RC 20410 (03/15/96)
Computer Science/Mathematics

IBM Research Report

MMPacking: A Load and Storage Balancing Algorithm for Distributed Multimedia Servers

Dimitrios N. Serpanos

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

Leonidas Georgiadis

Dept. of Electrical Engineering
Univ. of Thessaloniki
Thessaloniki, Greece

Tasos Bouloutas

Bank of Boston
Treasury Department
Boston, Massachusetts

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

MMPacking: A Load and Storage Balancing Algorithm for Distributed Multimedia Servers

D. N. Serpanos* L. Georgiadis† T. Bouloutas‡

Abstract

In distributed multimedia servers where client requests for different video streams may have different probabilities, placement of video streams is an important parameter, because it may result in unbalanced requests to the system's stations, and thus to high blocking probabilities of requests. We present a method, MMPacking, to balance traffic load and storage use in a distributed server environment. Since different video streams are requested by clients with different rates, video stream replication is used to balance the traffic patterns of the stations; thus, the requests and I/O usage of the stations is balanced, since replication allows requests for the same video stream to be routed to different stations. MMPacking achieves load balancing by producing at most $N-1$ replicas of video streams in a system with N servers. These replicas are distributed among the stations, so that storage balancing is achieved as well, since no station stores more than 2 video streams than any other station in the system.

*IBM, T.J. Watson Research Ctr., P.O. Box 704, Yorktown Heights, New York 10598, USA

†Dept. of Electrical Eng., Univ. of Thessaloniki, Thessaloniki, GREECE

‡Bank of Boston, Treasury Department, Mail-Stop: 01-12-08, 100 Federal St., P.O. Box 2016, Boston, MA 02110, USA

1 Introduction

Servers are an integral part of multimedia environments such as digital libraries, in-house training systems, video-on-demand or near-video-on-demand services, etc. In a typical environment, a multimedia server is connected through an interconnection to clients which request information on demand. The servers not only store and provide information to clients, but they perform various management operations as well, such as billing, accounting, encryption, etc.

One of the most important elements (objects) of a multimedia service environment are video streams. These objects require high I/O bandwidth at the server for their delivery as well as large amounts of memory for their storage. Due to these demanding characteristics, we focus on video streams in this paper, although in principle our method applies to other types of objects as well.

Servers can be classified in general in two large categories: centralized and distributed. In centralized servers, a high-end system stores, serves and manages the video streams, while in a distributed server, a collection of workstations or PC's may constitute the server. As it is not clear which type of server provides the most cost-effective solution [1] [8], a lot of effort has been concentrating on distributed server environments [5] [7] [4] [2] [3]. In this paper, we provide a method to balance load and storage in a cluster of workstations or PCs constituting a multimedia server.

The most important parameters in a multimedia server are its I/O bandwidth requirements and its storage requirements. The I/O bandwidth designates how many clients can be simultaneously served, while the available amount of storage determines the number of video streams that can be stored in the server. In a centralized server, one can easily observe that no client request for a video stream can be blocked (denied) as long as there is I/O bandwidth available. On the other hand, this is not necessarily true for a distributed server: assume that a client request arrives at the server for a stream that is stored on a station whose I/O bandwidth is allocated to other served streams; although there may be aggregate I/O bandwidth available in the system, the request has to be blocked. One solution to resolve such problems is to provide copies of video streams in multiple stations, so that alternative stations can serve a request. This demonstrates that placement and replication of video streams is an important issue in distributed servers. Considering that replication increases storage requirements and that different video streams are requested by clients with different probabilities, we need to develop methods that minimize excess storage requirements in distributed multimedia server environments, while they serve the required number of clients. This is the problem we address in this paper.

We provide a method, MMPacking, that achieves load and storage balancing in the distributed server by replicating a small number of video streams in the server, and by employing a weighted scheduling algorithm to achieve load balancing. This results to the probability of a video request being the same for all stations in the distributed server. When client requests are uniformly routed to disks (or servers, assuming that a server corresponds to one disk), then the blocking probability of a client's request is minimized as has been shown in [7], although no method of movie assignment to disks or servers is given in the reference. In addition to traffic load balancing, MMPacking achieves storage balancing, since it results to approximately equal number of video streams stored in all stations; more precisely, the number of video streams stored on any two workstations differs by a number less or equal to 2.

The paper is organized as follows. Section 2 provides a formal presentation of the problem and introduces the notation used. Section 3 describes MMPacking and proves that balancing is achieved, while Section 4 describes a possible straightforward implementation of MMPacking. We conclude in Section 5.

2 Problem Definition and Notation

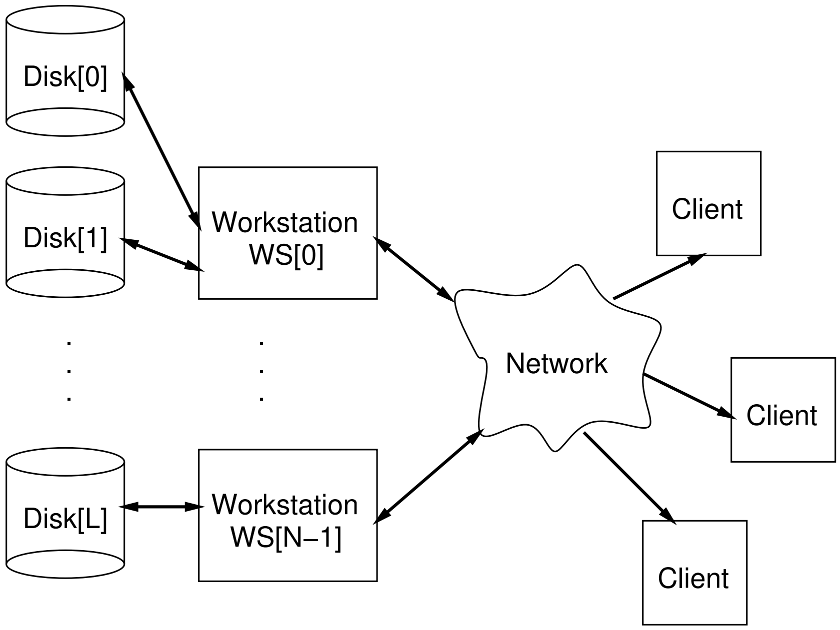

Figure 1: Distributed Server Configuration

We consider an environment as shown in Figure 1, where a distributed video server is employed to deliver video services to clients. The server is composed of N stations, $WS[0], \ldots, WS[N-1]$, storing a total of M different video streams, $VS_0, \ldots, VS_{M-1}$, and is connected to a total of C clients through a network as shown in the figure. Naturally, as is the case in real systems, we assume that $M \gg N$.

Clients generate requests for video streams to the server. Video streams are requested with different probabilities: video stream VS_i is requested with probability P_i. Clearly, $\sum_{i=0}^{M-1} P_i = 1$.

Given a placement of video streams to the workstations, the probability that a client's request is routed to a certain workstation $WS[i]$ is equal to the cumulative probability, SP_i, of the video streams stored on $WS[i]$. MMpacking achieves a cumulative probability $SP_i = \frac{1}{N}$ for $0 \leq i \leq (N-1)$. Denoting as NS_i the number of streams stored on workstation $WS[i]$, MMPacking also achieves storage balancing by resulting in a configuration with $|NS_i - NS_j| \leq 2$, for all i, j.

3 MMPacking

MMPacking is a method that achieves load and storage balancing with two basic operations:

1. placement of video streams in server stations and replication of a small subset of video streams;
2. weighted scheduling of client requests for the replicated video streams.

We select a small number of video streams for replication in the following fashion. We sort the video streams in ascending order using their corresponding probabilities, P_i, as the key. Without loss of generality, we assume that the given order of video streams $VS_0,\ VS_1,\ \ldots,\ VS_{M-1}$ is already sorted; so, for the corresponding probabilities of request $P_0,\ P_1,\ \ldots,\ P_{M-1}$ we have: $P_0 \leq\ P_1 \leq\ \ldots \leq\ P_{M-1}$.

Assuming an ordering of the workstations, $WS[0],\ \ldots,\ WS[N-1]$, we place the video streams in a round robin fashion to all the workstations, so that VS_0 is first placed in $WS[0]$, VS_1 is then placed in $WS[1]$, and so on. In this fashion, the placement of the video streams occurs in rounds. In round R_1 the first N video streams are placed on the N workstations: VS_0 on $WS[0]$, VS_1 on $WS[1]$, ..., and VS_{N-1} on $WS[N-1]$; in round R_2 the next N video streams are placed on the N workstations: VS_N on $WS[0]$, VS_{N+1} on $WS[1]$, ..., and VS_{2N-1} on $WS[N-1]$; and so on. With this scheme, in round R_k, workstation $WS[i]$ will store video stream $VS_{(k-1)N+i}$. So, MMPacking will place VS_0, VS_N, VS_{2N}, $\ldots$ on $WS[0]$, VS_1, VS_{N+1}, VS_{2N+1}, $\ldots$ on $WS[1]$, VS_2, VS_{N+2}, VS_{2N+2}, $\ldots$ on $WS[2]$, and so on.

As each video stream is placed in the server on a workstation, the cumulative probability of the stored video streams is calculated: when $VS_{(k-1)N+i}$ is placed on $WS[i]$ during round R_k, the cumulative probability SP_i^k becomes:

$$SP_i^k = SP_i^{k-1} + P_{(k-1)N+i} = P_i + P_{N+i} + \ldots + P_{(k-1)N+i} = \sum_{l=1}^{k} P_{(l-1)N+i}$$

Corollary 3.1 *Since all the video streams are ordered in ascending order, the following holds for the cumulative probabilities:*

$$SP_0^k \leq SP_1^k \leq SP_2^k \leq \ldots \leq SP_{N-1}^k \leq SP_0^{k+1} \leq \ldots$$

The placement of the streams takes place as above, as long as the cumulative probability of the stations is smaller than or equal to $\frac{1}{N}$. If all streams are assigned to workstations without the cumulative probability of any workstation becoming larger than $\frac{1}{N}$, then all the cumulative probabilities will be equal to $\frac{1}{N}$. To see this notice that in this case the sum of the cumulative probabilities is equal to the sum of the probabilities of the streams, i.e., equal to one. Since $SP_i^k \leq \frac{1}{N}$, $0 \leq i \leq N-1$, this can occur only if $SP_i^k = \frac{1}{N}$, $0 \leq i \leq N-1$.

Replication takes place as soon as the cumulative probability of a station exceeds $\frac{1}{N}$, in the following fashion:
Assume that the first time the cumulative probability of a workstation, workstation $WS[j]$, becomes larger than $\frac{1}{N}$ occurs at round k. Therefore, when the k-th video stream, i.e. stream $VS_{j_1} = VS_{(k-1)N+j}$, is placed on $WS[j]$ in round R_k, we have, $SP_j^k > \frac{1}{N}$. In this case, we replicate the video stream VS_{j_1} and place the second copy on $WS[j+1]$ (if $j = N-1$ then $j+1$ is replaced with 0), so that VS_{j_1} appears on both workstations $WS[j]$ and $WS[j+1]$. We consider that VS_{j_1} is replaced by two copies: $VS_{j_1}^1$, and $VS_{j_1}^2$ with probabilities $P_{j_1}^1 = \frac{1}{N} - SP_j^{k-1}$ and $P_{j_1}^2 = Pj_1 - P_{j_1}^1$, respectively (therefore, $P_{j_1} = P_{j_1}^1 + P_{j_1}^2$). $P_{j_1}^1$ is added to SP_j^{k-1} so that $SP_j^k = \frac{1}{N}$, and $P_{j_1}^2$ is added to SP_{j+1}^{k-1}. If $SP_{j+1}^{k-1} + P_{j_1}^2 > \frac{1}{N}$, we repeat the replication phase to the next workstation $WS[j+2]$ (and set $SP_{j+1}^k = \frac{1}{N}$ accordingly), until we reach a workstation $WS[r]$ with $SP_r^l \leq \frac{1}{N}$ after storing the replicated copy of VS_{j_1}. We next proceed with the placement of the remaining streams following the same algorithm whenever the cumulative probability of a workstation exceeds $\frac{1}{N}$.

Considering that replication starts for the first time at round R_k, the workstation $WS[r]$ where replication of VS_{j_1} ends is reached either in the same round, R_k, and $j < r \leq (N-1)$, or in the subsequent round, R_{k+1}, and $0 \leq r < j$, as the following lemma shows.

Lemma 3.1 *Using MMPacking, no video stream will be replicated twice on the same workstation.*

Proof: Assume that replication occurs at round R_k, and video stream VS_{j_1} is replicated to $WS[j+1]$, i.e. workstation $WS[j]$ is the first one reaching cumulative probability larger than $\frac{1}{N}$. Then, the cumulative probabilities calculated in all stations before placement of VS_{j_1} on $WS[j]$ are:

$$SP_i^k = \sum_{l=1}^{k} P_{(l-1)N+i}, \quad 0 \leq i < j$$
$$SP_i^{k-1} = \sum_{l=1}^{k-1} P_{(l-1)N+i}, \quad j \leq i \leq N-1.$$

The sum of all these cumulative probabilities is: $SP = \sum_{i=0}^{(k-1)N+(j-1)} P_i$, i.e., it is the sum of the distinct probabilities of request of all the $(k-1)N+(j-1)$ video streams placed before VS_{j_1}.

If VS_{j_1} is replicated to workstation $WS[j]$ twice, then there are N copies of VS_{j_1}, which have been placed on stations $WS[j], WS[j+1], ..., WS[N-1], WS[0], ..., WS[j-1]$, and have probabilities $P_{j_1}^1, P_{j_1}^2, ..., P_{j_1}^N$, respectively. Since VS_{j_1} has to be replicated on $WS[j]$ for a second time, then $P_{j_1} > \sum_{i=1}^{N} P_{j_1}^i$, while by the design of algorithm, all the cumulative probabilities of all workstations have reached $\frac{1}{N}$:

$$SP_j^k = SP_{j+1}^k = ... = SP_{N-1}^k = SP_0^{k+1} = SP_1^{k+1} = ... = SP_{j-1}^{k+1} = \frac{1}{N}$$

This implies that $SP_j^k + ... + SP_{N-1}^k + SP_0^{k+1} + ... + SP_{j-1}^{k+1} = 1$, i.e., $\sum_{i=0}^{(k-1)N+(j-1)} P_i + \sum_{i=1}^{N} P_{j_1}^i = 1 < \sum_{i=0}^{(k-1)N+j} P_i$. So, we reach a contradiction. $\Box$

So, if $WS[r]$ is the last workstation where VS_{j_1} is replicated, then either $j < r \leq (N-1)$ and replication of VS_{j_1} ends in round R_k or, $0 \leq r < j$ and replication of VS_{j_1} ends in round R_{k+1}. Actually, all replication of video streams ends either in round R_k at $WS[r]$ with $j < r \leq (N-1)$, or in round R_{k+1} at station $WS[r]$ with $0 \leq r < j$. Moreover, when the placement of all streams ends, the cumulative probability of any workstation is $\frac{1}{N}$. This is shown in the next theorem.

Theorem 3.1 *If the first replication of MMPacking occurs at round R_k, and requires that the video stream VS_{j_1} is replicated and placed on workstation $WS[j+1]$, then all video stream placement and replication will end, either during R_k at $WS[r]$ with $j < r \leq (N-1)$ or, during R_{k+1} at $WS[r]$ with $0 \leq r < j$. Moreover, if the placement of all streams ends at round R_m ($m = k$ or $m = k+1$), then $SP_i^m = \frac{1}{N}$, $0 \leq i \leq r$, and $SP_i^{m-1} = \frac{1}{N}$, $r < i \leq (N-1)$.*

Proof: Let us assume that VS_{j_1} is replicated on $WS[j], \ldots, WS[s]$, $j < s \leq (N-1)$, in round R_k, so that $SP_s^k \leq \frac{1}{N}$ after placement of the copy of VS_{j_1} as MMPacking requires. If $SP_s^k = \frac{1}{N}$, i.e. the cumulative probability of $WS[s]$ at round R_k has already reached $\frac{1}{N}$, we proceed to $WS[s+1]$, otherwise we proceed with $WS[s]$. In the following we assume that we continue with $WS[s]$, i.e. $SP_s^k < \frac{1}{N}$. [1]

Placement of VS_{j_1+1} on $WS[s]$ will lead to replication of VS_{j_1+1}, since the cumulative probability of $WS[s]$ becomes:

$$SP_s^k = SP_s^{k-1} + P_{j_1}^{N-1-s} + P_{j_1+1} > \frac{1}{N}$$

To see this recall that, $SP_j^{k-1} + P_{j_1} > \frac{1}{N}$, since VS_{j_1} was replicated to $WS[j+1]$; so,

$$SP_s^k = SP_s^{k-1} + P_{j_1}^{N-1-s} + P_{j_1+1} > SP_s^{k-1} + P_{j_1+1} \geq SP_j^{k-1} + P_{j_1}$$

since $SP_s^{k-1} \geq SP_j^{k-1}$ and $P_{j_1+1} \geq P_{j_1}$ as mentioned above.
This results to $SP_s^k > \frac{1}{N}$ which implies that VS_{j_1+1} will be replicated to $WS[s+1]$, and probably to other workstations as well[2]. So, when replication starts at round R_k, all video streams after VS_{j_1+1} will be replicated.

Based on the above, to prove the lemma, we need to show that all replication will finish before the round-robin placement of streams returns to $WS[j]$ in round R_{k+1}. This is easily proven by contradiction: as placement and replication occurs moving from $WS[j]$ towards $WS[N-1]$ in round R_k, all cumulative probabilities SP_j^k, SP_{j+1}^k, ... become equal to $\frac{1}{N}$. If placement continues in round R_{k+1} for $WS[i]$, such that $i < j$, then $SP_i^{k+1} = \frac{1}{N}$ as well. If eventually a stream VS_q required replication on $WS[j]$ at round R_{k+1}, then the sum of probabilities of all streams $P_0, \ldots, P_q$ should be:

$$\sum_{i=0}^{q} P_i > 1$$

which is a contradiction, since $\sum_{i=0}^{M} P_i = 1$.
Finally, since at any step of the algorithm the cumulative probability of any station does not exceed $\frac{1}{N}$ and when replication of all streams ends (assume that it ends at workstation $WS[r]$) the sum of the cumulative probabilities is equal to 1, we must necessarily have:

$$SP_{r+1}^{m-1} = \ldots = SP_{N-1}^{m-1} = SP_0^m = \ldots = SP_r^m = \frac{1}{N}.$$

□

[1]The same argument applies if we proceed with $WS[s+1]$.

[2]The same argument holds, if we assume that replication of VS_{j_1} ends at round R_{k+1}, and $0 \leq s < j$.

The previous theorem shows that the proposed algorithm equalizes the cumulative probabilities of the streams on all workstations. The next theorem shows that the algorithm achieves storage balancing.

Theorem 3.2 *The maximum number of replicas in the system is $(N-1)$, and the difference of the number of video streams stored in any two stations is at most* 2.

Proof: Based on Theorem 3.1, it is clear that the number of visited stations after (and including) the first replication is $(N-1)$. As only one replica is produced in every visit to a workstation, the total number of replicas is bounded by $(N-1)$.

Let us denote as NS_i^k the number of streams that have been placed on $WS[i]$ in all rounds up to (and including) R_k. If replication starts at round R_k with the first replica placed on station $WS[j+1]$, then all stations $WS[l]$, $0 \leq l \leq j$, at R_k have $NS_l^k = k$ streams each. Every workstation visited by MMPacking after the first replication receives at most 2 streams: either one new stream or a replica and a new stream[3].

So, if MMPacking ends during R_k at $WS[r]$, $j < r \leq (N-1)$, then for the workstations we have:

$$\begin{aligned} NS_0^k &= k \\ &\dots \\ NS_j^k &= k \\ k \leq NS_{j+1}^k &\leq (k+1) \\ &\dots \\ k \leq NS_r^k &\leq (k+1) \\ NS_{r+1}^k = NS_{r+1}^{k-1} &= k-1 \\ &\dots \\ NS_{N-1}^k = NS_{N-1}^{k-1} &= k-1 \end{aligned}$$

Clearly, $|NS_i^k - NS_j^k| \leq 2$, for all i, j.

[3]Only one stream is placed on a workstation $WS[r]$, if the probabilities are such that after placement of a replicated copy on $WS[r-1]$, the cumulative probability of $WS[r-1]$ is: $SP_{r-1}^k = \frac{1}{N}$.

If MMPacking ends during R_{k+1} at $WS[r]$, $0 \leq r < j$, then at the end of R_k:

$$\begin{aligned} NS_0^k &= k \\ &\ldots \\ NS_j^k &= k \\ k \leq NS_{j+1}^k &\leq (k+1) \\ &\ldots \\ k \leq NS_{N-1}^k &\leq k-1 \end{aligned}$$

As MMPacking may add at most 2 streams in every workstation visited in R_{k+1}, and given that MMPacking ends at $WS[r]$:

$$\begin{aligned} k \leq NS_0^{k+1} &\leq k+2 \\ &\ldots \\ k \leq NS_r^{k+1} &\leq k+2 \\ NS_{r+1}^{k+1} = NS_{r+1}^k &= k \\ &\ldots \\ NS_j^{k+1} = NS_j^k &= k \\ NS_{j+1}^{k+1} = NS_{j+1}^k &\leq k+1 \\ &\ldots \\ NS_{N-1}^{k+1} = NS_{N-1}^k &\leq k+1 \end{aligned}$$

So, clearly, $|NS_i^k - NS_j^k| \leq 2$, for all i, j. $\square$

The previous theorem shows that storage balancing is achieved by the proposed algorithm. To achieve load balancing as well, the scheduling of video stream requests from clients is weighted according to the probabilities assigned to each stream by the algorithm. Scheduling of requests for non-replicated video streams is straightforward since there is only one choice. For replicated streams, assume that stream VS_i is replicated between $WS[k]$, $WS[k+1]$, $\ldots$, $WS[r]$, and that the corresponding probabilities are P_i^1, P_i^2, $\ldots$, P_i^{r-k+1}. Then the proportion P_i^1/P_i of requests for video stream VS_i is routed to workstation $WS[k]$, the proportion $\frac{P_i^2}{P_i}$ is routed to workstation $WS[k+1]$, $\ldots$, and the proportion $\frac{P_i^{(r-k+1)}}{P_i}$ is routed to workstation $WS[r]$[4]. There are various methods

[4] $P_i = P_i^1 + P_i^2 + \ldots + P_i^{r-k+1}$.

for achieving this weighted round robin, e.g., see [6]. Notice that since the cumulative probability of streams on any workstation is $\frac{1}{N}$, the proportions of requests routed to each workstation is the same and load balancing is achieved.

To summarize, the combination of placement and replication with the weighted round-robin results to:

1. uniform probability of requests ($\frac{1}{N}$) routed to each workstation;

2. similar storage requirements to all workstations.

4 Implementation

```
Sort P_j in ascending order
for (i= 1; i <  M; i++)
{
   WS = (WS+1) modN;
   SP[WS] = SP[WS] + P_i;
   while (SP[WS] >  1/N)
   {
      tmp= SP[WS]-1/N;
      SP[WS]= 1/N;
      WS= (WS+1) modN;
      SP[WS] = SP[WS] + tmp;
   }
}
```

Figure 2: MMPacking Placement and Replication Pseudocode

The scheme can be implemented fully and easily in software without any need of any special hardware. The pseudo-code for the algorithm choosing the video streams to replicate and calculating the corresponding variables is given in Figure 2.

5 Conclusions

MMPacking is a powerful technique that achieves traffic load and storage balancing in distributed video servers. In this fashion it achieves minimal blocking probability of a client's request, while keeping the extra memory required for replicated copies in the system minimal, since it places at most $(N-1)$ replicas of video streams in a system of N workstations —the number of replicas is independent of the total number of video streams stored in the server. Furthermore, the total number of video streams and replicas is divided almost evenly among the stations of the server. Achievement of storage balancing leads to symmetric systems that are easily scalable.

References

[1] A. Bouloutas and D. N. Serpanos. A Comparison of Multimedia Server architecture. In *Proceedings of the International Conference on Multimedia Communications (MEDIACOMM'95), Southampton, U.K.*, pages 136–138, April 11-12 1995.

[2] A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and Caching in Large Scale Video Servers. Technical Report RC 19903, IBM Research, January 1995.

[3] A. Dan and D. Sitaram. Application Routing in Distributed Multimedia Database Systems. Technical Report RC 20284, IBM Research, Yorktown Heights, 1995.

[4] J. K. Dey-Sircar, J. D. Salehi, J. F. Kurose, and D. Towsley. Providing VCR Capabilities in Large-Scale Video Servers. In *Proceedings of ACM Multimedia'94*, pages 25–32, October 1994.

[5] K. D. Huynh and T. M. Khoshgoftaar. Performance Analysis of Advanced I/O Architectures for PC-based Video Servers. *ACM/Springer Multimedia Systems*, 2(1):36–50, June 1994.

[6] J.D. Aman, et al. Goal-Oriented CPU Scheduling for MVS Algorithms and Experimental Results. Technical Report RC 17978, IBM Research Division, 5/7/92.

[7] T. D. C. Little and D. Venkatesh. Popularity-Based Assignment of Movies to Storage Devices and Video-on-Deman System. *ACM/Springer Multimedia Systems*, 2(6):280–287, January 1995.

[8] D. N. Serpanos and T. Bouloutas. Centralized vs distibuted multimedia servers. Technical Report RC 20411, IBM Research Division, T. J. Watson Research Center, March 1996.