Data Placement in Intermittently Available Environments

Yun Huang and Nalini Venkatasubramanian

Dept. of Information & Computer Science, University of California, Irvine
Irvine, CA 92697-3425, USA
{yunh, nalini}@ ics.uci.edu

Abstract. In this paper, we address the problem of data placement in a grid based multimedia environment, where the resource providers, i.e. servers, are intermittently available. The goal is to optimize the system performance by admitting maximum number of users into the system while ensuring user Quality of Service (QoS). We define and formulate various placement strategies that determine the degree of replication necessary for video objects by using a cost-based optimization procedure based on predictions of expected requests under various time-map scenarios and QoS demands. We also devise methods for dereplication of videos based on changes in popularity and server usage patterns. Our performance results indicate the benefits obtained the judicious use of dynamic placement strategies.

1 Introduction

Global grid infrastructures [1, 11] enable the use of idle computing and communication resources distributed in a wide-area environment. Multimedia applications are resource intensive and can effectively exploit idle resources available on a grid. Many multimedia applications, e.g. streaming media, must ensure continuous access to information sources to maintain Quality-of-Service requirements. However, systems on a computational grid are not continuously available. Our objective is to ensure application QoS and effective resource utilization in "intermittently available" environments. We define "intermittently available" systems as those in which servers and service providers may not be available all the time. Effective load management in such an intermittently available environment requires: (1) Resource discovery and scheduling mechanisms that ensure the continuity of data to the user [9]; (2) Data placement mechanisms to ensure data availability. In this paper, we focus on the second problem.

Generally, a placement policy for a multimedia (MM) system will address: (1) how many replicas are needed for each MM object; (2) which servers the replicas should be created on; (3) when to replicate. Placement decisions directly affect requests acceptance ratios for different MM objects. A bad placement policy will deteriorate the system performance, since it causes unused replicas to occupy premium storage resources. An

ideal placement policy must be capable of adjusting the mapping of replicas on the servers according to the run-time request pattern. Previous work has addressed issues in data placement for servers that are continuously available [4]. In this paper, we propose placement strategies that consider the intermittent availability of servers in addition to the request patterns and resource limitations. Basically, we determine the degree of replication of an object by using a cost-based optimization procedure based on predictions of expected requests for that object. To enforce the replication decisions, the placement and dereplication strategies proposed take into consideration the time maps of server availability in addition to object popularity and server utilization.

We illustrate the system architecture in Section 2. Section 3 proposes a family of placement strategies for intermittently available environment. Section 4 introduces the time-aware predictive placement algorithm for dynamic object placement. We evaluate the performance of the proposed approaches in Section 5 and conclude in Section 6.

2 System Architecture

The envisioned system consists of clients and multimedia servers distributed across a wide area network (see Fig 1). The resources provided by the distributed servers include high capacity storage devices to store the multimedia data, processor, buffer memory, and NIC (network interface card) resources for real-time multimedia retrieval and transmission. Server availability is specified using a server time-map that indicates specific times when a server is available. The availability of resources on servers can vary dynamically due to request arrivals and completions; the stored data on a server also changes dynamically due

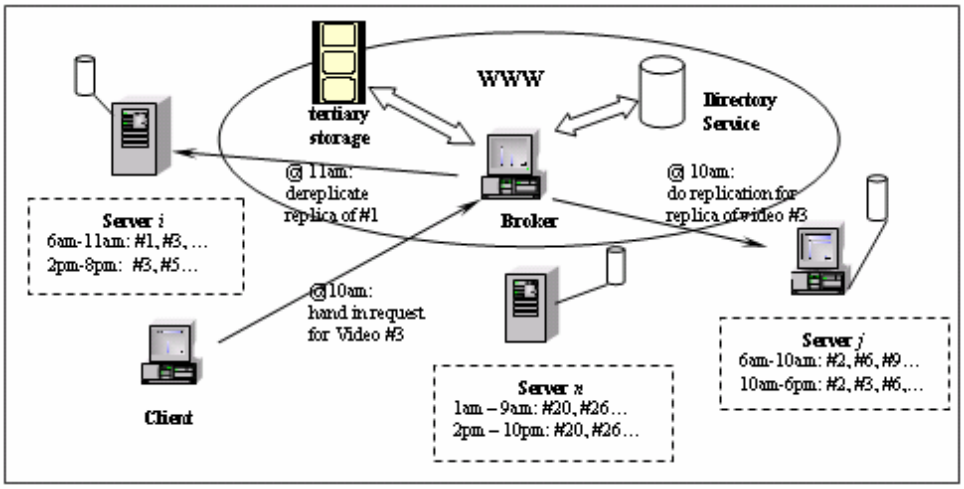

Fig. 1. System Architecture

to replication and dereplication of MM objects. To accommodate a large number of video objects, the environment includes tertiary storage. The key component of this architecture is a *brokerage service.* Specifically, the broker: determines an initial placement of video objects on servers; discovers the appropriate set of resources to handle an incoming request; coordinates resource reservation and schedules these requests on the selected resources; and initiates replication and dereplication of multimedia objects to cater to changes in request pattern and load conditions. Information required for effective data placement and resource provisioning include server resource availabilities, server time maps, replica maps, network conditions etc. This information is held in a directory service (DS) that is accessed and updated suitably by the broker.

Client requests for MM objects are routed to the broker that determines whether or not to accept the request based on current system conditions and request characteristics. Using state information in the DS, the broker determines a candidate server or a set of servers that can satisfy the request. Once a solution for the incoming request (i.e. scheduled servers and times) has been determined, the broker will update the directory service to reflect the allocated schedule. The goal is to improve the overall system performance and increase the number of accepted requests.

3 Placement Strategies

Given the time map and server resource configurations, an effective placement strategy will determine the optimal mapping of replicas to servers, so that the overall system performance is improved and accepted requests will be guaranteed QoS. Specifically, a placement policy for a MM system will provide decisions on which objects to replicate, how many replicas are required, where and when to replicate. Placement decisions can be made statically (in advance) or dynamically changed at runtime. We propose and compare a family of static and dynamic placement policies that can be used in intermittently available environments. Specifically, we devise and evaluate a Time-aware Predictive Placement (TAPP) algorithm for dynamic placement in a multimedia grid.

Static Placement Strategies: are determined at system initialization time; this placement is not altered during the course of request execution. Static placement policies may be popularity based where information on request popularity is taken into account when determining the replication degree of a MM object. We propose deterministic and non-deterministic policies to place the replicas on servers. Three static placement policies have been studied in this paper: (1) **SP1:** *Cluster-based static placement* – We cluster the servers into groups so that the total available service time of each group covers the entire day. Each video object is associated with exactly one group, so that every server in the group has a replica of this video object. (2) **SP2:** *Popularity-enhanced deterministic placement* – Here, we classify video objects into two groups – *very-popular* and *less-popular*. A replica of a very-popular video object is placed on every server, assuming resource availability. Less-popular video objects are evenly placed in the remaining

storage space. (3) **SP3:** *Popularity-based random placement* – Here, we choose the number of replicas for a video object based on its popularity, however, these replicas are randomly distributed among the feasible servers, i.e those that have available disk storage.

Dynamic Placement Strategies consider the available disk storage, the current load on the servers, and the changing request patterns for access to video objects so as to dynamically reconfigure the number of replicas for each video object and their placement. We model the request R from a client as: R: < VID_R , ST_R , ET_R , QoS_R >, where VID_R corresponds to the requested video ID; ST_R is the request start time; ET_R is the end time by which the request should be finished; QoS_R represents the resources required by the request, such as: the required disk bandwidth (R_{DBW}), required memory resource (R_{MEM}), required CPU (R_{CPU}) , and the required network transfer bandwidth (R_{NBW}); and the duration for which these resources are required (D_v). In order to deal with the capacity of each server over time in a unified way and represent how much a request will affect the server during the requested period of time, we define a *Load Factor*. Initially, we define a *Load Factor* (*R, S, t*) for a request r on server s at a particular time unit *t*, as:

$$\begin{aligned} \textit{Load Factor}\ (R, S, t) &= \text{Max}\ [\ CPU_a\ ,\ MEM_a\ ,\ NBW_a\ ,\ DBW_a\ ] \\ CPU_a &= R_{CPU} / S_{AvailCPU}(t),\ MEM_a = R_{MEM} / S_{AvailMEM}(t), \\ NBW_a &= R_{NBW} / S_{AvailNBW}(t),\ DBW_a = R_{DBW} / S_{AvailDBW}(t). \end{aligned} \quad (1)$$

Thus, the *Load Factor* of a server *S* at time *t*, LF(*R, S, t*), is determined by the bottleneck resource at time *t*. However, the duration of the requested period from ST_R to ET_R may cover multiple time units; we therefore use the average *Load Factor* over all time units (between ST_R and ET_R) during which the server is available. For example, if the granularity of the time units in a day is 24 (24 hours/day), and ST_R is 5am and ET_R is 10am, we consider

$$LF\ (S) = \text{Average}\ (LF_5, LF_6, LF_7, LF_8, LF_9, LF_{10}). \quad (2)$$

Dynamic placement strategies can be initiated on-demand to satisfy an individual request, or issued in advance based on predicted request arrivals. The efficacy of the on-demand technique depends on (a) the startup latency, i.e. how long a user will wait for a request to start and (b) the available tertiary storage bandwidth for replication. With low startup latencies, large amounts of server bandwidth and tertiary storage bandwidth are required to create replicas; hence the on-demand strategy may not always be feasible. In the following section, we propose a time-aware predictive placement approach that integrates server time-map information with request access histories to design a data placement mechanism for intermittently available environments.

4 The Time-Aware Predictive Placement (TAPP) Algorithm

The generalized *Time-Aware Predictive Placement* (TAPP) algorithm (Figure 2) has three main steps - *Popularity-Estimation, Candidate-server-selection, Pseudo-Replication*. An additional *Pseudo-Dereplication* phase may be added if sufficient storage space is not

available[1]. We assume that the TAPP algorithm executes periodically with a predefined prediction-period.

```
   Time-Aware Predictive Dynamic Placement Algorithm:
1  Obtain a snapshot of replica-maps and server state informaiton
   /* (2) to (5) below execute on the snapshot information and do not modify the actual data.*/
2  Initialize
   /*set  candidate_servers, add_replicas, del_replicas, replication_set, dereplication_set  = null */
3  Popularity-Estimation /* update add_replicas and del_replicas */
4  candidate_servers = Candidate-Server-Selection( )
5  if (storage-biased and  candidate_servers == null) then
   dereplication_set += Pseudo-Dereplication (system_wide) /* update  candidate_servers*/
6  while ( candidate_servers != null and add_replicas!= null) do
        replication_set = Pseudo-Replication()
        if (replication_set != null) then
            for each combination (Vi , Sj) in replication_set do
                if (Sj does not have enough storage for a replica of V, ) then
                    dereplication_set += Pseudo-Dereplication (server_specific)
        /* update replication_set, add_replicas and candidate_servers appropriately*/
7  if ( replication_set != null and dereplication_set != null) then
       do dereplication of video objects from appropriate real servers
       do replication of video objects on appropriate real servers
```

Fig. 2. Time-aware Predictive Placement Algorithm (TAPP)

Popularity-Estimation decides which video objects need more replicas based on a *rejection-popularity (RP)* factor defined for each video object V_i. Let #rejection(i) be the number of rejections of V_i in the last time period, #rejections be the total number of rejections for all videos within last time period, and #requests(i) be the number of requests for V_i in the last time period. Then $RP(i)$ is defined as:

$$RP(i) = \frac{\#rejections(i)}{\#rejections} * \#requests(i) . \quad (3)$$

Hence, the larger the *RP* is, the more problematic this video object is; which implies the necessity to add replicas. Furthermore, we group videos with rejection ratio > 5% into *add_replicas* list maintained in decreasing order of the *RP*. We also group videos with rejection ratio < 0.5% into a list *del_replicas*, maintained in ascending order of the RP.

Candidate-Server-Selection determines which servers will be considered for the creation of new replicas. It may be implemented using two approaches. A *Bandwidth – biased* approach chooses candidate servers using the current load and service time availability as primary criteria and ignores information about storage availability on servers. The *Storage – biased* approach uses the storage availability as a primary criterion

[1] Replication and dereplication decisions are not executed until a final mapping has been achieved – hence the names pseudo replication and pseudo dereplication.

in choosing candidate servers for new replicas. If there are no servers with sufficient disk storage, we choose servers that have replicas on the *del_replicas* list and attempt to dereplicate the less problematic replicas using the *Pseudo-Dereplication* process described later.

Table 1. Placement Cost Matrix (PCM)

	S_1	S_2	S_3
V_1	Min (N_1 ,1/LF(R_1, S_1)*T_1)	Min (N_1 ,1/LF(R_1, S_2)*T_2)	Min (N_1 ,1/LF(R_1, S_3)*T_3)
.	…	…	…
V_n	Min (N_n ,1/LF(R_n, S_1)*T_1)	Min (N_n ,1/LF(R_n, S_2)*T_2)	Min (N_n ,1/LF(R_n, S_3)*T_3)
	Max(PCM(V_n, S_1))	Max(PCM(V_n, S_2))	Max(PCM(V_n, S_3))

Pseudo-Replication: Using the set of servers from Step 2 (*candidate server selection*) and the *add_replicas* list obtained from Step 1 (*Popularity-Estimation)* we build a placement cost matrix, PCM (Table 1) in order to derive a mapping of video objects to data servers. The matrix represents the relative costs of servicing subscriber requests from each of the data servers. The columns represent data servers and rows represent video objects.

If server S_j already has a replica for video *Vi*, then the placement matrix entry is set as null, that is, this combination will not be considered for replication. Othewise, 1/ LF(R_i, S_j) represents the average number of requests similar to R_i that data server S_j can service per time-unit. To account for the intermittent availability of servers, we introduce a factor T_j to represent the duration of time for which server j is available in the next prediction-period. Then, the value (1/LF(R_i, S_j) * T_j) represents the average number of concurrently executing requests for video *i* that a server *j* can accept during the next prediction-period. N_i represents the number of rejections in the last period. We therefore set $PCM(V_i, S_j) = Min(N_i, (1/LF(R_i, S_j) * T_j)$ that represents the benefit that can accrue from allocating V_i to S_j.

After calculating the entries in the placement matrix, we choose the maximum value Max(PCM(V_i, S_j)). This gives us the object-server combination that is expected to improve request acceptance in the next prediction-period. We shrink the matrix by either deleting a column (if N_i >= 1/LF(R_i, S_j)*T_j) correspondingly decreasing the value of N_i by 1/LF(Ri, S_j)*tj; or deleting a row (if N_i <= 1/LF(R_i, S_j)*T_j) appropriately decreasing the available network bandwidth of S_j. We then repeat the pseudo-replication process until there are no more columns or rows left. At the end of the *Pseudo-Replication*, we have an ordered list of replication decisions that the broker can initiate.

Pseudo-Dereplication marks video objects in the *del_replicas* list (obtained during Popularity Estimation) as dereplicatable if they are not currently in use and/or have not been reserved for future use. Dereplication decisions can be made on a system wide basis or on a specific server. *System_wide dereplication* may be invoked prior to *Pseudo-Replication*. The goal is to dereplicate the least popular replica from lightly loaded servers

with more service time available. For each video V_k in *del_replicas*, we order the servers that have a copy of V_k in ascending order of their load-factors. We then pick the most beneficial replica for dereplication. The number of replicas that are selected for dereplication can be tailored to meet the storage needs of popular objects. *Server_specific dereplication* may be issued to ensure replication decisions made by the *Pseudo-Replication* process. In order to provide enough space for a new replica on a specific server, we may need to dereplicate appropriate number of videos from the *del_replicas* list that are not in use or have not been reserved for future use on that server.

5 Performance Evaluations

In this section, we evaluate the performance of the proposed placement strategies under various time-maps scenarios and server resource configurations.

Simulation Environment: We characterize incoming multimedia requests using a Zipfian distribution [6,13,14], with the request arrivals per day for each video V_i given:

$$\text{Pr. } (V_i \text{ is requested}) = \frac{K_M}{i}, \text{ where } K_M = \left(\sum_{i=1}^{M} \frac{1}{i} \right)^{-1}. \quad (4)$$

We compute the probability of request arrival in an hour j to be: $p_j = c/(j^{1-f})$ for $1 \le j \le 24$, where Φ is the degree of skew and assumed to be 0.8 and $c = 1/(\sum (1/j^{1-f})), 1 \le j \le 24$. Hence, the number of requests that arrive in each hour for each video V_j are computed.

The basic video server configuration used in this simulation includes 20 data servers each with 100 GB storage and 100Mbps network transfer bandwidth. For simplicity, the CPU and memory resources of the data servers are assumed not to be bottlenecks; besides the 100Mbps network bandwidth available for video streaming, a fraction of server bandwidth is reserved for replications into the server. In the following simulations, we set the duration of each video to be three hours; each video replica requires 2 GB disk storage, and 2 Mbps for network transmission bandwidth. In fact, these parameters will be changed for different purposes during the simulation.

In such an intermittently available system, specifically for each server, we use the service *time map* to keep the information of when the servers will be available during the span of a day. We study three approaches to model the time map of each server: (1) *T1*: *Uniform availability* – All the servers are available for an equal amount of time; and the servers are divided into groups, so that within each group, the time distribution covers the entire 24-hour day. (2) *T2*: *Random availability* – How long and when the servers are available are all randomly determined. (3) *T3: Total availability* – All the servers are available all the time. In the remainder of this paper, we will use T1, T2 and T3 to identify the three time map strategies.

Performance Results: Given the time map information and resource configuration of the video server system, we study the system performance by applying the scheduling polices for discovering intermittently available resources (DIAR) described in [9] and use the number of rejections (i.e. success of the admission control process) as the main metric of evaluation. In order to focus on the placement problem, we present our simulation results by using a scheduling policy where requests are initiated immediately (albeit startup latency) and continuously serviced without interruption by (possibly) multiple servers.

Overall performance of static placement under different time map patterns: Intuitively, when servers are always available (T3), the multiple server cases should always have a better acceptance rate, as in Fig 3. This is because, in general, the single server case is a constrained version of the multiple server case and therefore has fewer options for resource selection. With time-maps T2 and T1, we observe that the number of rejections of policy SP3 is much smaller than SP2 or SP1. Although both SP2 and SP3 take request popularity into account, SP3 is better than SP2, because the random distribution enlarges the possibility for a replica to be created on more servers with different time maps. Although SP1 (cluster-based placement) does not take the request popularity into account, it works well in conjunction with the T1 time-map since the clustering technique meshes well with the T1 time-map.

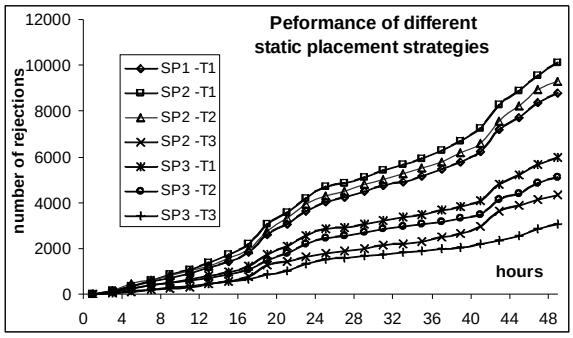

Fig. 3. Performance of different static placement strategies. The caption of "SPx-Ty" implies the SPx static placement and Ty time map model used.

Comparing dynamic placement strategies with static approaches: In order to avoid the influence of randomness, we choose the SP1 and T1 configurations. From Fig 4a, we observe that all the dynamic placement policies outperform SP1 with less number of rejections. The bandwidth biased TAPP seems to perform similar to the storage biased variant. However, comparing with Fig 4b, we observed that with more network bandwidth, the bandwidth-biased TAPP performs marginally better than storage-biased. We attribute this to the fact that when bandwidth is sufficient, the impact of effective storage utilization is less, since more effective load balance is achieved.

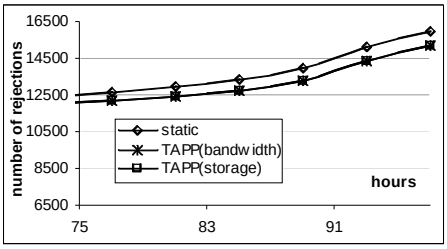

Fig. 4a. Placement strategies under limited network bandwidth resources.

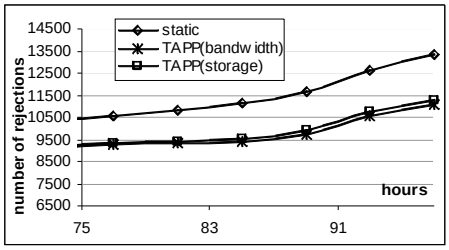

Fig. 4b. Placement strategies under large network bandwidth resource.

We also observe that more storage provides greater possibility of replicating popular videos yielding better acceptance ratios. However, our experiments indicate that beyond a point, merely increasing storage is insufficient to improve performance significantly. Other factors such as network bandwidth and server time-maps are bottlenecks to performance at high storage levels. We studied several factors that may influence the system performance such as time-map continuity, the number of data servers, etc. The TAPP policy exhibited low sensitivity to variations in time-map continuity; we believe that this policy is well suited to grid environments where systems exhibit randomized availability patterns. Furthermore, dynamic placement exhibits better performance with larger numbers of servers (as Fig 5) indicating the enhanced benefits that can be obtained as the size of the grid increases. We conclude that effective data placement can significantly improve system performance. Furthermore, dynamic placement is particularly effective when network bandwidth is limited.

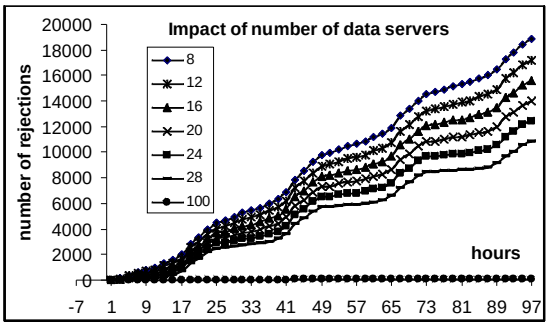

Fig. 5. The impact of number of data servers for TAPP (bandwidth-biased)

6. Related work and Future Research Directions

Data placement has been studied extensively in the context of distributed systems. [2] proposes a dissemination tree for replica placement algorithms which reduces the number of replicas deployed. Static data management policies for general purpose applications

(WWW) under resource constraints have been developed [12,3]. Research in data placement for multimedia servers addresses techniques to replicate frequently accessed videos to provide high data bandwidth and fault tolerance [10,7]; such algorithms are closely tied to the architectural configuration of the server. Dynamic segment replication [5] strategies have been proposed for cluster based video servers to replicate segments of files in order to be responsive to quick video load requests. Our prior work on dynamic placement policies for video servers based on predictions of future requests does not consider the intermittently availability of servers. In this paper, we have proposed and evaluated a family of data placement strategies to enable QoS-based services in a grid-based environment. We intend to conduct further performance studies with heterogeneous servers and combinations of request patterns. We also plan to explore effective middleware support in situations where server availability is not known ahead of time.

References

1. R.J. Allan. Survey of Computational Grid, Meta-computing and Network Information Tools, report, Parallel Application Software on High Performance Computers. (1999)
2. Y. Chen, R. H. Katz, J. Kubiatowicz, Dynamic Replica Placement for Scalable Content Delivery, 1st International Workshop on Peer-to-Peer Systems. (2002)
3. I. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. 12th ACM-SIAM Symposium on Discrete Algorithms (SODA), (2001)
4. C. Chou, L. Golubchik, and J.C.S. Lui, A Performance Study of Dynamic Replication Techniques in Continuous Media Servers, ACM (1999)
5. A. Dan, M Kienzle, D Sitaram, Dynamic Policy of Segment replication for load-balancing in video-on-demand servers. ACM Multimedia Systems, (1995)
6. A.L.Chervenak. Tertiary Storage: An Evaluation of New Applications, Ph.D. Thesis, University of California at Berkeley, December, (1994)
7. X. Wei, N. Venkatasubramanian, Predictive fault tolerant Placement in distributed video servers, ICME (2001)
8. N. Venkatasubramanian and S. Ramanathan, Effective Load Management for Scalable Video Servers, HP Laboratories Technical Report, (1996)
9. Y. Huang, N. Venkatasubramanian. QoS-based Resource Discovery in Intermittently Available Environments. 11th IEEE High Performance Distributed Computing, (2002)
10. M Chen, H. Hsiao, C. Li and P.S. Yu, Using Rotational Mirrored De-clustering for Replica Placement in a Disk-Array-Based Video Server, ACM Multimedia, (1995)
11. I. Foster, C. Kesselman. The Grid: Blueprint for a New Computing Infrastructure, book, preface, (1998)
12. C. Krick, H. Räcke, M. Westermann Approximation Algorithms for Data Management in Networks. 13th ACM Symposium on Parallel Algorithms and Architectures. 237-246, (2001)
13. A. Dan and D.Sitaram. An online video placement policy based on bandwidth to space ration (bsr). In *SIGMOD* '95, pages 376-385, (1995)
14. A. Dan, D. Sitaram, P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching, 2th ACM Multimedia Conference and Exposition, (1994).