Data Management for Multi-User Access to Digital Video Libraries*

J. Leon Zhao
School of Business and Management
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Doron Rotem
College of Business
San Jose State University
San Jose, CA 95192, USA
and
Data Management Group
Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA

Su-Shing Chen
Department of Computer Engineering and Computer Science
University of Missouri
Columbia, MO 65211, USA

Abstract

To support heterogeneous application types a video digital library will contain a large number of video objects with various lengths and display requirements. Multi-user access to the same video objects is required in order to increase the availability of video information and to make full use of the limited computing and storage resources. The access frequency and delay sensitivity of video objects require special methods to guarantee smooth playback of video objects and to minimize average waiting time. We propose an integrated approach to buffer and disk management for dynamic loading and simultaneous delivery of multiple video objects to multiple users. The allocation of buffer and disk resources in this study is based on quality of service variables such as average waiting time, display continuity, and viewer enrollment.

To Appear in the Special Issue of the Journal of Parallel and Distributed Computing on Distributed Multimedia Systems

*This research was supported in part by the Department of Energy under contract DE-AC03-76SF00098 and by the Hong Kong University of Science and Technology under contract DAG95/96.BM79.

Contents

1 Introduction

Storage and delivery of video data have been the topic of many recent research works and prototype implementations, but much of this research is directed towards the technology needed to build Video on Demand (VOD) servers for entertainment purposes [8, 9]. However, we address the unique requirements of video storage and delivery in digital libraries since there are major differences between the VOD and digital library services in terms of the characteristics of the video data and the viewing frequencies.

VOD servers are geared towards servicing a large number of users who view a limited collection of videos with relatively homogeneous characteristics such as length, viewing frequencies, etc. In contrast, the collection of videos actively viewed by users can be much larger in digital libraries than that in VOD, and the video lengths in digital libraries can vary from a few seconds (scientific animations) to a few minutes (product demonstrations) and to a couple of hours (lecture recordings and full feature videos). Furthermore, the viewing frequencies of videos in digital libraries may exhibit a large variance as some videos may be highly popular whereas others only requested occasionally.

The main challenges in managing video objects arise from their large size and real-time playback requirements [4, 10, 18]. One of the most common video compression standards currently in use is called MPEG [11]. A typical one hour video in MPEG compressed form requires about 1 to 2 Gigabytes of disk storage [2, 12] and must be delivered at the rate of 1.5 Megabits/second to avoid "jerkeness" [1, 4, 8, 9]. Furthermore, in many cases a video object must be delivered synchronously with its associated sounds, thus complicating the storage and delivery problem [2, 5]. A video server that needs to deliver multiple videos at the above rates requires considerable resources. Therefore, an important objective in designing video management systems is to utilize the limited computing resources and to improve quality of service for as many viewers as possible.

Due to the shear sizes of video objects, digital libraries must be built on mass storage systems consisting of main memory, secondary storage such as magnetic disks, and tertiary devices such as CDs and tapes. In this paper, we address the quality of service issues at the local area network level. Although OC-3 (and higher) fiber optic backbone networks can deliver high bandwidth video data, local area networks are limited to T-1 capacity. It is still not economical to set up OC-3 connection in local area networks. The main memory of the lcoal video server must be refreshed constantly to transfer the video objects from disks as the memory size is small such that no whole video can be contained in it. In addition, the disks must also be reloaded frequently with video objects requested by viewers since only a limited number of video objects can be placed on disks at one time. In other words, video data management requires two levels of buffering, main memory buffering and secondary storage buffering. For simplicity, we refer to the former as buffer management while the latter as disk management.

In recent years, there has been increasing interest in managing of multimedia data on disk and in memory for efficient delivery of video and audio materials. Many research problems have been studied such as the following: principles of retrieval and storage of delay-sensitive multimedia data [6, 5], the collocational storage of multimedia strands [18, 7], memory minimized storage architecture for video-on-demand servers [15, 16], buffer sharing for continuous media servers [3, 13, 14], and file allocation on disk arrays [17, 19, 20, 21].

In this paper, we propose an analytical method for designing buffer and disk management that is extensible to a digital library environment. First, we propose and analyze several buffer management strategies in terms of how buffer space is allocated and shared among multiple viewers of the same video. Secondly, we define several fundamental concepts for video data allocation on secondary storage, develop a mathematical algorithm for optimal placement of multiple video objects, and propose a methodology for dynamic loading of videos on multiple disks.

The remainder of the paper is organized as follows. In Section 2, we present a video storage and delivery architecture. Section 3 proposes three buffer management strategies: single buffer refreshing, static buffer partitioning, and adaptive buffer partitioning. The subsequent section describes a simulation study in which the three strategies are compared. Section 5 presents a new approach on managing multiple video objects in a matrix data placement method, where video lengths and viewing frequencies can vary. In section 6, we develop a mathematical model that optimizes the design of the video data matrix for multiple videos with various lengths and viewing frequencies. In section 7, we apply the mathematical model to delelop a technique that eliminates seek delays by utilizing data striping over multiple asynchronous disks and supports dynamic loading of new videos without interrupting the display of ongoing videos. Finally, Section 8 summarizes the findings in this study and outlines future research directions.

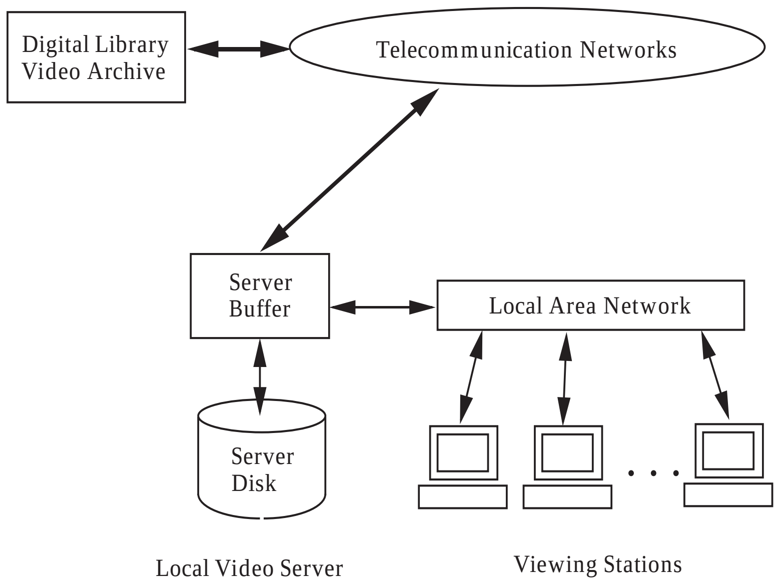


Figure 1: The architecture of a video delivery system

2 System Architecture for Distributed Video Delivery

The digital video delivery system is likely to be comprised of a hierarchy of storage and delivery devices. At the top of the hierarchy is the video archive where all video objects are stored on tapes, and at the bottom of the hierarchy are the viewing stations serving the users. In between, there is a local video server that supply video data to the viewing stations linked via a local area network. When a video requested by viewers is not currently installed on the local server disk, it is retrieved from tapes in the video archive of the digital libary and placed on the server disk(s). The video server then transfers the video data from disk(s) to buffer and then delivers the video from buffer to the viewing stations (Figure 1). The server disks can also store a limited number of videos that are presently in high demand.

In this paper, we concentrate on the lower level of the hierarchy – the local area network level, where a video server receives the necessary materials from a video archive and stores the data on disk. In general, the video server should be capable of delivering many videos at the same time, and the same video can be viewed by many viewers simultaneously. Supporting multi-user access to many video objects through a video server requires sophisticated video delivery techniques both at the disk level and the buffer level. At the disk level, a number of video objects must be stored and transferred to the buffer in ways that satisfy the consumption rates at which video objects are

Notation	Meaning
B	Buffer size measured in bytes
b	the unit size of data blocks transferred between disk and buffer
b_M	the size of a compressed MPEG block in bytes
C	the total number of viewing channels on a server
c_i	the number of viewing channels dedicated to a video
D	the width of a video matrix measured in minutes of display time
d	The display rate in number of MPEG blocks per second
f_i	the relative viewing frequencies of the ith video
l	the length of video measured in minutes
n	The number of users being served
n_c	the number of MPEG blocks refreshed per cycle
R	The transfer rate from disk to memory in bytes per second
r_i	the number of replicates for the ith video
S_{disk}	the capacity of the disk(s)
s	The total of latency and seek time in seconds
W	the width of the enrollment window
w_i	the maximal waiting time for the ith video

Table 1: List of notations used in this paper

displayed at the viewing stations. Similarly, the buffer must be refreshed on time to satisfy the viewing activities.

For convenience, we assume that all videos are stored in a compressed MPEG format although the results of our research are independent of the specific compression standard. In MPEG compression, there are three types of frames: Intrapictures called I-frames, Predicted pictures called P-frames, and Interpolated pictures, also called B-frames, used for bidirectional prediction [2, 12]. The I-frames allow random access points into the video and are only moderately compressed using JPEG techniques. The other types of frames represent only differences between the last I-frame and the current one and offer a large amount of compression. As a result of the connections between the P and B-frames to their respective I-frames, they must be decoded together as a unit. For that reason, there is a minimum number of video frames that must be delivered as a unit from the disk to a decoder for purposes of decompression; we refer to this number of frames as *the MPEG block*.

We define the symbols used in later sections in Table 1 for ease of reference; they will also be discussed at the first point of usage.

3 Buffer Management Strategies

Supporting simultaneous multi-entry and multi-user accesses to the same video object requires sophisticated buffer refreshment policies that are not found in conventional buffer systems. For instance, many video objects may be buffered only partially in memory due to space limitation. As a video object is viewed by users, its buffered slices of data blocks must be refreshed slice by slice so that early users can proceed without delay and late users are being served properly as well. Here, users are ordered by their relative starting points on the video object.

In the optimization of buffer management strategies, we focus on two factors in video delivery: *display quality* and *average waiting time*. The display quality factor requires that the video data is supplied to the viewing stations in such a rate that there is no jerkiness, and our objective is to minimize the average waiting time for all viewers. A very high average waiting time has two consequences in practice: (1) some viewers may have to wait until the next showing period, and (2) a viewer in waiting may decide to withdraw (also called *barking* in queueing theory). In other words, the quality factor acts as the constraint, and the waiting time factor is the objective function to be minimized.

In this section, we present three buffer management strategies aimed at minimizing average waiting time. The first strategy maximizes the number of viewers served by refreshing the buffer in a smart way. The second and the third strategies try to achieve the same objective by partitioning the buffer into several sections. The second strategy partitions the buffer statically, and the third strategy partitions the buffer adaptively as needed.

3.1 Strategy S1: No Partitioning

In this strategy, the whole buffer is used as a single partition to display the video. The objective is to minimize the average waiting time. The following assumptions are made:

- buffered data are compressed video MPEG frames
- the number of MPEG frames that can be decompressed independently are called a MPEG block
- the user station is equipped with the capability of decompression
- buffer refresh is done periodically as a multiple of MPEG blocks

- refresh rate is equal to transfer rate from disk to buffer, and
- all viewers have the same display rate.

The buffer management strategy is developed based on the analysis of the buffer refreshing process. In this process, five unique stages are identified as illustrated in Figure 2.

Stage 1: The buffer is filled with the beginning sequence of frames of the video, and the first user joins the video show and starts viewing the video.

Stage 2: While the first user is viewing the video, other users arrive randomly and join the viewing process. Since the video frames are stored in memory, each enrolled user can view the video independently, assuming that the operating system can handle all of the viewers without delay.

Stage 3: A critical point is reached such that no more viewers should be admitted to view the video. Otherwise, refreshing the buffer will overwrite MPEG frames which some viewers are yet to view. In other words, we say that the *viewer enrollment window* is closed.

Stage 4: At another point in time, the first viewer of the video is about to exhaust the video frames and in need of new frames that have not been retrieved into the buffer. At this point, we say a refresh of the buffer is needed. Note that refreshing for the first user will overwrite some frames that other users might have not finished viewing yet if viewer enrollment is not managed properly. That is, we should stop viewer enrollment some time before the refreshing starts.

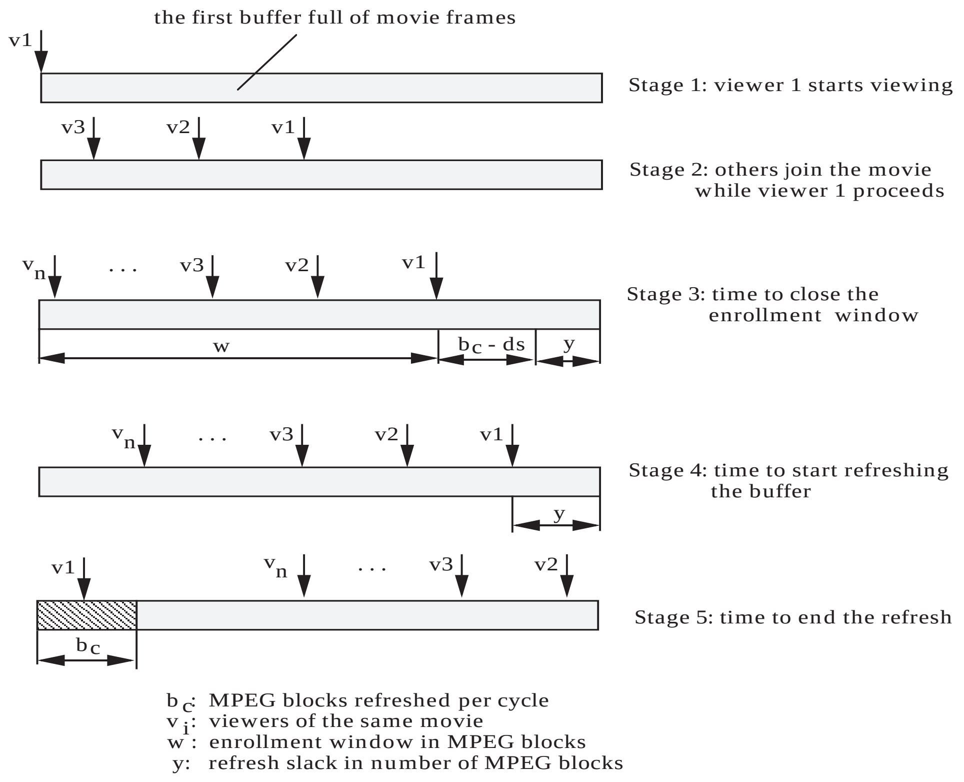
Stage 5: The buffer is refreshed for some number of MPEG blocks, and all viewers keep on viewing the frames in the buffer without hiccups. This process of buffer refreshing will be repeated over and over again until the video is done. The refresh cycle time is derived next.

Our problem at hand is to determine the cycle time, the size of viewer enrollment window, and the minimal number of MPEG blocks refreshed per cycle, as defined next. The objective is to enroll as many viewers as possible before the first buffer refreshment so that average waiting time is minimized.

Cycle time of buffer refreshing (t_c) is the number of MPEG blocks refreshed per cycle (n_c) divided by the display rate (d):

$$t_c = \frac{n_c}{d} \tag{1}$$

Minimal refresh ahead (y) is the size of remaining unconsumed data at the start of the next buffer refreshing cycle, which is equal to the display rate (d) times the sum of the seek (s) and the



the first buffer full of movie frames
v1
Stage 1: viewer 1 starts viewing
v3 v2 v1
Stage 2: others join the movie while viewer 1 proceeds
vn . . . v3 v2 v1
w
bc - ds
y
Stage 3: time to close the enrollment window
vn . . . v3 v2 v1
y
Stage 4: time to start refreshing the buffer
v1 vn . . . v3 v2
bc
Stage 5: time to end the refresh
bc: MPEG blocks refreshed per cycle
vi: viewers of the same movie
w : enrollment window in MPEG blocks
y: refresh slack in number of MPEG blocks

Figure 2: The Buffer Refresh Process

time required to transfer one MPEG block (b_M/R).

$$y = d(s + \frac{b_M}{R}) \tag{2}$$

The enrollment window size (W) in number of MPEG blocks, defines the cutoff point of new viewer enrollment; no new viewers will be admitted after viewer one has displayed all the frames in the enrollment window.

$$W = \frac{B}{b_M} - n_c - y + ds \tag{3}$$

where B is the buffer size in bytes, b_M is bytes per MPEG block, n_c is MPEG blocks refreshed per cycle, y is the refresh ahead in MPEG blocks.

Notice that the term ds is added to the window width because during the seek time, no frames are refreshed in the buffer. Next, we determine the optimal value of n_c, the number of MPEG blocks transferred from the disk to the buffer per cycle. This is done by observing the profit maximizing principle, the display continuity requirement, and the disk transfer constraint.

The profit maximizing principle is that one should serve as many viewers as possible in each viewing session. (Note that viewers may leave if they are tired of waiting and decide to do something else instead.) That is, we want to maximize the size of the enrollment window so that more viewers can be served. From figure 2, one can see that the size of enrollment window is negatively proportional to that of n_c as all other terms are fixed. Therefore, n_c must be minimized in order to maximize the enrollment window.

Note that the data in the buffer are compressed MPEG frames and the continuity requirement states that video frames must be transferred from the buffer to user stations in MPEG blocks so that the B and P-frames can be decompressed in relation to the I-frames in the same block. Consequently, the continuity requirement states

$$n_c \geq 1 \tag{4}$$

The data transfer constraint determines the data transfer cycle, the disk busy time and idle time. Assuming only the current video is in display, the disk idle time T_i is equal to the cycle time minus the seek time and the transfer time

$$\begin{aligned} T_i &= \frac{n_c}{d} - s - \frac{b_M n_c}{R} && (5) \\ &= \frac{R - b_M d}{Rd} n_c - s && (6) \end{aligned}$$

It is worth noting that seek time is incurred in each refresh cycle, and the smaller the value of n_c, the larger the total seek time will be. We know that T_i must be nonnegative and the linear relationship between T_i and n_c says that n_c is minimized if T_i is. Consequently,

$$n_c = \frac{Rds}{R - b_M d} \tag{7}$$

is obtained when T_i is equated to zero and that n_c is minimized.

In light of both equations 4 and 7, the optimal value of n_c is equal to

$$n_c = max(1, \frac{Rds}{R - b_M d})$$

which maximizes the enrollment window, satisfies the disk transfer constraint, and meets the display continuity requirement.

Strategy S1 allows viewers to start viewing the video with a flexible start and end time. This is an important departure from broadcasting services in analog VOD where viewing time is fixed. Under the flexible viewing schedule, the viewing time can be given as a range, say, starting from between 1:00 and 1:20 pm and ending between 2:40 and 3:00 pm.

3.2 Strategy S2: Static Partitioning

In the first strategy, viewers of the video can enroll only if they come before the window is closed. As a result, the expected waiting time for a viewer can be very long. Strategy S1 is effective where viewers' interarrival interval is small. In case viewer arrival is sparse, however, the strategy suffers from low buffer utilization. In an extreme case, suppose that there is only one viewer that has arrived before the viewing window is closed. Subsequent viewers must wait for the first viewer to finish the current viewing period. It is obvious that the buffer is not fully utilized. We now discuss another strategy called the static partitioning strategy in which buffer utilization rate can be improved by partitioning the buffer.

Imagine that there is only a single viewer in Strategy S1, the minimal buffer size required to serve the viewer (B_{min}) is

$$B_{min} = \begin{cases} b_M \left(n_c + d\frac{b_M}{R} \right) & \text{if } n_c = 1 \\ b_M n_c & \text{if } n_c > 1 \end{cases} \tag{8}$$

which is obtained by letting W be zero in Equation 3 and substituting terms for y using Equation 2. When $n_c = 1$, the first term in the parenthesis is the number of MPEG blocks refreshed per cycle, and the second term is the number of MPEG blocks required to serve the viewer while the buffer is refreshed, b_M/R is the time required to refresh one MPEG block, and db_M/R is the number of MPEG blocks displayed during that time. When $n_c > 1$, the second term is removed as the additional buffer space can be saved without affecting the viewing process.

Assuming that the overall buffer size is sufficiently large, we can partition the buffer into several sections. Each section will serve a group of viewers so that average waiting time is reduced. This can reduce the average waiting time greatly. Consider a video of 100 minutes and a buffer of size B that can contain 20 minutes of video. Assuming viewer arrival rate follows a Poisson distribution and Strategy S1 is used, the average waiting time would be about 32 minutes (.2(0)+.8(40)) with maximal waiting time of 80 minutes. Assume that we can partition the buffer into 5 sections of 4 minutes each. Upon arrival, viewers will be put into one of the 5 channels according to their time of arrival. This way, the average waiting time will be around 6.4 minutes (.2(0)+.8(8)) rather than 32 minutes. The maximal waiting time is now 16 minutes rather than 80 minutes.

Figure 3 illustrates the refreshing process of Strategy S2 after partitioning the buffer. However, one limitation of this strategy is that it creates more disk arm movement because now it must seek back and forth to serve the five groups of viewers. In the current strategy, we also assume that the

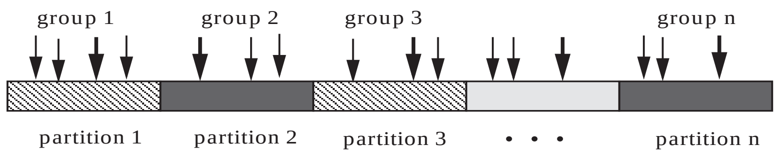


Figure 3: An Illustration of Buffer Partition

viewers come to view the video in an infinite cycle and the interarrival interval between viewers is exponentially distributed (due to the Poisson arrival distribution).

In this strategy, a grouping technique is used to increase the number of viewers served. Let us rank the buffer partitions according to the relative position of the video frames they contain at a point of time from the beginning to the end. For example, if partition i contains the 50th-100th frames, while partition j contains the 200th-250th frames, then we say partition i is closer to the beginning of the video than partition j. The buffer partition containing the beginning frames of the video is called the first partion and the one containing the ending frames, the last partition. When all frames have been viewed in a partition, it will change from being the last partition to become the first partition as its contents change dynamically.

When a viewer arrives after the viewing window has closed for the first partition, the viewer will be queued to wait for the last partition to restart the video. The thick arrow in figure 3 illustrates the grouped viewers.

Similar to Equation 8, the minimal buffer size per group is

$$B_{min} = \begin{cases} b_M \; (\; n_c + d\frac{b_M}{R} \;) & \text{if } n_c \; = \text{n} \\ b_M n_c & \text{if } n_c \; > \; n \end{cases} \tag{9}$$

except for

$$n_c = max(n, \frac{nRds}{R - nb_M d}) \tag{10}$$

which is similar to equation 7 except that the disk now must serve all n number of partitions. Therefore, n_c is a function of n.

The maximal number of partitions for a given buffer size B is

$$\begin{aligned} n_{max} &= \frac{B}{B_{min}} \\ &= \begin{cases} B/[b_M(1 + d\frac{b_M}{R})] & \text{if } n_c \; = \; n \\ B/(b_M n_c) & \text{if } n_c \; > \; n \end{cases} \end{aligned}$$

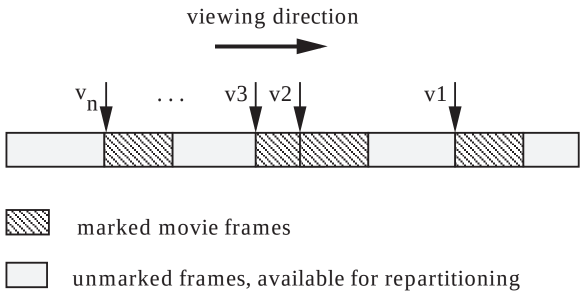


Figure 4: Available Spaces for Adaptive Partitioning

We know that when

$$\frac{nRds}{R - nb_M d} \leq n$$

the optimal value of n_c is equal to n and n_{max} can be easily solved. However, when

$$\frac{nRds}{R - nb_M d} > n$$

$$n_c = \frac{nRds}{R - nb_M d}$$

and it is required to solve the following quadratic equation:

$$n = B/(\frac{nRb_M ds}{R - nb_M d})$$

The solution is:

$$n_{max} = \frac{-Bb_M d + \sqrt{Bb_M d(Bb_M d + 4R^2 s)}}{2b_M dRs}$$

Figure 5 illustrates the variations of n_{max} and n_c as the transfer rate changes. The values used in the computation for B, b_M, d, s are 250 Mbytes, 60 Kbytes, 4 MPEG blocks, and 0.015 sec, respectively.

3.3 Strategy S3: Adaptive Partitioning

We now discuss another strategy called the adaptive refreshing strategy in which buffer utilization rate can be improved by partitioning the buffer adaptively.

The main idea of the strategy is to allow repartitioning of the buffer when a new viewer arrives after all viewing windows are closed. The basic idea of adaptive partitioning is that some buffer

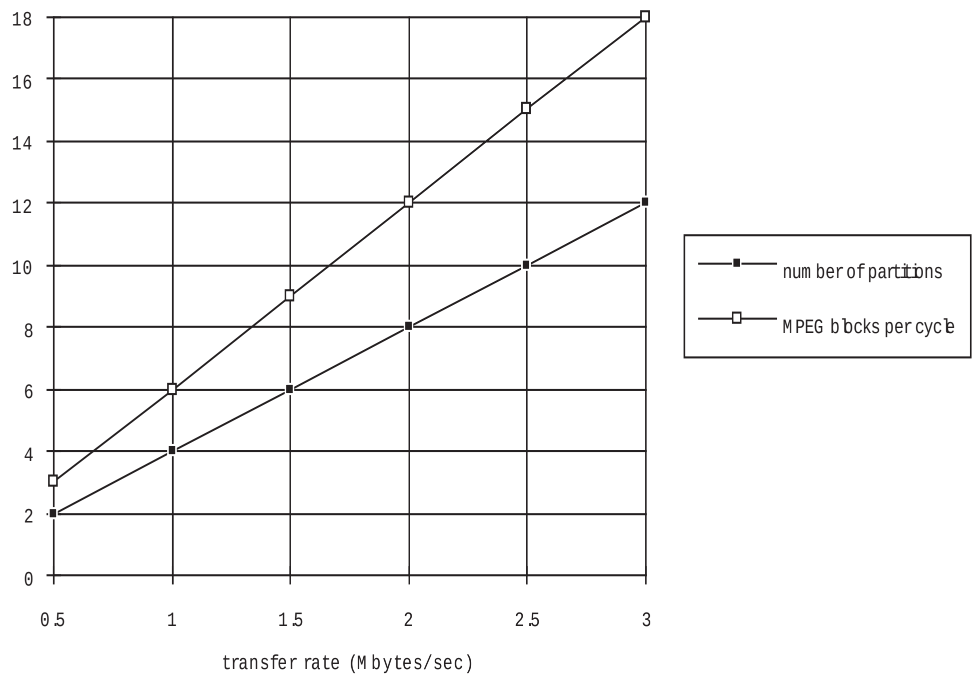


Figure 5: The values of n_{max} and n_c

space can be reclaimed for the new viewer without hindering the viewing process of existing viewers of a partition when the viewer arrival is sparse. As shown in Figure 4, a viewer can be served adequately as long as it has the minimal number of MPEG blocks (which is equal to B_{min} as derived previously) in reserve. We call this minimal number of MPEG blocks as the marked ones. The unmarked blocks will be available for repartitioning provided the number of partitions does not exceed the maximal number of partitions n_{max} derived earlier.

The adaptive partitioning algorithms is as follows:

1. Mark the minimal number of MPEG blocks that are required to ensure a continuous display for each viewer in session.

2. Merge those marked blocks that are interconnected.

3. Select one of the unmarked buffer spaces that are larger than the minimal buffer size to create a new partition.

4. The new viewer is placed in the new partition.

This buffer repartitioning process can be repeated for the next new viewer until no more buffer

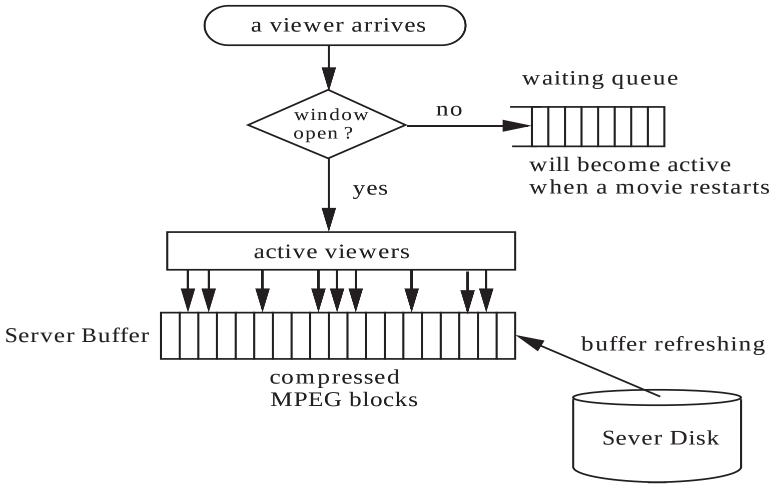


Figure 6: The Single Buffer Refresh Model

space is available for repartitioning. Repartition ceases when the total space of unmarked buffer pages is less than the minimal buffer size for a viewer as in step 3 above.

This strategy is effective when the buffer is large relative to the minimal buffer requirement per viewer and the viewers arrive in a sparse pattern so that repartitioning is possible. However, the strategy is ineffective when repartitioning is seldomly possible due to a high viewer arrival rate.

4 A Simulation Study of Buffering Strategies

To compare the three different buffer management strategies, namely, no partition (S1), static partitioning (S2), and adaptive partitioning (S3), a simulation study was conducted. The simulation program was written using an object-oriented discrete event simulation language called MODSIM. We present the simulation models and results in this section.

As illustrated in Figure 6, viewers arrive in an random process (Poisson process is used in the simulation). Upon arrival, a viewer joins other active viewers for the video if the viewing window is open. Otherwise, the viewer is placed into the waiting queue. To simplify the simulation, we assume that viewers will not "bark" from the waiting queue. As said earlier, the video data in the buffer are compressed MPEG blocks, therefore, their sizes may vary.

Figure 7 illustrates the situation where the buffer is partitioned into several sections. A newly arrived viewer joins other active viewers for the video if there is a section where the viewing window is open. Otherwise, the viewer is placed into the waiting queue. In case of dynamic partitioning,

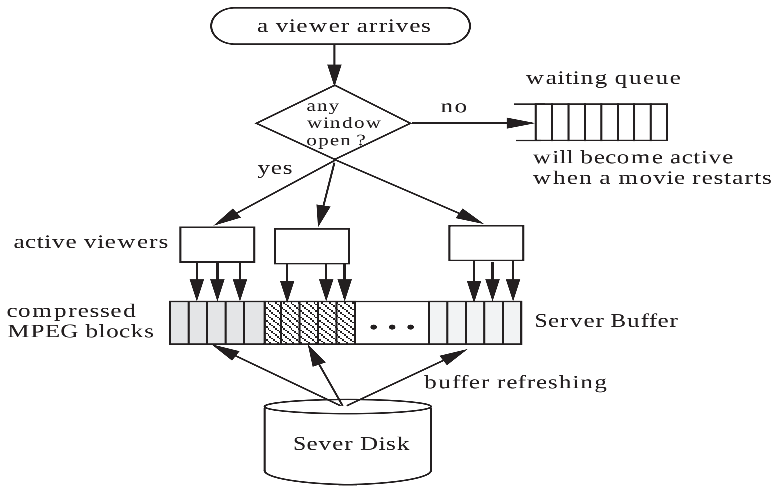


Figure 7: The Multiple Buffer Refresh Model

Notation	Value	Meaning
B	250 (Mbytes)	buffer size in Megabytes
b_{max}	60 (Kbytes)	maximal size of a compressed MPEG block
b_{min}	40 (Kbytes)	minimal size of a compressed MPEG block
d	4 (1/sec)	viewer display rate in number of MPEG blocks per second
R	1 (Mbytes/sec)	transfer rate from disk to buffer
l	100 (min)	video size in number of viewing minutes
s	0.015 (sec)	the total of latency and seek time in seconds
x	600 (sec)	viewer interarrival interval

Table 2: Default parameter values used in the simulation

the new viewer may cause a repartition of a buffer section if there is no open window and the total number of partitions is less than the maximal number of partitions. Static partitioning is done before any viewer arrives. The number of partitions is set as the maximal number of partitions. However, a partition will remain inactive until a viewer arrives while all active partitions are closed.

Table 2 contains the default values for the parameters used in the simulation. The size of a MPEG block b_M is assumed to be uniformly distributed between b_{min} and b_{max}, and viewers arrive in a Poisson process characterized by the interarrival interval x. The simulation was run for 200 minutes, twice the length of a video.

Figures 8 through 10 show the results of simulation by plotting the average waiting time against buffer size, viewer interarrival interval, and disk transfer rate respectively. In Figure 8, the size

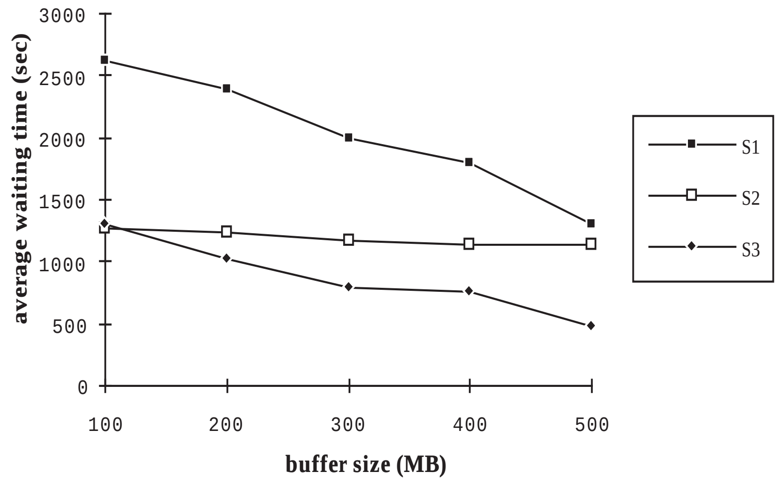


Figure 8: The Effect of Buffer Size on Average Waiting Time

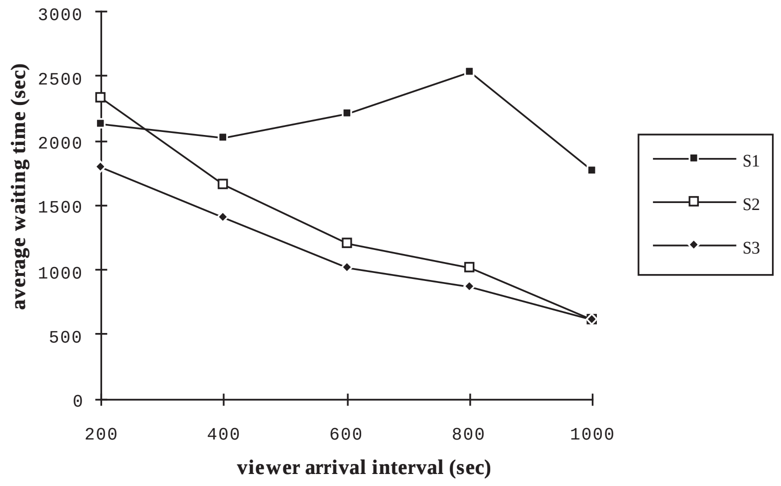


Figure 9: The Effect of Interarrival Interval on Average Waiting Time

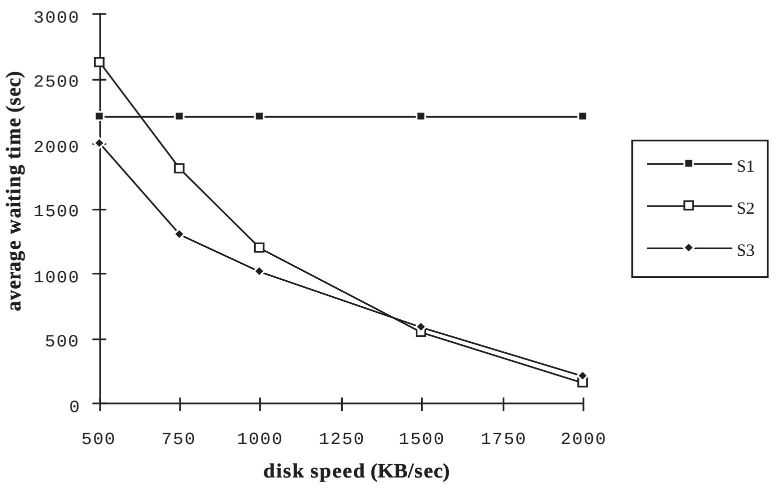


Figure 10: The Effect of Transfer Rate on Average Waiting Time

of the buffer runs from 100 through 500 megabytes. The results demonstrate that the partitioning strategies are always better than no partitioning. The adaptive partitioning showed roughly 50% improvement for the average waiting time. The static partitioning is better than adaptive partitioning at the lower end of buffer size, while the adaptive partitioning becomes better at the higher end of buffer size. This is because a smaller buffer size reduces the opportunity for dynamic partitioning, and therefore, the static partitioning wins.

Figure 9 gives the simulation results where the viewer arrival interval is the variable. It is shown that the average waiting time in Strategy S1 is not proportional to the interarrival rate because the arrival sequence is random. However, the strategies with partitioning show a linear effect. That is, the fewer the viewers, the shorter the average waiting time. This is an interesting result, which means that partitioning increases our control over the average waiting time.

Figure 10 displays the effect of disk transfer rate on the average waiting time. Results indicate that Strategy S1 is insensitive to the disk speed as long as it is faster than the display rate. On the other hand, the faster the disk, the better the partitioning strategies perform. This is because faster disk allows more partitions, and thereby more viewers can join the viewing process without waiting, further reducing the average waiting time. The results also show that static partitioning is slightly better than the adaptive strategy for a faster disk.

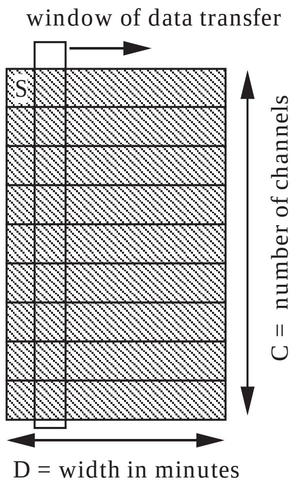


Figure 11: The single video matrix

5 Disk Management Techniques

The previous sections have focused on buffer management techniques while assuming a certain value of disk transfer rate that can feed data from a single video to multiple buffer channels. However, in this section, we develop techniques that can be used to serve multiple videos on one server by feeding multiple videos simultaneously from disk(s) to main memory. The main memory can then be partitioned into regions for different videos; the buffer management techniques developed in the previous sections can be applied to each such region. We first review the single video technique by Ozden et al [16] in the next subsection and then extend it to multiple video objects with heterogeneous characteristics.

5.1 The Single Video Matrix

We observe that the disk transfer rate R is typically orders of magnitude faster than the rate, d, at which the video is displayed on the user workstation. For that reason, the disk transfer rate can support multiple channels (up to $\frac{R}{d}$) simultaneously. The situation is best described by a matrix structure as in [16]. We refer to this matrix as the single video matrix as illustrated in Figure 11 and Table 3.

This matrix has C rows (each called a channel), and n columns, and each cell in the matrix is a block which in our context is the unit of video data transferred from the buffer to the viewing stations at each time. The block size b, is a size that guarantees smooth display without hiccups (the

b(1)	b(2)	b(3)	...	b(n)
b(n+1)	b(n+2)	b(n+3)	...	b(2n)
b(2n+1)	b(2n+2)	b(2n+3)	...	b(3n)
...	...	...	...	...
b((C-1)n+1)	b((C-1)n+2)	b((C-1)n+3)	...	b(C×n)

Table 3: Video blocks in the single video matrix

size depends on the MPEG compression rate [11]). Consequently, the size of the video contained in the matrix is equal to $b \times C \times n$ bytes.

During the time it takes a workstation to display (and the viewer to watch) one cell of the matrix, the disk can transfer a whole column (consisting of C cells) to the buffer. Figure 11 illustrates this process; the letter ”S” marks the first block of the video.

The symbol $b(i)$ in Table 3 denotes the ith block of the video. Let us assume that the first group of viewers (also called a channel) start watching block $b(1)$, each time they view a new block a complete column is transferred so that by the time this group has watched block $b(n)$ the whole video has been transferred to the buffer and a new showing of the video starts. The first group is then moved to channel 2 where $b(n+1)$ is currently displayed and a new group can enter and watch the video from the beginning on channel 1. In this way we can have (after some initial period) up to C different viewer groups watching the same video where in general two consecutive groups are watching portions of the video that are n blocks apart from each other.

Example 5.1 *For instance, a video of 100 minutes long contains about 1.125 Gigabytes data. With a disk transfer rate of 10 Megabytes/second and display rate 1.5 Megabits/second, C=53 (=10*8/1.5) channels can be supported. Therefore, the matrix contains 53 rows. Let the block size, b, be 50 Kilobits as chosen in [16], the number of columns, n, in the single video matrix is 3396, computed as* $1.125 * 8 * 10^9/(53 * 5 * 10^4)$. *The total number of blocks for this video is the number of channels times the number of columns, amounting to 180,000 (=53*3396). Furthermore, supporting the 53 channels requires 5.3 Megabits of buffer space (=.05*2*53). The factor 2 indicates that two blocks of buffer space are needed for each channel so that when one block is refreshed, the other block can be displayed by viewers of that channel.*

Once a video object is organized into the matrix, its data blocks are then placed on disk

b(1)	b(n+1)	...	b((c-1)n+1)	b(2)	b(n+2)	...	b((c-1)n+2)	...	b(c×n)

Table 4: The sequence of video blocks on disk

continuously in a *column first order* as illustrated in Table 4. In this way, the video blocks can be retrieved into the buffer column by column without incurring random seek times [16]. Note that there is one random seek after transferring the last column moving back to the first one. We will ignore this as it can be dealt with using some small additional buffer space as shown in [16].

In this scheme, the maximum waiting time a new user will experience is equal to the time required to display one row in the matrix. This follows since the matrix is transferred into buffer in every cycle and a viewer can start viewing whenever block $b(1,1))$ is in the buffer. In [16], only single videos are considered and assumed to be displayed continuously. The authors suggest that the same matrix can be duplicated for each video if multiple videos are to be supported. This of course will require an additional disk bandwidth of R for each such matrix.

One drawback of the single video matrix approach is that the special data structure essential for eliminating seek time requires excessive time when loading the video on disk. Loading basically requires permuting a huge matrix from row order to column order representation (see the next example). While this approach is suitable for static situations, it is problematic when viewers request videos that are not loaded on disk.

Example 5.2 *As computed in Example 5.1, the total number of blocks in the single video matrix for the 100 minutes long video is 180,000. Assuming the random seek time is 15 ms, the total seek time needed to load this video is therefore 2700 seconds (=180,000×0.015) or 45 minutes (assuming one seek per block). The total loading time for the video amounts to 46.875 minutes, including 1.875 minutes (1.125/10/60) transfer time from tape to disk and the 45 minutes total seek time. For convenience, we refer to this problem as the* **initial loading delay** *problem.*

To summarize, the single video matrix, although having its merits for VOD, cannot be directly applied to video storage and delivery in digital libraries because:

1. Although highly demanded videos can be displayed in continuous cycles, there are also many videos that are merely used on "one time only" basis, making invalid the assumption that all videos should be displayed continuously.

2. In digital libraries, dynamic loading of videos is absolutely necessary as there is no way of predicting perfectly the times of requests for specific videos. High initial loading delays will make video loading a bottleneck in a digital library.

5.2 Multiple Videos of the Same Length

To illustrate how to support multiple videos concurrently, let us first examine the case of videos with equal length. Multiple videos can be organized into a matrix using a variation to the single video matrix. We first illustrate the multiple video matrix with Examples 5.3 through 6.1; a general optimization model for the design of multiple video matrices is given next.

Example 5.3 *Suppose there are 3 videos that have equal length, assuming all are viewed equally frequently and the video server can support 9 channels simultaneously. These videos can then be placed into a matrix of 9 rows where each video takes 3 rows. Figure 12(a) shows the multiple video matrix, where the symbol "S" marks the first block of each video. Now, the 3 videos can be displayed simultaneously using a single server. However, the matrix width, D, and consequently the maximal waiting time, are three times as long as those when allocating all 9 channels to only one of the videos.*

In Example 5.3, we also assumed implicitly that all three videos have the same level of demand (measured by viewing frequency, for instance) so that the channels are equally distributed. The result is that all videos have the same maximal waiting time. In case that some videos are in higher demand, we should then consider allocating more channels to these videos to reduce the waiting time.

Example 5.4 *Let us now place 2 videos in the matrix, but video 1 is assumed to be twice as popular as video 2. In this case, we can place each of the two videos in 3 channels and use the remaining 3 channels to replicate the first video that is in higher demand. In order to reduce the waiting time, the first block of the second copy of video 1 should be placed in a column near the middle of the matrix so that the second copy restarts when the middle column of the matrix is transferred (about $D/2$ minutes from the start of the first copy). This situation is illustrated in Figure 12(b). In effect, the second copy cuts the waiting time by half. This technique is referred to as* **staggered replication**.

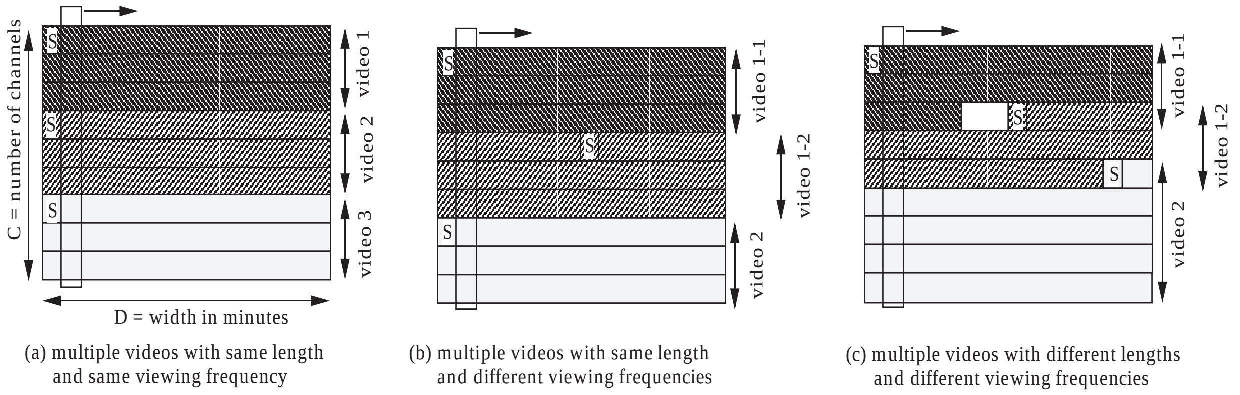

C = number of channels
D = width in minutes
S
video 1
video 2
video 3
(a) multiple videos with same length and same viewing frequency
video 1-1
video 1-2
video 2
(b) multiple videos with same length and different viewing frequencies
video 1-1
video 1-2
video 2
(c) multiple videos with different lengths and different viewing frequencies

Figure 12: Multiple video matrices

5.3 Multiple Videos of Different Lengths

In reality, few videos are of the same length, and therefore video data organization is more complex as illustrated in the next example.

Example 5.5 *We consider now a similar scenario as in Example 5.4 except that video 2 is longer than video 1. Suppose the best matrix design turns out to be what is shown in Figure 12(c). Here row 3 (from the top) is shared by the two copies of video 1, and row 5 is shared by the second copy of video 1 and video 2. We refer to this case as* **channel sharing**. *The figure also illustrates that some space (the white space in the figure) might be left unused at the end of a video in order to start the second copy of video 1 near the middle column of the matrix. This unused space will have only a minor effect on disk utilization, as the number of videos on the same disk is generally small. Furthermore, the open space may be filled with very short videos. Channel sharing is an important technique as it says that the number of channels allocated to each video does not have to be an integer. This will simplify considerably the optimization model developed next.*

6 A Mathematical Model for Minimizing Average Waiting Time

The video data organization problem can be formulated as a mathematical optimization problem that minimizes the average maximal waiting time by allocating the limited number of channels to the videos to be displayed. This requires to determine the best value for the width of the matrix measured in minutes (D) and number of copies and channel allocation for each video.

This problem can be formulated as follows:

Objective function (P1):

$$min(\sum_{i=1}^{m} f_i w_i) \tag{11}$$

Subject to (for i = 1 to m):

$$w_i = \frac{D}{r_i} \tag{12}$$

$$r_i \in \{1, 2, 3, ...\} \tag{13}$$

$$\sum_{i=1}^{m} c_i \leq C \tag{14}$$

$$c_i = \frac{l_i \times r_i}{D} \tag{15}$$

$$\sum_{i=1}^{m} l_i \times r_i \leq S_{disk} \tag{16}$$

where m is the number of videos, C is the total number of channels, D is the width of the matrix, S_{disk} is the size of the disk(s), c_i is the number of channels allocated to the ith video, f_i is the viewing frequency for the ith video. l_i is the size of the ith video, r_i is the number of replications for the ith video, and w_i is the maximal waiting time for the ith video.

This formulation can be simplified by merging Equation 12 into the objective function P1, and by combining Equations 14, and 15 into Equation 18. The constraints defined by Equation 12 will be used to determine the maximal waiting times and the width of the matrix. All these substitutions result in:

Objective Function (P2):

$$min(\sum_{i=1}^{m} D\frac{f_i}{r_i}) \tag{17}$$

Subject to (for i = 1 to m):

$$[\sum_{i=1}^{m}(l_i \times r_i)] \leq C \times D \tag{18}$$

$$[\sum_{i=1}^{m}(l_i \times r_i)] \leq S_{disk} \tag{19}$$

$$r_i \in \{1, 2, 3, ...\} \tag{20}$$

Example 6.1 *Consider three videos with lengths {105, 88, 119} in displaying minutes, and relative viewing frequencies {10.75, 2.75, 1.0}, respectively. Let the number of channels be 53 and the disk size be 10 Gigabytes. Determine the width of the matrix and the video loading patterns that minimizes the weighted total maximal waiting time. The disk size in bytes is equivalent to a transfer time of 888 minutes by assuming* $1.125 * 10^7$ *Bytes per minute of video.*

The mathematical model for this specific situation is therefore:

Objective Function (P3):

$$Z = min\{D(\frac{10.75}{r_1} + \frac{2.75}{r_2} + \frac{1}{r_3})\}$$

Subject to:

$$\begin{aligned}(105r_1 + 88r_2 + 119r_3) &\leq 53D \\ (105r_1 + 88r_2 + 119r_3) &\leq 888\end{aligned}$$

The optimal solution for this mathematical program is:

$$\begin{aligned}\text{Z} &= 68.58 \\ r_1 &= 3 \; copies \\ r_2 &= 2 \; copies \\ r_3 &= 1 \; copy \\ w_1 &= 3.84 \; minutes \\ w_2 &= 5.75 \; minutes \\ w_3 &= 11.51 \; minutes \\ c_1 &= 27.37 \; channels \\ c_2 &= 15.29 \; channels \\ c_3 &= 10.34 \; channels \\ D &= 11.51 \; minutes\end{aligned}$$

It is also interesting to note that the disk space used is equal to $(105r_1{+}88r_2{+}119r_3){*}1.125/100 =$ *6.86 Gigabytes. That is, even though the disk size is 10 Gigabytes, the optimization result indicates that the extra disk space does not help minimize the value of the objective function. Notice also that the optimization results in 3 copies of video 1, 2 copies of video 2, and a single copy of video 3. This result shows that simply allocating the available channels without replication of videos does not necessarily minimize the average waiting time.*

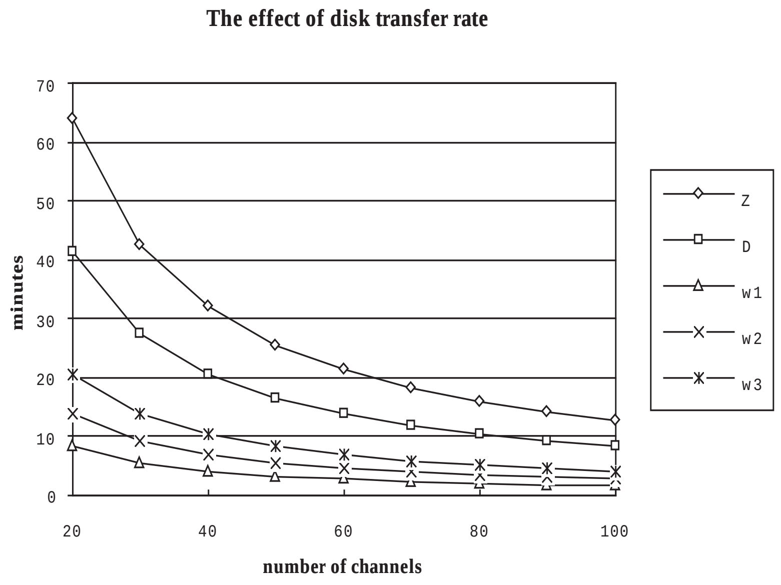


Figure 13: The Optimization Results

To extend the optimization results to more general cases, we analyze the effect of disk transfer rate on the average waiting time, matrix width and maximal waiting time for each video. The results are given in Figure 13. The figure uses the number of channels as an indicator for disk transfer rate since these two factors are linearly correlated ($C = R/d$). The results illustrate that a faster disk leads to shorter waiting times, smaller value of the optimization function (Z), as well as the matrix width (D).

7 Managing Video Objects Using Multiple Asynchronous Disks

In [16], video data are loaded onto the disk in column first order so that video blocks of multiple channels can be retrieved into the main memory column by column without incurring random seek time. However, this causes the initial loading delay problem discussed in Example 5.2. Next, we propose an alternative approach that is free from the initial loading delay problem.

Consider the case of two disks. Even though each disk may incur seek time when retrieving a video block, the CPU does not have to wait for the disk if any two consecutive video blocks in the same column are located on different disks. Using this method, the seek time does not cause any discontinuity of transferring the video; while the server transfers a block from the first disk, the second disk can seek ahead for the next block in the same column. Note that this requires that the

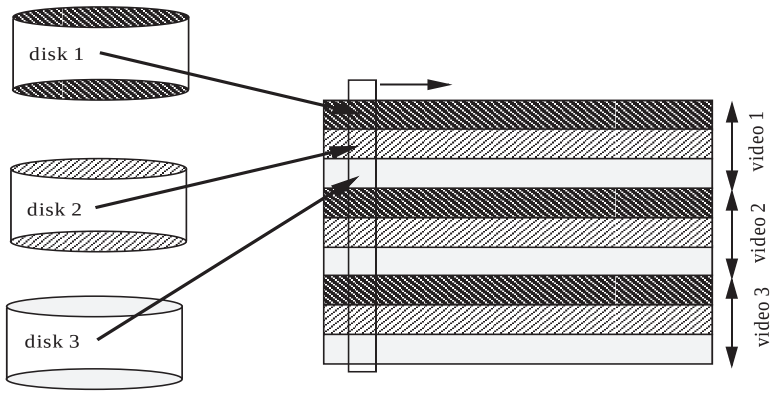


Figure 14: The ROSE approach using multiple asynchronous disks

seek time be shorter than the time of transferring one block. We call this approach as the *read one seek else* approach, or simply *the ROSE approach*.

Figure 14 illustrates the case of three disks; the consecutive rows (shown in different shades) in the matrix are loaded on disks one through three alternately. That is, with the ROSE approach, the video matrices can be loaded on disks row by row rather than column by column. This way, video data can be loaded much faster as no seek times are incurred for the blocks in the same row of video data. A simple computation shows that a 100 minute video can be loaded in 1.875 minutes (=1.125/10/60) as opposed to 46.875 minutes as illustrated in Example 5.2. Furthermore, the results presented in previous sections can be combined with the ROSE approach.

One limitation of the ROSE approach is that the size of transfer blocks is now constrained by the number of disks as the time to retrieve one block must be longer than the seek time if there are only two disks. The consequence is that a larger RAM may be needed. In the case of two disks with expected seek time of 15 ms and transfer rate of 10 Megabytes, the block size must be larger than or equal to 150 Kilobytes (=$10^4 * 15 * 10^{-2}$). To support 53 channels as in Example 5.1 would require 15.9 Megabytes of buffer (=.15*2*53), where the factor of 2 indicates that two blocks are the minimal buffer size for each channel. The size of the buffer can be reduced by using more disks.

Note that the size of transfer blocks is limited by the seek time divided by the number of disks N_{disk} minus one. This is because when one disk reads, the rest of the $N_{disk} - 1$ disks can seek ahead. In the case of four disks, the block size is reduced to 50 Kilobytes [=150/(4-1)], and the minimal buffer requirement is reduced to 5.3 Megabytes (=.05*2*53). In summary, the ROSE approach loads the matrix in row order rather than in column order, and therefore, the initial loading delay

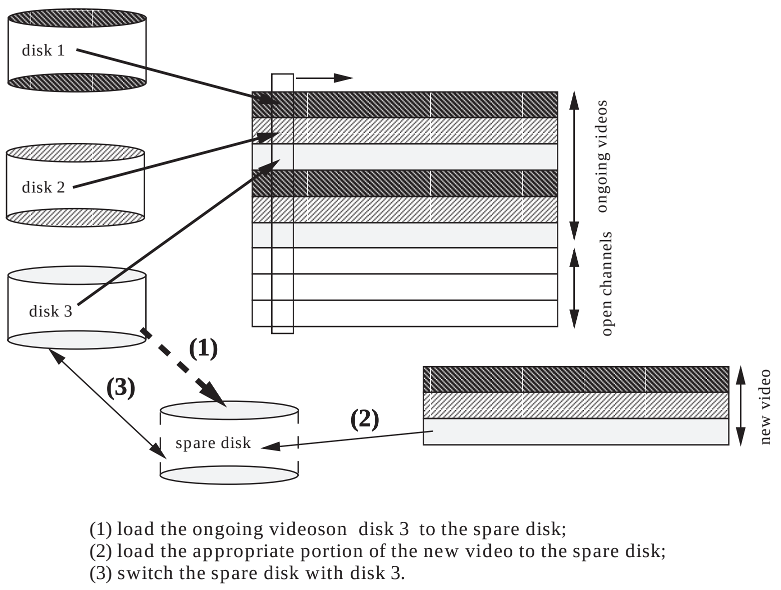

disk 1
disk 2
disk 3
ongoing videos
open channels
(1)
(3)
(2)
spare disk
new video
(1) load the ongoing videoson disk 3 to the spare disk;
(2) load the appropriate portion of the new video to the spare disk;
(3) switch the spare disk with disk 3.

Figure 15: An illustration of dynamic loading of a new video

problem is eliminated.

Even if viewers' video preferences are highly predictable, occasionally videos that are not currently on disk will be requested by users. *Dynamic loading* of videos, i.e., the capability of loading new videos and retiring existing ones without interrupting the display of ongoing videos, is therefore a necessary feature. In digital libraries, requests leading to *dynamic loading* are likely to be more frequent than in VOD services, and therefore, it is a critical requirement.

We propose a dynamic loading approach using a spare disk as shown in Figure 15. The new video is segmented into three portions, each of which will be loaded onto a separate disk (identified by the difference in shade). The general procedure is explained below.

1. *identify the open channels:* In Figure 15, the last three channels (open channels) are currently not used by any video.

2. *transfer data to the spare disk:* The useful video data currently on disk 3 are reloaded from tape to the spare disk.

3. *load the new video:* We also load the portion of the new video that has to reside on disk 3 to the spare disk.

4. *switch with the spare disk:* The spare disk is switched with disk 3. We then repeat steps 2

through 4 with respect to disks 1 and 2.

5. *start displaying the new video:* When all disks are loaded with the appropriate portions of the new video, the new video can then be displayed.

This loading scheme can support fast replacement of videos on disk without interfering the ongoing video displayment.

8 Conclusions

Efficient management of video data is a fundamental requirement in future digital libraries. The sheer volumes of the video objects (in the order of Gigabytes) and the continuous display requirements impose very high demand on the computational resources. The delay sensitivity of video objects require special data storage and delivery techniques to guarantee smooth playback of video data while making full use of the limited computing resources. In this paper, we investigated the problem of managing multiple video objects in digital libraries and proposed unique data management techniques in memory and on disk to enable dynamic loading and simultaneous delivery of multiple video objects on a single video server.

In memory management, we proposed three buffer management strategies to minimize the average waiting time while ensuring display without jerkiness. A simulation study was conducted, and the performance evaluation concludes that partitioning helps to reduce the average waiting time in most situations. The savings in waiting time can be more than 50%; however, when the arrival rate is high, static partitioning may not help. This is because the optimization algorithm does not take into account the viewer arrival rate. Nevertheless, the current algorithm has shown that partitioning is beneficial when the interarrival interval is long (i.e., arrival rate is low). When the viewer arrival rate is high, adaptive partitioning may not be possible because unmarked buffer spaces may not be found.

In this paper, we also presented new techniques for storing and delivering multiple videos. A mathematical model for optimal data placement of multiple videos on disk was presented, taking into account the disk transfer rate, viewing frequencies, and lengths of individual videos. The solution of the model gives the width of the matrix, the number of copies and the resulting waiting time for each video. The model can also be used to determine system resources such as types of disk drives and RAM needed in order to meet requested waiting times. Next, we showed that by using multiple asynchronous disks, the initial loading delay can be avoided by alternating successive rows from the matrix on different disks. We developed a novel dynamic loading procedure that enables loading new videos without interrupting the services of ongoing videos.

Future work in this area involves determining the initial selection of videos among the candidates based on certain economic criteria. Another open question is how to group candidate videos into different matrices each supported by an independent server. The problem of video data placement in mass storage system will also be investigated where data must be properly segmented and placed on a parallel disk system so that interference between videos can be minimized.

References

[1] M. Anderson. VCR quality video at 1.5 Mbits/s. *National Communication Forum*, Chicago, 1990.

[2] ISO/MPEG Committee. Coding of moving pictures and associated audio. *Committee Draft of Standard ISO11172: ISO/MPEG 90/176*, December, 1990.

[3] A. Dan et al. Buffering and caching in large-scale video servers. In *Compcon - Technologies for the Information Superhighway, Digest of Papers*, pages 217–224, Jan. 1995.

[4] C. Federighi and L. A. Rowe. A distributed hierarchical storage manager for a video-on-demand system. *EECS, Univerisity of California, Berkeley, CA 94720*, 1993.

[5] J. Gemmell and S. Christodoulakis. Principles of delay-sensitive multimedia data storage and retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, 1992.

[6] J. Gemmell et al. Multimedia storage servers: a tutorial. *IEEE Computer*, pages 41–49, May 1995.

[7] S. Ghandeharizadeh, S. H. Kim, and C. Shahabi. On configuring a single disk continuous media server. In *Proceedings of the ACM SIGMETRICS/PERFORMANCE*, May 1995.

[8] A.D. Gleman, S. Halfin, and W. Willinger. On buffer requirements for storage-and-forward video on demand service circuits. In *Proceedings of the IEEE GLOBECOM'91*, pages 976–80, Vol. 2 1991.

[9] W. Hodge, S. Mabon, and J. T. Powers, Jr. Video on demand: architecture, systems, and applications. *SMPTE Journal*, September:791–803, 1993.

[10] B. Jabbari et al. Statistical characterization and block-based modeling of motion-adaptive coded video. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(3), June:199–207, 1993.

[11] D. Le Gall. MPEG: a video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991.

[12] M.L. Liou. Overview of the px64 kbps video coding standard. *Communications of the ACM*, 34(4), 1991.

[13] D. J. Makaroff and R. T. Ng. Buffer sharing schemes for continuous-media systems. *Information Systems*, 20(6):445–464, 1995.

[14] F. Moser, A. Kraiss, and M. Klas. L/MRP: A buffer management strategy for interactive continuous data flows in a multimedia dbms. In *Proc. of the 21st International Conference on Very Large Data Bases*, Sep. 1995.

[15] B. Ozden et al. A disk-based storage architecture for movie on demand servers. *Information Systems*, 20(6):465–482, 1995.

[16] Banu Ozden et al. A low-cost storage server for movie on demand databases. In *Proceedings of the 20th VLDB Conference*, pages 594–605, Santiago, Chile 1994.

[17] Banu Ozden, R. Rastogi, and A. Silberschatz. Disk striping in video server environments. In *IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[18] P.V. Rangan and H.M. Vin. Efficient storage techniques for digital continuous multimedia. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):564–573, August, 1993.

[19] H. M. Vin, S Rao, and P. Goyal. Optimizing the placement of multimedia objects on disk arrays. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 158–165, May 1995.

[20] H. M. Vin, P Shenoy, and S. Rao. Analyzing the performance of asynchronous disk arrays for multimedia retrieval. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 14–17, August 1994.

[21] Yuewei Wang et al. Video file allocation over disk arrays for video-on-demand. In *Proceedings of The third IEEE International Conference on Multimedia Computing and Systems*, June 1996.