TRADING MEMORY FOR DISK BANDWIDTH IN VIDEO-ON-DEMAND SERVERS *

Weifeng Shi and Shahram Ghandeharizadeh
Computer Science Department
University of Southern California
Los Angeles, California 90089
{wfshi,shahram}@cs.usc.edu

KEYWORDS

Video-on-demand; data sharing; cost-effectiveness; system configuration.

ABSTRACT

In a Video-on-Demand server, requests from different clients are independent of each other and may arrive at random time. Commercial systems may contain hundreds to thousands of clients and thus providing an individual stream for each client may require very high disk bandwidth in the server. Therefore the disk bandwidth may become a bottleneck resource, restricting the number of concurrent displays in the system. In this paper, we propose a scheme that trades memory for disk bandwidth and strikes a balance in order to prevent either memory or disk bandwidth from becoming a bottleneck resource. Moreover, we obtain the balance point of trading memory for disk bandwidth which leads to the most cost-effective system configuration.

1 INTRODUCTION

During the past few years, advances in computing and communication technologies have made it feasible to implement a Video-on-Demand(VOD) server which can provide continuous delivery of thousands of video streams to different clients. Video is quite different from conventional data types like text and image. First, video objects are typically large in size and have high bit rate. For example, a two hour MPEG-2 encoded video may require 3.6 Gigabytes (GB) of storage and 4 Megabits per second (Mbps) of bandwidth for its display. Second, video data need to be retrieved and delivered to client at a pre-specified rate, otherwise the display may observe disruptions and delays.

*This research was supported in part by the National Science Foundation under grants IRI-9258362 (NYI award) and ERC grant EEC-9529152, and a Hewlett-Packard unrestricted cash/equipment gift.

We consider a VOD architecture depicted in Figure 1. The

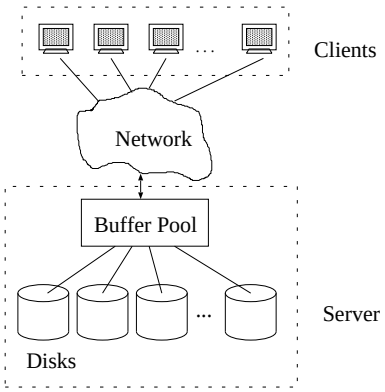


Figure 1: A typical architecture of a VOD system

server includes a number of disks and a buffer pool. We assume there is no network constraints. Each video is striped into equi-sized data blocks. The size of each block is denoted as B and the data blocks are assigned to the disks in a round-robin manner to balance the load imposed by a display evenly across the available disks [6, 9, 15]. The buffer pool is also divided into frames of size B. We assume an environment where all video objects have the same constant bandwidth requirement (this assumption will be relaxed as part of our future research directions). To guarantee a continuous display, a certain amount of disk bandwidth (referred to as a *disk stream*) must be reserved. Each disk stream may serve one or (as we shall see later) more displays. We adopt the cycle-based approach [2, 10, 13, 15] that retrieves data blocks into memory at regular time intervals. Each active disk stream stages one data block into the buffer pool during one cycle and transmits the block to the client by the end of the cycle. The display of this block starts from the beginning of next cycle and lasts for one cycle. This paradigm ensures a continuous display of each video.

The playback requests for videos from different clients are independent of each other and may arrive at random time, and therefore a new video display could be started to satisfy each request. In a commercial VOD system which may contain thousands of active clients [1], this individual service may lead to very high disk bandwidth requirement in the server. Thus the disk bandwidth, as opposed to disk storage capacity, is the more scarce resource, restricting the number of concurrent displays in the system. There have been several studies on buffer sharing techniques to overcome the disk bandwidth restriction [3, 4, 8, 11, 12, 14]. The basic idea behind buffer sharing is as follows: if two clients request the same video at different points in time, the server may service the latter one by using the data which is read into the buffer pool on behalf

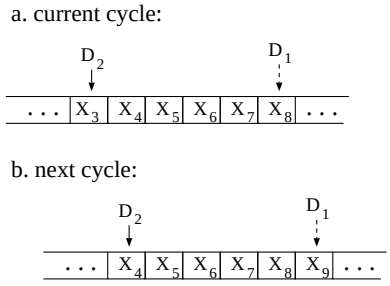


Figure 2: An example of buffer sharing

of the former one. Therefore the referenced video is read from disk only once, while the system supports two simultaneous displays. An example of buffer sharing is shown in Figure 2 where two displays (D_1 and D_2) of video X are supported by a single disk stream. In the current cycle (see Figure 2.a), two displays, D_1 and D_2, are consuming blocks 8 and 3 respectively. During the next cycle (see Figure 2.b), they consume blocks 9 and 4, respectively. If two displays are referencing the same video, one consuming block $p1$ and the other one consuming block $p2$ ($p1 < p2$), then the *distance* between them is defined as $p2 - p1$. In the example of Figure 2, the distance is 5 blocks. While the blocks referenced by D_1 and D_2 change as a function of time, the distance between them remains constant until one display ends[1]. By retaining the blocks between them in buffer pool to cover the distance, D_2 is supported with no disk access.

Ideally, a buffer sharing technique should enhance the overall performance of a system. However, all the existing buffer sharing techniques may exhaust the available buffer space, degrading the system performance. To illustrate, assume a server that consists of: 1) a disk drive with sufficient bandwidth to provide five disk streams, and 2) enough buffer space to stage five data blocks in memory. Without buffer sharing (Figure 3.a), the server can support five simultaneous displays since each display needs one disk stream and one buffer block. With buffer sharing, there could be excessive use of buffer space for sharing purpose and therefore make the buffer pool become a bottleneck. In the worst case scenario shown in Figure 3.b, the entire buffer space is used to enable sharing between two displays of Y. Other requests are queued due to lack of buffer space while 80% of the disk bandwidth sits idle. The queued requests may have to wait for hours before one of the active displays finishes.

Another important aspect of VOD systems is cost-effectiveness. For a commercial system, cost-effectiveness is critical. Buffer sharing is essentially a method of trading memory for disk bandwidth. Since memory is not a free resource, it is important to study how much memory should be traded for disk bandwidth in order to minimize the overall system cost. To the best of our knowledge, no study has addressed this issue before, even though it has a substantial impact on the cost-effectiveness of the buffer sharing technique.

In this paper, we introduce the concept of *distance threshold* (denoted d_t), which limits the number of blocks between two adjacent displays referencing the same video. None of the existing studies have proposed this concept before. If the distance between two displays exceeds d_t, then they are not allowed to share one disk stream using memory. In fact, the value of d_t defines how much memory could be used to trade for disk bandwidth. We further present the *Controlled Buffer Sharing* (CBS) scheme that (a) *prevents* the buffer pool from becoming a bottleneck and (b) reduces the disk bandwidth requirement effectively by sharing data blocks. The CBS scheme consists of two parts: a buffer management algorithm with the concept of distance threshold (*termed* BMDT *algorithm*), and a configuration planner that takes d_t as one of its input parameters to determine the amount of required buffer and disk bandwidth in support of a desirable performance objective. These two parts work together to guarantee there is no bottleneck resource (either buffer space or disk bandwidth) in the system. Moreover, we obtain the optimal d_t value which leads to the most cost-effective system configuration. **We also prove that the optimal d_t value is determined based on the cost of memory and disk bandwidth, and is independent of either the arrival rate of requests or the access frequency of each video.**

[1] We assume no VCR functions such as Pause, Resume, Fast-Forward and Fast-Rewind.

The rest of this paper is organized as follows. In Section 2, we describe the related work. In Section 3, we present the CBS scheme. In Section 4, we explain how to choose the optimal value for distance threshold to minimize the system cost. Experimental results are presented in Section 5. The conclusions and future research directions can be found in Section 6.

2 RELATED WORK

Except buffer sharing, there are two other approaches to overcome the I/O bottleneck:

- *batching [5, 16]:* in this method, requests are delayed until they can be merged with other requests for the same video. These merged streams then form one physical stream from the disk and consume only one set of buffers. Only on the network will the streams split at some point for delivery to the individual client. Although batching does not require additional resources, it may result in longer waiting time, which is unacceptable in some circumstances. Fox example, when performing the VCR functions, clients usually expect to get the response immediately, and even a few seconds waiting seems to be undesirable.
- *adaptive piggybacking [7]:* in this approach streams for the same video are adjusted to go slower or faster by a few percent, such that it is imperceptible to the viewer, and the streams eventually merge and form one physical stream from the disks.

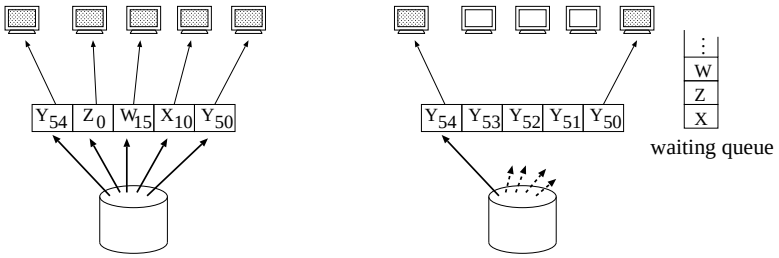
Batching and adaptive piggybacking are orthogonal to buffer sharing, and we will not investigate them any further in this paper.

3 CONTROLLED BUFFER SHARING SCHEME

In this section, we present the controlled buffer sharing (CBS) scheme. The CBS scheme consists of two parts: the BMDT algorithm and the configuration planner. These two parts work together to guarantee that there is no bottleneck resource in the system.

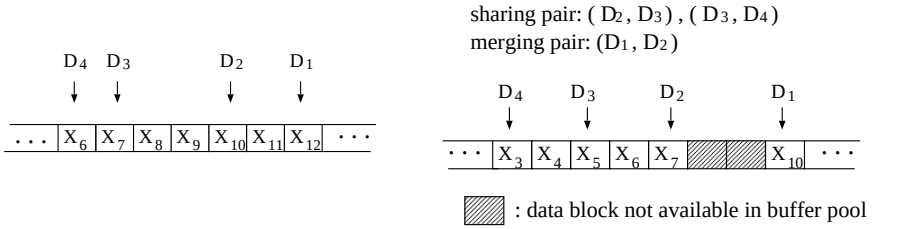
3.1 Sharing Pair and Merging Pair

We start by introducing the concept of sharing and merging pair in order to describe BMDT. For any video object X, multiple simultaneous displays of X might consume different blocks of X concurrently. For a given request referencing block X_j, its position is defined to be j (the block id that it is currently displaying). Given the distance threshold d_t, we define a $group$ of video X as a sequence of displays D_1, D_2, ..., D_n, where (1) D_i ($0 < i \leq n$) is displaying video X, (2) this sequence is ordered in terms of decreasing position of each display, (3) the distance between D_i and D_{i+1} ($0 < i < n-1$) does not exceed d_t, (4) there is no display of X that is ahead of D_1 within the distance of d_t, and (5) there is no display of X that is fewer than d_t blocks behind D_n. For each pair of (D_i, D_{i+1}) ($0 < i < n-1$) in the sequence, D_i is termed *preceding display* of D_{i+1}. D_{i+1} is termed the *succeeding display* of D_i. Figure 4.a shows an example of a group of video X consisting of four displays.



a. without buffer sharing b. with buffer sharing

Figure 3: Buffer sharing could degrade the system performance



a. A group of video X (d_t = 5) b. Sharing pair and merging pair (d_t = 5)

Figure 4: Buffer sharing

Consider a pair of preceding and succeeding displays in a group. The distance between these two displays does not exceed d_t (otherwise, they would not be in the same group based on the above definition). If the succeeding display of this group is served from the buffer pool with no disk access, we term these two a *sharing pair*. In this case, the data blocks between this pair must be retained in buffer pool to support the continuous data delivery to the succeeding display and cannot be discarded when buffer replacement is necessary. The buffer requirement of a sharing pair is determined by the distance between them. On the other hand, if the succeeding display is being served by a disk stream, then we term these two displays a *merging pair*. A merging pair can evolve to become a sharing pair when all the blocks between this pair are available in buffer pool. An example of sharing pair and merging pair is shown in Figure 4.b. Note that disk stream D_2 supports three simultaneous displays (D_2, D_3 and D_4) because the two adjacent sharing pairs are connected to one another. In fact, if a video is popular enough, then it may have a large number of consecutive blocks retained in the buffer pool because many sharing pairs of this video begin to share one disk stream. When this happens, one single disk stream will support many displays of the same video.

3.2 BMDT Algorithm

A system must satisfy the buffer requirement of a sharing pair to ensure continuous display of this pair. Moreover, each active disk stream will read one new data block into buffer pool during each cycle. Let M be the size of buffer pool in terms of the number of blocks, d_s be the sum of the distances between each sharing pair (i.e., the total number of buffer blocks required by all sharing pairs), d_m be the number of buffers occupied by merging pairs, and S be the number of active disk streams. Then the buffer constraint

$$d_s + S \leq M \tag{1}$$

must be satisfied to guarantee the continuous display for each client. The variable d_m does not appear in inequality (1) because those buffers occupied by merging pairs are not mandatory to provide continuous displays. The upper bound of d_m is $M - d_s - S$. Whenever d_m exceeds this bound, some buffers occupied by merging pairs will be discarded to reduce the value of d_m to be within its upper bound. In addition, there also exists disk bandwidth constraint, which can be expressed as

$$S \leq I \tag{2}$$

where I is the maximum number of disk streams supported by the system. Inequality (2) limits the number of active disk streams based on the disk bandwidth resource in the system. The system should manage the buffer and disk bandwidth resources in the presence of new requests, active displays ending and exiting the system, or merging pairs evolving to sharing pairs.

We propose the BMDT algorithm as follows. Applied in a system configured according to the planner (Section 3.3), this algorithm guarantees that there will be no queued requests, i.e., each request is served immediately upon its arrival in the server (see the result in Section 5). The BMDT algorithm is applied at the end of each cycle. The admission control is implied in this algorithm so that the continuous display for each client is guaranteed.

1. Free the disk stream that has reached the end of a display.
2. Evolve as many of the merging pairs to sharing pairs and free the disk stream of the succeeding display while: a. all the blocks between the merging pair are available in buffer; and b. inequality (1) is satisfied.

   Inequality (1) is a condition of this step because evolving merging pairs to sharing ones increases the value of d_s and impacts this inequality.
3. Admit as many of the new requests while both inequality (1) and (2) are satisfied.

   Inequalities (1) and (2) are involved because a new display may need both buffer and disk bandwidth resources.
4. Perform buffer replacement and allocation algorithm.

The buffer replacement and allocation algorithm in step *4* is further detailed as follows. It will decide which buffer should be discarded when buffer replacement is necessary.

1. Free the buffers which have no potential for sharing.

   This step discards all the buffers which do not fall in between merging/sharing pair, i.e., no active display will obtain data from these buffers in the future.

2. While $(d_s + d_m + S > M)$ Do

   i. choose the merging pair with the longest distance as the victim.

   ii. free those buffers that fall in between this merging pair.

   iii. decrement d_m by the number of buffers freed in ii.

   This step will discard the buffers that fall in between the victim merging pair to create buffer space for either sharing pairs or new displays.

3. Assign one free buffer to each active disk stream.

   This step specifies the address to which the new data block will go.

So far the BMDT algorithm is complete. However, setting a threshold for buffer sharing solely might not prevent the undesirable condition shown in Figure 3.b. If the buffer pool is too small, it could still form a bottleneck; if the buffer pool is too large, some buffer space may be wasted. We need to quantify how much buffer space is actually required. Moreover, in the presence of BMDT, the I/O requirement will be reduced, and therefore we also need to quantify how much disk bandwidth is actually needed.

3.3 Configuration Planner

The configuration planner computes the buffer and disk bandwidth requirement based on a number of input parameters. These values are determined with the objective to minimize system cost. It is applied at system generation to determine its configuration parameters. Once configured, the BMDT algorithm is employed at run-time. The planner and the BMDT algorithm constitute the CBS scheme.

The input to the planner includes the arrival rate of requests, the access frequency distribution, the number of videos available, the system utilization factor, and the distance threshold (d_t). The output is the number of buffer blocks and the number of disk streams required. Except d_t, all the input parameters reflect the physical characteristic of the system. Theoretically the planner could take any value of d_t as its input. Different d_t values will lead to different buffer and disk bandwidth requirements, resulting in different system cost. In the next section (Section 4), we will describe how to choose the optimal d_t value which brings the system cost down to the minimum.

Assume the request arrival process follows Poisson distribution with rate λ. Let V denote the number of different videos in the system. Assume the length of each video is l, and let μ be $\frac{1}{l}$. Upon arrival of a request to the system, the client chooses to watch video j $(1 \leq j \leq V)$ with probability p_j. Therefore, for each video j $(1 \leq j \leq V)$, the request arrival follows a Poisson process with rate λ_j where $\lambda_j = p_j \lambda$ and $\sum_{j=1}^{V} \lambda_j = \lambda$. Then the expected number of displays that the system must support is

$$m = \frac{\lambda}{\mu} \tag{3}$$

and the expected number of displays that reference video j is

$$m_j = \frac{\lambda_j}{\mu} \tag{4}$$

Consider the disk bandwidth requirement first. With no buffer sharing, m is also the expected number of disk streams in the system. Now consider m', the expected number of disk streams in a system employing BMDT algorithm. Let t_p denote the length of a cycle, then the time interval between a sharing pair should not exceed $d_t \cdot t_p$. According to Poisson process, the probability of two requests for video j arriving within $d_t \cdot t_p$ is

$$p_{s_j} = 1 - e^{-\lambda_j \cdot d_t \cdot t_p} \tag{5}$$

Let I_j be a random variable denoting the number of disk streams which are retrieving the data of video j. We already know the expected number of displays of video j is m_j. Let $D_1, D_2, ..., D_{m_j}$ denote these m_j displays. From Equation (5) we know the probability that D_i and D_{i+1} $(1 < i < m_j)$ are bridged up to form a sharing pair is p_{s_j}, therefore, the probability that there is no bridge between D_i and D_{i+1} $(1 < i < m_j)$ is $1 - p_{s_j}$. If there are only one out of m_j displays served from disk, it means all these m_j displays are bridged up. Let $P[I_j = k]$ be the probability that k out of m_j displays are served from disk while the other $m_j - k$ displays being served from the buffer pool. Then we get $P[I_j = 1] = p_{s_j}^{m_j - 1}$ since there are totally $m_j - 1$ bridges. If there are two out of m_j displays served from disk, it means there are $m_j - 2$ bridges among the $m_j - 1$ display pairs and there is only one display pair which is not bridged up. There are totally $\begin{pmatrix} m_j - 1 \\ 1 \end{pmatrix}$ combination of the pair with no bridge. Then we have $P[I_j = 2] = \begin{pmatrix} m_j - 1 \\ 1 \end{pmatrix} \cdot p_{s_j}^{m_j - 2} \cdot (1 - p_{s_j})$. By generalizing the computation, we have

$$P[I_j = k] = \begin{pmatrix} m_j - 1 \\ k - 1 \end{pmatrix} \cdot p_{s_j}^{m_j - k} \cdot (1 - p_{s_j})^{k-1} \tag{6}$$

and the expected value of I_j can be expressed as

$$E[I_j] = \sum_{k=1}^{m_j} k \cdot P[I_j = k] \tag{7}$$

$E[I_j]$ is actually the expected number of disk streams which are displaying video j. Then the expected number of disk streams in a system employing BMDT is

$$m' = \sum_{j=1}^{V} E[I_j] \tag{8}$$

Now Consider the buffer requirement of the system. With no buffer sharing, each display needs one buffer and the expected number of required buffers w is

$$w = m \tag{9}$$

The measurement of buffer requirement under BMDT can be achieved as follows. First, consider the sharing pairs of video j. Let d_j be a random variable denoting the distance between a sharing pair of video j. If $d_j = k$, it means that the time interval between the sharing pair is less than $k \cdot t_p$ but greater than $(k-1) \cdot t_p$. The probability of $d_j = k$ can be expressed as

$$P[d_j = k] = (1 - e^{-\lambda_j \cdot k \cdot t_p}) - (1 - e^{-\lambda_j \cdot (k-1) \cdot t_p}) \tag{10}$$

where $1 \leq k \leq d_t$. Since we are considering the buffers occupied by a sharing pair of video j, the probability that this sharing pair requires k buffers is

$$P[d_j = k | d_j \leq d_t] = \frac{P[d_j = k \cap d_j \leq d_t]}{P[d_j \leq d_t]} \tag{11}$$

where the event $d_j \leq d_t$ means this is a sharing pair. Since $1 \leq k \leq d_t$, the event $d_j = k$ implies $d_j \leq d_t$, therefore $P[d_j = k \cap d_j \leq$

$d_t] = P[d_j = k]$. From Equation (5) we know $P[d_j \leq d_t] = p_{s_j}$. Then Equation (11) can be rewritten as

$$P[d_j = k | d_j \leq d_t] = \frac{P[d_j = k]}{p_{s_j}} \quad (12)$$

Now we have the expected number of buffers occupied by one sharing pair of video j as follows:

$$E[d_j] = \sum_{k=1}^{d_t} k \cdot P[d_j = k | d_j \leq d_t] \quad (13)$$

From Equation (7) we obtain the expected number of disk streams of video j, $E[I_j]$, then the expected number of displays which are served from the buffer pool is $m_j - E[I_j]$. So the expected number of buffers required by sharing pairs of video j is

$$b_j = (m_j - E[I_j]) \cdot E[d_j] \quad (14)$$

and the expected number of buffers required by all sharing pairs in the system is

$$b = \sum_{j=1}^{V} b_j \quad (15)$$

Besides the buffer requirement of sharing pairs, each active disk stream will read one new data block into the buffer pool during each cycle as we explained previously, therefore the expected number of buffers required by the whole system is

$$w' = b + m' \quad (16)$$

Equation (8) and (16) are the solution of the expected number of disk streams and the expected number of buffer blocks. Given the system utilization factor, we may further get the actual number of disk streams and buffer blocks through a simple multiplication. For example, if the expected number of disk streams is 4000 and the utilization factor is 80%, then the system should be configured with 5000 disk streams.

Experimental results (see Section 5) shows that if a system is configured according to this planner and the BMDT algorithm is applied at the system run-time, then there will be no bottleneck resources in the system.

4 CHOOSING THE OPTIMAL DISTANCE THRESHOLD

From Section 3.3, the buffer and disk bandwidth requirements are impacted by the value of d_t. If d_t is too large, more buffer space may be needed to enable sharing, increasing the memory cost. On the other hand, a small value of d_t may discourage sharing all together, causing the disk cost to become significant. In a real system it is important to choose the optimal d_t value that minimizes the system cost with respect to memory and disk resources.

Let $M_\$$ denote the cost of memory for each data block, and $I_\$$ denote the cost of a disk stream. $I_\$$ can be obtained through the cost of one disk drive and the number of disk streams it supports. For example, if one disk drive costs \$1000 and it can support as many as 10 disk streams, then $I_\$$ will be \$100. We found the optimal d_t value is $\frac{I_\$}{M_\$}$ when it is an integer value. (The discussion for $\frac{I_\$}{M_\$}$ as a real number is at the end of this section.) The formal proof is as follows.

Let a_i denote the number of displays that are i blocks apart from the immediate prior displays of the same video. When i does not exceed d_t, the cost of these a_i displays is $a_i \cdot i \cdot M_\$$ because they are all served from buffer pool with no disk access. When i is greater than d_t, the displays must be served from disk and they each cost $I_\$ + M_\$$ due to the requirements of one disk stream and one buffer. Then the cost of the whole system C can be expressed as

$$C = \sum_{i=0}^{d_t} a_i \cdot i \cdot M_\$ + (m - \sum_{i=0}^{d_t} a_i) \cdot (I_\$ + M_\$) \quad (17)$$

Let d_{opt} be $\frac{I_\$}{M_\$}$ (assuming it is an integer), then:

$$C_{opt} = \sum_{i=0}^{d_{opt}} a_i \cdot i \cdot M_\$ + (m - \sum_{i=0}^{d_{opt}} a_i) \cdot (I_\$ + M_\$) \quad (18)$$

For those d_t values greater than d_{opt}, the difference between C and C_{opt} is

$$C - C_{opt} = \sum_{i=d_{opt}+1}^{d_t} a_i \cdot i \cdot M_\$ - \sum_{i=d_{opt}+1}^{d_t} a_i \cdot (I_\$ + M_\$) \quad (19)$$

which can be rewritten as

$$C - C_{opt} = \sum_{i=d_{opt}+1}^{d_t} a_i \cdot (i \cdot M_\$ - (d_{opt} \cdot M_\$ + M_\$)) \quad (20)$$

The value is no less than zero and demonstrates that the system cost does not decrease with those d_t values greater than d_{opt}. In reality $C - C_{opt} = 0$ only when d_t is equal to $d_{opt} + 1$. (This is because each display needs *1* buffer block without sharing. Take a simple example where $d_{opt} = 10$ and two displays of X have a distance of 11 blocks. If d_t is set to d_{opt}, then the cost for these two displays is $c = 2 \cdot I_\$ + 2 \cdot M_\$$ because each of them needs one disk stream and one buffer block. If d_t is set to $d_{opt} + 1$, then they can share one disk stream. The cost of the preceding display is $I_\$ + M_\$$, and the cost of the succeeding one is $11 \cdot M_\$$. Therefore the total cost of the two displays is $c' = I_\$ + 12 \cdot M_\$$. Obviously $c = c'$ since $\frac{I_\$}{M_\$} = 10$.)

Now consider those d_t values less than d_{opt}. For these values, the difference between C and C_{opt} is

$$C - C_{opt} = \sum_{i=d_t+1}^{d_{opt}} a_i \cdot (I_\$ + M_\$) - \sum_{i=d_t+1}^{d_{opt}} a_i \cdot i \cdot M_\$ \quad (21)$$

which can be rewritten as

$$C - C_{opt} = \sum_{i=d_t+1}^{d_{opt}} a_i \cdot (d_{opt} \cdot M_\$ + M_\$ - i \cdot M_\$) \quad (22)$$

This value is greater than 0 because $i \leq d_{opt}$. It completes the proof that the system with the distance threshold of d_{opt} achieves the lowest system cost.

In this proof, we applied the same d_t value on each video. Similarly, it can also be proved that applying any d_t value other than d_{opt} or $d_{opt} + 1$ on any video in the system will lead to a system cost higher than C_{opt} (the proof is similar to the above and is eliminated here due to lack of space).

From the procedure of the above proof, it is shown that the optimal d_t value is impacted only by the cost of memory and disk stream, and interestingly it is independent of the arrival rate of requests, the access frequency distribution, and the access frequency of each individual video (the experimental results in Section 5 also demonstrate this).

The optimal d_t value $\frac{I_\$}{M_\$}$ indicates that one disk stream is equivalent to d_{opt} memory blocks as far as cost is concerned. When using a d_t value larger than d_{opt}, some displays may be served from the buffer pool by using d_t memory blocks each, which cost more than serving them directly from disk streams. On the other hand, when using a d_t value smaller than d_{opt}, some displays with the distance of d_t from their immediate prior display will be served by disk streams, while otherwise they could be served from the buffer pool and cost less. Essentially d_{opt} defines a balance point of trading memory for disk bandwidth which leads to the most cost-effective system configuration. In the extreme case where memory is very cheap (i.e., $M_\$$ becomes very small), d_{opt} may go to infinity, which means the entire video could be retained in memory for sharing between different clients.

When $\frac{I_\$}{M_\$}$ is not an integer, we first configure the system according to the planner by taking $\lfloor \frac{I_\$}{M_\$} \rfloor$ as the distance threshold, and compute the system cost ($m' \cdot I_\$ + w' \cdot M_\$$). Then we configure the system using $\lceil \frac{I_\$}{M_\$} \rceil$ and compute the system cost in this case. The optimal d_t will be the value which results in the lower cost.

5 EXPERIMENTAL RESULTS

For experimental purposes, we assume an open simulation study that employs a Poisson arrival rate of requests. Upon arrival of a request, if the system cannot admit the request due to lack of resources, then it is rejected. The server contains 100 constant bit rate MPEG-2 encoded videos, each requiring 4 Mbps for its display. Each clip is 120 minutes long. The length of each cycle is 2 seconds and the size of a data block is 1 MB. The frequency of access to each video clip is based on a Zipf distribution[2] with parameter 0.271 (see Figure 5) which has been shown to match with the empirical data from video rental stores in a particular week [5]. The system utilization factor is set to 90%.

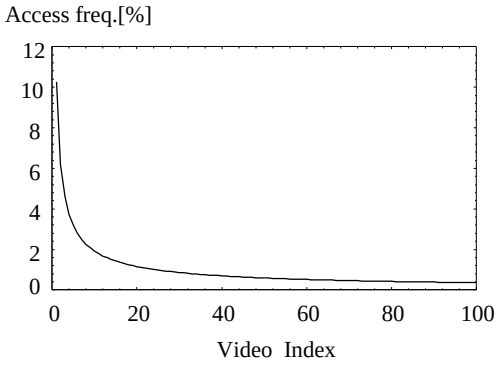


Figure 5: Access frequency distribution (Zipf with parameter 0.271)

We start with experiments which compare the performance of different buffer sharing techniques. The value of $I_\$$ and $M_\$$ is set to be \$92.00 and \$8.00 respectively according to the market price. For example, one Seagate Barracuda ST19171WB disk drive is about \$1470 and can provide 16 disk streams in our experiments. Thus $I_\$$ is around \$92.00. $M_\$$ is the cost for 1 MB memory in our experiments. First we compute the optimal d_t value which is 12 ($\lceil \frac{92}{8} \rceil$) in our case. Now consider the arrival rate of 20 requests per minute. We configure the system according to the planner in Section 3.3. Then, under the same system configuration, we perform three experiments using different buffer management techniques: 1) the CBS scheme, 2) buffer sharing with no distance threshold and 3) no buffer sharing at all. Each experiment is run for 100 hours. We measure the system throughput in each experiment. The experiments for other arrival rates (30, 40, ...,70) are conducted by following the same steps. It could happen that two requests for the same video arrive within one cycle, however, the chance is very low (less than 1%) in all the experiments and is neglected in our study. The results presented in Figure 6 show that the CBS

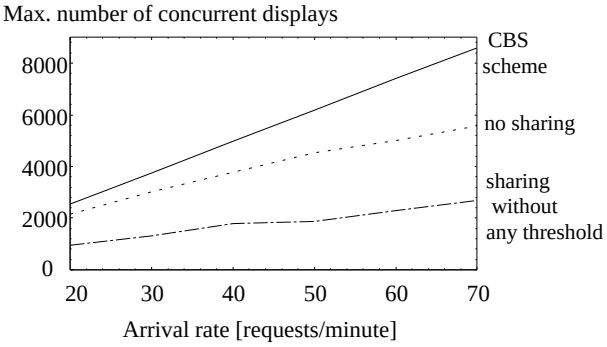


Figure 6: Comparison of system performance

scheme achieves the best performance, and buffer sharing with no distance threshold is even worse than no buffer sharing at all due to the exhausted buffer usage. We also monitor the percentage of requests which are rejected due to lack of buffer space and/or disk bandwidth in each experiment. The results are shown in Figure 7.

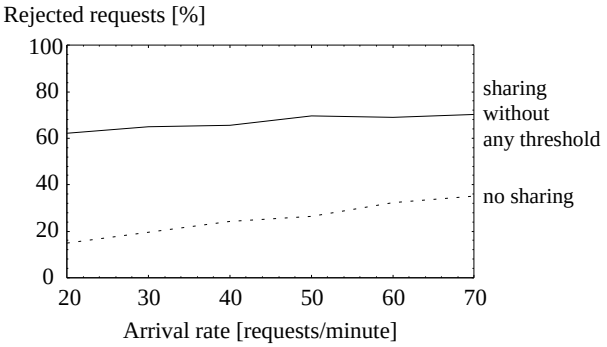


Figure 7: Comparison of the percentage of rejected requests

The CBS scheme does not appear in Figure 7 because there is no request rejected in the experiments using CBS. This demonstrates that there is no bottleneck in a system employing CBS.

We then study the characteristic of the optimal d_t value.

In Section 4, it is proved that the optimal d_t value is determined by the ratio between $I_\$$ and $M_\$$. To verify this, we fix $I_\$$ at \$92.00 and vary the $M_\$$ value to be \$6.00, \$8.00, and \$10.00 respectively. The arrival rate is 50 requests per minute. The system is configured for each d_t value from 0 to 25 according to the planner. Note that each configuration may have different amount of buffer and disk bandwidth, but the number of concurrent displays supported by the system is the same. The system cost ($m' \cdot I_\$ + w' \cdot M_\$$) is measured in each case and depicted (Figure 8) as compared with the case of $d_t = 0$. Note that when $d_t = 0$ there is no sharing and the system cost is marked as 1. In Figure 8, the lowest system cost is achieved when d_t is 16, 12, and 9 with the $M_\$$ value as \$6.00, \$8.00 and \$10.00 respectively, which exactly verified our study on how to choose the optimal value.

Figure 9 shows the system cost under different arrival rates

[2] In a Zipf distribution, if the videos are sorted according to the access frequency then the access frequency for the i^{th} video is given by $f_i = \frac{c}{i^{(1-\theta)}}$, where θ is the parameter for the distribution and c is the normalization constant.

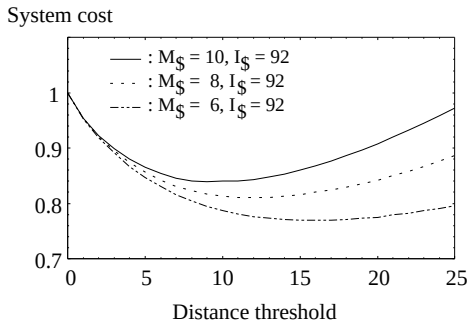


Figure 8: Optimal d_t value for different memory price

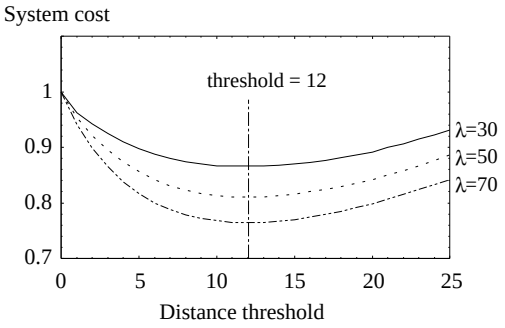


Figure 9: Optimal d_t for different arrival rate

and distance threshold values. The value of $I_\$$ and $M_\$$ are \$92.00 and \$8.00 respectively. We try three different arrival rates (30, 50 and 70). For each arrival rate, we configured the system according to the planner by using various distance threshold. The result in Figure 9 shows the lowest system cost is achieved when $d_t = 12$ for every arrival rate, and demonstrates that the optimal d_t value is independent of the arrival rates.

We also study the relationship between the optimal d_t value and the access frequency distribution. Still $I_\$ = 92.00$ and $M_\$ = 8.00$. The arrival rate is fixed at 50 requests per minute. Except the Zipf with parameter 0.271 which matches well with the empirical data on video rental, we try two other parameters 0.135 and 0.012 which are slightly more skewed than that of parameter 0.271. (The actual customer demand may be more skewed than the empirical rental frequencies because video stores can stock only a limited number of copies of popular videos and customers who wish to rent a popular video may be forced to rent a less popular video.) For each distribution, the system is configured according to the planner by using various distance threshold. The system cost in each case is measured and presented in Figure 10, which shows that the optimal d_t is independent of the access frequency distribution.

A misconception about distance threshold is that setting a different threshold for the popular videos would reduce the system cost. In fact, using a d_t value either larger or smaller than the optimal value on popular videos will increase the system cost. We perform the experiments in the following approach. First, we identified the 10 most popular videos in the system. For the remaining 90 videos we fix d_t at 12 which is the optimal value. Then for each number from 1 to 30, we configure the system specially by applying this number as the distance threshold on the 10 popular videos. In all cases the arrival rate is 50 and the access frequency is Zipf with parameter 0.271. Figure 11 shows the system cost vs. the threshold applied on the popular videos. The system cost of no sharing is marked as 1. It is clear that only when the threshold applied on popular videos is 12 does the system reach the lowest cost. This result shows the optimal d_t value is independent of the access frequency of the individual video.

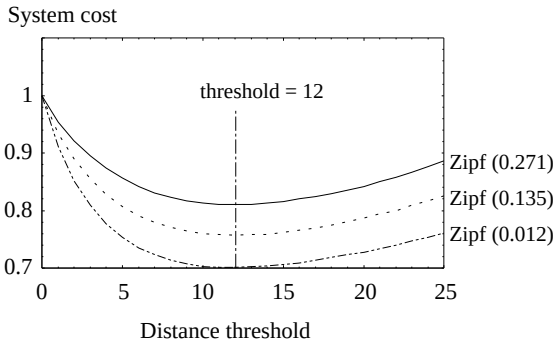


Figure 10: Optimal d_t for different access freq. distribution

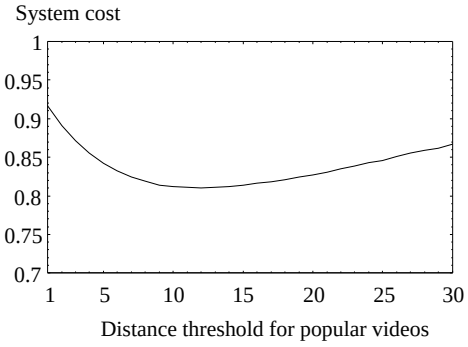


Figure 11: Optimal d_t is independent of video popularity

6 CONCLUSION AND FUTURE RESEARCH DIRECTIONS

Buffer sharing could be effective to reduce the I/O requirement in VOD systems. However, the existing buffer sharing techniques may exhaust the available memory space to form new bottleneck. In this paper we introduced the concept of distance threshold and presented the CBS scheme which guarantees there is no bottleneck resource (either memory or disk bandwidth) in the system. Moreover, we obtained the optimal value of distance threshold which minimizes the system cost. This optimal value is independent of the arrival rate of requests, the access frequency distribution, and the access frequency of each individual video.

Presently, we are pursuing two research directions. First, we are extending the CBS scheme to mixed media types, i.e., there could be videos of different bit rates in the system. Second, we are attempting to support VCR functions in the presence of controlled buffer sharing.

References

[1] *Electronic Engineering Times*, page 72, March 1993.

[2] D.P. Anderson, Y. Osawa, and R. Govindan. A File System for Continuous Media. *ACM Transactions on Computer Systems*, 10(4):311–337, November 1992.

[3] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and Caching in Large-Scale Video Servers. In *Proc. of COMPCON*, 1995.

[4] A. Dan and D. Sitaram. Buffer management policy for an on-demand video server. *U.S. Patent No. 5572645*, November 1996.

[5] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Sever with Batching. In *Proceedings of the ACM Multimedia*, pages 391–398, 1994.

[6] S. Ghandeharizadeh, R. Zimmermann, W. Shi, R. Rejaie, D. Ierardi, and T.W. Li. Mitra: A Scalable Continuous Media Server. *Kluwer Multimedia Tools and Applications*, pages 79–108, July 1997.

[7] L. Golubchik, J. Lui, and R. Muntz. Reducing I/O Demand in Video-On-Demand Storage Servers. In *Proceedings of the ACM SIGMETRICS*, pages 25–36, 1995.

[8] M. Kamath, K. Ramamritham, and D. Towsley. Continuous Media Sharing in Multimedia Database Systems. In *Proceedings of the 4th International Conference on Database Systems for Advanced Applications*, pages 79–86, 1995.

[9] P. Lougher and D. Shepgerd. The Design of a Storage Server for Continuous Media. *The Computer Journal (special issue on multimedia)*, 36(1):32–42, February 1993.

[10] R.T. Ng and J. Yang. Maximizing Buffer and Disk Utilizations for News On-Demand. In *Proceedings of the International Conference on Very Large Databases*, September 1994.

[11] B. Özden, R. Rastogi, and A. Silberschatz. Demand Paging for Video-on-demand Servers. *IEEE International Conference on Multimedia Computing and Systems*, May 1995.

[12] B. Özden, R. Rastogi, and A. Silberschatz. Buffer Replacement Algorithms for Multimedia Databases. *IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[13] P. Rangan and H. Vin. Efficient Storage Techniques for Digital Continuous Media. *IEEE Transactions on Knowledge and Data Engineering*, 5(4), August 1993.

[14] D. Rotem and J.L. Zhao. Buffer Management for Video Database Systems. In *Proceedings of International Conference on Database Engineering*, pages 439–448, March 1995.

[15] F.A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID-A Disk Array Management System for Video Files. In *First ACM Conference on Multimedia*, August 1993.

[16] J. Wolf, H. Shachnai, and P. Yu. DASD Dancing: A Disk Load Balancing Optimization Scheme for Video-on-Demand Computer Systems. In *Proceedings of the ACM SIGMETRICS and Performance*, May 1995.