1 ms	1049 ms
(30, 3 GB)		(35, 3.5 GB)		(40, 4 GB)		(45, 4.5 GB)	
TPHA	lp_solve	TPHA	lp_solve	TPHA	lp_solve	TPHA	lp_solve
1 ms	1127 ms	1 ms	2450 ms	2 ms	6324 ms	1 ms	26.465 ms

not change abruptly [39], [40]. For example, most accesses occur during working hours and very few during the night; most accesses last for quite a long time [39], [40]. Therefore, the speed of a disk can be kept constant over quite a long period. Thus we can select a set of speeds for the disks in a server, which we expect to last for a certain period, based on historic data. Each disk then stays as the assigned speed throughout the entire period if the bandwidth that it provides is sufficient. However, if the workload exceeds the bandwidth available from that configuration, then a new set of speeds needs to be chosen even though the period has not expired.

VII. EXPERIMENTAL RESULTS

A. Simulation Environment

We evaluated the effectiveness of our scheme through simulations of an array of Hitachi 7K400 disks. These have a maximum data transfer rate of 61.4 MB/s and a typical seek time of 8.5 ms [16]. We considered a range of up to five speeds between 2880 and 7200 RPM, so as to be able to see the effects of various speed configurations. The characteristics of each speed level are shown in Table I. When a disk stops spinning completely, it enters standby mode, which only consumes 2.5 W. The energy required to change speed or to stop spinning is 152 J [6]. The transition time T_t between two speeds is assumed to be 7 s [16].

The arrival of client requests has been found to have a diurnal pattern [39]–[41], which follow a Poisson process. The parameters of this process depend on the nature of the video service [41]. We thus model clients' arrivals as a poisson process with arrival times that vary during the day. Videos typically last between 1 and 2 h, and the length of each video is selected randomly in this range. We profiled the energy consumption of the disk array over 24 h. Unless otherwise stated, our algorithm runs and potentially determines a new set of disk speeds every 2 min, but it may also be invoked if the bandwidth is inadequate. Unless otherwise stated, we set N_l to 2. We will initially consider two sets of speeds, $S2 = \{2880, 7200\}$ and $S5 = \{2880, 3960, 5040, 6120, 7200\}$, and the set of round lengths $S = \{1\,s, 1.5\,s, 2\,s, 2.5\,s, 3\,s, 3.5\,s, 4\,s\}$.

Unless otherwise stated, we consider four sets of disk configurations, identified by the number of disks and the buffer size as follows: (20, 2 GB), (30, 2 GB), (30, 3 GB), and (40, 3 GB). The access probability follows a Zipf distribution with parameter $\theta = 0.271$, which was a measured for a real VoD application [42]. Different video encodings have characteristic bit-rates and we considered three of the most common: 1.5 Mb/s (MPEG1), 5 Mb/s (MPEG2 for DVD), and 9 Mb/s (MPEG2 HD) [48]. We consider a workload consisting of 1000 video files, 300 MPEG1, 400 MPEG2 DVD, and 300 MPEG2 HD, where each video file is stored in a round-robin fashion across the disks.

B. Effectiveness of the Heuristic Algorithm

We found the optimal solutions to $\mathcal{RSSP}$, by running an lp_solve program [36], and compared TPHA with the optimal solutions for eight sets of disk configurations: (10, 1 GB), (15, 1.5 GB), (20, 2 GB), (25, 2.5 GB), (30, 3 GB), (35, 3.5 GB), (40, 4.0 GB), and (45, 4.5 GB). Here we are considering the $S5$, with five speeds, and the average time between the arrivals of video requests is 4 s.

Table III shows the percentage difference between the result from TPHA and the optimal value. These results suggest that TPHA can provide a near-optimal solution to $\mathcal{RSSP}$ with an 0.04% of power difference on average.

Table IV shows the average time taken to produce these results running both the lp_solve program and TPHA on a personal computer with a quad-core 2.66 GHz Intel i5-750 CPU and 3.3 GB of RAM; it clearly shows the advantage of TPHA is suitable for $\mathcal{RSSP}$ when speeds and round lengths need to be selected quickly.

Due to the constraint of the prefetching buffer size, we limited the number of speed changes to N_l, which affects the round lengths and speeds that we selected. To find the absolutely minimal energy consumption, we relax this constraint. We therefore performed more simulations in which neither the number of simultaneous speed changes nor the buffer space was limited. Table V shows that TPHA continues to provide a near-optimal solution, with an 0.04% average difference on average in the power requirement.

C. Effects of Adjusting the Round Length

To evaluate the effectiveness of our scheme, we will compare it with two other methods.

1) Control of variable speeds by selecting the highest (CVH): operates like a standard video server, in that it does not allow round length adjustment, and always selects the highest disk speeds. The round length is calculated as follows: using bit-rates from Section VII-A,

TABLE V
COMPARISON BETWEEN TPHA AND OPTIMAL RESULTS WITH UNLIMITED SPEED CHANGES

(10, 1 GB)	(15, 1.5 GB)	(20, 2 GB)	(25, 2.5 GB)	(30, 3 GB)	(35, 3.5 GB),	(40, 4 GB)	(45, 4.5 GB)
0.01%	0.05%	0.07%	0.02%	0.05%	0.05%	0.04%	0.01%

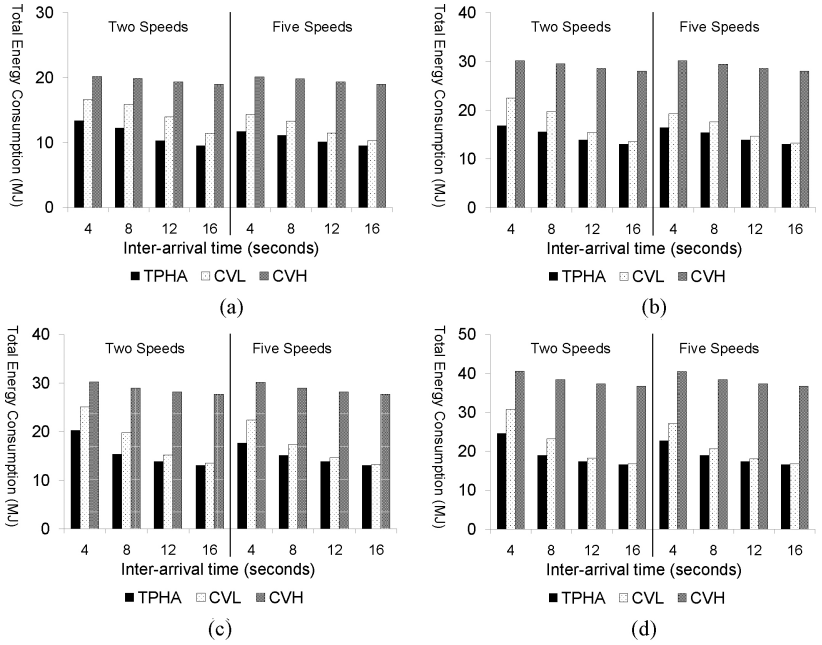

Fig. 3. Energy consumption over 24 h against aggregate access rate. (a) (20, 2 GB). (b) (30, 2 GB). (c) (30, 3 GB). (d) (40, 3 GB).

we determine an average bit-rate of 5.15 Mb/s across all the videos, and hence obtain a round length that can be expected to balance disk bandwidth against buffer size [29] in the aggregate; the closest round length in the list S is then selected. This is effectively the standard scheme for an array of single-speed disks running at 7200 RPM.

2) Control of variable speeds by selecting the lowest (CVL): is similar to CVH in that it does not allow round length adjustment, but it does allow changes in disk speeds, and selects the lowest speed level that does not exceed the disk utilization limit. This keeps power consumption as low as possible without any need to change the round length.

1) *Impact of Aggregate Access Rate:* The aggregate access rate determines the disk loads, and hence affects energy consumption. We examined how this effect depends on buffer size and the number of disks, with the results shown in Fig. 3. The TPHA scheme exhibits the best performance under all workloads, using between 1% and 26% less energy than CVL and between 33% and 55% less than CVH. On average, it saves 13% more energy than CVL, and 48% more than CVH. It is clear that adjusting the round length gives the disks more opportunities to reduce their speeds.

In more detail, we note the following differences between the performance of TPHA and CVL: a) except that the load is high (e.g., interarrival time is 4 s); the energy gap is more pronounced when only two speeds are allowed instead of five speeds; b) when the resource is adequate (40, 3 GB) and the load is light, the energy gap is negligible; and c) the gap increases as loads increase when five speeds are supported. We can also see that the workload has little effect on the energy consumption with CVH, because the difference between idle and active power is small. Reducing disk loads does allow the disks to stay in idle mode for longer, but this saves little energy. Conversely, the aggregate access rate has a great impact on both CVL and TPHA. We also observe that TPHA and CVL use less energy with five disk speeds rather than two, because more accurate matching of speeds to workload is possible.

2) *Diurnal Access Patterns:* Video server loads typically follow a diurnal cycle, with peaks usually occurring between 11 A.M. and 4 P.M. [39]–[41]. We reflect this by hourly changes to the rate at which requests arrive [41]. The interarrival times are varied between α and β seconds, where α is the peak load. We consider the following four combinations of α and β: (3, 9), (4, 12), (3, 15), and (4, 18).

Fig. 4 shows how the energy consumption of the three schemes depends on (α, β). The TPHA scheme exhibits the best performance under all workloads, using between 5% and 27% less energy than CVL, and between 37% and 53% less than CVH. We also observe that the differential between TPHA and CVL is generally greater than the case when the interarrival time is fixed, which implies that our scheme is adapting to changing workloads effectively.

D. Effects of Different Speed Sets

We examined how the energy consumption depends on the number of available speeds, and considered two additional sets of speeds: $S3 = \{2880, 5040, 7200\}$ and $S4 = \{2880, 5040, 6120, 7200\}$. From now on, we consider a diurnal

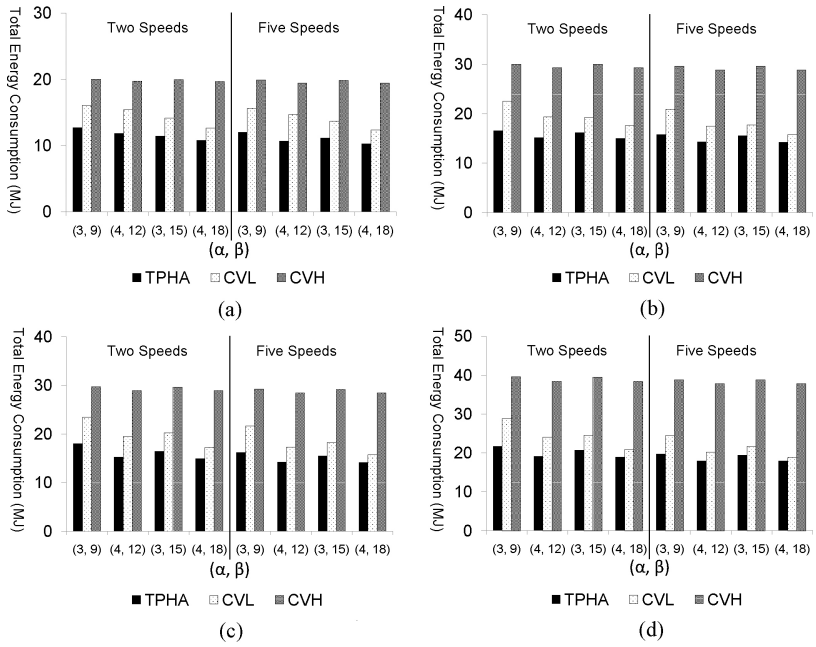

Fig. 4. How energy consumption depends on (α, β). (a) (20, 2 GB). (b) (30, 2 GB). (c) (30, 3 GB). (d) (40, 3 GB).

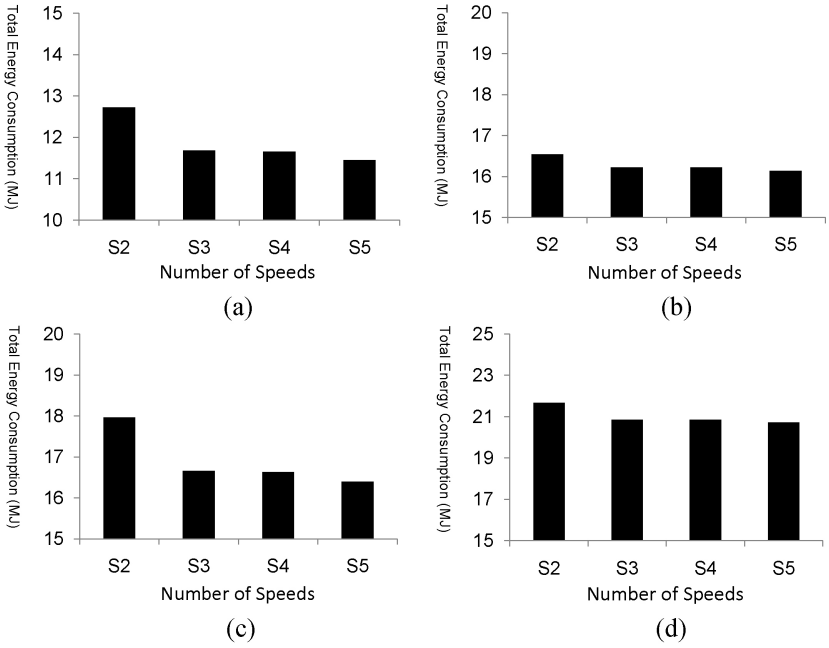

Fig. 5. Energy consumption against the number of speed sets. (a) (20, 2 GB). (b) (30, 2 GB). (c) (30, 3 GB). (d) (40, 3 GB).

access with arrival times determined by (3, 9). The results are shown in Fig. 5, and confirm our expectation that more allow better matching to workloads and hence reduce energy consumption; e.g., $S5$ uses 2.4% and 10% less energy than $S2$, and between 0.5% and 2% less than $S3$ and $S4$. However, we see that the energy gap between $S3$ and $S5$ is much smaller that that between $S2$ and $S5$, suggesting that increases in the number of speeds are subject to the law of diminishing returns.

E. Number of Speed Changes

1) *Effects of* (α, β)*:* We assessed the average number of speed changes per disk for different values of α and β, with the results shown in Fig. 6. Obviously, increasing the workloads generally increases the number of speed changes. If the workload is heavy but the bandwidth is inadequate (20, 2 GB), then the number of speed changes is relatively high. We can also see that there is no clear relationship between the number of speeds available and the number of speed changes.

2) *Effect of the Time Period:* In the foregoing tests, our algorithm was executed every 2 min, and when there is a disk bandwidth shortage. We now examine how this period affects the number of speed changes when α is 3 and β is 9. The periods we tried were 4, 8, 20, 60, 120, and 240 s, and the results are shown in Fig. 7. As the period increases, the number of speed changes decreases. We also assessed how

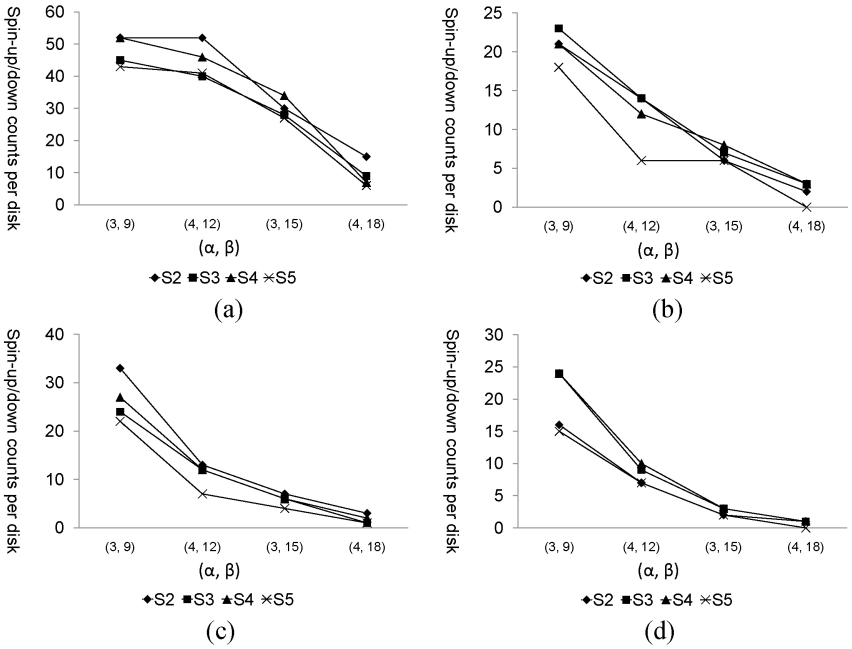

Fig. 6. Average number of speed changes per disk for different values of α and β, and different numbers of available speeds. (a) (20, 2 GB). (b) (30, 2 GB). (c) (30, 3 GB). (d) (40, 3 GB).

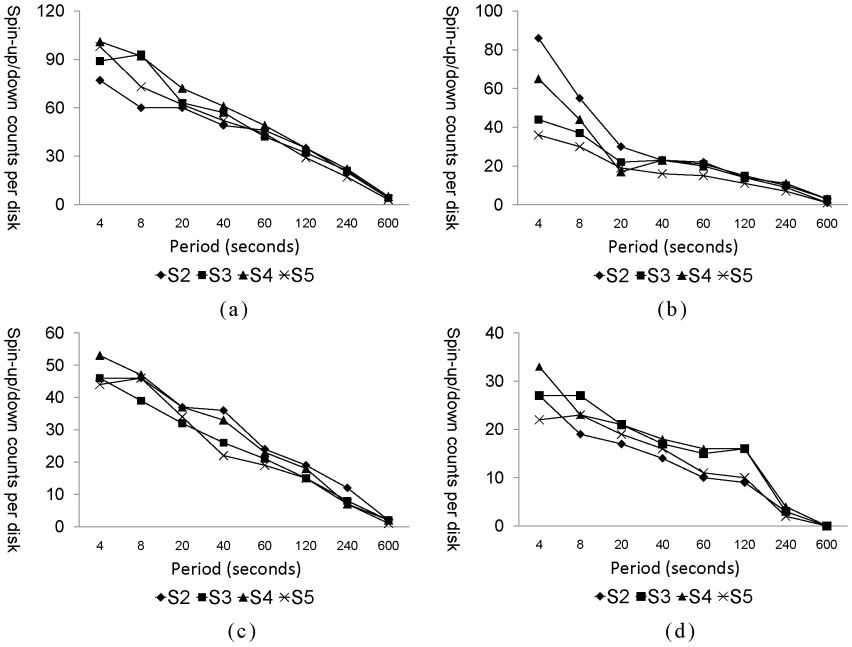

Fig. 7. Average number of spin-up/downs per disk against period between algorithm runs. (a) (20, 2 GB). (b) (30, 2 GB). (c) (30, 3 GB). (d) (40, 3 GB).

the energy consumption depends on the period, with the results shown in Fig. 8. Increasing the period also reduces the energy consumption because speed changes require significant energy, but these savings gradually tail off and they disappear entirely when the period is long. Increasing the period beyond 240 s is pointless, because a shortage of disk bandwidth is always certain to occur within 240 s.

F. Impact of N_l Values

We now examine how the value of N_l affects the energy consumption and the number of clients admitted when five speeds are supported, using the results given in Table VI. As N_l increases, the energy consumption per client increases. This is because increasing N_l requires more contingent buffer space to store prefetched data, which reduces the available buffer space, resulting in a shorter round length. The average round length selected is given in Table VI, and we see that increasing N_l decreases the round length. Because a shorter round may increase power consumption, as shown in Fig. 2, increasing N_l may have an adverse effect on energy consumption even though this is very small, especially when the buffer space is sufficient. That is because optimal speed and round length pairs do not change drastically since the pattern of user access to videos does not change abruptly [39], [40].

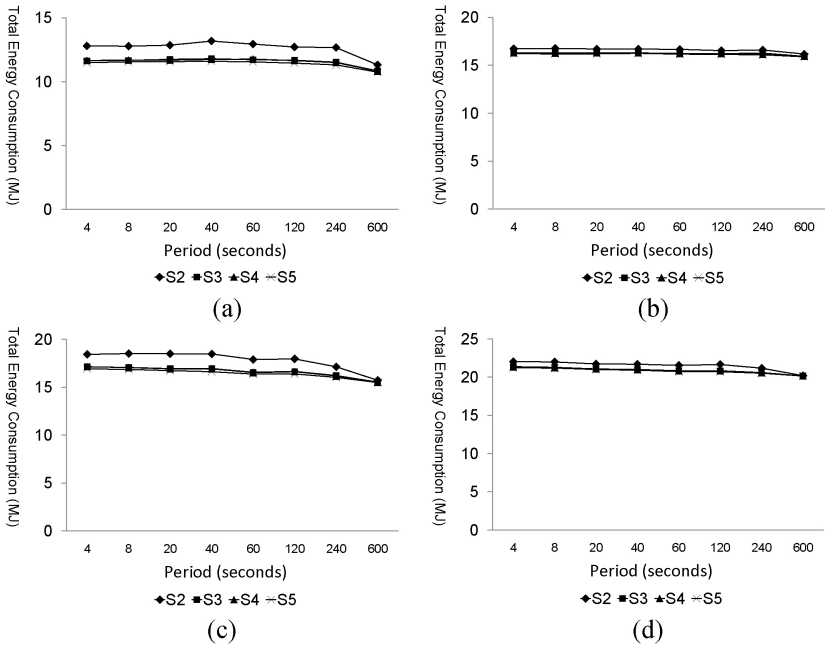

Fig. 8. How energy consumption depends on the time periods. (a) (20, 2 GB). (b) (30, 2 GB). (c) (30, 3 GB). (d) (40, 3 GB).

G. Discussion

We draw the following conclusions from our evaluation.

1) TPHA provides a near-optimal solution to $\mathcal{RSSP}$ and runs in a matter of milliseconds.
2) The power consumption of single-speed disks is largely unaffected by reductions in load, whereas multispeed disks can respond effectively to the changing workloads expected in video servers, yielding economies when loads are light.
3) Adjusting the round length and disk speeds is effective in reducing energy consumption when the workload follows the diurnal pattern expected in most VoD services.
4) Increasing the number of available disk speeds saves energy, but the benefits gradually tail off.
5) The number of speed changes that occur increases as the number of clients and the number of speeds increase. If light loads are expected, a wide range of speeds is desirable; but if loads are likely to be heavy, fewer speeds are actually more effective.
6) Increasing N_l may require more contingent buffer space to store prefetched data. This reduces the available buffer space, and reduces the average round length, increasing the energy consumption, but this effect is small.

VIII. Extensions to Our Scheme

A. Support of VBR Video

Although our scheme was developed for constant bit-rate video, we now describe how it might be extended to support variable bit-rate (VBR) streams. The bit-rate of a VBR video may change during a round and this affects the disk bandwidth and buffer requirements. We can address this issue by estimating the extent of the variability which can be done by deterministic, statistical, or predictive approaches [45], [47]. We now suggest three ways to extend our scheme to VBR.

1) A deterministic method in which the server uses the peak data-rate requirement to calculate the bit-rate of a video. Then, our scheme can be extended in a straightforward manner by replacing the bit-rate of each video by its peak rate. This approach guarantees to meet the real-time requirements of the video; but the number of clients is limited and server resources are underutilized.
2) A statistical method provides statistical service guarantee. It has been shown that the sum of data bit-rates of all the clients approaches a normal distribution using the central limit theorem [45]. This means that the disk bandwidth and the buffer utilizations in (1)–(4) also follow normal distributions. Based on this, we can calculate the probability that a frame is not read in time. To illustrate, let us define two random variables for each disk k: $rd_k(j,m)$ is the disk bandwidth utilization with a round length index of j and a speed level of m, and rc_k is the contingent disk bandwidth utilization. Let γ be the probability that $\gamma\%$ of the requested frames are read in time. We can derive a list of candidate round and speed pairs for disk k as follows:
$$\{(j,m)|\ R_j \in S \text{ and } P(rd_k(j,m) + rc_k \leq 1) \leq \gamma)\}.$$
3) A predictive method in which we calculate resource requirements based on measurements of recent utilization. By adding the average bandwidth requirement of a newly requested stream to the recent average bandwidth requirements, we can estimate the future bandwidth requirement.

The future energy requirement may fluctuate due to the characteristics of VBR videos; but we can estimate it using two approaches: first, we can replace the bit-rate of each video by its average rate, and then, estimate the energy requirement using (5), or we can estimate the energy requirement based

TABLE VI
ENERGY CONSUMPTION AND NUMBER OF CLIENTS ADMITTED AGAINST N_l VALUES

	(20, 2 GB)				(30, 2 GB)			
N_l	1	2	3	4	1	2	3	4
Energy (MJ)	11.6	11.45	11.57	11.8	16.4	16.14	16.29	16.51
No. of clients	9500	9274	8762	7894	11 980	11 473	10 647	9742
Average round length (s)	3.3	2.9	2.6	2.5	2.2	2	1.9	1.88
	(30, 3 GB)				(40, 3 GB)			
N_l	1	2	3	4	1	2	3	4
Energy (MJ)	16.36	16.39	16.45	16.53	20.65	20.71	20.79	20.88
No. of clients	12 226	12 224	12 224	11 949	13 826	13 826	13 791	13 683
Average round length (s)	4	3.6	3.3	3	3.5	3	2.8	2.6

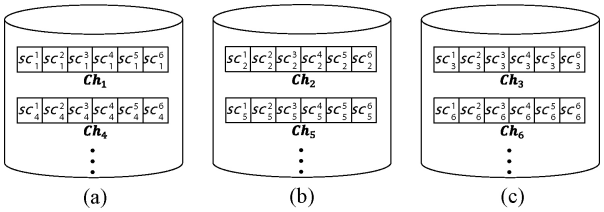

Fig. 9. Example of data placement when striping is used. (a) Disk 1. (b) Disk 2. (c) Disk 3.

on the past average energy requirement. For example, when a client requests a new stream, the recent power consumption plus the average power requirement for a new client can be used as an estimate of the subsequent power consumption. This approach is based on the observation that the average bandwidth requirement does not change significantly [45].

B. Support of Striping

If a video file is stored contiguously on a single disk, then any data read from that file during a round involves the overhead of a single seek. But if a file is striped across several disks, then changing the round length may incur additional seek overheads because the data retrieved during a round may be stored on more than one disk. Our scheme can be extended to striping in a straightforward manner provided that we ensure that each access involves only one seek.

We can guarantee that only one seek is required in each round by splitting each video into chunks, and the selecting the chunk size to reflect the data-rate of each video. Suppose a video is split into chunks ch_m, each composed of N_u subchunks sc_m^n ($n = 1, \ldots, N_u$) which contain the amount of data that can be retrieved during a round of length sr. Thus, the size of each subchunk of video stream V_i is $b_i sr$. We store N_u subchunks contiguously in a chunk, and each chunk is placed on the disks in round-robin fashion, as shown in the example of Fig. 9, in which $N_u = 6$ and the striping width is 3. We will use dv_j to denote the jth divisor of N_u ($j = 1, \ldots, N_I$), where N_I is the total number of such divisors.

We can now define a list of feasible round lengths $S = \{R_j | R_j = dv_j sr\}$, which is sorted into ascending order. For example, if $N_u = 6$, then $S = \{sr, 2sr, 3sr, 6sr\}$. The selection of round lengths other than R_j is not allowed because this may lead to two or more seeks being required for one read.

If a round length is chosen from S, then each request during a round is guaranteed to access only one disk. For example, if the round length is set to $3sr$, then accesses are redirected to the next disk in the array every two rounds. Conversely, if $4sr$ is selected, then two disks may be accessed simultaneously, as shown in Fig. 9. In this example, chunks sc_1^5, sc_1^6, sc_2^1 and sc_2^2 need to be read simultaneously.

IX. CONCLUSION

We have presented a new disk energy saving scheme for video servers that use multispeed disks. We analyzed how disk and buffer utilization depends on the retrieval period, the buffer size, and the disk speed. We then proposed a new data retrieval and buffer management scheme in which the retrieval period and the speed of each disk can be dynamically changed to adjust disk utilization, with the aim of minimizing the total energy consumed by the disks while limiting the number of speed changes. We also extended our scheme to reduce the number of disk speed changes and to support striping.

Experimental results have shown that: 1) our scheme achieves appreciable energy savings under all workloads, and 2) increasing the number of available speeds reduces energy consumption, but increases the number of speed changes. These observations provide a guideline for the design of video servers with multispeed disks.

There are several potential areas where our scheme could be extended in our future work. First, we are in the process of adding support for servers that have to provide both video and transactional data. One possible way to design an algorithm for these servers is to reserve some fraction of the disk bandwidth to handle the transactional workload. Second, we are interested in extending our scheme to servers which use parity information for error correction, such as those based on RAID5. We are hoping to implement our scheme on real video servers, allowing us to measure actual disk energy consumption.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] R. Bianchini and R. Rajamony, "Power and energy management for server systems," *IEEE Comput.*, vol. 37, no. 11, pp. 177–190, Nov. 2004.
[2] Q. Zhu and Y. Zhou, "Power aware storage cache management," *IEEE Trans. Comput.*, vol. 54, no. 5, pp. 587–602, May 2005.

[3] Z. Ge, P. Ji, and P. Shenoy, "Design and analysis of a demand adaptive and locality aware streaming media server cluster," *ACM/Springer Multimedia Syst. J.*, vol. 13, no. 3, pp. 235–249, Sep. 2004.
[4] *Youtube Bandwidth: Terrabytes Per Day* [Online]. Available: http://blog.forret.com/2006/05/youtube-bandwidth-terabytes-per-day
[5] B. Moore. (2002, Aug.). Taking the data center power and cooling challenge. *Energy User News* [Online]. *27*. Available: http://www.sustainablefacility.com/articles/taking-the-data-center-power-and-cooling-challenge
[6] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes, "Hibernator: Helping disk arrays sleep through the winter," *ACM Oper. Syst. Rev.*, vol. 39, no. 5, pp. 177–190, 2005.
[7] American Power Convention, *Determining Total Cost of Ownership for Data Centers and Network Room Infrastructure*, White Paper #6, 2003.
[8] S. Gurumurthi, "Power management of enterprise storage systems," Ph.D. dissertation, Dept. Comput. Sci. Eng., Pennsylvania State Univ., University Park, Aug. 2005.
[9] *Symmetrix 3000 and 5000 Enterprise Storage Systems Product Description Guide* [Online]. Available: http://www.emc.com/1999
[10] F. Chen and X. Zhang, "Caching for bursts (c-burst): Let hard disks sleep well and work energetically," in *Proc. ACM Symp. Low Power Electron. Des.*, Aug. 2008, pp. 141–146.
[11] A. Papathanasiou and L. Scott, "Energy efficient prefetching and caching," in *Proc. USENIX Annu. Tech. Conf.*, Jun. 2004, pp. 255–268.
[12] Y. Won, J. Kim, and W. Jung, "Energy-aware disk scheduling for soft real-time I/O requests," *Multimedia Syst. J.*, vol. 13, no. 5, pp. 409–428, Feb. 2008.
[13] T. Xie, "SEA: A striping-based energy-aware strategy for data placement in RAIS-structured storage systems," *IEEE Trans. Comput.*, vol. 57, no. 6, pp. 748–761, Jun. 2008.
[14] M. Song, "Dynamic buffer allocation for conserving disk energy in clustered video servers which use replication," in *Proc. EUC Conf.*, Aug. 2006, pp. 193–203.
[15] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "Reducing disk power consumption in servers with DRPM," *IEEE Comput.*, vol. 36, no. 12, pp. 59–66, Dec. 2003.
[16] Hitachi Global Storage Technologies, *Hitachi Power and Acoustic Management: Quietly Cool*, White Paper, Mar. 2004.
[17] H. Yada, H. Ishioka, T. Yamakoshi, Y. Onuki, Y. Shimano, M. Uchida, H. Kanno, and N. Hayashi, "Head positioning servo and data channel for HDDs with multiple spindle speeds," *IEEE Trans. Magnet.*, vol. 36, no. 5, pp. 2213–2215, Sep. 2000.
[18] Nexsan [Online]. Available: http://www.nexsan.com/sataboy/automaid.php
[19] E. Pinheiro and R. Bianchini, "Energy conservation techniques for disk-array-based servers," in *Proc. ACM/IEEE Conf. Supercomput.*, Jun. 2004, pp. 88–95.
[20] E. Pinheiro, R. Bianchini, and C. Dubnicki, "Exploiting redundancy to conserve energy in storage systems," *ACM Performance Eval. Rev.*, vol. 4, no. 1, pp. 15–26, Jan. 2006.
[21] X. Yao, H. Zhu, and J. Wang, "Exploiting in-memory and on-disk redundancy to conserve energy in parity disk array," *IEEE Trans. Comput.*, vol. 57, no. 6, pp. 733–747, Jun. 2008.
[22] S. Son and M. Kandemir, "Energy-aware data prefetching for multi-speed disks," in *Proc. ACM Conf. Comput. Frontiers*, 2006, pp. 105–114.
[23] M. K. R. Rao, and S. Vrudhula, "Disk drive energy optimization for audio-video applications," in *Proc. ACM Conf. Compilers, Architect. Syn. Embedded Syst.*, Sep. 2004, pp. 93–103.
[24] W. G. X. Liu, and P. Shenoy, "A time series-based approach for power management in mobile processors and disks," in *Proc. ACM Workshop Netw. Oper. Syst. Support Digital Audio Video*, Jun. 2004, pp. 74–79.
[25] L. Cai and Y. Lu, "Energy management using buffer memory for streaming data," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 24, no. 2, pp. 141–152, Jul. 2005.
[26] N. Pettis, L. Cai, and Y. Lu, "Statistically optimal dynamic power management for streaming data," *IEEE Trans. Comput.*, vol. 55, no. 7, pp. 800–814, Jul. 2006.
[27] E. Chang, "Storage and retrieval of compressed video," Ph.D. dissertation, Dept. Electric. Eng. Comput. Sci., Univ. California Berkeley, Berkeley, 1996.
[28] E. Chang and H. Garcia-Molina, "Effective memory use in a media server," in *Proc. VLDB Conf.*, Aug. 1997, pp. 496–505.
[29] M. Song and H. Shin, "Replication and retrieval strategies for resource-effective admission control in multi-resolution video servers," *Multimedia Tools Applicat. J.*, vol. 28, no. 3, pp. 89–114, Mar. 2006.
[30] H. Vin, S. Rao, and P. Goyal, "Optimizing the placement of multimedia objects on disk arrays," in *Proc. IEEE Int. Conf. Mutimedia Comput. Syst.*, May 1995, pp. 158–165.
[31] C. Chou, L. Golubchik, and J. Lui, "Striping doesn't scale: How to achieve scalability for continuous media servers with replication," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Apr. 2000, pp. 64–71.
[32] M. Reisslein, K. Loss, and S. Shrestha, "Striping for interactive video: Is it worth it?" in *Proc. IEEE Int. Conf. Multimedia Comput. Syst.*, Jun. 1999, pp. 635–668.
[33] E. Chang and A. Zakhor, "Disk-based storage for scalable video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 5, pp. 758–770, Oct. 1997.
[34] M. Song, "Energy-aware data prefetching for multi-speed disks in video servers," in *Proc. ACM Multimedia Conf.*, Oct. 2007, pp. 755–758.
[35] D. Pisinger, "Algorithms for knapsack problems," Ph.D. dissertation, Dept. Comput. Sci., Univ. Copenhagen, Copenhagen, Denmark, 1995.
[36] *Introduction to lp_solve 5.5.2.0* [Online]. Available: http://lpsolve.sourceforge.net/5.5
[37] T. Bisson, S. Brandt, and D. Long, "A hybrid disk-aware spin-down algorithm with I/O subsystem support," in *Proc. IEEE Int. Performance, Comput. Commun. Conf.*, Apr. 2007, pp. 236–245.
[38] *Panel Computer Hard Disk Drive Precautions* [Online]. Available: http://www.pro-face.com/support/technical/00apr3.htm
[39] J. Almeida, J. Krueger, D. Eager, and M. Vernon, "Analysis of educational media server workloads," in *Proc. Netw. Oper. Syst. Supports Digital Audio Video*, Jun. 2001, pp. 21–30.
[40] F. Johnsen, T. Hafse, C. Griwodz, and P. Halvorsen, "Workload characterization for news-on-demand streaming services," in *Proc. Int. Performance Comput. Commun. Conf.*, Apr. 2007, pp. 314–323.
[41] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, "Medisyn: A synthetic streaming media service workload generator," in *Proc. Netw. Oper. Syst. Supports Digital Audio Video*, Jun. 2003, pp. 12–21.
[42] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *ACM/Springer Multimedia Syst. J.*, vol. 4, no. 3, pp. 112–121, 1996.
[43] J. Go and M. Song, "Adaptive disk power management for portable media players," *IEEE Trans. Consum. Electron.*, vol. 54, no. 4, pp. 1755–1760, Nov. 2008.
[44] J. Kim, A. Yang, and M. Song, "Exploiting flash memory for reducing disk power consumption in portable media players," *IEEE Trans. Consum. Electron.*, vol. 55, no. 4, pp. 1997–2004, Nov. 2009.
[45] H. Vin, P. Goyal, and A. Goyal, "A statistcal admission control algorithm for multimedia storage servers," in *Proc. ACM Multimedia*, Oct. 1994, pp. 33–40.
[46] M. Song and H. Shin, "A QoS degradation policy for revenue maximization in fault-tolerant multi-resolution video servers," *IEEE Trans. Consum. Electron.*, vol. 49, no. 2, pp. 392–402, May 2003.
[47] X. Jiang and P. Mohapatra, "Efficient admission control algorithms for multimedia servers," *ACM Multimedia Syst. J.*, vol. 7, no. 4, pp. 294–304, Jul. 1999.
[48] *Bit Rate* [Online]. Available: http://en.wikipedia.org/wiki/bitrate

Manjong Kim (S'11) received the B.S. degree in computer engineering from Inha University, Incheon, Korea, in 2009. He is currently pursuing the M.S. degree from the School of Computer Science and Information Engineering, Inha University.

His current research interests include embedded software and multimedia systems.

Minseok Song (M'07) received the B.S. and M.S. degrees in computer engineering in 1996 and 1998, respectively, and the Ph.D. degree in electrical engineering and computer science in 2004, all from Seoul National University, Seoul, Korea.

Since September 2005, he has been with the School of Computer Science and Information Engineering, Inha University, Incheon, Korea, where he is currently an Associate Professor. His current research interests include real-time, embedded systems and multimedia systems.