An Optimal Multimedia Object Allocation Solution in Transcoding-Enabled Wide-Area Storage Systems

Wenyu Qu, Kazuo Goda, and Masaru Kitsuregawa
Institute of Industrial Science
The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, Japan

ABSTRACT

Together with the information explosion era comes the problem of high energy consumption in data centers. Among the various components of a data center, storage is one of the biggest energy consumers. Many efforts have been put on reducing energy consumptions of data centers/storage systems, from server workflow management to storage data allocation. However, most works focus on local storage systems and pay no attention to other structures of storage systems. With the exponential expanding on both the depth and width of the Internet, wide-area storage systems attract more and more attention as they are reliable, load-balancing, and easy to collaborate. In this paper, we study the energy consumption in wide-area storage systems by using multimedia objects as our special application area. We develop a multimedia object distributing algorithm based on the access pattern to multimedia objects and the topology of storage systems. We also present an in-depth analysis that provides valuable insight into the characteristics of multimedia object distribution strategies and leads to precision guarantees.

Keywords

Multimedia object, allocation, wide-area storage system.

1. INTRODUCTION

The far-reaching significance of the Internet and dramatic advances in computer technology has resulted in a proliferation of applications that are server-centric. Data centers are playing a key role in this new architecture since they are commonly used to provide a wide variety of services, such as Web hosting, application services, and so on.

Typically, data centers have very high power requirements. The steady growth of data centers makes the problem of energy consumption much serious. According to EUN (Energy User News) [18], today's data centers have power requirements that range from 75 W/ft^2 for typical service providers. As Eric Schmidt, the CEO of Google, said, "what matters most to the computer designers at Google is not speed but power, because data centers can consume as much electricity as a city." In the future, the power requirement is predicted to increase to 200-300 W/ft^2 . These increasing power requirements are driving energy costs up as much as 25 percent annually and making it a growing consideration in the TCO (total cost of ownership) for a data center [17]. Besides and maybe the most important, high energy consumption also has negative environmental implication that has attracted a large amount of attentions. Storage device is one of the biggest components in data center and accounts for almost 27 percent of the total energy consumed by a data center [17]. Currently, storage demand is growing by 60 percent annually [21] and is expected to be 10 times as much as they are today by 2008. This problem is exacerbated by the availability of faster disks with higher power needs as well as the increasing shift from tape backups to disk backups for better performance.

There are many studies on power conservation of storage systems [10, 25, 24, 22]. Some addressed the problem for the workloads and the I/O subsystems in server environments, including RAID configuration, number of disks, striping unit, etc. Others focused on file allocation among storage disks. In [23], energy was conserved by concentrating requires on some disks and leaving others idle for a longer time. However, all these works are done for centralized single-server storage systems. Wide-area storage systems such as FarSite [1], Glacier [2], and OceanStore [11] rely on data redundancy to ensure durable storage despite of node failures. They have attracted considerable attentions from data centers such as Google and Baidu in recent years for their merits of being reliable, load-balancing, and easy to collaborate. Although most experts are working on the synchronicity and consistency of such kind of storage systems, we don't pay attention to these problems but concentrate on the energy conservation problem instead, which is another important problem to be solved.

In this paper, we address the problem of energy conservation by distributing requires among storage devices in wide-area storage systems. We develop a multimedia object distribution algorithm that is fully dynamic (supporting any system topologies and access patterns), efficient (processing distribution and redistribution with very low computational overhead), and optimal (producing solutions with small processing latency and energy consumption). The contributions of this paper are twofold. We not only devise a multimedia object distribution algorithm for a wide-area storage system by inserting the function of transcoding onto storage

servers, but also, in light of the concerns on energy consumption from industrial park, optimize the overall effect on both energy conservation and access latency. To the best of our knowledge, none of previous works has addressed the energy conservation problem of wide-area storage systems for multimedia object applications.

The rest of the paper is organized as follows. Section 3 preliminarily describes the problem to be solved. Section 4 and Section 5 mathematically model and solve the described problem and propose the distribution algorithms. Section 8 concludes the paper with directions for future work. Due to the space limitation, we present our idea here but omit the theoretical proof of the proposed theorem and experimental results.

2. RELATED WORKS

Previous studies has shown that the aggregate effect of storing multiple versions of the same multimedia object is not the simple summation of that of storing individual versions of the multimedia object, but rather, depends on the relationship among these versions according to the technology of transcoding [9].

In [9], the authors proposed an algorithm on exploring the aggregate effect of placing multiple versions of the same multimedia object in transcoding-enabled storage devices. However, the authors have not considered the case in which multiple versions of the same multimedia object are placed in many storages at the same time. In [26], the authors studied the case of general objects, which can be viewed as single-version multimedia objects. Thus, the problem addressed in [26] can be viewed as a special case of ours when a multimedia object has only one version. From the following example, we can see that the network performance will be greatly reduced if the solution proposed in [26] is applied to deal with multimedia objects, i.e., view different versions of the same multimedia object as different objects. Suppose that a multimedia object has three versions ($A1$, $A2$, and $A3$) and the relationship among these versions is shown in Figure $1-(b)$. The number in the figure denotes the transcoding cost from one version to another, which is defined as the delay for executing the transcoding between two different versions of the same multimedia object. Figure $1-(c)$ is a simple network with four nodes among which node 0 is the server. Suppose that the transmission cost on each edge for A_1, A_2, and $A-3$ are the same. The transmission cost on each link and the access frequency from each node for each version are shown in Figure $1-(b)$. For example, the transmission costs from node 1 to node 0 for $A1$, $A2$, and $A3$ are all 0.6 and the access frequencies for $A1$, $A2$, and $A3$ from node 2 are $2, 3, 0$, respectively. Based on the solution proposed in [26], $A2$ should be cached at nodes $1, 2, 3$ such that the maximal delay saving is $2\times0.6+3\times(0.6+0.7)+1\times(0.6+0.7+0.8) = 7.2$. Obviously, it is not optimal when we consider the relationship among A_1, A_2, and $A-3$. Accordingly, the maximal delay saving is $1\times0.6+2\times(0.6+0.7)+0\times(0.6+0.7+0.8)+2\times(0.6-0.5)+3\times(0.6+0.7-0.5)+1\times(0.6+0.7+0.8-0.5)+1\times(0.6-0.7)+0\times(0.6+0.7-0.7)+2\times(0.6+0.7+0.8-0.7) = 8.7$ by caching $A1$ at nodes 1, 2, and 3, respectively.

In [9, 5, 3], the authors studied the object replacement problem from different point of views. In [8], the authors addressed the problem for content adaption. They presents cooperative architectures and algorithms for discovery and transcoding of multi-version content. In [27], the authors addressed the problem of distributing computational load caused by object transcoding throughout a hierarchical system structure. In [28, 14, 26, 12], the authors studied the problem of placing single-version multimedia objects for different kinds of network topologies from different point of view. However, these solutions are not applicable for multimedia objects since different versions of the same multimedia object can not be simply views as different web objects. In [9], the authors proposed an object replacement algorithm by investigating the aggregate effect of placing multiple versions of the same multimedia object. Another aspect that affects the performance of transcoding-enabled storage systems is the storage distribution placement, which has been well studied [29, 19, 4, 16].

3. PROBLEM CHARACTERISTICS

The topology of a wide-area storage system is defined as $G = (V, E)$, where $V = \{v_i, i = 1, 2, \cdots\}$ is the set of storages and E is the set of links. A multimedia object A may have a set of different versions $A_{version} = \{A_j, j = 1, 2, \cdots, m\}$, which are distributed in the storage systems, and b_{A_j} is the size of A_j. Requests on multimedia objects may come from different devices/users in various preferences on content presentation, which may divergent of sizes, weight, input/output capabilities, network connectivity, and computing power. As keeping all versions of a same multimedia object in one storage could take too much space and is not flexible in dealing with changing clients' needs, servers in the storage system are enabled the capability of transcoding for transforming the multimedia object to proper versions to meet the diverse needs.

The full object version is denoted as A_1 whereas the least detailed version which cannot be transcoded any more is denoted as A_m. $D(A_j)$ is the set of all versions that can be transcoded from A_j. If A_i can be transcoding from $A_j (i \neq j)$, then we call A_i is a less detailed version than A_j and A_j is a more detailed version than A_i. As shown in Fig. 1, A_1 is the full object version and A_3 is the least detailed version. $D(A_1) = \{A_2, A_3\}$, $D(A_2) = \{A_3\}$, and $D(A_3) = \phi$. The versions in $D(A_i)$ $(i = 1, 2)$ are less detailed versions than A_1 and A_2, respectively, while A_i $(i = 1, 2)$ are the more detailed versions than the versions in $D(A_i)$ $(i = 1, 2)$, respectively. A_3 has two more detailed versions, i.e., A_1 and A_2 and has not any less detailed version. That is, A_3 can not be transcoded any more. Let $f_{v_i}(A_j)$ denote the access frequency for A_j through v_i per unit time. We use B_{v_i} to denote the version stored or to be placed at v_i. Obviously, we have $B_{v_i} \in A_{version}$. Fig. 1-(b) is a simple system with four storages among which storage 0 has the original version of the multimedia object. For example, the transmission costs from storage 1 to storage 0 for A_1, A_2, and A_3 are all 0.6 and the access frequencies for A_1, A_2, and A_3 from storage 2 are $2, 3, 0$, respectively.

Suppose that each storage receives a large number of requirements on multimedia objects per time unit, which are either satisfied by local versions of the multimedia objects or have to be forwarded to other storages. The distribution of multimedia objects in the system will certainly influence the service efficiency of these requirements. Our first consideration on the distribution problem is the energy consumption of storages, which is decided by the system configurations and file allocations. The multimedia object distribu-

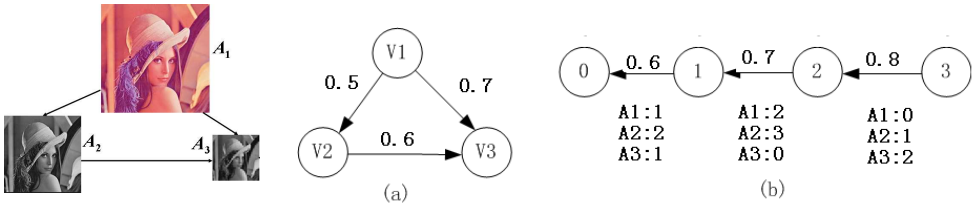

Figure 1: Multimedia Object Allocation in Transcoding Enabled Storages

tion problem can be formulated as:

$$\text{R1: select } A^*_{version} \text{ for } V^* \text{ s.t. } \min\left(E_{A^*_{version}\rightarrow V^*}\right) \quad (1)$$

where $A^*_{version}$ and V^* are subsets of $A_{version}$ and V respectively, $A^*_{version} \rightarrow V^*$ indicates a mapping between $A^*_{version}$ and V^*, and E is the total energy consumption of the system based on the allocation strategy. Specifically, there are two sub-problems to be solved. The first one aims at the selection of $A^*_{version}$ and V^*, whereas the second one tackles the mapping $A^*_{version} \rightarrow V^*$. On the other hand, the multimedia object distribution problem can be formulated from the clients' view (i.e., the faster, the better) as follows:

$$\text{R2: select } A^*_{version} \text{ for } V^* \text{ s.t. } \min\left(L_{A^*_{version}\rightarrow V^*}\right) \quad (2)$$

where L expresses the total access latency of the system based on the allocation strategy.

It is desirable to save much energy with a high performance. To achieve this goal, we combined the above two considerations into one multi-objective optimization problem. The major difficulty in this problem lies in the fact that changing the location of one version of a multimedia object may result in a redistribution of multimedia objects in this storage and in the system. In particular, within the limitation of the storage space, the distribution decision should guarantee that the benefit of placing a new version in a storage should be greater than the lost of removing other versions from the storage to make room for the new one. Another challenge is that the requirements to a multimedia object may change over time. In this case, a good algorithm should be able to adopt this change dynamically.

The working mechanism of the proposed system can be briefly summarized as follows:

- There are n multi-disk storages in the storage system, each of which has multiple power modes (including active, idle, and standby), a time threshold, and a hit threshold.
- Disks staying in the idle mode provide a shorter response time to disk requests but consume much energy. On the other hand, the standby mode consumes less energy compared to higher speed modes but requires for a longer response time and additional energy to spinning the disk up. Modes shift according to the time threshold. That is, if no request arrives on a disk for a period longer than or equal to the time threshold, the disk will transit its mode to a lower power mode.
- Files in the storage system are distributed based on the optimization problem $R1$ in (1) and R2 in (2). User requests are served by a dynamic programming-based algorithm. Section 4 and 5 will provide detailed descriptions.
- Files on disks are ordered by their popularity. The most frequently accessed data are concentrated to a subset of the disks so that other disks may become idle longer and more often and can be set to low-power modes to conserve energy. If the total hit frequency is greater than or equal to the hit threshold, the least popular data will be moved out and migrate to the less popular disk.

4. MATHEMATICAL MODELING

Based on the analysis in Section 3, the distribution problem of multimedia objects in wide-area storage systems can be modeled as an optimization problem as

$$\text{R3: } \min\left\{L_{A^*_v\rightarrow V^*}, E_{A^*_v\rightarrow V^*}\right\} \text{ where } A^*_v \subset A_v,\ V^* \subset V. \quad (3)$$

In the following, we will first mathematically characterize the optimization problem and then try to find out the optimal solution of this problem.

4.1 Energy Consumption

The considered energy cost for a client request mainly comes from two aspects, i.e., transmission cost and transcoding cost. The transmission cost, denoted as $Em_{v_i,v_j}(A_k)$, is defined as the energy consumption for transmitting A_k from v_j to v_i, whereas the transcoding cost, denoted as $Ec_{v_i}(A_p, A_q)$, is defined as the energy consumption for transcoding A_p into A_q. Let $v_i^+(A_p)$ be the nearest higher level storage of v_i (including v_i) that contains a more detailed version than A_p (including A_p) and B_{v_i} be the exact version of A on v_i, the energy cost for a request on A_p that is received from v_i, denoted as $E_{v_i}(A_p)$, can be expressed as the sum of the transmission cost and transcoding cost as follows:

$$E_{v_i}(A_p) = Em_{v_i,v_i^+(A_p)}(A_p) + Ec_{v_i^+}(B_{v_i^+(A_p)}, A_p). \quad (4)$$

Fig. 2(a) illustrates an example where a request on A_1 is received from v_1. The denotation in the circle indicates the version of object A that is available on this storage. Since v_1 is the nearest higher level storage of v_1 (including v_1) that contains a more detailed version than A_1 (including A_1) and the exact version of A on v_1 is $B_{v_1} = A_1$, $Em_{v_i,v_i^+(A_p)}(A_p) = Em_{v_1,v_1}(A_1) = 0$ due to the fact that no transmission between two storages is needed. Also, since $B_{v_1} = A_1$, $Ec_{v_i^+(A_p)}(B_{v_i^+(A_p)}, A_p) = Ec_{v_1}(A_1, A_1) = 0$ due to the fact that no transcoding between two versions is needed. Therefore, the access latency of this request is $E_{v_1}(A_1) = 0$.

It is possible that a client request cannot be directly satisfied on the current storage. Fig. 2(b) shows an example

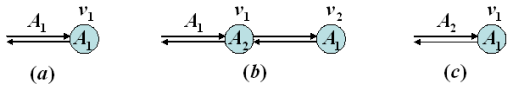

Figure 2: Example 2

that the version on v_1 is less detailed than the required version. In this case, the client request can only be satisfied on $v_1^+(A_1) = v_2$ and $Em_{v_i,v_i^+(A_p)}(A_p) = Em_{v_1,v_2}(A_1)$. Besides, wince $B_{v_2} = A_1$, we have $Ec_{v_i^+(A_p)}(B_{v_i^+(A_p)}, A_p) = Ec_{v_2}(A_1, A_1) = 0$ and $E_{v_1}(A_1) = Tm_{v_1,v_2}(A_1)$. Fig. 2(c) expresses the case that the version on v_1 is more detailed than the required version. Thus, the request can be satisfied on v_1 by transcoding. I.e., $v_1^+(A_2) = v_1$, $B_{v_1} = A_1$, $Em_{v_i,v_i^+(A_p)}(A_p) = Tm_{v_1,v_1}(A_2) = 0$, $Ec_{v_i^+(A_p)}(B_{v_i^+(A_p)}, A_p) = Ec_{v_1}(A_1, A_2)$, and $E_{v_1}(A_2) = Tc_{v_1}(A_1, A_2)$.

Suppose that originally there is a version B_{v_i} on v_i, then, the additional energy cost for accessing A_p if B_{v_i} is removed from v_i, denoted as $a.E_{v_i}(A_p)$, satisfies

$$a.E_{v_i}(A_p) = Em_{v_i,v_i^+(A_p)}(A_p) + Ec_{v_i^+(A_p)}(B_{v_i^+(A_p)}, A_p) - Ec_{v_i}(B_{v_i}, A_p). \quad (5)$$

Fig. 3(a) gives an example to the additional energy cost. A client request on A_2 is supposed to be received from v_1. As the exact version that is required can be found on v_1, we have $E_{v_1}(A_2) = 0$, which can be deduced from a similar discussion to Fig. 2(a). If A_2 is removed from v_1, all requests on A_2 cannot be satisfied on v_1 and additional transmission cost may be necessary. For the case shown in Fig. 3(b), the request can be satisfied directly by the version stored on v_2. $v_1^+(A_2) = v_2$, $B_{v_1^+}(A_2) = A_2$, $Em_{v_i,v_i^+(A_p)}(A_p) = Em_{v_1,v_2}(A_2)$, $Ec_{v_i^+(A_p)}(B_{v_i^+(A_p)}, A_p) = Ec_{v_2}(A_2, A_2) = 0$, and $a.E_{v_1}(A_2) = Em_{v_1,v_2}(A_2)$. For the case shown in Fig. 3(c), the request can be satisfied by a more detailed version on v_2. Thus, $v_1^+(A_2) = v_2$, $B_{v_1^+}(A_2) = A_1$, $Em_{v_i,v_i^+(A_p)}(A_p) = Em_{v_1,v_2}(A_2)$, $Ec_{v_i^+(A_p)}(B_{v_i^+(A_p)}, A_p) = Ec_{v_2}(A_1, A_2)$, and $a.E_{v_1}(A_2) = Em_{v_1,v_2}(A_2) + Ec_{v_2}(A_1, A_2)$.

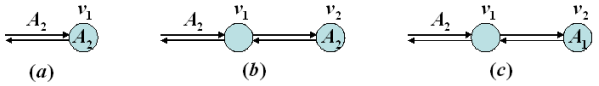

Figure 3: Example 2

Equation 5 explains the affect of removing A_p from v_i on those versions that can be transcoded from A_p. Formally, let $f_{v_i}(A_p)$ be the access frequency for A_p on v_i, the energy saving of placing A_p on v_i, denoted by $s.e_{v_i}(A_p)$, satisfies

$$s.e_{v_i}(A_p) = \sum_{A_x \in D(A_p)} f_{v_i}(A_x) \cdot a.E_{v_i}(A_x) \quad (6)$$

where $D(A_p)$ is the set of all versions that can be transcoded from A_p.

4.2 Access Latency

As shown in Section 1, a storage v_i has the form $\{RPM_i, C_i, O_i\}$, where the first element indicates the RPM (Rotation Per Minute) of this storage, the second denotes the capacity of the storage, and O_i corresponds to the set of the remaining attributes of the storage. RPM_i of v_i is an important parameter to the allocation problem in that a high RPM value indicates a short access latency (including transcoding latency and transmission latency) and a large amount of energy consumption. Let $Tc_{v_i}(A_p, A(q))$ be the access latency of v_i for transcoding A_p into A_q, it is congruous to our intuition that this function is a monotonously decreasing function on RPM_i. Similarly, $Tm_{v_i,v_j}(A_p)$, the transmission latency for transmitting A_p between v_i and v_j, is a monotonously decreasing function on RPM_i and a monotonously increasing function on b_{A_p}. By a similar deduction to $a.E_{v_i}(A_p)$, the additional transmission latency for accessing version A_p with respect to v_i, which is defined as the additional time cost of accessing A_p if B_{v_i} is removed from v_i, satisfies

$$T_{v_i}(A_p) = Tm_{v_i,v_j}(A_p) + Tc(B_{v_i^+(A_p)}, A_p) - Tc(B_{v_i}, A_p). \quad (7)$$

Consequently, the time saving of placing A_p at v_i, denoted by $st_{v_i}(A_p)$, can be defined as

$$st_{v_i}(A_p) = \sum_{A_x \in D(A_p)} f_{v_i}(A_x) \cdot T_{v_i}(A_x) \quad (8)$$

based on the fact that removing A_p from v_i will affect those versions that can be transcoded from A_p.

Taking into consider C_i, the limited capability of v_i, some objects will be evicted from v_i to make room for placing A_p at v_i, which raises the cost loss defined as follows. Let $\ell.e_{v_i}(A_p)$ and $\ell.t_{v_i}(A_p)$ denote the energy cost and time loss of placing A_p at v_i, respectively. We apply the following greedy heuristic to decide replacement candidates. Note that the normalized energy loss (NEL, i.e., the energy loss introduced by creating one unit of free space) of ejecting A_p is $s.e_{v_i}(A_p)/b_{A_p}$ and the normalized time loss (NTL) is $st_{v_i}(A_p)/b_{A_p}$. The objects in v_i are ordered by the value of $NELs$ and $NTLs$ and are selected sequentially, starting from the object with the smallest value, until enough space is created. The cost loss of placing a version A_p of a multimedia object A at a storage v_i, denoted by $\ell_{v_i}(A_p)$, which is a combination of $\ell.e_{v_i}(A_p)$ and $\ell.t_{v_i}(A_p)$, is calculated by all the selected candidates. For example, a convenient combination of $\ell_{v_i}(A_p)$ could be $\ell.e_{v_i}(A_p) + (1/theta)\ell.t_{v_i}(A_p)$ where $\theta \geq 0$ is a weighting coefficient. A_p will be ejected into v_i only when $\ell_{v_i}(A_p)$ is greater than the aggregation cost loss of removing objects.

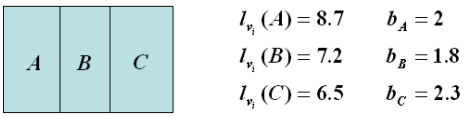

Figure 4: Example 3

In the example presented in Fig. 4, there are three objects on the storage v_i, each with a given cost loss and size. If there is an object D to be placed in this storage and the relevant parameters are known as $\ell_{v_i}(D) = 11, b_D = 3$, the normalized cost loss (denoted as NCL) of all candidates should be calculated. By the definition of cost loss, we have $NCL_{v_i}(A) = 8.7/2 = 4.35$, $NCL_{v_i}(B) = 7.2/1.8 = 4$, and $NCL_{v_i}(C) = 6.5/2.3 = 2.83$. Although C represents the minimum NCL value, its size is not large enough for making room for D since $b_D = 3 > b_C$. Therefore, B should also be removed from this storage. However, since the total cost loss of B and D, $\ell_{v_i}(B) + \ell_{v_i}(C) = 7.2 + 6.5 = 13.7$, is greater than that of D, $\ell_{v_i}(D) = 11$, i.e., the aggregate cost saving of placing D by removing B and C is negative, D will not be placed in v_i.

4.3 Problem Formulation

Now we begin to formulate the problem of placing multimedia objects in wide-area storage systems. Let $v_1, v_2, \cdots, v_n$ be a sequence of storages. Suppose that v_1 has the full object version of the requested multimedia object and v_n is the storage on which the client issued the request. $v_2, v_3, \cdots, v_n - 1$ are the intermediate storages on the path from v_1 to v_n. The energy saving of placing B_{v_i} at v_i, denoted by $Se_{v_i}(B_{v_i})$, is given by

$$Se_{v_i}(B_{v_i}) = \sum_{A_x \in D(B_{v_i})} \left(f_{v_i}(A_p) - f_{v_i^-(A_x)}(A_x) \right) \cdot s.e_{v_i}(A_x), \tag{9}$$

where $v_i^-(A_x)$ is the nearest lower level storage of v_i that has a less detailed version than A_x. The time saving of placing B_{v_i} at v_i, denoted by $St_{v_i}(B_{v_i})$, is given by

$$St_{v_i}(B_{v_i}) \sum_{A_x \in D(B_{v_i})} \left(f_{v_i}(Ap) - f_{v_i^-(A_x)}(A_x) \right) \cdot st_{v_i}(A_x). \tag{10}$$

The energy loss of placing B_i at v_i, denoted by $Le_{v_i}(B_i)$, is given by

$$Le_{v_i}(B_{v_i}) = \sum_{A_x \in D(B_{v_i})} \ell.e_{v_i}(A_x) \tag{11}$$

and the time loss of placing B_i at v_i, denoted by $Lt_{v_i}(B_i)$, is given by

$$Lt_{v_i}(B_i) = \sum_{A_x \in D(B_{v_i})} \ell.t_{v_i}(A_x). \tag{12}$$

For simplicity, we use $1, 2, \cdots, n-1, n$ to denote $v_1, v_2, \cdots, v_n - 1, v_n$ in the following analysis, respectively. Let $v_i, v_2, \cdots, v_k$ be a set of k storages such that $1 \le v_1 \le v_2 \le \cdots \le v_k \le n$. $Fe(n : v_1, v_2, \cdots, v_k)$, which is the aggregate energy profit of placing multiple versions of a multimedia object at $v_1, v_2, \cdots, v_k$, is defined as

$$Fe(n : v_1, v_2, \cdots, v_k) = \sum_{i=1}^{k} \left(Se_{v_i}(B_{v_i}) - Le_{v_i}(B_{v_i}) \right). \tag{13}$$

Similarly, $Ft(n : v_1, v_2, \cdots, v_k)$, which is the aggregate time profit of placing multiple versions of a multimedia object at $v_1, v_2, \cdots, v_k$, is defined as

$$Ft(n : v_1, v_2, \cdots, v_k) = \sum_{i=1}^{k} \left(St_{v_i}(B_{v_i}) - Lt_{v_i}(B_{v_i}) \right). \tag{14}$$

If $k = 0$, we define

$$Fe(n : \phi) = Ft(n : \phi) = 0. \tag{15}$$

Thus, the objective is to find k^* and $v_1, v_2, \cdots, v_{k^*}$ that maximizes $Fe(n : v_1, v_2, \cdots, v_k)$ and $Ft(n : v_1, v_2, \cdots, v_k)$ which is referred to as an n-optimization problem.

5. DYNAMIC PROGRAMMING-BASED ALGORITHM

Actually, the n-optimization problem to maximize $Fe(n : v_1, v_2, \cdots, v_k)$ and $Ft(n : v_1, v_2, \cdots, v_k)$ is a multi-objective optimization problem as follows:

$$\max_{k, v_1, v_2, \cdots, v_k} \{ Fe(n : v_i), Ft(n : v_i) | \, i = 1, \cdots, k \}. \tag{16}$$

By the weighting coefficient method, this problem can be transformed into a single-objective problem as follows:

$$\max_{k, v_1, v_2, \cdots, v_k} F(n, \theta : v_i | \, i = 1, \cdots, k) = \left\{ Fe(n : v_i) + \frac{1}{\theta} Ft(n : v_i) \middle| \, i = 1, \cdots, k \right\} \tag{17}$$

where $\theta \ge 0$ is a weighting coefficient. Obviously, when θ is small, the second item of the objective function is dominative. Then the gained solution emphasizes the requirement of reducing energy consumption. With the increase of θ's value, the effect of the first item increases; thus, a quick reply to the client request is the dominative objective.

In the following, we develop a dynamic programming based algorithm inspired by [4] to solve the problem formulated in the previous section. The following theorem shows that an optimal solution for the whole problem must contain optimal solutions to some subproblems.

THEOREM 1. *Suppose that $\{v_1, v_2, \cdots, v_I\}$ is an optimal solution to the n-optimization problem and $\{u_1, u_2, \cdots, u_l\}$ is an optimal solution to the $(v_I - 1)$-optimization problem, then $\{u_1, u_2, \cdots, u_l, v_I\}$ is also an optimal solution to the n-optimization problem.*

PROOF. On one hand, by the definitions in (13), (14), and (17), we have

$$\begin{aligned} & F(n, \theta : u_1, u_2, \cdots, u_l, v_I) \\ &= Fe(n : u_1, u_2, \cdots, u_l, v_I) \\ &\quad + \frac{1}{\theta} Ft(n : u_1, u_2, \cdots, u_l, v_I) \\ &= Fe(v_I - 1 : u_1, u_2, \cdots, u_l) \\ &\quad + \frac{1}{\theta} Ft(v_I - 1 : u_1, u_2, \cdots, u_l) \\ &\quad + [Se_{v_I}(B_{v_I}) - Le_{v_I}(B_{v_I})] \\ &\quad + \frac{1}{\theta} [St_{v_I}(B_{v_I}) - Lt_{v_I}(B_{v_I})] \\ &= F(v_I - 1, \theta : u_1, u_2, \cdots, u_l) \\ &\quad + [Se_{v_I}(B_{v_I}) - Le_{v_I}(B_{v_I})] \\ &\quad + \frac{1}{\theta} [St_{v_I}(B_{v_I}) - Lt_{v_I}(B_{v_I})] \end{aligned} \tag{18}$$

Since $\{u_1, u_2, \cdots, u_l\}$ is an optimal solution to the $(v_I - 1)$-optimization problem, it is obvious that

$$F(v_I - 1, \theta : u_1, u_2, \cdots, u_l) \ge F(v_I - 1, \theta : v_1, v_2, \cdots, v_{I-1}) \tag{19}$$

Therefore, we have

$$\begin{aligned} & F(n, \theta : u_1, u_2, \cdots, u_l, v_I) \\ &\ge F(v_I - 1, \theta : v_1, v_2, \cdots, v_{I-1}) \\ &\quad + [Se_{v_I}(B_{v_I}) - Le_{v_I}(B_{v_I})] \\ &\quad + \frac{1}{\theta} [St_{v_I}(B_{v_I}) - Lt_{v_I}(B_{v_I})] \\ &= F(n, \theta : v_1, v_2, \cdots, v_{I-1}, v_I). \end{aligned} \tag{20}$$

On the other hand, since $\{v_1, v_2, \cdots, v_I\}$ is an optimal solution to the n-optimization problem, it is obvious that

$$F(n, \theta : u_1, u_2, \cdots, u_l, v_I) \le F(n, \theta : v_1, v_2, \cdots, v_{I-1}, v_I). \tag{21}$$

Therefore, we have

$$F(n, \theta : u_1, u_2, \cdots, u_l, v_I) = F(n, \theta : v_1, v_2, \cdots, v_{I-1}, v_I). \tag{22}$$

Hence, the theorem is proven. □

Before presenting the dynamic programming-based algorithm, we give the following definition. Define $F^*_{n,\theta}$ to be the maximum aggregate profit of $F(n, \theta : v_1, v_2, \cdots, v_k)$ obtained by solving the n-optimization problem and $I_{n,\theta}$ the maximum index in the optimal solution. If the optimal solution is an empty set, define $I_{n,\theta} = -1$.

Obviously, we have $I_{0,\theta} = -1$ and $F^*_{0,\theta} = 0$. From Theorem 1, it can be seen that if $I_{r,\theta} > 0$, $F_{i,\theta}$ satisfies

$$\begin{aligned} F_{I_r, \theta} = F_{I_r - 1, \theta} &+ \left[Se_{v_{I_r}}(B_{v_{I_r}}) - Le_{v_{I_r}}(B_{v_{I_r}}) \right] \\ &+ \frac{1}{\theta} \left[St_{v_{I_r}}(B_{v_{I_r}}) - Lt_{v_{I_r}}(B_{v_{I_r}}) \right] \end{aligned} \tag{23}$$

Therefore, we can check all possible locations of I_r $(0 \le r \le n)$ and select the one that maximizes $F(r, \theta : v_1, v_2, \cdots,$

v_k). So we have

$$\begin{cases} F^*_{0,\theta} = 0 \\ F^*_{r,\theta} = \max\limits_{1\le v_i\le r}\Big\{0, F^*_{v_i-1,\theta} + [Se_{v_i}(B_{v_i}) - Le_{v_i}(B_{v_i})] \\ \qquad\qquad + \frac{1}{\theta}\left[St_{v_i}(B_{v_i}) - Lt_{v_i}(B_{v_i})\right]\Big\} \end{cases}$$

and

$$\begin{cases} I_{0,\theta} = -1 \\ I_{r,\theta} = \begin{cases} -1 & \text{if } F^*_{r,\theta} = 0 \\ v & \text{if } F^*_{r,\theta} = F^*_{v-1,\theta} + [Se_v(B_v) - Le_v(B_v)] \\ & \qquad + \frac{1}{\theta}\left[St_v(B_v) - Lt_v(B_v)\right]. \end{cases} \end{cases}$$

The original problem can be solved using a dynamic programming based algorithm with the recurrences above. Theorem 1 ensures the correctness.

In the following, we present an analysis of the algorithm above. Let $g_{v_i,\theta}(B_{v_i})$ be the aggregate profit of placing B_{v_i} at storage v_i, i.e.,

$$\begin{aligned} g_{v_i,\theta}(B_{v_i}) &= Se_{v_i}(B_{v_i}) - Le_{v_i}(B_{v_i}) \\ &\quad + \frac{1}{\theta}\left[St_{v_i}(B_{v_i}) - Lt_{v_i}(B_{v_i})\right] \\ &= \sum_{A_x\in D(B_{v_i})} \Big[\Big(f_{v_i(A_p)} - f_{v_i^-(A_x)}(A_x)\Big)\,\Phi(v_i,\theta) - \Psi(v_i,\theta)\Big], \end{aligned}$$

where $\Phi(v_i,\theta) = s.e_{v_i}(A_x) + \frac{1}{\theta}st_{v_i}(A_x)$, and $\Psi(v_i,\theta) = \ell.e_{v_i}(A_x) + \frac{1}{\theta}\ell.t_{v_i}(A_x)$ and $h_{v_i,\theta}(B_{v_i})$ be the local profit of placing B_{v_i} at storage v_i, i.e.,

$$h_{v_i,\theta}(B_{v_i}) = \sum_{A_x\in D(B_{v_i})} \big[f_{v_i(A_p)}\cdot\Phi(v_i,\theta) - \Psi(v_i,\theta)\big],$$

The following theorem describes an important property of the algorithm.

THEOREM 2. *Suppose that $\{v_1, v_2, \cdots, v_I\}$ is an optimal solution to the n-optimization problem, then we have*

$$h_{v_i,\theta}(B_{v_i}) \ge 0 \quad \forall\ 1 \le i \le k$$

PROOF. From the definition of $g_{v_i,\theta}(B_{v_i})$ and $h_{v_i,\theta}(B_{v_i})$, it is ready to see that $g_{v_i,\theta}(B_{v_i}) \le h_{v_i,\theta}(B_{v_i})$. Suppose that there exists r such that $h_{v_r,\theta}(B_{v_r}) < 0$, then we have

$$\begin{aligned} &F(n,\theta : v_1, v_2, \cdots, v_I) \\ &= F(v_r - 1, \theta : v_1, v_2, \cdots, v_{r-1}) + g_{v_r,\theta}(B_{v_r}) \\ &\qquad + F(n - v_r, \theta : v_{r+1}, v_{r+2}, \cdots, v_I) \\ &\le F(v_r - 1, \theta : v_1, v_2, \cdots, v_{r-1}) + h_{v_r,\theta}(B_{v_r}) \\ &\qquad + F(n - v_r, \theta : v_{r+1}, v_{r+2}, \cdots, v_I) \\ &< F(v_r - 1, \theta : v_1, v_2, \cdots, v_{r-1}) \\ &\qquad + F(n - v_r, \theta : v_{r+1}, v_{r+2}, \cdots, v_I) \\ &\le F(n, \theta : v_1, \cdots, v_{r-1}, v_{r+1}, \cdots, v_I) \end{aligned} \tag{24}$$

which contradicts the fact that $\{v_1, \cdots, v_{r-1}, v_{r+1}, \cdots, v_I\}$ is an optimal solution to the n-optimization problem. Hence, the theorem is proven. □

Theorem 2 shows that we should only consider the storages where the local profit is beneficial.

6. SIMULATION MODEL

In this section, we describe the simulation model used for performance evaluation. We have performed extensive simulation experiments to compare the results of our model with those of existing models. We generated the simulation model from the empirical results presented in [6, 15, 13], which are considered to be reasonable.

The network topology was randomly generated by the Tier program [13]. We have conducted experiments for many topologies with different parameters and found that the performance of our model was relatively insensitive to topology changes. Here, we list only the experimental results for one topology due to space limitations. Table 1 shows the characteristics of this topology and the workload model, the parameters and their values used in our simulation.

We describe storage size as the total relative size of all objects available. We assume for our experiments that the object sizes follow a Pareto distribution. We also assume that each multimedia object has five versions and that the transcoding graph is as shown in Figure 5. The sizes of each version are assumed to be 100 percent, 80 percent, 60 percent, 40 percent, and 20 percent of the original object size. The transcoding cost is determined as the quotient of the object size to the transcoding rate. In our experiments, each storage randomly generates the requests, and the average request rate of each storage follows the distribution of $U(1, 9)$, where $U(x, y)$ represents a uniform distribution between x and y. The access frequencies of the objects maintained by a given storage follow a Zipf-like distribution [15, 20]. Specifically, the probability of a request for an object O in storage S is proportional to $1/(i^\alpha \cdot j^\alpha)$, where S is the i-th most popular storage and O is the j-th popular object in S. The delay between storage follows an exponential distribution, where the average delay is 0.23 seconds.

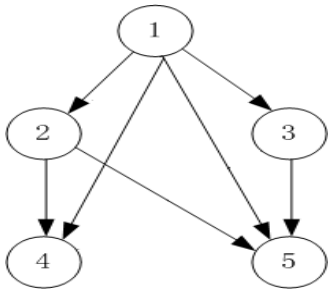

Figure 5: Transcoding Graph for Simulation

The transmission cost between storages is calculated by the access delay. For simplicity, the delay caused by sending the request and the relevant response for that request is proportional to the size of the requested object. The transcoding cost is calculated by the transcoding delay from one version to another during the processing of accessing. Similarly, the transmission energy cost and the transcoding cost are calculated by the computational ability of the storage that are proportional to the size of the requested object. Here, we consider the average object sizes for calculating both the transmission cost and the transcoding cost.

In addition to the model presented in Section 4.3, we also consider the mirror placement model for comparison purposes. In this model, each storage save the same objects. The performance metrics employed in the simulation include delay-saving ratio (DSR) [1], average access latency (AAL), request response ratio (RRR) [2], average hit ratio (OHR)[3], and exit hit ratio (EHR)[4]. In the following figures, TMP

[1] DSR is defined as the fraction of communication and server delays which is saved by satisfying the references from the storage instead of the server

[2] RRR is defined as the ratio of the access latency of the target object to its size .

[3] OHR is defined as the ratio of the number of requests satisfied by the storages as a whole to the total number of requests (including those requests that are served by transcoding in the storages).

[4] EHR is defined as the ratio of the number of requests satisfied by the storages as a whole to the total number of requests

Table 1: Parameters Used in Our Simulation

Parameter	Value
Number of WAN Nodes	500
Number of MAN Nodes	500
Delay of WAN Links	Exponential Distribution $p(x) = \theta^{-1}e^{-x/\theta}$ $(\theta = 0.48\ Sec)$
Delay of MAN Links	Exponential Distribution $p(x) = \theta^{-1}e^{-x/\theta}$ $(\theta = 0.58\ Sec)$
Number of Web Objects	1000 objects per server
Web Object Size Distribution	Pareto Distribution $p(x) = \frac{ab^a}{a-1}$ $(a = 1.2, b = 8596)$
Web Object Access Frequency	Zipf-Like Distribution $\frac{1}{i^\alpha}$ $(\alpha = 0.7)$
Relative Size Per Storage device	4%
Average Request Rate Per Storage	$U(1, 9)$ requests per second
Transcoding Rate	$25KB/Sec$

shows the results for the model proposed in Section 4.3 and MP shows the results for mirror placement.

7. PERFORMANCE EVALUATION

7.1 Impact of Storage Size

In this experiment set, we compare the performance of different models across a wide range of storage sizes, from 30 percent to 80 percent.

Figures 6, 7, 8 show the simulation results for DSR, AAL, and RRR. We can see that our model outperforms the MP model since our model considers placing multimedia objects in storages in a coordinated way, whereas MP considers placing multimedia objects only at a single storage. Clearly, the lower the AAL or the RRR, the better the performance. As we can see, all models provide steady performance improvement as the storage size increases. We can also see that $CMOTP$ significantly improves both AAL and RRR compared to MP, since our model determines the optimal locations for placing multimedia objects in a coordinated way, while the MP places multimedia objects at each storage.

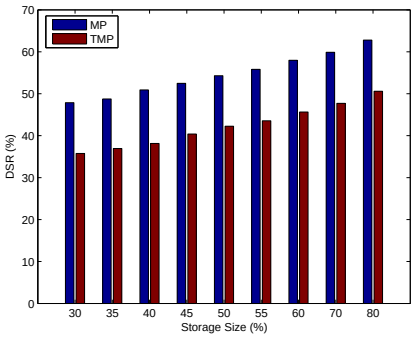

Figure 6: Simulation Results for DSR

Figures 9 and 10 show the results of AHR and EHR as a function of the relative storage size for different models. By computing the optimal locations, we can see that the results for our model can greatly outperform the MP model. We can also see that OHR steadily improves as the relative storage size increases, which conforms to the fact that more requests will be satisfied by the storage as the storage size becomes larger.

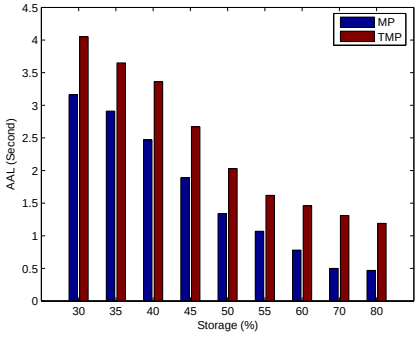

Figure 7: Simulation Results for AAL

7.2 Impact of Object Access Frequency

This experiment set examines the impact of object access frequency distribution on the performance of different models. Figures 11, 12, 13 show the performance results of DSR, RRR, and AHR for the values of Zipf parameter α from 0.1 to 0.9, respectively. We can see that $CMOTP$ consistently provides the best performance over a wide range of object access frequency distributions.

8. CONCLUDING REMARKS

This paper presented an energy-sensitive multimedia object distribution algorithm for wide-area transcoding-enabled storage systems. We proved that our technique provides optimal answers with small space and computational overhead. While our current focus is on storage systems that has only three modes (i.e., active, idle, and standby), we plan to investigate situations where storages with tunable RPM [7]. Further, it has been observed [23]that the energy consumption of storages can be reduced if requirement statistics are available in advance. The design of such "workload-aware" multimedia object distribution methods in storage systems constitutes an interesting topic. Finally, we would like to ex-

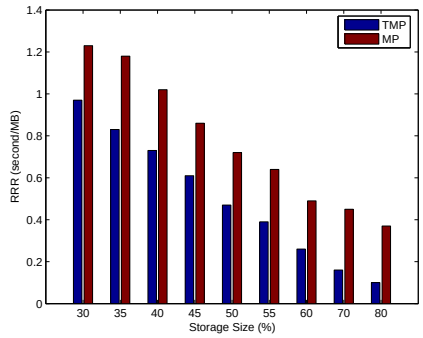

Figure 8: Simulation Results for RRR

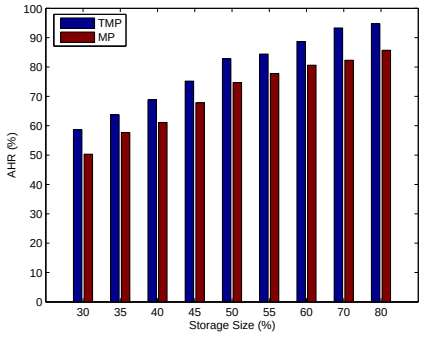

Figure 9: Simulation Results for OHR

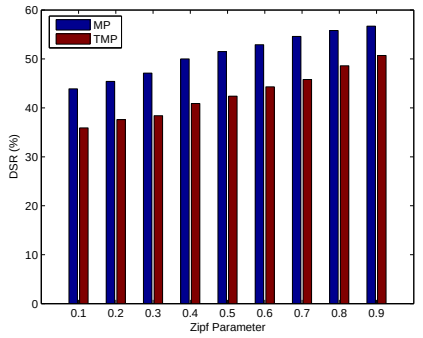

Figure 11: Simulation Results for DSR

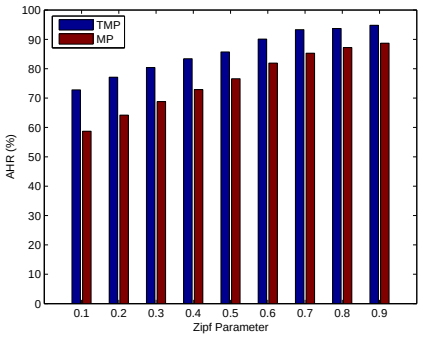

Figure 12: Simulation Results for RRR

plore the applicability of our distribution strategy for tracking the dynamically changing requirements of multimedia objects.

9. REFERENCES

[1] M. C. G. C. R. C. J. D. J. H. J. L. M. T. A. Adya, W.J. Bolosky and R. Watterhofer. Farsite: Federated, acailable, and reliable storage for an incompletely trusted environment. In *OSDI02 Conference Proceedings*.

[2] A. M. A. Haeberlen and P. Druschel. Glacier: Highly durable, decentralized storage despite massive correlated failures. In *NSDI05 Conference Proceedings*.

[3] K. R. A. Singh, A. Trivedi and P. Shenoy. Ptc : Proxies that transcode and cache. in heterogeneous web client environments. *WWW*, 7(1):7–28.

[4] G. F. I. X. D. B. Li, M. J. Golin and K. Sohraby. On the optimal placement of web proxies in the internet. In *INFOCOM99 Conference Proceedings*, pages 1282–1290.

[5] S. J. L. B. Shen and S. Basu. Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks. *IEEE Trans. on Multimedia*, 6(2):375–386.

[6] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *ACM SIGMETRICS'98 Conference Proceedings*, pages 151–160.

[7] R. Bianchini and R. Rajamony. Power and energy management for server systems. *IEEE Computer*,

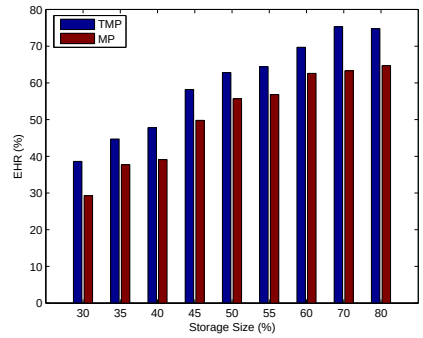

Figure 10: Simulation Results for ASL

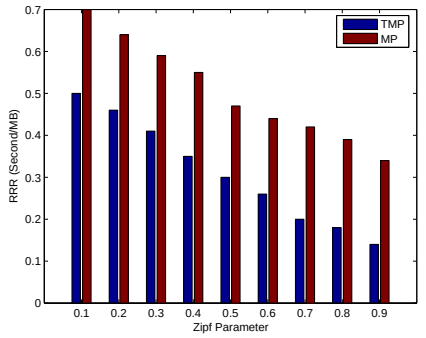

Figure 13: Simulation Results for OHR

37(11):68–74.

[8] M. C. R. L. P. S. Y. C. Canali, V. Cardellini. Cooperative architectures and algorithms for discovery and transcoding of multi-version contents. In *WCW 2003 Conference Proceedings*.

[9] C. Chang and M. Chen. On exploring aggregate effect for efficient cache replacement in transcoding proxies. *IEEE TPDS*, 14(6):611–624.

[10] E. P. E. V. Carrera and R. Bianchini. Conserving disk energy in network servers. In *ICS03 Conference Proceedings*, pages 86–97.

[11] Y. C. S. C. P. E. D. G. R. G. S. R. H. W. W. W. C. W. J. Kubiatowicz, D. Bindel and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *ASPLOS-IX Conference Proceedings*.

[12] B. L. J. Xu and D. L. Li. Placement problems for transparent data replication proxy services. *IEEE Journal on Selected Areas in Communications*, 20(7):1383–1398.

[13] M. B. D. K. L. Calvert and E. W. Zegura. Modelling internet topology. *IEEE Comm. Magazine*, 35(6):160–163.

[14] F. C. K. Li, H. Shen and S. Zheng. Optimal methods for coordinated en-route web caching for tree networks. *ACM TOIT*, 5(3):480–507.

[15] L. F. G. P. L. Breslau, P. Cao and S. Shenker. Web caching and zip-like distributions: Evidence and implications. In *IEEE INFOCOM'99 Conference Proceedings*, pages 126–134.

[16] K. Li and H. Shen. Optimal proxy placement for coordinated en-route transcoding proxy caching. *IEICE Trans. on IS*, E87-D(12):2689–2696.

[17] B. Moore. More power needed. *Energy User News*.

[18] B. Moore. Taking the data center power and cooling challenge. *Energy User News*.

[19] D. R. P. Krishnan and Y. Shavitt. The cache location problem. *IEEE/ACM TON*, 8(5):568–582.

[20] V. N. Padmanabhan and L. Qiu. The content and access dynamics of a busy site: Findings and implications. In *ACM SIGCOMM'00 Conference Proceedings*, pages 111–123.

[21] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *ICS04 Conference Proceedings*, pages 68–78.

[22] C. F. D. Z. L. Y. Z. Q. Zhu, F. M David and P. Cao. Power-aware storage cache management. *IEEE Trans. on Computers*, 54(5):587–602.

[23] L. T. Y. Z. K. K. Q. Zhu, Z. Chen and J. Wilkes. Hibernator: Helping disk arrays sleep through the winter. In *SOSP05 Conference Proceedings*, pages 177–190.

[24] A. S. H. F. N. V. S. Gurumurthi, J. Zhang and M. J. Irwin. Interplay of energy and performance for disk arrays running transaction processing workloads. In *ISPASS2003 Conference Proceedings*, pages 123–132.

[25] M. K. S. Gurumurthi, A. Sivasubramaniam and H. Franke. Drpm: Dynamic speed conrol for power management in server class disks. In *ISCA03 Conference Proceedings*, pages 169–179.

[26] X. Tang and S. T. Chanson. Coordinated en-route web caching. *IEEE TC*, 51(6):595–607.

[27] P. Y. V. Cardellini and Y. Huang. Collaborative proxy system for distributed web content transcoding. In *ACM ICIKM 200 Conference Proceedings*, pages 520–527.

[28] D. D. X. Jia, D. Li and J. Cao. On optimal replication of data object at hierarchical and transparent web proxies. *IEEE TPDS*, 16(8):1–13.

[29] X. H. W. W. X. Jia, D. Li and D. Du. Placement of web-server proxies with consideration of read and update cost on the internet. *The Computer Journal*, 46(4):378–390.