Dynamic Buffer Allocation for Conserving Disk Energy in Clustered Video Servers Which Use Replication

Minseok Song

School of Computer Science and Engineering,
Inha University, Korea
mssong@inha.ac.kr

Abstract. Reducing energy consumption is a key concern in video data centers, in which disk arrays consume a significant portion of the total energy. Disks typically support multiple power modes including a low-power mode in which they use considerably less energy than in any other mode. Therefore, extending the length of time that disks stay in low-power mode is important for energy conservation. We propose a new energy-aware buffer allocation scheme for clustered video servers which use replication. We first present a data retrieval scheme which adaptively retrieves data from the primary and backup copies so as to allow disks to go into low-power mode. We then analyze the relationship between the retrieval period and the buffer size assigned to a cluster, and examine how buffer allocation influences total energy consumption. Based on this, we propose a new buffer partitioning scheme in which the retrieval period for each cluster can be dynamically changed to adjust disk utilization, with the aim of increasing the number of disks that enter low-power mode. Simulations demonstrate that our scheme saves between 22% to 43% of the energy required for conventional video server operation.

1 Introduction

Recent advances in multimedia and network technologies make it feasible to provide multimedia-on-demand (MOD) services for application areas such as digital libraries, education-on-demand, distance learning and movie-on-demand. In a MOD system, video data is housed in a storage server and delivered to clients where requested. Due to the high bandwidth and large storage requirements of video data, video servers are built on disk arrays which may consist of hundreds of disks.

The increasing demands for multimedia data makes the energy consumption of servers a significant problem. *Energy User News* [6] recently suggests that the power requirements of typical service providers are now 150-200 W/ft^2 and will be 200-300 W/ft^2 in the near future. These growing power demands are a serious economic problem for service providers. For instance, a medium-sized 30,000 ft^2 data center requires 15 MW, which currently costs \$13 million per year [10, 11]. Another problem with such high power consumption is *heat* [4, 7]. It has been shown that running at 15 °C above ambient can double the failure rate of a disk drive [4]. But cooling systems for high heat densities are prohibitively expensive and the cooling system itself adds significantly to the power cost of a data center.

Among the components of a server, storage is one of the biggest energy consumers. A recent report [7] indicates that storage devices consume 27% of the total power. It has also been shown [7] that the energy consumed by a disk array can easily surpass that of the rest of the system, depending on the array size. This problem is exacerbated by the availability of faster disks which need more power. To reduce power consumption, modern disks have multiple power modes [4, 10, 11]: in *active mode* the platters are spinning and the head is reading or writing data; in *seek mode* the head is seeking; in *idle* mode the disk spins at full speed but there is no disk request; and in *low-power* or *standby* mode the disks stops spinning completely and consumes much less energy than in any other mode. A commonly used method of energy reduction is to transition a disk into low-power mode after the disk has been idle for a while. If a request arrives while a disk in low-power mode, then it immediately transitions to active mode to service the request. But this method of power saving is not readily applicable to video servers because the length of a video usually exceeds 1 hour and the server rarely goes to low-power mode.

We propose a new energy-aware buffer allocation (EBA) scheme for clustered video servers which use replication. We will first present a data retrieval scheme which adaptively retrieves data from the primary and backup copies so as to allow disks to go to low-power mode. We will then analyze the relationship between the retrieval period and the buffer size allocated to each cluster, and examine how buffer allocation affects energy consumption. Finally, we go on to propose a dynamic buffer partitioning algorithm in which the buffer size for each cluster is dynamically changed with the aim of minimizing the total energy consumption.

The rest of this paper is organized as follows. We explain the background to our work in Section 2. We propose a energy-aware buffer allocation scheme in Section 3. We validate the proposed scheme through simulations in Section 4 and conclude the paper in Section 5.

2 Background

2.1 Multimedia-on-Demand Servers

We use round-based scheduling for video data retrieval: time is divided into equal-sized periods, called rounds, and each client is served once in each round. We partition a video object into blocks and distribute them over multiple disks. We will refer to this scheme as *striping*, and a segment denotes the maximum amount of contiguous data that is stored on a single disk. To reduce seek overhead, data retrieved during a round are grouped into a segment, and each segment is stored in a round-robin fashion across the disks [9]. In addition, for scalability, the disk array is generally partitioned into several clusters, each of which independently forms a striping group, and each video is then striped within a cluster [3].

For fault-tolerance, we use a replication technique where the original data is duplicated on separate disks. We refer to the original data as the primary copy and call the duplicated data the backup copy. The server retrieves data from the primary copy when all disks are operational; but in degraded mode when a disk fails, it reads the backup copy. Among various replication schemes, chained declustering (CD) shows the best performance [9]. Suppose that each cluster consists of Q homogeneous disks and that the number of clusters is C. In the CD scheme, a primary copy on D_k^i, the i^{th} disk of cluster k will have a backup copy on $D_k^{(i+1) \bmod Q}$. We place the backup copy on one disk as in the CD scheme. Let us assume that a video V_i is divided into finite numbers of sequential segments ($S_{i,1}$, $S_{i,2}$, ...). We can now see, in Fig. 1, how data placement works using the CD scheme, with videos, V_i stored in cluster 1 and V_j, in cluster 2.

	cluster 1				cluster 2			
	D_1^1	D_1^2	D_1^3	D_1^4	D_2^1	D_2^2	D_2^3	D_2^4
primary copy	$S_{i,1}$	$S_{i,2}$	$S_{i,3}$	$S_{i,4}$	$S_{j,1}$	$S_{j,2}$	$S_{j,3}$	$S_{j,4}$
	$S_{i,5}$	$S_{i,6}$	$S_{i,7}$	$S_{i,8}$	$S_{j,5}$	$S_{j,6}$	$S_{j,7}$	$S_{j,8}$
	...	...	...	...	...	...	...	...
backup copy	$S_{i,4}$	$S_{i,1}$	$S_{i,2}$	$S_{i,3}$	$S_{j,4}$	$S_{j,1}$	$S_{j,2}$	$S_{j,3}$
	$S_{i,8}$	$S_{i,5}$	$S_{i,6}$	$S_{i,7}$	$S_{j,8}$	$S_{j,5}$	$S_{j,6}$	$S_{j,7}$
	...	...	...	...	...	...	...	...

Fig. 1. An example of data placement using the CD scheme in a clustered video server with $Q = 4$ and $C = 2$.

2.2 Conventional Power Management in Servers

Even though disk power management in mobile devices has been studied extensively, relatively few attempts have been made to reduce the energy consumption of storage servers. Most existing schemes involve switching disks to low-power mode whenever that is possible without affecting performance. These schemes primarily aim to extend the period during which a disk is in low-power mode [4, 7, 10, 11]. However, because returning from low-power to active mode involves spinning up the disk, the energy saved by putting the disk into low-power mode needs to 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*.

As we have already described, saving power by going to low-power mode is hardly relevant to current video servers, due to the long duration of video streams. To emphasize this, let us consider a video that lasts for an hour and playing on a server configuration like that shown in Fig. 1, with a round length of 2 seconds and disks which have a break-even time of 16 seconds. Even if there is only one request for the video, no disk can go to low-power mode because the server needs to access each disk once every 8 seconds.

3 Energy-Aware Buffer Allocation

3.1 Adaptive Data Retrieval

We now present an adaptive data retrieval scheme that permits to video server disks to enter low-power mode by supporting dynamic data retrieval from either the primary copy or the backup copy depending on the disk loads. Let CDU_k^i be the disk bandwidth utilization of D_k^i, the i^{th} disk of cluster k, in the situation that all data is being retrieved from the primary copy ($k = 1, ..., C$ and $i = 1, ..., Q$). We define two states: if $\forall i$, $CDU_k^i \leq 0.5$, then cluster k is in *energy reduction (ER)* state; otherwise, cluster k is in *normal* state. In the normal state, data is retrieved from the primary copy on every disk in the cluster; while in the ER state, data is retrieved only from odd-numbered disks. Since the data on disk D_k^i is replicated as a backup copy on $D_k^{(i+1) \bmod Q}$, the disk loads incurred in reading the primary copies on D_k^{2i} are shifted to the backup copies on disks $D_k^{(2i+1) \bmod Q}$, $(i = 1, ..., \lfloor \frac{1}{2} Q \rfloor)$. D_k^{2i} is able to go into low-power mode because it is not being accessed. Even though $D_k^{(2i+1) \bmod Q}$ now carries the load from D_k^{2i}, its utilization does not exceed 1 because $\forall i$ $CDU_k^i \leq 0.5$. Fig. 2 illustrates how disk loads are shifted at the moment when the server changes from normal to ER states, and we see that D_k^2, D_k^4 and D_k^6 can all go to low-power mode because they are not being accessed.

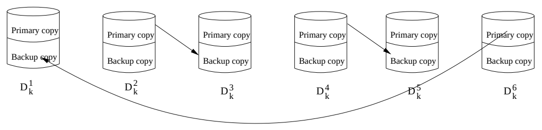

Fig. 2. Example movement of disk loads when changing from normal to ER state.

3.2 The Variation of Energy Consumption with the Round Length

The round length plays an important role in determining the requirements for server resources (i.e. disk bandwidth and buffer) [5, 9]. Increasing the round length decreases disk bandwidth utilization because retrieving a large amount of data during a single round reduces the impact of disk latency; but this increases buffer requirements because a lot of buffer space is needed to store the data retrieved during a round. Reducing disk bandwidth utilization below 0.5 allows a transition to the ER state, so it is clear that a long round is advantageous in terms of energy conservation.

The main idea of our scheme is to divide the entire buffer space into C partitions, one for each cluster, and to allocate buffer space dynamically with the aim of minimizing the total energy consumption. This approach is motivated by the observation that extending the round length is profitable in terms of energy reduction but increases the buffer overhead. This means that assigning more buffer space to a cluster may allow it to go to the ER state. But this requires judicious buffer partitioning methods, because buffer space is limited and is shared by clusters.

Changing the round length may also incur an additional seek overhead because the data retrieved during a round may not be stored contiguously. To remedy this, we split a data segment $S_{i,m}$ into NS sub-segments $ss_{i,m}^n$ $(n = 1, ..., NS)$, where the size of each sub-segment corresponds to the data retrieved during a basic round of length BR. The NS sub-segments are stored contiguously to constitute a segment, and each segment is placed in round-robin fashion, as depicted in Fig. 1. Fig. 3 illustrates the sub-segments that makes up a segment $S_{i,m}$ when $NS = 8$.

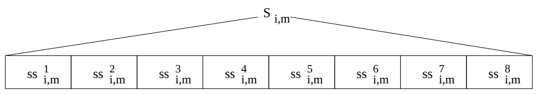

Fig. 3. A segment and its associated sub-segments.

We will use dv_j to the j^{th} divisor of NS $(j = 1, ..., ND)$, where ND is the number of divisors for NS, and the set of feasible round lengths $FS = \{fr_j | fr_j = BR \times dv_j\}$. We will assume that the elements of FS are sorted in ascending order. For example, if $NS = 6$, then $FS = \{fr_1 = BR, fr_2 = 2BR, fr_3 = 3BR, fr_4 = 6BR\}$. The selection of round lengths other than fr_j is not allowed because this may lead to the occurrence of two seeks for one read.

Let us see how the buffer and disk bandwidth requirements for a video V_i with data rate dr_i (bits/sec) depends on the round length. To ensure the continuous playback of all streams, the total time spent retrieving the streams must not exceed the round duration. The bandwidth utilization for a disk is usually defined as the ratio of the total service time to the round length [1]. We use a typical seek time model in which a constant seek time of T_s and a rotational delay of T_d are required for one read of contiguous data [1]. Retrieving a video stream V_i incurs an overhead of $T_s + T_d$ and a reading time of $fr_j \times \frac{dr_i}{tr}$, where tr is the data transfer rate of the disk. For ease of exposition, we are assuming that disk loads are evenly distributed across disks in the same cluster. If the round length is fr_j, and there are Q disks in a cluster, then servicing a video stream V_i increases the bandwidth utilization for a cluster by $DU_i(j)$, where

$$DU_i(j) = \frac{T_s + T_d + fr_j \times \frac{dr_i}{tr}}{fr_j \times Q}. \tag{1}$$

Let B be the total buffer size. We assume that SCAN scheduling is used to reduce the seek overhead. Since double buffering is used for SCAN scheduling [1], servicing a video stream V_i increases the buffer utilization by $BU_i(j)$, where

$$BU_i(j) = \frac{fr_j \times dr_i}{B}. \tag{2}$$

We now suppose that a client CL_i^m requests a video stream V_i. We partition the clients into C client groups (say CG_1,...,CG_C) where the clients in group CG_k receive streams from cluster k. We can now determine the disk bandwidth utilization $DS_k(j)$ for cluster k, as follows:

$$DS_k(j) = \sum_{CL_i^m \in CG_k} DU_i(j). \tag{3}$$

We also obtain the buffer utilization $BS_k(j)$ for cluster k as follows:

$$BS_k(j) = \sum_{CL_i^m \in CG_k} BU_i(j). \tag{4}$$

We will now examine how the energy consumption depends on the round length, fr_j. Let P_s be the power required to perform a seek, P_a the power to read or write data, P_i the power consumption in idle mode, and P_l the power consumption in low-power mode. We now formulate the following energy properties for a cluster k:

1. The total seek time during fr_j is $\sum_{CL_i^m \in CG_k} T_s$. Thus, the energy required to perform seeks during fr_j denoted by $E_k^s(j)$, is $\sum_{CL_i^m \in CG_k} T_s \times P_s$.
2. The energy required for reading or writing during fr_j, denoted by $E_k^a(j)$, is $\sum_{CL_i^m \in CG_k} (fr_j \times \frac{dr_i}{tr}) \times P_a$.
3. If no disk activity is taking place or a disk is waiting for a sector to arrive underneath a head, it is considered to be passive, and its energy consumption must be calculated in different ways depending on whether the server is in the normal or the ER state.
   – In the normal state, no disk goes to low-power mode so the energy consumed by passive disks during fr_j, denoted by $E_k^n(j)$, may be expressed as:

$$E_k^n(j) = P_i \times (\underbrace{Q \times fr_j}_{\text{total round length for Q disks}} - \underbrace{\sum_{CL_i^m \in CG_k} (T_s + fr_j \times \frac{dr_i}{tr})}_{\text{read or seek time}})\,.$$

   – In the ER state, $\lfloor \frac{1}{2} Q \rfloor$ disks go to low-power mode, and the energy consumed by passive disks during fr_j, denoted by $E_k^e(j)$, may be expressed as:

$$E_k^e(j) = \underbrace{P_i \times (\lceil \frac{1}{2} Q \rceil \times fr_j - \sum_{CL_i^m \in CG_k} (T_s + fr_j \times \frac{dr_i}{tr}))}_{\text{energy for disks in idle mode}} + \underbrace{P_l \times \lfloor \frac{1}{2} Q \rfloor}_{\text{energy for disks in low-power mode}}.$$

If $DS_k(j) > 0.5$, cluster k is in the normal state; otherwise, it is in the ER state. Based on the expressions above, we can now how the total energy consumption during BR, denoted by $E_k(j)$, depends on the round length fr_j:

$$E_k(j) = \begin{cases} (E_k^s(j) + E_k^a(j) + E_k^n(j)) \times \frac{BR}{fr_j} & \text{if } DS_k(j) > 0.5, \\ (E_k^s(j) + E_k^a(j) + E_k^e(j)) \times \frac{BR}{fr_j} & \text{if } DS_k(j) \leq 0.5. \end{cases}$$

3.3 A Dynamic Buffer Partitioning Algorithm

Let SL_k be a selection parameter indicating that the SL_kth element of FS, fr_{SL_k}, is selected as the round length for cluster k. For example, if $FS = \{BR, 2BR, 3BR, 6BR\}$ and $SL_k = 2$, then $2BR$ is selected as the round length. From Equations (1), (2), (3) and (4), we can easily see that higher values of fr_j decrease $DS_k(j)$ but increase $BS_k(j)$. Since decreasing the value of $DS_k(j)$ below 0.5 leads to the cluster entering the ER state, higher values of SL_k may produce the ER state in circumstances under which lower values of SL_k would result in the normal state. Since higher values of SL_k also increases the buffer requirements, SL_k should be selected carefully.

Our goal is to minimize the total energy consumption during a round BR while satisfying the disk bandwidth constraint $\forall\ k$, $DS_k(SL_k) \leq 1$, $(k = 1, ..., C)$ without exceeding the buffer limits $\sum_{k=1}^{C} BS_k(SL_k) \leq 1$. We now define this more formally as the buffer allocation problem.

Definition 1 The Buffer Allocation Problem ($\mathcal{BAP}$)
The $\mathcal{BAP}$ is to find SL_k $(SL_k = 1, ..., ND)$ for every cluster k $(k = 1, ..., C)$, which minimizes $\sum_{k=1}^{C} E_k(SL_k)$ while satisfying $\sum_{k=1}^{C} BS_k(SL_k) \leq 1$ and $\forall\ k$, $DS_k(SL_k) \leq 1$, $(k = 1, ..., C)$.

Note that $\mathcal{BAP}$ is a variant of the multiple-choice knapsack problem, in which each object has a finite set of items, and exactly one item from each object must be selected so as to maximize the total profit. Since the multiple-choice knapsack problem is NP-hard [8], $\mathcal{BAP}$ is also NP-hard, which implies that a heuristic approach will be necessary, especially as the values of SL_k must be determined quickly, so that the admission of new clients in not noticeably delayed. We will now outline our solution.

The server first checks whether changing the round length to fr_j would violate the disk bandwidth constraint, $DS_k(j) \leq 1$. Let SV_k be the smallest value of j that satisfies $DS_k(j) \leq 1$. We start by defining a set of parameters, $DF_k(m)$, $(m = SV_k, ..., ND-1)$ for cluster k, as follows:

$$DF_k(m) = \frac{E_k(m) - E_k(ND)}{BS_k(ND) - BS_k(m)}.$$

In this formulation the denominator represents the decrease in buffer requirements, while the numerator represents the increase in energy consumption for cluster k when selection parameter SL_k is decreased from ND to m. Based on this, we propose Algorithm 1, which we call a dynamic buffer partitioning algorithm (DBPA).

Algorithm 1 DBPA (Dynamic Buffer Partitioning Algorithm)

```
1:  Set of values of $DF_k(m)$: $SD$;
2:  Temporary variable: $IS$;
3:  $IS \leftarrow \sum_{i=1}^{NC} BS_k(ND)$;
4:  $SL_k \leftarrow ND$;
5:  while $IS > B$ and $SD \neq \phi$ do
6:     Find the lowest value, $DF_k(l) \in SD$;
7:     $SD \leftarrow SD - \{DF_k(l)\}$;
8:     if $l < SL_k$ then
9:        $IS \leftarrow IS - BS_k(SL_k) + BS_k(l)$;
10:       $SL_k \leftarrow l$;
11:    end if
12: end while
```

The array SD is a set of $DF_k(m)$ parameters and initially contains those of every video (line 1). DBPA initializes the value of SL_k to ND (line 4). Next, it chooses the lowest value of $DF_k(l)$ in SD and removes it from SD. If $l < SL_k$ (line 6), the value of SL_k is reduced to l. These steps are then repeated while $IS > B$ and $SD \neq \phi$ (lines 5-12).

4 Experimental Results

To evaluate the effectiveness of our scheme, we performed several simulations. Our server has 160 disks, each of which employs an IBM Ultrastar36Z15 disk [10], with the parameters shown in Table 1. The server is divided into 40 clusters, each of which composed of 4 disks. The arrival of client requests is assumed to follow a Poisson distribution. We also assume that all videos are 90 minutes long, and have the same bit-rate of 4.5MB/sec. The access probability follows a Zipf distribution with $\alpha = 0.271$ [2]. We calculated the round length that achieves a balanced use of disk bandwidth and buffer, as described in [9]; the resulting value of BR, is 0.85 seconds. The cluster location of each movie is chosen randomly. NS is assumed to be 8. To evaluate the efficacy of the EBA scheme, we will compare it with four other methods:

1. PRS: operates like a conventional video server, in that it does not allow data retrieval from the backup copy. The round length used for PRS is BR.
2. ORS: permits data retrieval from the backup copy depending on the disk utilization, but does not allow adaptive round adjustment. The round length used for ORS is BR.
3. RAN: allows adaptive round adjustment, but randomly assigns SL_k to each cluster subject to buffer constraints.
4. UNI: allows adaptive round adjustment but, unlike DBPA, initially assigns SL_k to SV_k. Then it uniformly increases the value of SL_k subject to buffer constraints.

Table 1. Parameter values for our video server.

Transfer rate (tr)	55 MB/s
Typical disk seek time (T_s)	7 ms
Typical disk rotation time (T_r)	4 ms
Storage space for each disk	18 GB
Idle power	10.2 W
Standby power (low-power mode)	2.5 W
Active power	13.5 W
Total buffer size (B)	8 GB

Note that the number of admitted clients is lower for PRS and ORS than for the UNI, RAN and EBA schemes. This is because PRS and ORS do not permit adaptive round adjustment so server resources may not be fully utilized. To reflect this, we assess the total energy consumption per client admitted. Fig. 4 shows how the energy consumption of the four schemes depends on the inter-arrival time of requests, compared to PRS. The EBA scheme exhibits the best performance under all workloads with an energy saving of between 22% to 43% compared to the conventional PRS scheme. This is because the EBA scheme allows up to half of the disks to go to low-power mode when the level of utilization permits. As the inter-arrival time increases, the performance gap increases because EBA gives the disks more opportunities to go to ER state when it is lightly loaded. EBA uses between 19% and 36% less energy than ORS. This is because EBA adjusts the round length, which results in more time in low-power mode. EBA also saves between 4.6% to 18% more energy than RAN, and 1.4% to 17% more than UNI, which implies that buffer allocation has a significant impact on energy consumption.

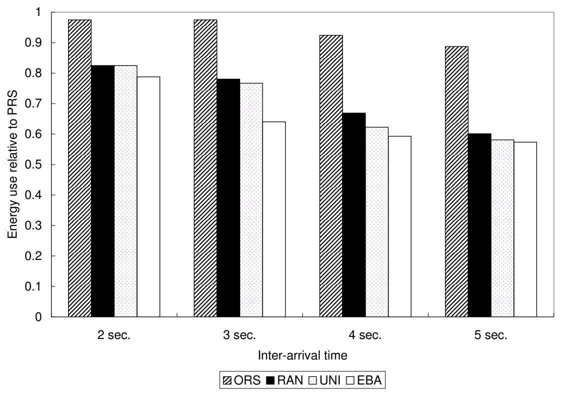

Fig. 4. Energy consumption per client relative to PRS for various inter-arrival times.

To evaluate the effectiveness of our heuristic algorithm, let us consider another partitioning algorithm called LB: this method relaxes the integrality constraints on variables (here, SL_k) and then calculates the energy consumption. Its performance corresponds to a lower bound on $\mathcal{BAP}$ [8][1]. Whenever a client requests or closes a video stream, we calculate the lower bound on $\mathcal{BAP}$ using the LB method, and then find the average value of that bound over 24 hours. We also examine the average energy consumption of DBPA, UNI and RAD over 24 hours. Table 2 shows the energy consumption relative to the LB method for DBPA, UNI and RAD as a function of inter-arrival time. From the table, we observe that the energy consumption of DBPA is very close to the lower bound. We also see that, compared with the lower bound, RAD consumes between 4.7% to 16.9% more energy, UNI uses between 2.5% to 14.7% more. Note that no algorithm can produce a better result than the lower bound obtained by the LB method, which implies that DBPA is a near-optimal solution to $\mathcal{BAP}$.

5 Conclusions

We have proposed a new dynamic buffer allocation scheme for reducing the energy consumption of clustered video servers which use replication. We have presented a data

[1] A relaxed version of $\mathcal{BAP}$ can be reduced to a linear multiple-choice knapsack problem for which the optimal solution can be obtained in polynomial time [8].

Table 2. Energy consumption relative to the LB method.

inter-arrival time	DBPA	UNI	RAD
1 second	1.007	1.065	1.065
2 seconds	1.007	1.070	1.077
3 seconds	1.007	1.147	1.169
4 seconds	1.001	1.086	1.113
5 seconds	1.0002	1.025	1.047

retrieval scheme in which data segment size can be dynamically selected to give the server more chance to operate in low-power mode. Based on this, we have analyzed how buffer allocation affects energy consumption. We have then gone on to propose a new buffer allocation scheme with the aim of minimizing total energy consumption. Experimental results show that our scheme enables a server to achieve appreciable energy savings under a range of workloads. They also demonstrate that our algorithm produces a practical and near-optimal buffer allocation.

Acknowledgements

This work was supported by INHA UNIVERSITY Research Grant. (INHA-33917)

References

1. E. Chang. *Storage and Retrieval of Compressed Video*. PhD thesis, University of California at Berkeley, 1996.
2. A. Dan, D. Sitaram, and P. Shahabuddin. Dynamic batching policies for an on-demand video server. *ACM/Springer Multimedia Systems Journal*, 4(3):112–121, 1996.
3. L. Golubchik, J. Lui, and M. Papadopouli. A survey of approaches to fault tolerant vod storage servers: Techniques, analysis, and comparison. *Parallel Computing*, 24(1):123–155, January 1998.
4. S. Gurumurthi. *Power Management of Enterprise Storage Systems*. PhD thesis, Pennsylvania State University, 2005.
5. K. Lee and H. Yeom. A dynamic scheduling algorithm for large scale multimedia server. *Information Processing Letters*, 68(5):235–240, March 1998.
6. B. Moore. Taking the data center power and cooling challenge. *Energy User News*, 27, August 2002.
7. E. Pinheiro and R. Bianchini. Energy conservation techniques for disk-array-based servers. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, pages 88–95, June 2004.
8. D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen, 1995.
9. M. Song and H. Shin. Replication and retrieval strategies for resource-effective admission control in multi-resolution video servers. *Multimedia Tools and Applications Journal*, 28(3):89–114, March 2006.
10. Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: helping disk arrays sleep through the winter. *ACM Operating Systems Review*, 39(5):177–190, 2005.
11. Q. Zhu and Y. Zhou. Power aware storage cache management. *IEEE Transactions on Computers*, 54(5):587–602, 2005.