J. Parallel Distrib. Comput. 75 (2015) 29–39

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Performance-constrained energy reduction in data centers for video-sharing services

Hang Yuan [a,*], Ishfaq Ahmad [b], C.-C. Jay Kuo [a]

[a] Ming Hsieh Department of Electrical Engineering, University of Southern California, United States
[b] Computer Science and Engineering Department, The University of Texas at Arlington, United States

HIGHLIGHTS

- A new model for disk idle time in video-sharing services (VSSs) is proposed.
- We optimize the selection of disk power modes with delay constraints.
- We propose efficient energy-aware caching and prefetching schemes for VSSs.
- Significant energy saving is achieved for VSSs under very low service delay.

ARTICLE INFO

Article history:
Received 14 April 2014
Received in revised form
3 October 2014
Accepted 20 October 2014
Available online 24 October 2014

Keywords:
Energy efficiency
Video-sharing servers
Parallel storage systems

ABSTRACT

Energy saving in large-scale video sharing data centers is an important yet daunting challenge due to the conflicting goal of providing real-time guarantees. Simple energy reduction techniques can result in excessive delay and severely affect the quality-of-service. This paper aims to optimize energy consumption while ensuring service delay constraints in data centers that provide large-scale video-sharing services. However, this broader goal requires three challenges that must be holistically addressed rather than in isolation. First, we propose a generic model to accurately characterize the disk behavior in a VSS by taking into account the unique characteristic of parallel video workloads. Second, the paper proposes a prediction-based algorithm that formulates and solves a constrained optimization problem for determining optimal selections of disk power modes in VSSs. Third, two novel caching algorithms are proposed that achieve additional energy saving through optimizing cache utilization. Experiments reveal that the proposed 3-component scheme achieves a significant amount of energy saving under the same delay level as compared to traditional energy management schemes.

1. Introduction

YouTube and other large-scale video-sharing services (VSSs) have come to dominate the Internet traffic and continue to grow rapidly [5]. Unlike traditional Internet applications, VSSs deal with millions of videos and users on a daily basis. To support the scale and growth of both video content and users, large-scale storage systems have been deployed, resulting in ever-growing energy consumption and, hence, cost. Reports show that 20%–40% of energy in data centers is consumed by disk drives [19]. Furthermore, around 82%–94% of this amount of energy is dissipated when disks are idle [14], indicating very low energy efficiency in disks. These observations call for effective measures to reduce energy and cut costs.

To provide high throughput and scalability, VSSs employ parallel architectures for storage and retrieval [15,16]. In parallel servers, a video is divided into chunks and distributed across multiple disks. By splitting a relatively long video file into smaller units, multiple requests can be served concurrently [9]. A further benefit is that both bandwidth and capacity can be scaled up by including additional servers into the system [15]. With the requirement of parallel storage and real-time delivery, energy management in modern VSSs becomes an intricate challenge that has only been marginally addressed by previous research works.

One of the main difficulties in the endeavor of reducing energy for VSSs pertains to the need for low-delay solutions for ensuring good performance. As videos in VSSs such as YouTube tend to be short (less than 10 min), users have become increasingly sensitive to delays. Traditional disk energy reduction techniques

* Corresponding author.
E-mail addresses: rhagn.ryu@gmail.com (H. Yuan), iahmad@cse.uta.edu (I. Ahmad), cckuo@sipi.usc.edu (C.-C.J. Kuo).

http://dx.doi.org/10.1016/j.jpdc.2014.10.008
0743-7315/

Table 1
Low power modes and corresponding recovery penalty.

Mode	Description	Power savings	Recovery times (s)
L1	Active idle	0%	0
L2	Unloaded heads	23%	0.5
L3	Reduced speed	35%	1
L4	Stopped motor	54%	8

rely on spinning down the disk whenever a long idle period is expected [21,31]. However, this approach induces very heavy penalties in service delay – the key performance constraint – preventing it from being a viable choice for VSSs [14]. Recently, disks with multiple power modes have become increasingly popular [20,24]. These power modes are achieved by unloading the heads to drive ramp and slowing down drive speed. In these modes, the disk is not be able to serve any requests. Therefore, they are sometimes referred to as *sleep modes*. Generally, they incur smaller penalty in order to bring the disk back to active state than a complete spin-down operation.

Although energy management algorithms based on similar disk models have been proposed [13,34], they do not consider constraints on service delay that is crucial in VSSs. This paper proposes schemes that minimize energy consumption under delay constraints for large-scale video sharing servers. Moreover, caching is an integral part of video delivery systems and thus must be taken into account. A scheme based on disk models alone will be insufficient to ensure energy reduction in video sharing servers. The paper starts with an overview of critical issues in energy saving for a large-scale VSS with a diverse video database and parallel storage systems, and makes the following major contributions.

1. The paper proposes a generic model for disk idle time for video-sharing servers by taking into account the unique characteristics of parallel video workloads. The model is based on time-varying Poisson processes and accurately characterizes the disk behavior in a VSS.
2. The paper formulates a constrained optimization problem for energy reduction while ensuring minimum delays, and proposes a prediction-based algorithm that utilizes a novel Lagrangian formulation for determining optimal selections of disk power modes in VSSs. The algorithm takes into account specific delay constraints while solving the energy optimization problem. The proposed algorithm consumes less energy than the traditional approaches under the same delay constraints.
3. In conjunction with the above approaches, two novel caching algorithms are proposed that achieve additional energy saving through optimizing cache utilization. Compared to traditional caching algorithms, these schemes incur lower energy under very low delay constraints.

For demonstrating the applicability of the proposed framework in real environments, we adopt the Seagate® PowerChoice™ [24] in our disk power model, as summarized in Table 1. However, our approach is general and therefore applicable to any disk that uses multiple power modes, including disks with multiple low speed modes [2,10]. As explained below, more energy saving can be achieved with increased number of intermediate modes.

The rest of this paper is organized as follows. Section 2 presents related background and previous work. Section 3 describes the scheduling of heterogeneous video workloads in a parallel storage system, and the modeling of disk behaviors under such scheduling. Section 4 includes the details of the proposed prediction-based mode decision algorithm. Section 5 presents the proposed energy-aware caching algorithms. Section 6 provides the details of the evaluation of the proposed algorithms through simulation. Section 7 wraps the paper with concluding remarks and future directions.

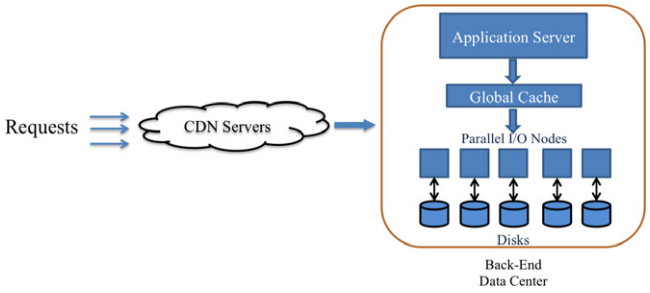


Fig. 1. The architecture of video sharing services.

2. Background and related work

2.1. Large-scale video sharing services

Large-scale VSSs such as YouTube employ a hierarchical delivery cloud that contains at least one level of cache servers, which are placed close to end users [33]. These cache servers, mostly part of a content delivery network (CDN), store and deliver the most popular video files. The architecture of a VSS with one level of cache servers is shown in Fig. 1. Unlike traditional server workload, content popularity in a YouTube-like VSS does not follow a Zipf-like distribution [3]. Fig. 2 depicts the relationship between the popularity ranks of videos and their normalized popularities. It can be observed while head part of the curve is Zipf (linear in log-scale), the tail part exhibits an exponential distribution. This work focuses on the back-end data center, which serves the non-Zipf part of the curve, corresponding to videos with low to medium popularities [33]. Since there is a large number of less popular videos in the data center, we can observe the "long-tail" effect [1] and anticipate a low cache hit rate [33].

As shown in Fig. 1, the back-end data center employs a parallel storage architecture. Each storage node is an I/O server, which can have either a single disk or a disk array. In this paper, we assume that a single disk is attached to each node. However, our scheme is based on a very general system model and, therefore, can be easily applied to the case of disk arrays. One simple way to do this is to treat each disk array as a single logical disk attached to an I/O server node.

2.2. Review of energy-efficient system design

A number of algorithms have been proposed for utilizing low-power modes in disks to reduce the energy consumption in storage servers. Pinheiro and Bianchini [22] proposed to concentrate workload onto a subset of disks so that the rest can use more low power modes. Xie [32] designed an algorithm that divides disks into two zones according to the load and then applies different power modes. Son and Kandemir [27] used compiler-directed data prefetching to allow some disks to stay longer in low power modes. Zhu and Zhou [34] developed an efficient cache replacement algorithm to enable reduction in disk access. This technique creates longer idle periods and enables more frequent use of low power modes. Manzanres et al. [18] proposed a scheme to prefetch popular content into the cache so that disks can stay in the standby mode for longer periods.

The above techniques exploit low-power modes to achieve energy saving. However, none of them are suitable for real-time Internet services, where the time constraint is critical. More recently, a couple of schemes have been developed for energy saving in the context of real-time video services. For example, Forte and Srivastava [7] studied data ordering and placement onto a single disk drive with an objective of supporting more concurrent accesses

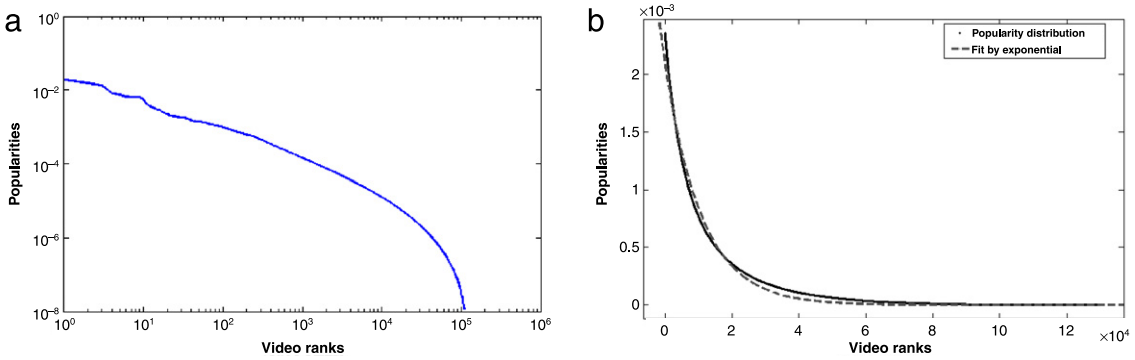


Fig. 2. The distribution of video popularities for (a) all videos and (b) moderately played and tail content.

while consuming less energy. Kim and Song [14] proposed to adjust the disk speed and round length to save energy in video servers.

Although the energy saving problem for video servers was addressed in [14,7], our current work significantly differs from theirs in the following areas. First, we deal with large-scale Internet video-sharing workloads that consist of a diverse repository of videos with different bit-rates and lengths. In contrast, video databases in [14,7] are homogeneous with a much smaller number of videos, which oversimplified the real workload characteristics. Second, we deal with a huge number of user requests mainly for short videos. This makes delay a crucial factor, which is previously unexplored but highly critical in a realistic environment. Finally, our system model assumes parallel storage of video contents, which is not addressed previously but almost a norm in modern VSSs.

A few studies have been reported that took advantage of file replications and applied workload redirection to increase the length of disk idle intervals [19,30]. These approaches are fundamentally orthogonal to ours, wherein each workload is directed to one single disk, and hence can work together with our algorithms. These techniques can be used first to decide which disk should be accessed for each workload; next, our algorithms can be applied to optimize the mode selection and caching decision after the target disk is selected.

3. Workload scheduling and modeling

3.1. Placement and delivery of video content

To facilitate video delivery, video data files are often divided into fixed size blocks in video servers [23]. In parallel storage systems, these blocks can be placed onto different serves to improve performance. While data placement algorithms are important, they are not the focus of this paper. To strike a balance between server throughput and energy-saving opportunities, we adopt a heuristic file assignment policy that creates a reasonable level of imbalance in load distribution. Among video blocks served by the back-end data center, we place those with highest access rates (top 5%) evenly across all disks. For the remaining blocks, we place them in a skewed fashion (i.e. popular video are concentrated in the first few disks).

When a user requests a video for playback, the system dispatcher locates all blocks of the video. One workload is then created for each block, and added to the global service queue. Next, the shared memory cache is searched for each of the workloads. If the requested block is found in the cache, data are fetched directly from the memory cache. Otherwise, the workload is added to the disk service queue and disk access is initiated.

Due to the periodic nature of video retrieval, video servers usually adopt a round-based scheduling strategy [14,9,7]. To facilitate disk scheduling and admission control, time is divided into equal-sized rounds. Since a fixed block size is used, we can simply impose a cap on the number of blocks to be served per round to ensure that disks are not overloaded. Also, the whole delivery process is pipelined [26]. If a request arrives at round a, its first block is fetched from disk to memory in round $a+1$, and from memory to the network interface controller in round $a+2$.

3.2. Scheduling of parallel video workload

At the disk level, the system should decide when to initiate accesses when cache misses occur. Disk access has to be launched immediately if the pending workload is created for the leading block of a certain video; otherwise, service delay increases. For other workloads, there is no need to fetch the blocks right away because data transfer rate is typically much higher than the playback rate of any video. We define the longest time that a disk access operation can be delayed as the *maximum waiting time* (MWT), which equals the playback time of all blocks preceding the requested one. The *MWT* of the jth block of video i can be written as

$$MWT_{i,j} = \frac{(j-1)b}{r_i}, \tag{1}$$

where b is the block size and r_i the bit-rate of video i. If the workload is to access the jth block of video i, its deadline can be found via

$$Deadline_{i,j} = t_{req} + MWT_{i,j} + AD_{i,j-1}, \tag{2}$$

where t_{req} is the arrival time of the request, and $AD_{i,j-1}$ is the accumulated delay of the request up to the time the previous block is delivered. Given Eq. (2), we can apply the earliest-deadline-first (EDF) scheduling algorithm that maximizes disk utilization and minimizes delay [17]. However, we do not process a workload whose deadline is too far from the current time for two reasons. First, accessing these blocks in an unconstrained manner may lead to many out-of-order deliveries. Second, under EDF, the disk is likely to be kept busy fetching blocks for future workloads and has less opportunity to switch to low-power modes.

To prevent these undesirable effects, we apply a window constraint to the EDF scheduling. In other words, only workloads with deadlines within the window is scheduled in the current service cycle. While EDF can achieve optimal utilization, use of windowed EDF scheduling does not exhibit performance degradation with our algorithms, as we will discuss in Section 6.2.

3.3. Modeling of disk idle time

The disk idle time plays a central role in energy management [34]. The arrival of client requests in video servers typically follows the Poisson process [14,28]. Therefore, if only a single server is considered, the disk idle time is likely to be exponentially

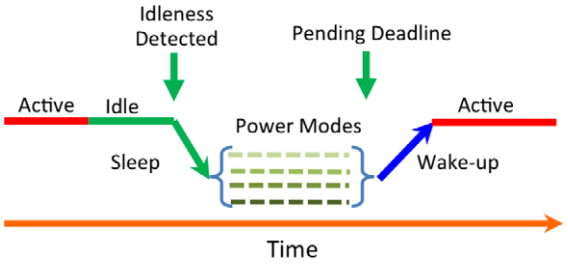


Fig. 3. Operation of disks with multiple power modes.

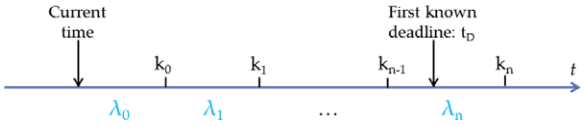


Fig. 4. The modeling of disk wake-up.

distributed. However, a more complicated model is needed for a parallel storage system.

Fig. 3 illustrates the operation of disks with multiple low power modes. A disk enters a sleep mode once it has been idle for a few rounds. Then, the disk has to switch back to the active mode when a pending deadline is approaching. The deadline that triggers this power saving operation can arise from either a past or a future request. For a workload from a past arrival, its corresponding deadline is given by Eq. (2). For a future workload, since the deadline can be delayed by at least *MWT* seconds after its arrival time, we have the following constraint

$$Deadline_{i,j} > t_{req} + MWT_{i,j} \tag{3}$$

in accessing the jth block of video i. With such a constraint on each of the blocks, we can model the appearances of the *unknown* deadlines as arrivals following a time-varying Poisson process.

Fig. 4 illustrates the model. For each disk, if we sort all non-leading blocks that are not cached according to their *MWT*s (denoted by $k_0, k_1, \ldots$), the future timeline can be divided into a finite number of intervals. Within the first interval, the arrival rate equals the aggregate request rate of all uncached leading blocks in the disk. Beyond that window, we need to include the arrival rates of uncached non-leading blocks: between k_i and k_{i+1}, the arrival rates of blocks with *MWT*s equal to k_i are added. We denote the arrival rates as λ_0, λ_1, etc.

If there are pending workloads that have not yet been processed in the current service cycle, the length of the future idle interval of the disk is bounded deterministically by t_D, the earliest deadline of all pending workloads. By denoting the disk idle time and the time until the appearance of the first unknown deadline as T_s and T_r, respectively, we have $T_s = \min(T_r, t_D)$. If there is no pending workload, $T_s = T_r$.

If we first assume $t_D \in (k_{n-1}, k_n]$, the time-varying Poisson process consists of n segments, within each of which the arrival rate is a constant. The pdf of T_s can be obtained by combining the time-varying Poisson process and the deterministic bound as

$$p(t) = \begin{cases} \lambda_0 e^{-\lambda_0 t}, & \text{if } t < k_0, \\ \alpha_i \lambda_i e^{-\lambda_i t}, & \text{if } t \in [k_{i-1}, \min(k_i, t_D)), \\ \alpha_n \delta(t - t_D) e^{-\lambda_n t_D}, & \text{if } t \geq t_D, \end{cases} \tag{4}$$

where $i = 1, 2, \ldots, c, n$ and $\alpha_x = \prod_{j=0}^{x-1} e^{-(\lambda_j - \lambda_{j+1})k_j}$.

If the deadline of the first known arrival is so early that $t_D < k_0$, we have $p(t) = \lambda_0 e^{-\lambda_0 t}$. On the other extreme, if $t_D > k_m$ where k_m is the largest *MWT* of all uncached blocks in the disk, we set $n = m$. Finally, if we do not have any pending workload, t_D goes to infinity. In this case, the last segment becomes $t \in (k_{m-1}, \infty)$, and no deterministic upper bound is imposed.

Table 2
Notation and meanings.

Symbols	Meanings
T	Length of one scheduling round
n_l	Maximum number of blocks a disk can process in one round
T_s	Idle interval length of the disk
T_r	Time until random-triggered wake-up
t_D	Time until the first unscheduled known deadline
k_i	The ith smallest *MWT* of the un-cached blocks in the disk
λ_i	The arrival rate in the ith time interval
P_i	Power level of the ith mode
R_i	Recovery time of the ith mode
O_i	Energy cost of disk wake-up from the ith mode
p_i	Average request rate of the ith video
$MWT_{i,j}$	The maximum waiting time of the jth block of the ith video
μ	The Lagrangian multiplier that imposes rate constraints

Eq. (4) serves as the foundation of the overall energy-delay optimization framework to be detailed in Sections 4 and 5. Its derivation is omitted here due to space constraint. Table 2 lists important notations used in this paper.

4. Optimized mode decisions with delay constraints

For the purposes of saving energy and preserving service quality, it is desirable to switch disks to low-power modes when such an action is unlikely to induce excessive service delay. It is therefore vital to decide the power mode whenever disks become idle. In this section, we describe two mode selection algorithms. First, we introduce the threshold-based approach that is widely applied in disk energy management [14,13,34,2,8]. Next, we propose the Prediction-based Mode Decision (PMD) algorithm and discuss how it achieves optimal power mode selections.

4.1. Threshold-based mode decision (TMD)

The threshold-based power mode decision (TMD) policy is built upon the concept of *break-even time* [2,34], which is defined as the minimum amount of idle time that justifies a power saving operation. For each power mode i and potential length of idle interval t, the energy consumption is $E_i(t) = P_i(t - R_i) + O_i$, where P_i, R_i and O_i are the power consumption, recovery time and recovery energy for mode i, respectively. Then, the minimum energy consumption can be obtained by plotting lines for $E_i(t)$ and finding the lower envelope of all lines [34] as shown in Fig. 5. The intersections t_1, t_2, t_3 are the break-even times.

If we could predict the future perfectly through a certain Oracle, it is possible to minimize disk energy by choosing the power mode that fits the lower envelop. However, since it is not feasible to know the future disk idle time in reality, a practical approach is to switch a disk into the next lower power mode if it has stayed idle at the current mode for a certain threshold time. It was proved in [11] that, when the break-even times are used as thresholds, its energy cost can be no worse than 2 times the energy cost of the perfect Oracle-based energy minimization scheme.

4.2. Prediction-based mode decision (PMD)

Although TMD exploits disk idleness, it does not take into consideration the characteristics of service workloads. Here, we leverage the model presented in Section 3 to predict the *optimal* power mode.

To evaluate the effect of different power modes on disk energy, we need to estimate how energy consumption varies when we choose different modes. To simplify the analysis, we assume that mode decisions do not affect the total service time and the average power consumption for the active periods. The assumption is valid

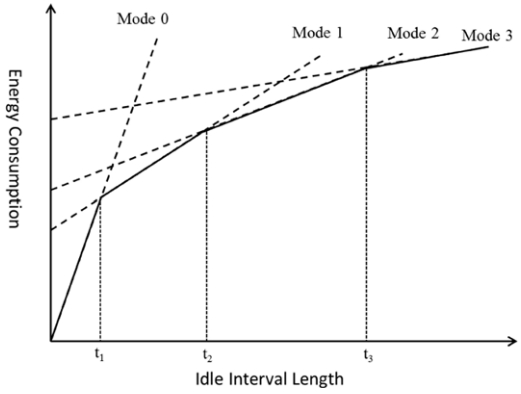


Fig. 5. Energy consumption for a 4-mode disk with three break-even times.

if we consider the disk behavior over a long period of time. Under this assumption, we use the *average power* of non-active periods to measure the energy cost of each power mode, which is in form of

$$\overline{P_i} = \frac{P_i \cdot T_s + O_i}{T_s + R_i}. \tag{5}$$

Another shortcoming of the TMD scheme is that it focuses on energy minimization but ignores the effect of low-power modes on service delay. To address this issue, we need to predict how service delay can be affected by choosing different modes. As such, we consider two types of delay. The first type arises from the fact that a disk is inaccessible during mode transition periods. Thus, all workloads due within such periods incur extra delays. The second type occurs when the number of pending workloads exceeds the per-round admission limit. If this happens, the requests that arrive later will suffer from further delay.

We employ a simple approach to estimate the sum of these two delay types. Let λ_k be the arrival rate in the segment where mode transition takes place (according to the time-varying Poisson model in Section 3.3), and assume that $a = \frac{\lambda_k R_i}{n_l}$ is an integer, the total delay caused by mode transition can be written as

$$D_i = n_l \sum_{j=1}^{a} \left(\frac{R_i}{2} + j * T \right) = \frac{TR_i^2}{2n_l}\lambda^2 + \frac{(T+R)R_i}{2}\lambda, \tag{6}$$

where T is the round length and n_l is the admission limit per round.

In general, deeper power saving modes lead to lower energy consumption but incur more mode transition penalties. To achieve energy minimization without heavy delay penalties, we can impose a constraint on service delay, denoted by D_c. This constrained minimization takes the following form:

$$\min\{E\}, \quad \text{subject to } D < D_c. \tag{7}$$

The constrained optimization problem in Eq. (7) can be solved using the Lagrangian formulation [6]. That is, it can be converted to an unconstrained minimization problem in form of

$$\min\{C\}, \quad \text{where } C = E + \mu D, \tag{8}$$

where μ is a parameter called the Lagrangian multiplier. By minimizing the Lagrangian cost function, C, for a specific μ, we equivalently solve the optimization problem for a particular D_c in Eq. (7).

In the proposed PMD scheme, we estimate the energy-delay cost function for different power modes and choose the one that minimizes it, thereby optimizing the overall system performance which combines the effects of energy and delay. To do so, we make use of the distribution of the disk idle time T_s as formulated in Eq. (4) and estimate the Lagrangian cost function in Eq. (8). We use $\overline{P_i}$ in Eq. (5) as the energy cost metric and D_i in Eq. (6) as the delay metric. The final objective function J_i is defined as the expected energy-delay cost for power mode i, which can be written as

$$\begin{aligned} J_i &= \mathbb{E}(\overline{P_i} + \mu D_i) = \int_t (\overline{P_i} \mid T_s = t) p(t)\, \mathrm{d}t + \mu \mathbb{E}(D_i) \\ &= (O_i - P_i R_i) \int_t \frac{p(t)}{t + R_i} \mathrm{d}t + P_i + \mu \mathbb{E}(D_i). \end{aligned} \tag{9}$$

As shown in Eq. (6), the delay term D_i is a quadratic function of both λ_k and R_i. While R_i is a constant, λ_k will change along time. Then, the expected delay cost can be written as

$$\mathbb{E}(D_i) = \frac{TR_i^2}{2n_l^2}\overline{\lambda^2} + \frac{(T+2R)R_i}{2n_l}\overline{\lambda}, \tag{10}$$

where $\overline{\lambda^r} = \int_t \lambda^r(t) p(t) \mathrm{d}t$. The term $\overline{\lambda^r}$ can be obtained by substituting the pdf of T_s (Eq. (4)) into Eq. (10). Then, we can obtain the cost function J_i for each mode i. To select the optimal power mode M, we minimize the objective function J:

$$M = \operatorname{argmin} J_i.$$

Since the distribution of idle intervals changes with cache replacement operations and the arrival sequence of requests, the objective function J_i has to be evaluated in real-time. One issue is that the number of time segments defined by $k_0, k_1, \ldots, c$ can be large if we have a lot of video blocks stored on the disk. To simplify the calculation, we can group blocks with similar *MWT*s together and reduce the number of segments. In addition, the integral in Eq. (9) is not bounded. Fortunately, we do not need very precise calculation when comparing a small number of power modes. Therefore, fast numerical methods suffice to serve our purpose.

The PMD algorithm is applied with the following steps once a disk has been idle for a short period of time (chosen to be 4 rounds in our experiment):

1. Update the disk wake-up model as shown in Fig. 4 by evaluating the time intervals, the arrival rates and the deterministic bound.
2. Calculate the delay cost metric using Eq. (10).
3. Calculate the energy-delay cost function according to Eq. (9) for each power mode. Select the optimal mode that minimizes the cost. The Lagrangian multiplier μ can be adjusted to control the delay level. A larger μ will lead to lower service delays.
4. Switch the disk to the selected mode in the beginning of next round.

PMD demands some workload information including the total arrival rate and the popularity distribution of videos. These values are not expected to change frequently over time [12] and can be obtained by examining history statistics and applying simple prediction techniques such as the moving average model.

5. Energy-aware cache management

We examined how to make the optimal power mode decisions by evaluating the energy-delay cost associated with each mode in the last section. In addition to optimal power mode selection, the disk access pattern is crucial to overall energy efficiency [34].

One way to alter the disk access sequence is to change the size of the scheduling window. As we show in Section 6.2, an enlarged scheduling window increases the length of the disk idle period and, therefore, improves energy efficiency to some extent. However, its effect is limited since scheduling does not change the workload arrival sequence. To schedule a workload, the system should know its deadline, which is *MWT* seconds after its request arrival time. Thus, workloads with shorter *MWT* are more difficult to schedule in a timely manner and tend to trigger mode transitions. The transition overhead caused by these *less schedulable* workloads cannot be reduced by increasing the scheduling window size, making it a

major bottleneck to energy efficiency. More details on the effects of scheduling window size will be discussed in Section 6.2.

To reshape the disk access sequence for further energy saving, we can remove the bottleneck of mode transition overhead by employing energy-aware caching algorithms. In this section, we propose two such algorithms. Unlike traditional caching algorithms such as LFU and LRU which only reduce disk access, our algorithms optimize cache utilization for energy efficiency. First, we derive the cost functions for disk access and recovery. Then, we develop a new caching utility for cache replacement based on these costs. Finally, we design a prefetching algorithm that effectively reduces the number of mode transitions for disks.

5.1. Disk access and mode transition costs

Disk access operations fetch video blocks to the cache. Cache replacement algorithms decide which blocks to be kept in the cache and which ones to be discarded. While traditional cache replacement algorithms reduce the number of disk access and the corresponding energy consumption, they do not address the mode transition overhead, which can be equally important in overall energy efficiency. Essentially, we need a caching policy that takes into account both the disk access and the mode transition costs.

To consider the energy and delay effects jointly, we adopt the same Lagrangian cost function in the following derivation:

$$C = E + \mu D, \tag{11}$$

where the energy term needs to be represented by the average power. The energy-efficient cache utility function should measure the reduction of the joint energy-delay cost if a block is in the cache. This is equivalent to the additional cost the block incurs on the system if it is *not* in the cache. Using this guideline, we can derive the cache utility as follows.

First, we calculate the cost of the block if it is requested and causes disk accesses. To simplify the derivation, we assume that this happens when the disk is already awake. The scenario of the workload waking up the disk will be discussed later. Energy-wise, the disk processes one more workload. Without the workload, the disk can process anything from 1 to $n_l - 1$ workloads in that round. We thereby use the average increment in average power to quantify this effect. For the delay cost, assuming there is no additional queuing delay, the workload incurs a one-round cache miss delay only if it is a leading block. Summarizing the above, the cost of accessing block (i, j) (the jth block of video i) is:

$$C_a(i,j) = \frac{1}{n_l}\sum_{w=0}^{n_l-1}(P_a(w+1) - P_a(w)) + \mu T \mathbb{I}_{j=0}, \tag{12}$$

where $P_a(x)$ is the average power of a round when x workloads are accessed by a disk, and $\mathbb{I}_{j=0}$ is 1 only when it is a leading block and stays 0, otherwise.

Next, we calculate the cost of a mode transition when it takes place. If a workload happens to cause a disk that operates at mode m to wake up, the additional delay for the workload associated with block (i, j) can be written as

$$l_{i,j} = \max(T + R_m - MWT_{i,j}, 0), \tag{13}$$

where T is the scheduling round length and R_m stands for the recovery time. In addition to the delay penalty, we need to calculate the energy term in the cost function, which is simply the average power of transitional rounds. Obviously, the energy cost depends on the current disk mode. However, even if the disk is currently active, cache replacement decisions can still affect mode transitions in the future. To account for this effect, we use the power mode of the last sleep cycle if the disk is in active mode. If being triggered by block (i, j), the mode transition cost becomes

$$C_r(i,j) = \frac{O_m}{R_m} + \mu \cdot l_{i,j}, \tag{14}$$

where m is the current power mode or the previous sleep mode if the disk is currently active.

5.2. Joint Energy-Delay Optimized Cache (EDOC) Replacement

An ideal utility should combine the above two cost functions. To do so, we calculate the expected number of each of these events.

For the former event (accessing the disk for block (i, j)), the expected number of accesses should be proportional to the average request rate of video i, denoted by p_i, if we look at long-term statistics. However, the existence of pending workloads shapes the access patterns in short term, which in turn affects cache efficiency. The impact of this short-term information varies for different blocks. The most popular blocks, for example, are expected to be requested a lot more often in the long run than the least popular blocks in the cache. Therefore, short-term knowledge (i.e. the deadlines of pending workloads) are not so important for the popular blocks as for the unpopular ones.

To make the best use of both long-term statistics and the existing information of pending deadlines, we define the access frequency as the expected number of workloads over a certain time window. The size of the window decides how much short-term knowledge will be taken into account. The window size will vary from block to block, depending on how long the block is expected to stay in cache. For popular blocks, we use a larger window so that we can benefit from the long-term expectation that they will be accessed often. For unpopular blocks, the turnover rate is expected to be high, which means that the block can be swapped out of the cache very soon. In this case, the available deterministic information brings more benefits than its popularity. Thus, it is advantageous to apply a small window for them.

In short, the size of the time window should relate to the popularities of videos, which follow an exponential distribution, as shown in Fig. 2(b). Therefore, it is reasonable to define the time window based on an exponential function. But if the block has pending requests, the time window should be large enough to include the earliest time all these workloads can be served. Based on these two principles, we design the window function as follows.

For each of the cached blocks (i, j), we can obtain a rank, $r_{i,j}$, based on its popularity. If the block has pending requests, the deadline of the latest workload is denoted by $T_f(i, j)$. Then, the time window is calculated by

$$L_{i,j} = \max(\alpha e^{\beta r_{i,j}}, T_f(i,j) - T_w), \tag{15}$$

where T_w is the size of scheduling window we discussed in Section 3.2, and α and β are trained parameters (selected to be 2000 and -4.6×10^{-4}, respectively, in our experiments).

After getting the window size, we can estimate the *access frequency* by combining the deterministic and stochastic information. Recalling the constraints on deadlines due to MWT, we have

$$f_{i,j} = \begin{cases} \dfrac{n}{L_{i,j}} & \text{if } L_{i,j} \le MWT_{i,j}, \\ \dfrac{p_i\lambda(L_{i,j} - MWT_{i,j}) + n}{L_{i,j}} & \text{otherwise,} \end{cases} \tag{16}$$

where n is the number of pending workloads for block (i, j).

The latter event (disk waking up by block (i, j)) is part of the disk re-activation process. Its expectation is proportional to the contribution of block (i, j) to the time-varying Poisson process as described in Section 3.3. With the inclusion of block (i, j), we can

recalculate the new pdf of disk idle time T_s and its expectation using Eq. (4). Assuming that this new disk idle time is T_s', the expected increase in the number of mode transitions is proportional to

$$\delta\lambda_{i,j} = \frac{1}{\mathbb{E}(T_s')} - \frac{1}{\mathbb{E}(T_s)}. \tag{17}$$

Finally, we can combine the two costs with proper weights by their expected numbers of occurrences to obtain the final utility function:

$$C_{i,j} = f_{i,j}C_a(i,j) + \delta\lambda_{i,j}C_r(i,j). \tag{18}$$

Note that as *MWT* becomes smaller, the weight of C_r grows more quickly. In particular, if *MWT* is 0, which means the block is a leading block, $\delta\lambda_{i,j}$ is very close to p_i. In this case, the mode transition penalty has almost the same weight as the disk access cost. Furthermore, the disk access cost for leading blocks is greater as there will be one-round cache miss delay. Thus, the utility function biases greatly towards leading blocks. On the contrary, blocks with very long *MWT*s have little contribution to the wake-up penalty.

In the Energy-Delay Optimized Cache Replacement (EDOC) algorithm, Eq. (18) is evaluated whenever there is a cache miss and the cache is full. The block with the least caching utility is victimized and swapped out of the cache. In this way, the cache replacement decision leads to a lower *expected* energy-delay cost than LRU/LFU. The operation of EDOC is interconnected with PMD. On the one hand, the caching utility of each block depends on the selected power modes. On the other hand, EDOC reshapes the disk access patterns, which affect future power mode decisions. We present the performance of EDOC in Section 6.

5.3. Prediction-based energy-efficient prefetching (PEEP) algorithm

The main idea of EDOC is to put less schedulable blocks into the cache so that the disk-access workload is easier to schedule. The cache replacement operations, however, only occurs when there are cache misses, which are rare under low service loads. To further improve the "energy awareness", we develop a proactive caching policy that fetches less schedulable blocks selectively into the cache before disks go to sleep. Since a disk will stay idle for a few rounds before it enters the sleep mode as described in Section 4, the time period can be utilized for prefetching.

It is important to point out that the number of disk accesses may increase with prefetching, as additional blocks need to be fetched from disks. Thus, there is a trade-off between the number of disk accesses and workload schedulability (or mode transition penalties). To find a balance, we need to decide the number of blocks to be prefetched. As the time of prefetching decision is close to the time when the system decides for a disk to go to a sleep mode, the two decisions can be combined together. Mathematically, we can incorporate the cost of prefetching into the Lagrangian cost function in Eq. (9).

Recall that the energy cost is the average power of non-active periods, which include sleep and transition periods. To incorporate the prefetching effect, its associated energy and time need to be added. If we prefetch m blocks and switch the disk to the ith mode, the energy cost can be evaluated as

$$\overline{P}(i,m) = \frac{E_p(m) + P_i \cdot T_s(m) + O_i}{T_p(m) + T_s(m) + R_i}, \tag{19}$$

where $E_p(m)$ is the prefetching energy and $T_p(m)$ is the prefetching time. These two terms only depend on m and can be easily calculated from the disk power model. The sleep time, T_s, is a function of m since the prefetching decision affects the disk idle time distribution.

The delay term in the cost function has the same form as that in Eq. (10), although the delay measure D becomes a function of both mode i and the number of prefetching blocks, m.

In the proposed prefetching algorithm, we estimate the expected cost function for each selection of power mode i and each possible number of prefetching blocks m. With prefetching incorporated into the formulation, the arrival rates $\lambda_0, \lambda_1, \ldots$ will vary with different prefetching decisions. Also, the distribution $p(t)$ depends on m. Following the similar procedures as we did for PMD in Section 4.2, the objective function can be written as

$$\begin{aligned} J(i,m) &= \mathbb{E}(\overline{P}(i,m) + \mu D(i,m)) \\ &= (E_p(m) + O_i - P_i R_i) \int_t \frac{p_m(t)}{t + R_i + T_p(m)} \mathrm{d}t \\ &\quad + P_i + \mu\mathbb{E}(D(i,m)). \end{aligned} \tag{20}$$

The calculation of $p_m(t)$ for different m is straightforward as we only need to change the set of arrival rates $\lambda_0, \lambda_1, \ldots$. With the same set of arrival rates, we can use Eq. (10) to derive $D(i,m)$ for different m. Before making a prefetching decision, we need to find the set of blocks, which reside in the disk, to be prefetching candidates. For this purpose, we can use the caching utility function in Eq. (18).

Since prefetching incurs unnecessary accesses to disks for the benefit of better schedulability, it calls for precaution. In particular, we should not allow too many prefetching accesses. Moreover, there is a risk that the disk may not enter any sleep mode after prefetching. In this case, these extra accesses can be wasted. To address these two issues, we may limit the number of blocks to be prefetched and postpone the prefetching operation. They are implemented as follows.

Recall that we allow a fixed number of blocks, n_l, to be accessed from a disk in each round. To maximize utilization, we should make the disk as busy as possible. Thus, the prefetching time can be decided by the number of prefetched blocks via

$$T_p(m) = \left\lceil \frac{m}{n_l} \right\rceil T. \tag{21}$$

To restrict the number of prefetched blocks, we impose a cap on the number of prefetching rounds. Let m_p denote the maximum number of prefetching rounds and n_p be the actual number of prefetching rounds ($n_p = \lceil \frac{m}{n_l} \rceil$). The disk does not enter any sleep mode until it has been idle for a few rounds, and we use n_t to denote this threshold. To ensure maximum disk sleep time, we demand $m_p < n_t$.

The whole prefetching process can be summarized below.

1. When the disk has been idle for $n_t - m_p$ rounds, we go through the list of blocks and find the uncached ones with the highest m_p caching utilities as our prefetching candidates.
2. Perform the prefetching decision by calculating the prefetching utilities for all modes and all prefetching options (from 1 to m_p) using Eq. (20). Decide the optimal decision pair (the ith mode, m blocks to prefetch).
3. Calculate the actual prefetching rounds $k = \lceil \frac{m}{n_l} \rceil$.
4. If the disk has been idle for $n_t - k$ rounds, start prefetching blocks.
5. If there are new workloads due in the current round, abort prefetching operations. Otherwise, wait until prefetching finishes and switch the disk to the ith low power mode.

The prediction-based energy-efficient prefetching (PEEP) algorithm performs the joint decision of power mode selection and prefetching. It requires the decision routine to be run by $m_p \cdot n_t$ times, while PMD only requires n_t iterations. The complexity of the two decision routines is almost the same. Since the complexity of PMD is low, PEEP will not be computationally intensive either.

Table 3
Simulation parameters.

Number of disks	20
Idle power	11.6 W
Active power	16.3 W
Cache memory size	32 GB
Block size	1.5 MB
Round length	0.15 s
Admission limit per round	4

In this work, we focus on the disk subsystem. One potential issue with PEEP is that prefetching might increase energy consumption of other components (e.g. CPU and cache). According to our measurement, prefetching only accounts for less than 5% of all cache replacement operations; yet the number of cache hit from these pre-fetched blocks constitutes more than 30% of those from all the replaced blocks. Therefore, the benefit of prefetching is likely to outweigh the potential overhead. Furthermore, the energy models for other components can be easily incorporated into Eq. (20), thereby making more accurate prefetching decisions.

6. Experimental results and evaluation

In this section, we evaluate the performance of three proposed schemes:

1. PMD with LFU cache replacement (PMD);
2. PMD with EDOC cache replacement (PMD+EDOC);
3. PEEP with EDOC cache replacement and up to 8 blocks to prefetch each time (PEEP+EDOC).

The benchmark algorithm is the threshold-based mode decision (TMD) scheme as introduced in Section 4.1.

6.1. Simulation environment

To evaluate the proposed algorithms, we simulated a storage server system consisting of an application server, a global cache and 20 storage nodes, each of which contains one disk. Our algorithm runs on the application server, which not only stores the metadata, but also calculates schedules (including prefetching schedules) and mode decisions and passes them down to individual I/O nodes. After receiving the information, each I/O server node can take care of disk scheduling and mode transitions accordingly. The placement of video blocks has been discussed in Section 3.1.

The disk specifications were taken from the data sheet of the Seagate Cheetah 15k.7 model [25]. For system parameters such as the number of disks, the round length and the block size, we set them according to the size of our dataset. The admission limit is obtained by calculating the number of video blocks the disk can read per round, taking into account the seek and rotational overhead. Depending on the workload, the active energy consumption per round is calculated using the model proposed in [29]. All related disk and system parameters are shown in Table 3. The low power modes were simulated according to Table 1.

We generated client requests using an online YouTube dataset available at [4]. We used the newest available dataset which contains the view counts, sizes and bit-rates for 161,085 videos. Since the typical video request arrival pattern follows a Poisson process [14,28], we generated requests by directing exponential arrivals to videos according to their relative popularities.

6.2. Effects of scheduling window

The scheduling window concept was introduced in Section 3.2. The window size determines the degree of workload concentration in time. Workloads are more scattered around with a smaller window, while they tend to be served together with a larger one.

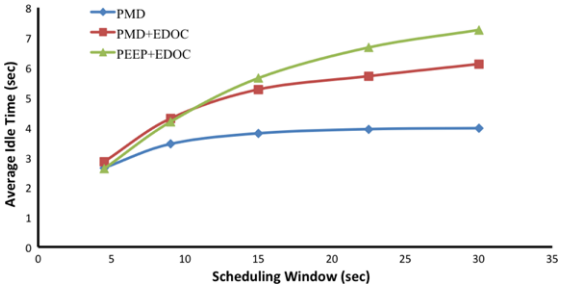


Fig. 6. The average idle time as a function of the scheduling window size for three proposed algorithms.

For energy management, the scheduling window size plays an important role because it affects the length of the disk idle time. We show the average idle time as a function of the scheduling window size for the three proposed algorithms in Fig. 6. We see that a larger scheduling window will increase the average disk idle time in all algorithms. This is advantageous because it allows us to apply low power modes more aggressively. However, the potential saving achieved by these low power modes might be offset by the incurred penalties. This tradeoff is the main issue in traditional disk energy management algorithms that do not have a delay control mechanism.

To evaluate how PMD performs under different scheduling window sizes, we plot the energy-delay curves in Fig. 7(a) by simulating the system under PMD for 30 min. Service delays were calculated for each request by summing up the initial latency and all extra delays incurred on non-leading blocks. Different average delay levels were achieved using different values of μ. We see from Fig. 7(a) that there is a considerable amount of improvement of the energy-delay performance in the mid- to high-delay regions when the window size increases from 3 to 22.5 s. Moreover, the energy consumption levels under these two window sizes are virtually the same in the low-delay regions. This indicates that PMD can take the advantage of a large scheduling window size without incurring penalties on service delay.

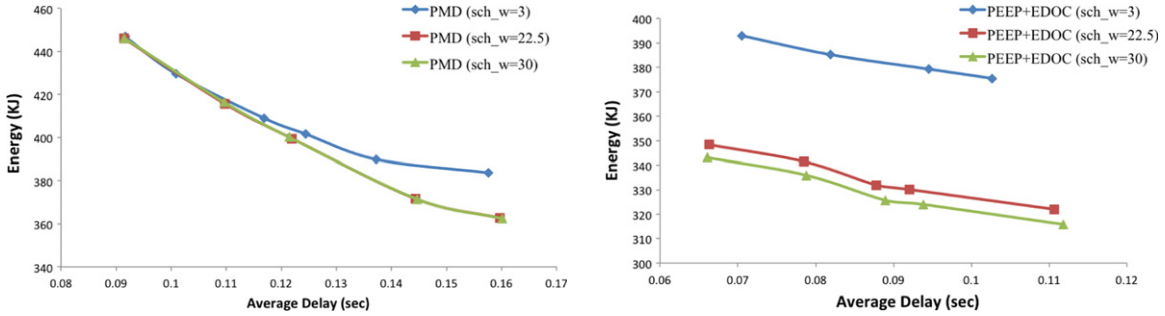
Furthermore, we see that the length of disk idle period saturates in Fig. 6 when we increase the window size beyond 20 s if PMD alone is applied. This is consistent with the fact that there is no further improvement in energy-delay performance when we increase the window size from 22.5 to 30 s as shown in Fig. 7(a). This is caused by the bottleneck of the mode transition overhead, which cannot be removed by enlarging the scheduling window size, as explained in Section 5.

However, with PMD+EDOC and PEEP+EDOC, the average disk idle time continues to increase well beyond 20 s as shown in Fig. 6. As a result, their energy-delay performance continues to improve as the window size increases. One example is shown in Fig. 7(b), where the performance of PEEP+EDOC continues to improve when the window size increases from 22.5 to 30 s.

To conclude, the proposed algorithms enable us to apply a large scheduling window to achieve more energy saving without sacrificing the service delay. In particular, PMD+EDOC and PEEP+EDOC lead to better cache utilization and improved disk access sequence since disk-access workloads become more schedulable. As a result, we can take more advantage of large scheduling window sizes.

6.3. Comparison of energy-delay performance

Based on the discussion in the last subsection, we use a scheduling window of 22.5 s for PMD and increase the window size to 30 s for PMD+ EDOC and PEEP+EDOC. We evaluate the performance



(a) PMD. (b) PEEP+EDOC.

Fig. 7. The energy-delay performance with different scheduling window sizes.

Table 4
Energy saving of PMD over TMD.

Average delay	30 ms	40 ms	50 ms	100 ms
0.12 s	5.7%	7.5%	8.5%	11.0%
0.14 s	7.6%	9.7%	11.2%	15.0%

of all three proposed algorithms and the benchmarking TMD algorithm under different workload levels by plotting their energy-delay curves in Fig. 8. The inter-arrival time is decided by the percentage of CDN-cached videos, which is between 30 and 100 ms for CDN-caching percentages ranging from 3% to 15%.

The value of μ is adjusted to impose different constraints on service delay in the proposed algorithms. In contrast, there is no mechanism to control delay in TMD. Therefore, its scheduling window has to be changed to adjust the average length of the disk idle time, thereby achieving different service delay levels.

6.3.1. Performance of PMD

PMD consistently outperforms TMD in all settings. To better evaluate the performance, the percentages of energy saving of PMD over TMD with two averaged delay levels (0.12 and 0.14 s) and four inter-arrival times (30, 40, 50 and 100 ms) are shown in Table 4.

As shown in Table 4, the percentage of energy reduction of PMD over TMD ranges from 7.6% to 15.0% in the longer delay scenario (0.16 s), and from 5.7% to 11.0% in the shorter delay scenario (0.12 s). We see that PMD saves more energy for longer inter-arrival times, which indicates lower service loads. This is expected because the amount of disk idleness will inevitably reduce when we increase the load. Since PMD does not address energy consumption in active periods, the portion of energy to be optimized is smaller in higher load periods.

Energy saving is more significant if we allow longer delay in the system since there is more room to utilize the low power modes. In these regions, PMD makes more intelligent use of the lower power modes by prediction than TMD, under which energy saving achieved by the lower power modes is largely offset by the high delay penalty. Considerable energy reduction is also achieved in the shorter delay scenario. The main reason is that PMD allows the system to use a large scheduling window while having a firm control on service delay. However, PMD does not gain as much in this case because the delay constraint prevents the system from selecting the lower power modes.

6.3.2. Performance of PMD+EDOC and PEEP+EDOC

As compared to PMD with traditional caching that only achieves significant energy saving in high-delay regions, PMD+EDOC and PEEP+EDOC improve energy efficiency for all settings across all delay levels. The low-delay region is of particular interest since VSS users have become increasingly sensitive to service delays. To better visualize the benefits of EDOC and PEEP under tight delay constraints, we plot the energy consumption levels in Fig. 9 for average delay levels of 0.1 and 0.12 s. Also, the saving percentages are listed in Table 5, indicating that the saving for all service loads is significant and consistent.

Table 5
Energy saving under low delay levels.

	30 ms	40 ms	50 ms	100 ms
(a) Average delay: 0.1 s				
PMD+EDOC	23.0%	21.8%	23.2%	22.9%
PEEP+EDOC	27.0%	26.9%	28.6%	29.1%
(b) Average delay: 0.12 s				
PMD+EDOC	22.6%	23.1%	23.7%	24.0%
PEEP+EDOC	26.1%	27.9%	29.3%	28.7%

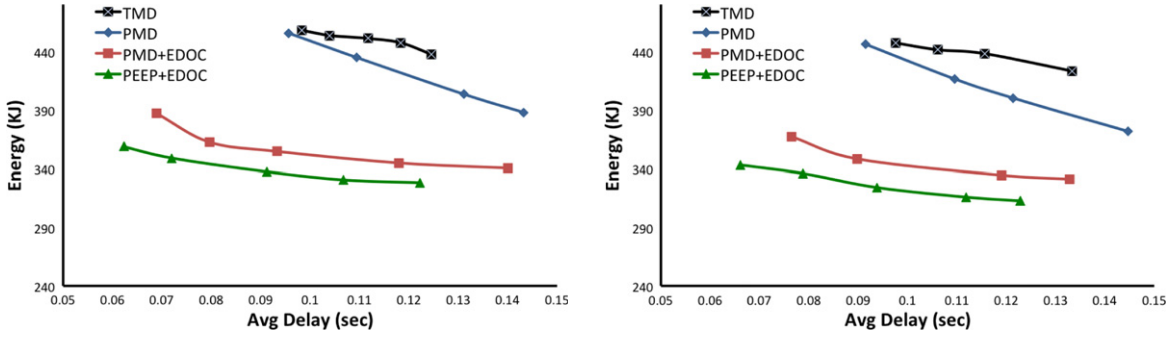
As mentioned earlier, PMD alone does not gain a lot of energy saving in low-delay regions because the lower power modes are unlikely to be selected due to heavy delay penalties of mode transitions. For PMD+EDOC and PEEP+EDOC, we can effectively reduce the number of mode transitions by getting the troublemakers (the less schedulable blocks) in the cache.

Besides, PMD+EDOC and PEEP+EDOC achieve consistent improvement across different service loads. Recall that PMD is less effective when the service load is higher. The reason is that idle periods make up a smaller percentage in this case. Since PMD with EDOC and PEEP can make non-active periods longer and more continuous, their improvement in the high load region is comparable to that in the low load region.

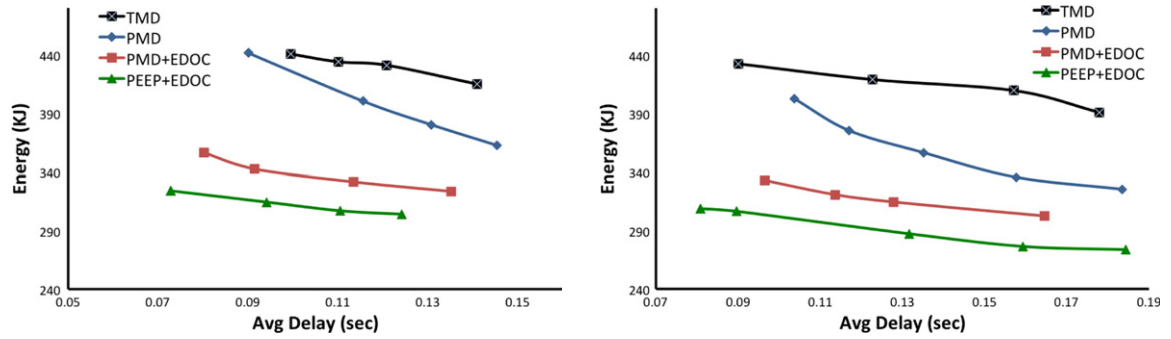
Furthermore, we observe that PMD+EDOC and PEEP+EDOC are able to achieve very low delay levels (0.05 to 0.07 s), which are not attainable with TMD or PMD alone. The ability to offer such low-delay guarantee makes our scheme a promising approach for practical real-time Internet services.

In addition to this current disk model, we also simulated our system for future disks that are equipped with more power modes, possibly utilizing the DRPM technology [10]. We simulated six power modes as listed in Table 6(a). Compared to® PowerChoice™ [24], two intermediate power modes are used. The resulting energy savings under PEEP+EDOC are reported in Table 6(b). As we can see, with the introduction of additional intermediate power modes, we are able to save considerably more energy.

To summarize, PMD+EDOC and PEEP+EDOC optimize cache utilization for the benefit of energy efficiency. The increased schedulability of workloads leads to longer average idle time and reduced mode transition overheads. As a result, we observe consistent and significant energy saving by both PMD+EDOC and PEEP+EDOC in the low delay and high service load scenario, which is not achievable using PMD alone. Finally, we find out that our algorithms can work better if more power modes are provided.

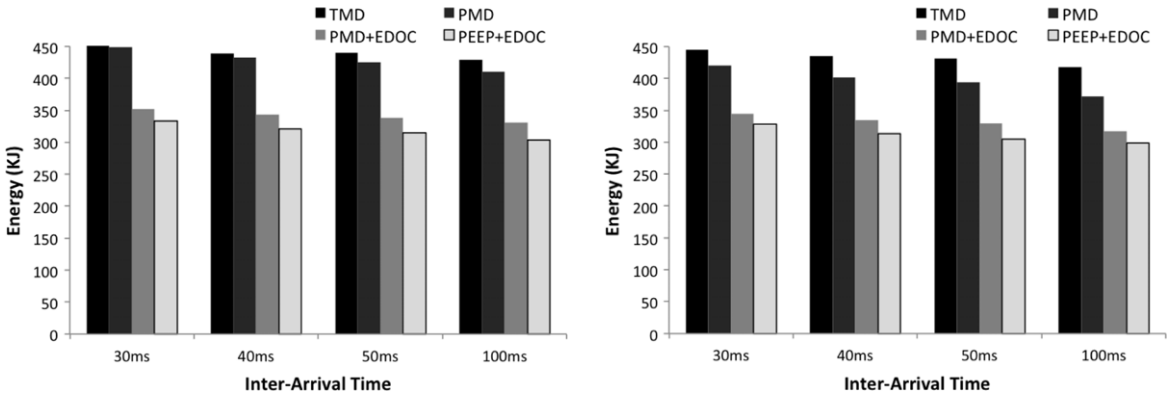


(a) Inter-arrival time: 30 ms. (b) Inter-arrival time: 40 ms.



(c) Inter-arrival time: 50 ms. (d) Inter-arrival time: 100 ms.

Fig. 8. Comparison of the energy-delay performance of four algorithms with different inter-arrival times.



(a) Average delay: 0.1 s. (b) Average delay: 0.12 s.

Fig. 9. Comparison of energy consumption with low delay levels.

Table 6
Parameters and results for a possible future disk model.

(a) Power modes					
Mode	L1	L2	L3	L4	L5
Idle power (W)	10.3	9.28	7.54	6	5.3
Recovery time (s)	0.15	0.3	0.9	3	8

(b) Energy saving under PEEP+EDOC				
Delay	30 ms	40 ms	50 ms	100 ms
0.1 s	31.2%	33.0%	34.1%	35.1%
0.12 s	28.7%	32.1%	33.5%	36.1%

7. Conclusion and future work

As compared to the threshold-based mode decision (TMD) approach, our prediction-based mode decision (PMD) algorithm reduces the energy consumption by 5.7% $\sim$ 15.0% under different service load and delay constraints. Furthermore, the proposed cache management algorithms offer additional energy efficiency. Overall, the disk subsystem saves 22% $\sim$ 29% of the total energy consumption by applying EDOC and PEEP under very tight delay requirements. The saving percentage can go up to 36% for disks with two additional power modes.

This work can be extended in several ways: First, better data placement policies can be developed to further improve the energy-delay performance. Second, one can investigate the energy consumption of memory cache. It will also be interesting to consider the storage and service of scalable video with a multi-layer representation.

References

[1] C. Anderson, The Long Tail: Why the Future of Business Is Selling Less of More, Hyperion, 2006.

[2] E.V. Carrera, E. Pinheiro, R. Bianchini, Conserving disk energy in network servers, in: Proceedings of the 17th Annual International Conference on Supercomputing, ICS'03, ACM, New York, NY, USA, 2003, pp. 86–97. http://doi.acm.org/10.1145/782814.782829 http://dx.doi.org/10.1145/782814.782829.
[3] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, S. Moon, I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system, in: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC'07, ACM, New York, NY, USA, 2007, pp. 1–14. http://doi.acm.org/10.1145/1298306.1298309 http://dx.doi.org/10.1145/1298306.1298309.
[4] X. Cheng, C. Dale, J. Liu, Dataset for statistics and social network of youtube videos, 2008, http://netsg.cs.sfu.ca/youtubedata/.
[5] Cisco Systems Inc., Cisco visual networking index—forecast and methodology, 2010–2015, Cisco White Paper, 2011. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf.
[6] Hugh Everett III, Generalized lagrange multiplier method for solving problems of optimum allocation of resources, Oper. Res. 11 (1963) 399–417.
[7] D. Forte, A. Srivastava, Energy-aware video storage and retrieval in server environments, in: Green Computing Conference and Workshops, IGCC, 2011 International, 2011, pp. 1–6. http://dx.doi.org/10.1109/IGCC.2011.6008589.
[8] R. Garg, S.W. Son, M. Kandemir, P. Raghavan, R. Prabhakar, Markov model based disk power management for data intensive workloads, in: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 76–83. http://dx.doi.org/10.1109/CCGRID.2009.67 http://dx.doi.org/10.1109/CCGRID.2009.67.
[9] D. Gemmell, H. Vin, D. Kandlur, P. Venkat Rangan, L. Rowe, Multimedia storage servers: a tutorial, Computer 28 (1995) 40–49.
[10] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, H. Franke, DRPM: dynamic speed control for power management in server class disks, in: Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on, 2003, pp. 169–179. http://dx.doi.org/10.1109/ISCA.2003.1206998.
[11] S. Irani, R. Gupta, S. Shukla, Competitive analysis of dynamic power management strategies for systems with multiple power savings states, in: Proceedings of the Conference on Design, Automation and Test in Europe, IEEE Computer Society, 2002, p. 117.
[12] X. Kang, H. Zhang, G. Jiang, H. Chen, X. Meng, K. Yoshihira, Measurement, modeling, and analysis of internet video sharing site workload: A case study, in: IEEE International Conference on Web Services, ICWS '08, 2008, pp. 278–285, http://dx.doi.org/10.1109/ICWS.2008.28.
[13] H. Kim, E.J. Kim, R.N. Mahapatra, Power management in raid server disk system using multiple idle states, in: Proceedings of International Workshop on Unique Chips and Systems, UCAS, 2005, p. 5359.
[14] M. Kim, M. Song, Saving energy in video servers by the use of multi-speed disks, Circuits and Systems for Video Technology, IEEE Trans. PP (2011) 1.
[15] J. Lee, Parallel video servers: a tutorial, Multimedia, IEEE 5 (1998) 20–28.
[16] J. Lee, P. Wong, Performance analysis of a pull-based parallel video server, IEEE Trans. Parallel Distrib. Syst. 11 (2000) 1217–1231.
[17] C.L. Liu, J.W. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, J. ACM 20 (1973) 46–61.
[18] A. Manzanres, X. Ruan, S. Yin, M. Nijim, W. Luo, X. Qin, Energy-aware prefetching for parallel disk systems: Algorithms, models, and evaluation, in: Eighth IEEE International Symposium on Network Computing and Applications, NCA 2009, 2009, pp. 90–97. http://dx.doi.org/10.1109/NCA.2009.29.
[19] X. Mountrouidou, A. Riska, E. Smirni, Adaptive workload shaping for power savings on disk drives, in: ICPE, 2011, pp. 109–120.
[20] Nexsan, Nexsan energy efficient automaid technology. http://www.slideshare.net/socialnexsan/nexsan-auto-maid-marketing-report, 2009.
[21] A.E. Papathanasiou, M.L. Scott, Energy efficient prefetching and caching, in: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '04, USENIX Association, Berkeley, CA, USA, 2004 pp. 22–22. http://dl.acm.org/citation.cfm?id=1247415.1247437.
[22] E. Pinheiro, R. Bianchini, Energy conservation techniques for disk array-based servers, Proceedings of the 18th Annual International Conference on Supercomputing ICS 04, 2004, p.68.
[23] D. Ronca, A brief history of netflix streaming, 2013.
[24] Seagate Technology, Reducing storage energy consumption by up to 75%. http://www.seagate.com/files/docs/pdf/whitepaper/tp608-powerchoice-tech-provides-us.pdf, 2011.
[25] Seagate, Seagate cheetah 15k.7 sas product manual, 2010. http://www.seagate.com/staticfiles/support/disc/manuals/enterprise/cheetah/15K.7/100516226a.pdf.
[26] D. Sitaram, A. Dan, Multimedia Servers: Applications, Environments, and Design, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.
[27] S.W. Son, M. Kandemir, Energy-aware data prefetching for multi-speed disks, in: Proceedings of the 3rd Conference on Computing Frontiers, CF'06, ACM, New York, NY, USA, 2006, pp. 105–114. http://doi.acm.org/10.1145/1128022.1128038 http://dx.doi.org/10.1145/1128022.1128038.
[28] W. Tang, Y. Fu, L. Cherkasova, A. Vahdat, Medisyn: a synthetic streaming media service workload generator, in: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, NOSSDAV '03, ACM, New York, NY, USA, 2003, pp. 12–21. http://doi.acm.org/10.1145/776322.776327 http://dx.doi.org/10.1145/776322.776327.
[29] E. Varki, A. Merchant, J. Xu, X. Qiu, Issues and challenges in the performance analysis of real disk arrays, IEEE Trans. Parallel Distrib. Syst. 15 (2004) 559–574.
[30] A. Verma, R. Koller, L. Useche, R. Rangaswami, Srcmap: Energy proportional storage using dynamic consolidation, in: FAST, 2010, pp. 267–280.
[31] Y. Won, J. Kim, W. Jung, Energy-aware disk scheduling for soft real-time I/O requests, Multimedia Syst. 13 (2008) 409–428. http://dx.doi.org/10.1007/s00530-007-0107-8.
[32] T. Xie, Sea: A striping-based energy-aware strategy for data placement in raid-structured storage systems, IEEE Trans. Comput. 57 (2008) 748–761.
[33] YouTube/Google, Inc, Google tech talks: Youtube scalability, Seattle Conference on Scalability, 2007. http://www.youtube.com/watch?v=ZW5_eEKEC28.
[34] Q. Zhu, Y. Zhou, Power-aware storage cache management, IEEE Trans. Comput. 54 (2005) 587–602.

Hang Yuan is a Ph.D. student in Electrical Engineering at University of Southern California. He received B.Eng. (Hons) degree in Electronic and Information Engineering from the Hong Kong Polytechnic University in 2008. His research interests include image/video technology and energy-efficient computing.

Ishfaq Ahmad received a B.Sc. degree in Electrical Engineering from the University of Engineering and Technology, Pakistan, in 1985, and an M.S. degree in Computer Engineering and a Ph.D. degree in Computer Science from Syracuse University, New York, U.S.A., in 1987 and 1992, respectively. Since 2002, he is a professor of Computer Science and Engineering at the University of Texas at Arlington (UTA). Prior to that, he was an associate professor of computer science at the Hong Kong University of Science and Technology. His research focus is on the broader areas of parallel and distributed computing systems and their applications, optimization algorithms, multimedia systems, video compression, and energy-aware green computing. Dr. Ahmad has received numerous research awards, including three best paper awards at leading conferences and 2007 best paper award for IEEE Transactions on Circuits and Systems for Video Technology, IEEE Service Appreciation Award, and 2008 Outstanding Area Editor Award from the IEEE Transactions on Circuits and Systems for Video Technology. His current research is funded by the US Department of Justice, National Science Foundation, SRC, Department of Education, and several companies. He is the Founding Editor-in-Chief of the new Journal, Sustainable Computing: Informatics and Systems, and co-founder of the International Green Computing Conference (IGCC). He is an editor of the Journal of Parallel and Distributed Computing, IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Parallel and Distributed Systems, and Hindawi Journal of Electrical and Computer Engineering. He is a Fellow of the IEEE.

C.-C. Jay Kuo received the B.S. degree from the National Taiwan University, Taipei, Taiwan, in 1980, and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1985 and 1987, respectively, all in electrical engineering. He is currently the Director of the Media Communications Laboratory and a Professor of electrical engineering, computer science, and mathematics at the University of Southern California, Los Angeles, and the President of the Asia–Pacific Signal and Information Processing Association (APSIPA). His current research interests include digital image/video analysis and multimedia data compression. He is a co-author of about 210 journal papers, 850 conference papers, and 10 books. Dr. Kuo is a fellow of IEEE, AAAS and SPIE.