Controlled Buffer Sharing in Continuous Media Servers *

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

October 24, 2002

Abstract

Continuous media servers manage delay sensitive data such as audio and video clips. Once a server initiates the display of a clip on behalf of a client, it must deliver the data to the client in a manner that prevents data starvation. Otherwise, its display may suffer from disruptions and delays, termed hiccups. A hiccup-free display is important to a number of applications such as video-on-demand for entertainment, distance learning, news dissemination, etc. Buffer sharing enables a server to trade memory for disk bandwidth to service multiple clients by sharing data in memory, using a single disk stream. However, an uncontrolled buffer sharing scheme may reduce system performance.

This paper presents Controlled Buffer Sharing (CBS) as a novel framework that facilitates sharing and supports both a hiccup-free display and VCR operations. It includes a configuration planner and a buffer pool management technique (applied at run time). CBS trades memory for disk bandwidth in order to meet the performance objectives of an application and minimize cost per stream. It uses bridging and merges two displays referencing the same clip when they are d_t blocks apart. One insight of this framework is that d_t is determined by market forces (cost of memory and disk bandwidth) and is independent of a clip's frequency of access. We use both analytical and simulation models to quantify the characteristics of CBS.

1 Introduction

Sharing of data in memory reduces the number of disk accesses. This enhances system performance because memory is significantly faster than disk; 1 million times faster assuming a latency of 60 nanoseconds for memory and 6 milliseconds for disk[1]. When many simultaneous clients reference the same data item in memory, memory's low latency enables the system to satisfy all requests in a short period of time. This also provides for network delivery optimization [ZT99, RHYE99, SRT99]. With continuous media, e.g., audio and video clips, many clients may reference the same video clip simultaneously. This video clip may have a display time of 90 minutes and consist of thousands of blocks. Moreover, requests might arrive staggered in time such that they reference different memory locations at any given instance in time. A simple "Use & Toss" strategy prevents these requests from sharing memory buffers [CAF+91]. A straightforward use of LRU or LRU-K [OOW93, LCK+99] may also prove inadequate [ORS95]. To illustrate, Figure 1 shows two displays,

*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.

[1]This also explains why the cost per megabyte of memory is higher than disk.

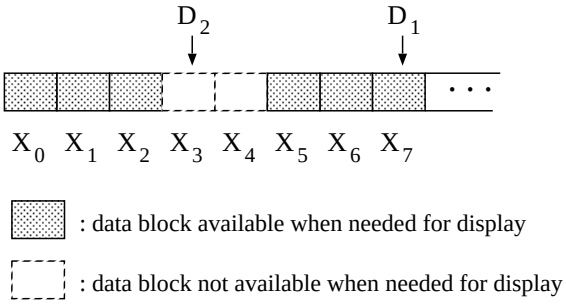


Figure 1: Two displays may compete for buffer frames with LRU

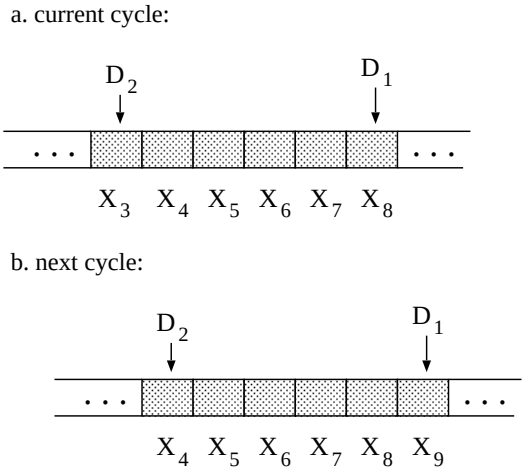


Figure 2: An example of bridging

D_1 and D_2, referencing different blocks of the same clip. If the system applies LRU to D_1 and "Use & Toss" to D_2 then these two displays merge so that a single disk stream would serve both displays [MKK95]. With LRU as a global buffer pool replacement policy, these two displays may not merge because the trailing blocks of D_1 are discarded in favor of trailing blocks of D_2.

Bridging [DDM+95, DS96, RZ95, SG97, SG98] is a more intelligent technique to facilitate sharing of data in memory. This technique forms a bridge between the data blocks staged by two different clients referencing the same clip, enabling them to share memory and use one disk stream. An example is shown in Figure 2 where two displays (D_1 and D_2) of video X are supported using a single disk stream. If two displays reference the same clip, one consuming block $p1$ and the other block $p2$ ($p1 \leq p2$) then their distance is $p2 - p1$. In Figure 2, the distance between D_2 and D_1 is 5. A bridging technique pins the intermediate data pages between two displays in the buffer pool, preventing the buffer pool manager from swapping these pages out. In Figure 2, D_1 pins data pages for consumption by D_2 while D_2 consumes and unpins these pages to free buffer frames.

A naive application of this approach may exhaust the available buffer space, degrading system performance [SG97, SG98]. To illustrate, assume a server that consists of: a) a disk drive with sufficient bandwidth to support 5 simultaneous clients, and b) enough buffer space to stage five data blocks in memory. Without bridging, see Figure 3.a, the server can support five simultaneous displays since each display consumes one memory frame and one disk stream[2]. Bridging might cause the buffer pool to become a bottleneck. In the

[2]Without loss of generality, we assume that a display requires at most one buffer frame for its continuous display. This assump-

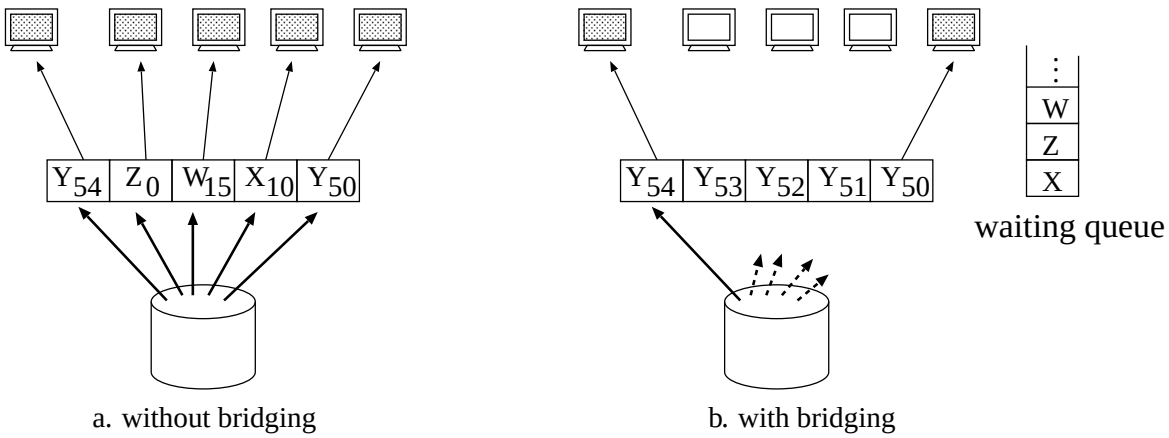


Figure 3: Bridging could degrade the system performance

worst case scenario, see Figure 3.b, it may pin data pages in all five buffer frames to enable bridging between two displays of a clip Y. With all buffer frames allocated, the system is forced to queue other requests that reference a different clip while 80% of the disk bandwidth sits idle.

This paper describes a framework for Controlled Buffer Sharing (CBS). It trades memory for disk bandwidth in order to accomplish two objectives: a) minimize cost per simultaneous stream supported by the system, and b) at run time, balance the memory and disk utilization in a manner that prevents memory frames from becoming system bottleneck (such as that depicted in Figure 3.b). This framework consists of a configuration planner and a buffer pool management technique. The first consumes the performance objectives of a target application and configures the system accordingly to meet this objective (with a minimum cost per stream). Next, the buffer management technique, termed BMDT, is used at run time to prevent memory from becoming system bottleneck. The *distance threshold*, d_t, is an important system parameter and captures the price of memory and disk bandwidth. This parameter is an input to both the configuration planner and BMDT.

The contributions of this study is CBS as a framework that can trade memory for disk bandwidth in order to minimize cost per stream. As detailed in Section 4, while BMDT shares similarities with other previous buffer sharing techniques, the framework that subsumes this strategy is novel. Our framework is validated using simulation studies. The results demonstrate: a) CBS minimizes cost per stream, b) VCR operations can be supported with negligible extra cost, c) the impact of CBS framework is more profound with larger ratios between $I_\$$ and $M_\$$, cost per disk bandwidth and megabyte of memory, respectively.

The rest of this paper is organized as follows. Section 2 describes the CBS framework. Section 3 presents a performance analysis of this framework. We survey the related studies in Section 4. Brief conclusions and future research directions are offered in Section 5.

2 Controlled Buffer Sharing Scheme

This section start with an overview of CBS, followed by its detailed description. The overview establishes distance threshold, d_t, as a parameter important to the overall framework. Section 2.2 derives the optimal value of d_t that minimizes system cost. We use this to detail CBS concisely.

tion simplifies the discussion about the CBS framework. Extensions of CBS to scheduling techniques such as Group Sweeping Scheme [YCK93] that require 2 buffers on behalf of each display is a straightforward extension of this work.

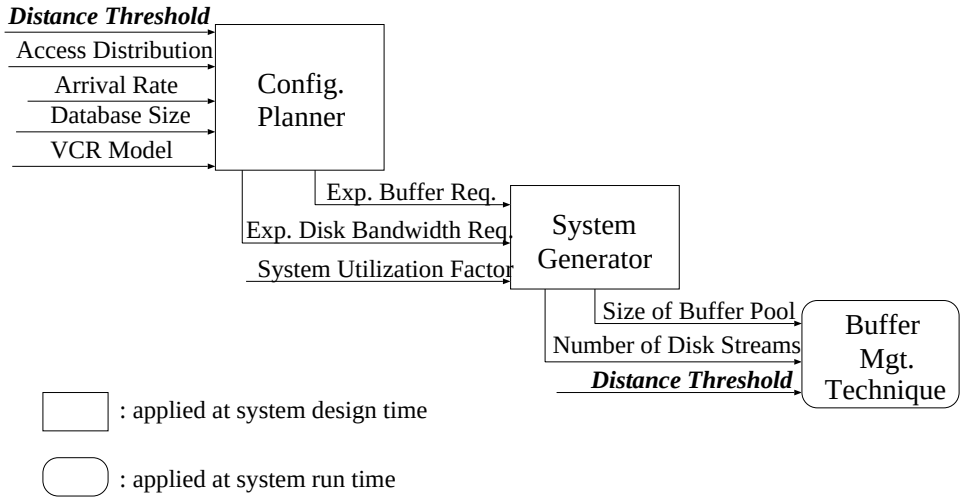


Figure 4: Overview of the CBS scheme

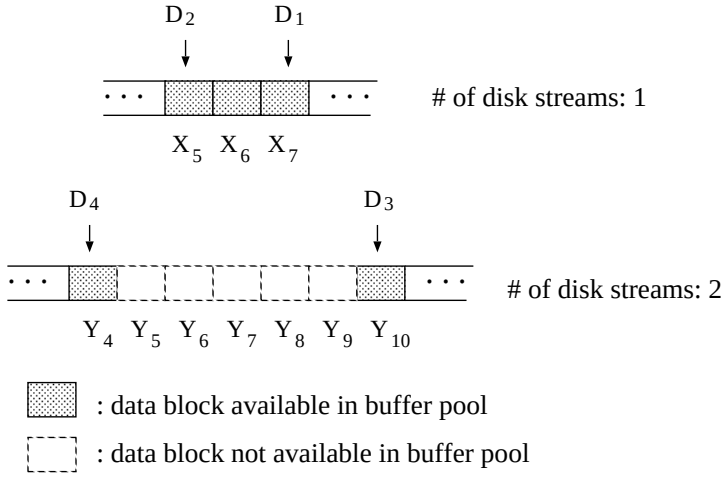


Figure 5: The effect of distance threshold (d_t = 5)

2.1 Overview of the CBS scheme

Figure 4 shows the three components of CBS: a configuration planner, a system generator, and a buffer pool management technique. The first two are applied off-line to determine system size. The third, termed *BMDT*, is utilized at system run time. The system generator is the simplest one and acts as a multiplier. For example, if the expected buffer requirement is 4000 blocks (input) and the system is to operate at 80% utilization (input), then the buffer pool should consist of 5000 blocks (output). The purpose of the configuration planner is to determine the amount of required buffer and disk bandwidth in support of a pre-specified performance objective. Both the configuration planner and BMDT consume the *distance threshold* (denoted d_t) as their input. Distance threshold is a major concept with CBS. It limits the number of pinned buffer blocks between two adjacent displays referencing the same video. If the distance between two adjacent displays exceeds d_t, then they are allowed to neither pin their intermediate data pages nor share one disk stream using memory. Figure 5 shows two different scenarios with distance threshold equal to five. In this figure, D_3 and D_4 cannot share one disk stream because their distance exceeds the specified threshold. However, D_1 and D_2 are allowed to share one disk stream because their distance is below the threshold. In essence, d_t specifies how

much memory can be traded for disk bandwidth. It also prevents memory from becoming system bottleneck in order to avoid the undesirable scenario of Figure 3.b. This framework assumes an "admission control" component that monitors and controls available resources. When resources (memory and disk bandwidth) are exhausted, it queues newly arrived requests in order to prevent hiccups. A number of studies have outlined the design and implementation of such a component [GM98, GZS+97, YCK93, OBRS94, WSY95, TPBG93].

This section begins with d_t and how to derive its optimal value in order to minimize system cost. Next, we present BMDT, followed by a discussion of the configuration planner. For each, we describe the impact of the VCR operations. The configuration planner and BMDT are designed to complement one another to prevent either the available memory or disk bandwidth from resulting in data starvation. Moreover, we demonstrate that our approach provides full VCR functionality with marginal extra costs.

2.2 Optimal distance threshold

The buffer and disk bandwidth requirements of a system are impacted by the value of d_t. If d_t is too large, additional buffer space might be required to facilitate sharing, increasing cost per stream. 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 system cost based on price of memory and disk resources.

Let $M_\$$ denote the cost of memory for each data block, and $I_\$$ denote the cost of a disk stream. Note that a larger block size increases both $M_\$$ and $I_\$$ [GM98]. $I_\$$ can be obtained through the cost of one disk drive and the number of disk streams it supports. For example, if a disk drive costs $\$1000$ and supports 10 simultaneous disk streams, then $I_\$$ is $\$100$. If the cost per megabyte of memory is one dollar and a block size is one megabyte in size then $M_\$$ is $\$1$. 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 intuition behind this claim is as follows. With buffer sharing, we may use d_t memory frames to free up a disk stream. Each has a cost and $\frac{I_\$}{M_\$}$ is the break even point to trade a disk stream for d_t memory frames. A larger d_t value ($I_\$$ and $M_\$$ ratio) complements the CBS framework and makes it more cost effective as demonstrated in Section 3.

The formal proof of $d_t = \frac{I_\$}{M_\$}$ 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

$$C = \sum_{i=0}^{d_t} a_i \cdot i \cdot M_\$ + (m - \sum_{i=0}^{d_t} a_i) \cdot (I_\$ + M_\$) \tag{1}$$

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

$$C_{opt} = \sum_{i=0}^{d_{opt}} a_i \cdot i \cdot M_\$ + (m - \sum_{i=0}^{d_{opt}} a_i) \cdot (I_\$ + M_\$) \tag{2}$$

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

$$C - C_{opt} = \sum_{i=d_{opt}+1}^{d_t} a_i \cdot i \cdot M_\$ - \sum_{i=d_{opt}+1}^{d_t} a_i \cdot (I_\$ + M_\$) \tag{3}$$

which can be rewritten as

$$C - C_{opt} = \sum_{i=d_{opt}+1}^{d_t} a_i \cdot (i \cdot M_\$ - (d_{opt} \cdot M_\$ + M_\$)) \quad (4)$$

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 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

$$C - C_{opt} = \sum_{i=d_t+1}^{d_{opt}} a_i \cdot (I_\$ + M_\$) - \sum_{i=d_t+1}^{d_{opt}} a_i \cdot i \cdot M_\$ \quad (5)$$

which can be rewritten as

$$C - C_{opt} = \sum_{i=d_t+1}^{d_{opt}} a_i \cdot (d_{opt} \cdot M_\$ + M_\$ - i \cdot M_\$) \quad (6)$$

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, we can also show 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 above proof shows only two factors impact the optimal d_t value, namely, the cost associated with memory and a disk stream. The optimal d_t value 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 3.2 also demonstrate this). This observation is consistent with those for relational database management systems [GG97, GS00].

When $\frac{I_\$}{M_\$}$ is not an integer, we invoke the planner with two possible d_t values: $\lfloor \frac{I_\$}{M_\$} \rfloor$ and $\lceil \frac{I_\$}{M_\$} \rceil$. We choose the one that minimizes system cost.

2.3 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 the same video clip 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 *producing display* of D_{i+1}. D_{i+1} is termed the *consuming display* of D_i.

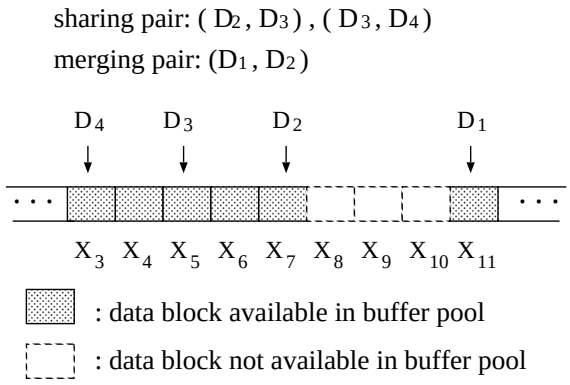


Figure 6: Sharing pair and merging pair (d_t = 5)

Consider a producing and consuming display 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 consuming 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 pinned in the buffer pool to support the continuous data delivery to the consuming 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 these two displays are being served by two different disk streams, then they are termed a *merging pair*. Figure 6 shows a sharing pair and a merging pair when d_t equals five. Note that the disk stream in support of D_2 services three simultaneous displays (D_2, D_3 and D_4) because the two adjacent sharing pairs have merged into one. In this figure, D_2 is a producing display while D_3 and D_4 are consuming displays. Assuming that there is no other request after D_4 that is d_t blocks behind, D_4 unpins the blocks that are pinned by D_2. If d_t is set to 2 then D_3 is both a consuming and an implicit producing display. It is consuming because it forms a group with D_2. It is an implicit producing because its presence is an absolute must to enable D_4 to share a single stream with D_2.

In Figure 6, D_1 and D_2 are a merging pair. They satisfy the concept of a group because they are less than 5 blocks apart (the pre-specified d_t value). This hint enables the BMDT replacement policy, see Section 2.4, to pin the blocks produced by D_1. After 4 time cycles, D_1 and D_2 become sharing pairs because they will start to share a disk stream (assuming both displays continue normal viewing). In fact, if a video is popular enough, BMDT may retain a large number of its consecutive blocks in the buffer pool because the algorithm constructs many sharing pairs of this video. When this happens, one single disk stream will support many displays of the same video. In sum, a merging pair becomes a sharing pair when all the blocks between this pair is pinned and available in buffer pool.

2.4 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 N be the size of buffer pool in terms of the number of blocks, d_s be the sum of the distances between all 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

$$d_s + S \leq N \tag{7}$$

must be satisfied to guarantee the continuous display for each client. The variable d_m does not appear in Equation 7 because those buffers occupied by merging pairs are not pinned (and not required for a continuous display). The upper bound of d_m is $N - d_s - S$. Whenever d_m approaches this bound (say within 2% of this limit), some buffers occupied by merging pairs will be unpinned 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

$$S \leq I \tag{8}$$

where I is the maximum number of disk streams supported by the available disk bandwidth with no sharing. Equation 8 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 (including the arrival of new clients and the VCR requests from existing clients), active displays ending and exiting the system, or merging pairs evolving to become sharing pairs.

When applied in a system configured according to the planner (see Section 2.7), BMDT 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 3.1). The BMDT algorithm is applied at the end of each cycle. The admission control is assumed implicitly 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. This disk stream could be supporting a display either in normal mode or in VCR mode.

2. Evolve as many of the merging pairs to sharing pairs and free the disk stream assigned to the consuming display while: a. all blocks between a merging pair are buffer resident; and b. Equation 7 is satisfied. Equation 7 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 clients by assigning a new disk stream to those displays that reference different clips while both Equations 7 and 8 are satisfied. These two equations ensure availability of sufficient resources to accommodate a new display.

4. Perform buffer replacement and allocation algorithm.

The processing of VCR operations and its impact on buffer management is further explained in Section 2.6. The buffer replacement and allocation algorithm in Step *4* is further detailed as follows. It decides victim pages that should be discarded when buffer replacement is necessary.

4.1. Unpin the buffers that have no potential for sharing.

This step unpins all the buffers that do not fall in between a merging/sharing pair, i.e., the likelihood of an active display obtaining data from these buffers sometimes in the future is slim.

4.2. While memory is over allocated, i.e., $(d_s + d_m + S > N)$, Do

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

ii. unpin all those buffers that fall in between this merging pair.

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

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

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

This step specifies the memory frame used by a stream to stage blocks from disk to memory.

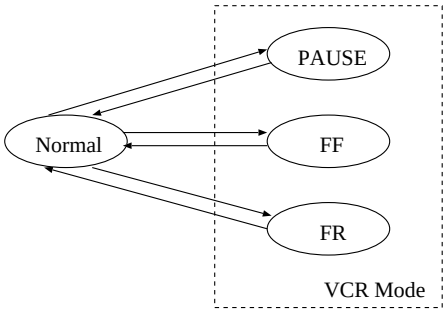


Figure 7: Mode transition diagram

2.5 User activity with VCR operations

Before describing how VCR operations are supported with buffer sharing, we analyze the user activity. Once a display is initiated, a client may change from normal mode to VCR mode, and vice versa. In normal mode, the video is retrieved and consumed at a prespecified rate. The client may switch to VCR mode using either pause, FF, or FR functionality. Once in VCR mode, the display may switch back to normal display mode. This mode transition may be repeated many times until the display ends. Figure 7 shows the possible modes of operation.

With the pause VCR functionality, the system is not required to display data. However, with both FF and FR, the browsing with FF/FR effect must be supported. There are two general paradigms to implement VCR functionalities: either (1) a single, normal-speed movie is processed accordingly in real-time during playback time, termed *online* processing [CKY94, DSSJT94, SV95], or (2) a clip is pre-processed by the content provider with separate files for FF and FR viewing, termed *offline* processing [And96, GZK$^+$97].

With the online approach, the system might either skip segments or subtract frames. Segment skipping [CKY94] skips a fixed number of blocks to achieve the desired playback rate. For example, to provide five times fast-forward, this method displays one block out of five consecutive blocks. The advantage of this approach is that it does not need extra disk bandwidth to support various fast rates. However, it results in unusual viewing effect because it skips blocks, not frames. Frame subtracting [DSSJT94, SV95] requires extra disk bandwidth to do frame reduction. In order to provide five times fast-forward, the server retrieves five data blocks during one cycle, and within each block, it transmits every fifth frame and drops the other four frames.

We choose the offline processing approach proposed in [And96, GZK$^+$97]. The basic idea is to implement separate fast-forward and fast-rewind videos by selecting and pre-processing frames. These clips are termed either trick or express (XPRS) files. For example, to create a five times fast-forward video, every fifth frame is selected from the original movie before compression. This collection of frames is encoded in the regular manner (e.g., using MPEG) and stored in a separate video file. In order to switch between the different versions, a method of jumping to the appropriate location in each clip must be provided. These locations are referred to as Random Access Points (RAPs) [And96]. Their cross-references, which map between the frames of different versions, must be maintained (see [GZK$^+$97] for details). When a client switches between normal mode and VCR mode, the cross-reference is used to determine the new display point. For example, if FF is invoked during a normal display, then the system follows the cross-reference to find the appropriate location in the FF version, and start to retrieve data from that location of the FF file. The client will receive data blocks of the FF file in support of the FF viewing. When the client transitions back to the normal playback, again the system will follow the cross-reference to find the appropriate location in the normal version, and stream the normal data blocks to the client from that point. In this study, we assume that the FF (or FR) video has the

same disk bandwidth requirement as the original video.

The main challenge posed by the off-line processing approach is as follows. A display that transitions from normal display might be a member of either a sharing or a merging pair. As a member of a sharing pair, it might be the producing display. In these cases, the buffers and disk streams must be managed intelligently in order to: a) prevent hiccups, b) minimize the delay observed by the client transitioning from normal display to a VCR operation, and c) maintain a balance between available memory and disk bandwidth in order to prevent either from becoming system bottleneck. In the following, we address items (a) and (c). Item (b) is investigated experimentally in Section 3.1. The obtained results show that using optimal d_t value minimizes observed delays.

2.6 Processing VCR requests

In this section, we describe the extensions to the BMDT algorithm (step 3) in support of a VCR request. Our proposed algorithm utilizes the data blocks available in the buffer pool to facilitate the VCR operations. The basic idea is that when a display invokes a VCR function, one disk stream may be assigned solely to this display to provide VCR browsing. When this display transitions back to normal mode, if the returning point falls on the data block available in buffer pool, then the disk stream used for VCR browsing will be freed and the display will be served from buffers. Even if the returning point cannot be found in buffer, it is still possible for this display to share a disk stream with other displays in the future using buffer sharing. Assuming that a client spends most of its time in normal playback with short invocations of VCR operations, at any given point in time, the average number of clients invoking VCR operations is only a small fraction of the total number of clients. Therefore applying buffer sharing on VCR streams is not expected to provide significant gains and is eliminated from further consideration.

In order to describe the details of how the system manages its buffer space with VCR operations, we introduce the concepts of *isolated* and *associated* displays. The former refers to a display that does not share its disk stream with others. The later refers to many displays that share a single disk stream. Note that an associated display might retrieve its data from either a disk stream or the buffer pool, i.e., it might be either a producing, a consuming, or dual role display. Consider the example in Figure 6 where D_2, D_3 and D_4 are all associated displays: D_2 is served using a disk stream (producing display) while D_3 and D_4 are served using the buffer pool (consuming displays).

2.6.1 From normal to VCR mode

For an isolated display, say D_i, if it switches from normal playback to pause, the system frees its disk stream in order to terminate production of data on behalf of D_i. With either FF or FR, since D_i is assigned a disk stream, this disk stream will be used to retrieve data from the XPRS file in support of either FF or FR browsing. For buffer management, if D_i is not part of a merging pair, then the system can free its buffer space (one block). Otherwise (D_i is a member of a merging pair), there are three cases: (1) D_i is a producing display, (2) D_i is a consuming display, and (3) D_i plays a dual role and is both a consuming and an implicit producing display. In case (1), the system unpins the data blocks between D_i and its succeeding display D_{i+1}. In case (2), the system unpins the data blocks between D_i and its preceding display D_{i-1} if and only if there is no other request D_{i+1} that forms a sharing pair with D_{i-1}. In case (3), there are two conditions depending on the distance between D_i's preceding display D_{i-1} and D_i's succeeding display D_{i+1}. If the distance exceeds the threshold d_t, then all the data blocks between D_{i-1} and D_{i+1} are unpinned, disallowing bridging. The result is shown in Figure 8.b. On the other hand, if the distance is within the threshold d_t, then the data blocks

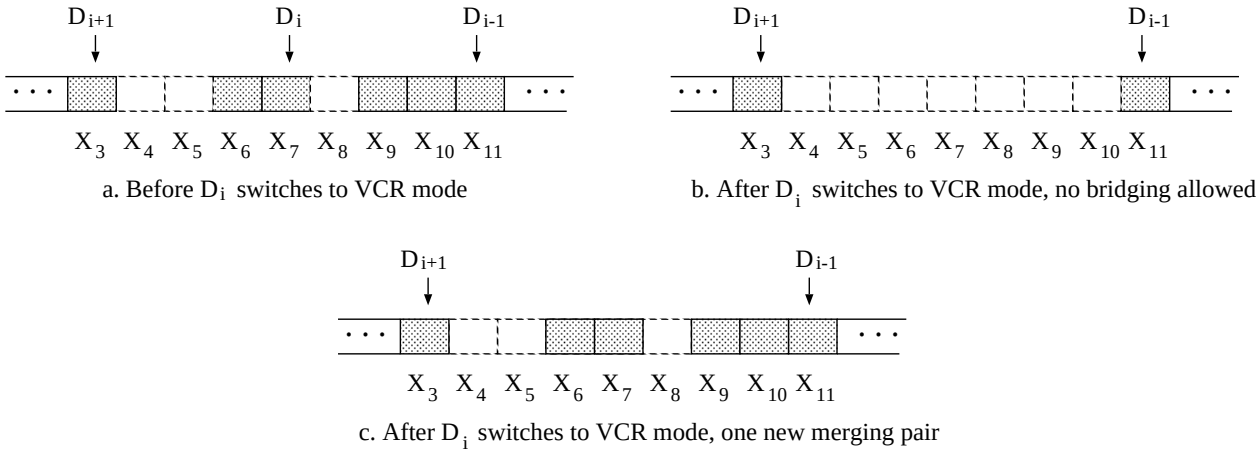


Figure 8: Isolated display: from normal to VCR (case 3)

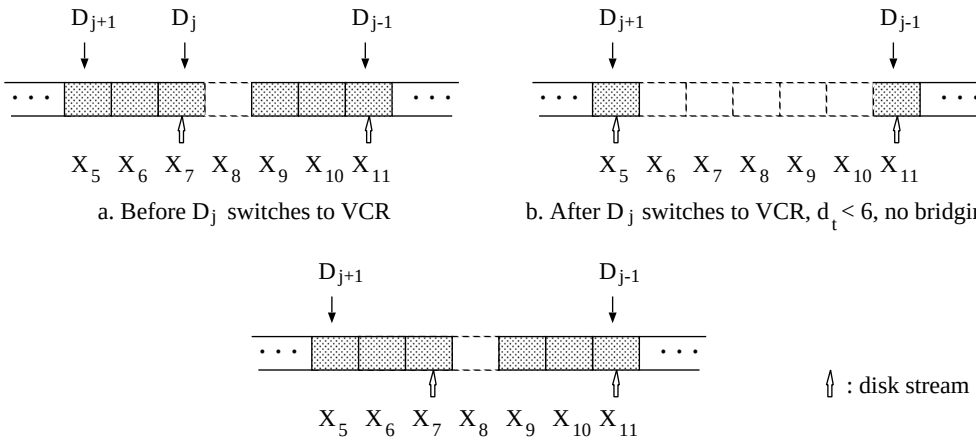
between D_i and D_{i+1} remain pinned, because D_{i-1} and D_{i+1} can form a new merging pair. The result is shown is Figure 8.c, where X_6 and X_7 remain pinned.

For an associated display D_j, the transition of D_j from normal to VCR mode may affect the assignment of disk streams to other displays since D_j shares one disk stream with other displays by definition. Once again, three possible cases arise: (1) D_j is a producing display only, (2) D_j is a consuming display only, and (3) D_j plays a dual role and is both an implicit producing display and a consuming display. In all three cases, if D_j switches to FF/FR, an individual disk stream is assigned to D_j to provide FF/FR browsing. Other displays are impacted in a different manner depending on which case is encountered. In case (1), the buffers between D_j and its succeeding display D_{j+1} are unpinned, and the disk stream serving D_j previously is used to serve D_{j+1}. In case (2), the buffers between D_j and its preceding display D_{j-1} are unpinned with no modification to the disk stream assignment. Case (3) is more complicated. There could be three conditions: (3a) D_j is the consuming display of a merging pair and the producing display of a sharing pair, see Figure 9.a, (3b) D_j is the consuming display of a sharing pair and the producing display of a merging pair, see Figure 10.a, and (3c) D_j is the consuming and an implicit producing display of sharing pairs, see Figure 11.a. Case (3a) is shown in Figure 9 where the disk stream serving D_j is used to serve D_{j+1}. If the distance between its succeeding display D_{j+1} and preceding display D_{j-1} is within d_t, then the buffers between D_j and D_{j+1} remain pinned because D_{j-1} and D_{j+1} can form a new merging pair. Otherwise all the data blocks between D_{j-1} and D_{j+1} are unpinned because bridging between D_{j-1} and D_{j+1} is not allowed. In case (3b), there is no change in disk stream assignment. If the distance between D_{j-1} and D_{j+1} is within d_t, then no buffer is unpinned since D_{j-1} and D_{j+1} can form a new merging pair; Otherwise all the blocks between D_{j-1} and D_{j+1} are unpinned. This is shown in Figure 10.

In case (3c), if the distance between D_{j-1} and D_{j+1} is within d_t, then no change is made to the current buffer assignment, see Figure 11. Otherwise a new disk stream must be assigned to serve D_{j+1}, and all the buffers between D_{j-1} and D_{j+1} are unpinned, see Figure 11.

2.6.2 From VCR to normal mode

Once D_k switches from FF/FR to normal mode, the system: a) computes D_k's reference frame for the normal display which maps to a block, termed D_k's reference block, and b) frees the disk stream in support of its FF/FR browsing. If D_k's reference frame for normal display falls on the data block available in buffer pool,

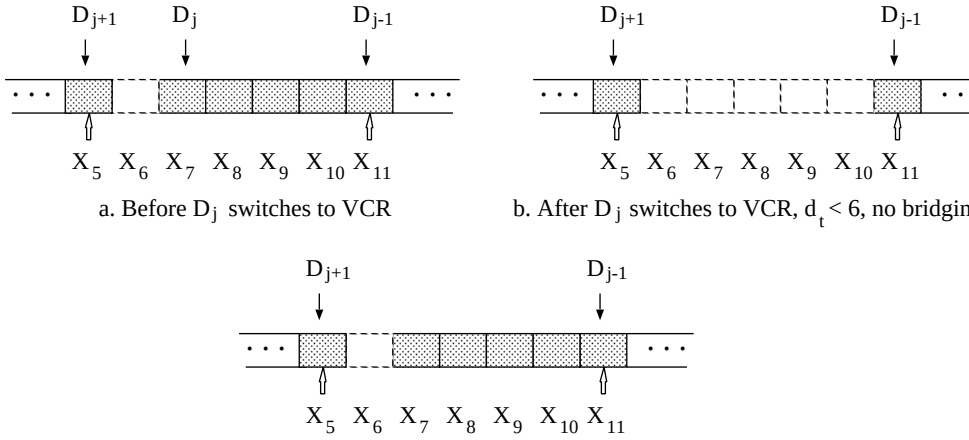


a. Before D_j switches to VCR

b. After D_j switches to VCR, $d_t < 6$, no bridging allowed

c. After D_j switches to VCR, $d_t >= 6$, one new merging pair

Figure 9: Associated display: from normal to VCR (case 3a)

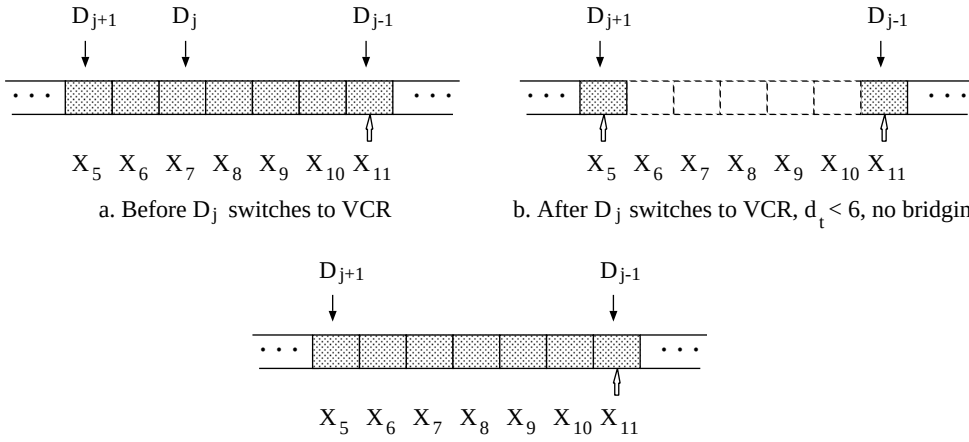


a. Before D_j switches to VCR

b. After D_j switches to VCR, $d_t < 6$, no bridging allowed

c. After D_j switches to VCR, $d_t >= 6$, one new merging pair

Figure 10: Associated display: from normal to VCR (case 3b)



a. Before D_j switches to VCR

b. After D_j switches to VCR, $d_t < 6$, no bridging allowed

c. After D_j switches to VCR, $d_t >= 6$, one new merging pair

Figure 11: Associated display: from normal to VCR (case 3c)

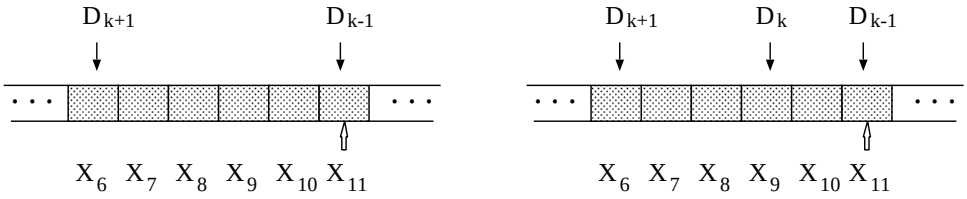


Figure 12: Back to normal mode: on sharing pair

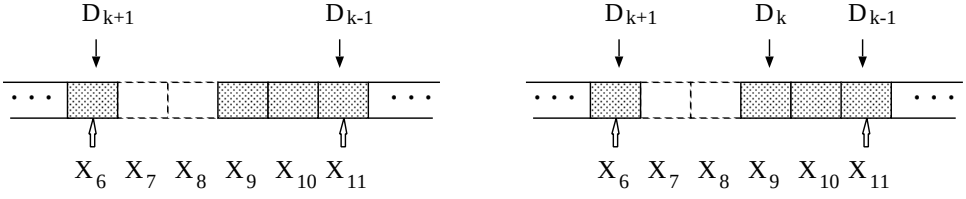


Figure 13: Back to normal mode: on merging pair

then its normal display begins immediately by accessing data from memory, and no extra disk stream is required. This is shown in Figures 12 and 13. In Figure 12 where the reference block falls in between a sharing pair, D_k forms one sharing pair with D_{k+1} and one sharing pair with D_{k-1}, and no extra disk stream and buffer space is required. In Figure 13, the reference block falls on a block between a merging pair, and as a result, D_k form one sharing pair with D_{k-1} and one merging pair with D_{k+1}.

If D_k's reference block is not buffer pool resident, then a new disk stream is assigned to D_k to facilitate its normal display. There could be three cases: (1) D_k is not part of a merging pair, (2) D_k becomes part of one merging pair and (3) D_k becomes a part of two merging pairs. Case (1) occurs when D_k's reference block is not d_t blocks apart relative to other displays referencing the same video. Case (2) implies that D_k's reference block is d_t blocks apart from one of the adjacent displays (either D_{k-1} or D_{k+1}), while case (3) means D_k's reference frame is d_t blocks apart from both D_{k-1} and D_{k+1}. In case (3), if D_{k-1} and D_{k+1} are originally a merging pair, the return of D_k will not impact buffer blocks in-between them, as illustrated in Figure 14. Note that although a new disk stream may be needed to support the normal display of D_k when it returns from VCR mode, the disk stream may be freed later when the merging pairs involving D_k evolve to become a sharing pair.

This discussion completes the BMDT algorithm. However, on its own, BMDT does not prevent the undesirable condition shown in Figure 3.b. If the buffer pool is too small, it could still form a bottleneck; if

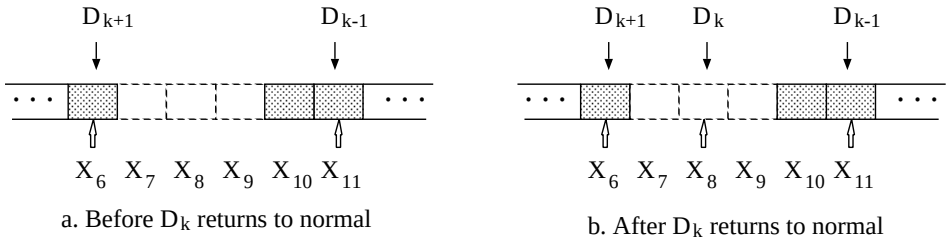


Figure 14: D_k switches to normal display and forms new merging pairs

the buffer pool is too large,it might waste some buffer space. We need to quantify how much buffer space is required. Moreover, in the presence of VCR operations, if the required disk bandwidth is not available when a VCR mode transition is invoked, the client may experience some wait time. Thus, we also need to quantify how much disk bandwidth is required to prevent these delays.

2.7 Configuration planner

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

The input to the planner includes the arrival rate of requests, the access frequency distribution, the number of videos available, the system utilization factor, the VCR model, and the distance threshold (d_t). Among them, the access frequency distribution specifies the popularity of each video in the system, and the VCR model describes how frequently the VCR operations are performed. However, except d_t, all input parameters reflect the physical characteristics of the system. Theoretically the planner could take any value of d_t as its input. Section 2.2 derived the optimal d_t value. The output of the planner is the expected number of buffer blocks and the expected number of disk streams.

We first analyze the memory and disk configuration in a system without VCR operations, then we compute the impact of VCR operations on the system configuration and obtain the analytical model for an interactive system.

Assume the request arrival process follows Poisson distribution with rate[3] α. Let V denote the number of different videos in the system. Assume the length of each video is l time units, and let β be $\frac{1}{l}$. Upon arrival of a new client to the system, she 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 α_j where $\alpha_j = p_j\alpha$ and $\sum_{j=1}^{V} \alpha_j = \alpha$. Then the expected number of displays that the system must support is

$$M = \frac{\alpha}{\beta} \tag{9}$$

and the expected number of displays that reference video j is

$$m_j = \frac{\alpha_j}{\beta} \tag{10}$$

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

$$p_{s_j} = 1 - e^{-\alpha_j \cdot d_t \cdot t_p} \tag{11}$$

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 11 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

[3]We reserve λ to model VCR operations.

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

$$P[I_j = k] = \begin{pmatrix} m_j - 1 \\ k - 1 \end{pmatrix} \cdot p_{s_j}^{m_j - k} \cdot (1 - p_{s_j})^{k-1} \tag{12}$$

and the expected value of I_j can be expressed as

$$E[I_j] = \sum_{k=1}^{m_j} k \cdot P[I_j = k] \tag{13}$$

$E[I_j]$ is the expected number of disk streams which are displaying video j. The total number of expected disk streams in a system employing BMDT is the sum of $E[I_j]$ for each video clip:

$$M' = \sum_{j=1}^{V} E[I_j] \tag{14}$$

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

$$W = M \tag{15}$$

The buffer requirement with BMDT can be estimated 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

$$P[d_j = k] = (1 - e^{-\alpha_j \cdot k \cdot t_p}) - (1 - e^{-\alpha_j \cdot (k-1) \cdot t_p}) \tag{16}$$

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

$$P[d_j = k | d_j \leq d_t] = \frac{P[d_j = k \cap d_j \leq d_t]}{P[d_j \leq d_t]} \tag{17}$$

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 11 we know $P[d_j \leq d_t] = p_{s_j}$. Then Equation 17 can be rewritten as

$$P[d_j = k | d_j \leq d_t] = \frac{P[d_j = k]}{p_{s_j}} \tag{18}$$

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

$$E[d_j] = \sum_{k=1}^{d_t} k \cdot P[d_j = k | d_j \leq d_t] \tag{19}$$

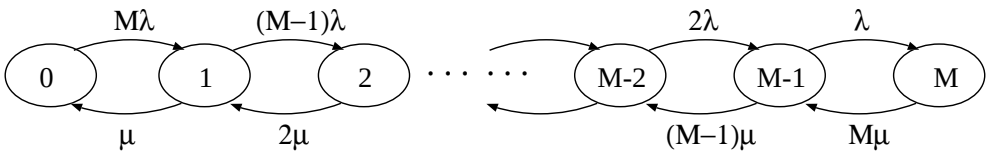


Figure 15: State-transition-rate diagram for an $M/M/\infty//M$ system

From Equation 13 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

$$b_j = (m_j - E[I_j]) \cdot E[d_j] \tag{20}$$

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

$$b = \sum_{j=1}^{V} b_j \tag{21}$$

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

$$W' = b + M' \tag{22}$$

Equations 14 and 22 estimate the expected number of disk streams and the expected number of buffer blocks in a system without VCR operations. Now we analyze the impact of VCR operations. From Section 2.5, a client may switch between normal mode and VCR mode repeatedly. For example, the user may spend T_n time on normal display, and then change to VCR mode for T_v time and then come back to normal display. The random variables T_n and T_v are assumed to be exponentially distributed with means $\frac{1}{\lambda}$ and $\frac{1}{\mu}$ respectively. Assume the total number of clients in VCR mode is k. Given the total number of clients M, and that we want to configure a system with no waiting time when VCR operations are invoked. This is equivalent to a $M/M/\infty//M$ system. The birth-death coefficients are as follows:

$$\lambda_k = \begin{cases} \lambda(M-k) & 0 \le k \le M \\ 0 & otherwise \end{cases} \tag{23}$$

$$\mu_k = k\mu \qquad k = 1, 2, ... \tag{24}$$

The state-transition-rate diagram is shown in Figure 15. This system is solved in [Kle75] and the expected number of users in the VCR mode is given by

$$E[k] = \frac{M\lambda/\mu}{1 + \lambda/\mu} \tag{25}$$

For a specific video j, the expected number of clients in VCR mode is

$$vcr_j = E[k] \cdot p_j \tag{26}$$

For each of these vcr_j displays, before it enters the VCR mode, it could be either isolated or associated, with the probability of $p_{isolated\_j}$ and $p_{associated\_j}$ respectively. By definition we obtain

$$p_{associated\_j} = p_{s_j} \tag{27}$$

and

$$p_{isolated\_j} = 1 - p_{associated\_j} \tag{28}$$

We first consider the effect of VCR operations on disk bandwidth requirement. If an isolated display enters VCR mode, the number of disk streams supporting normal displays will be reduced by one. We denote this effect as

$$\delta_1 = \sum_{j=1}^{V} vcr_j \cdot p_{isolated\_j} \tag{29}$$

For an associated display, say D_i, its invokation of a VCR mode will not effect the number of disk streams for normal display, except when the distance between D_{i-1} and D_i (denoted d_1) is within d_t, and the distance between D_{i+1} and D_i (denoted d_2) is within d_t, but the distance between D_{i-1} and D_{i+1} ($d_1 + d_2$) exceeds d_t, as illustrated in Figure 11.b. The probability of this scenario is

$$p_{exceed\_j} = P[d_1 \leq d_t \cap d_2 \leq d_t \cap (d_1 + d_2) > d_t] \tag{30}$$

which can be rewritten as

$$p_{exceed\_j} = \sum_{i_1=1}^{d_t} (P[d_1 = i_1] \cdot \sum_{i_2=d_t-i_1+1}^{d_t} P[d_2 = i_2]) \tag{31}$$

where

$$P[d_1 = i_1] = (1 - e^{-\lambda_j \cdot i_1 \cdot t_p}) - (1 - e^{-\lambda_j \cdot (i_1-1) \cdot t_p}) \tag{32}$$

$$P[d_2 = i_2] = (1 - e^{-\lambda_j \cdot i_2 \cdot t_p}) - (1 - e^{-\lambda_j \cdot (i_2-1) \cdot t_p}) \tag{33}$$

The effect of the above scenario on disk bandwidth requirement is denoted as

$$\delta_2 = \sum_{j=1}^{V} vcr_j \cdot p_{exceed_j} \tag{34}$$

If a display enters VCR mode for PAUSE, then no disk stream is required to support PAUSE; otherwise if it is for FF/FR (assume the probability is p_F), then one disk stream is needed to provide FF/FR browsing. This requirement is denoted as

$$\delta_3 = \sum_{j=1}^{V} vcr_j \cdot p_F \tag{35}$$

Now we obtain the expected number of disk streams in an interactive system as

$$M'' = M' - \delta_1 + \delta_2 + \delta_3 \tag{36}$$

The analysis of buffer requirement of an interactive system can be done as follows. We first compute how many shared buffers will be affected when a display D_i switches to VCR mode. If D_i is isolated, there will be no effect on shared buffers. If D_i is associated, the number of shared buffers will be reduced by $E[d_j]$ except in the scenario shown in Figure 11.b and Figure 11.c. In Figure 11.b, the number of reduced shared buffers is $2 \cdot E[d_j]$ since two sharing pairs are broken up; while in Figure 11.c, no shared buffers will be affected. We already know the probability of the scenario in Figure 11.b is p_{exceed_j}. To summarize the effect on shared buffers, we have

$$\delta_4 = \sum_{j=1}^{V} (vcr_j \cdot (p_{associated\_j} - p_{s_j}^2) \cdot E[d_j] + 2 \cdot vcr_j \cdot p_{exceed\_j} \cdot E[d_j]) \tag{37}$$

Then the number of shared buffers in an interactive system is

$$b'' = b - \delta_4 \tag{38}$$

and finally the total number of buffers is

$$W'' = b'' + M'' \tag{39}$$

3 Evaluation of Controlled Buffer Sharing

For experimental purposes, we assume an open simulation study that employs a Poisson arrival rate of requests. Upon the arrival of a new client, the system will reject this request if there is insufficient resources to support its display. Otherwise, the client is admitted and its display is initiated. If this client issues VCR operations and there is insufficient resources to satisfy its request then the client waits until resources become available (neither the client nor the request is rejected).

In our experiments, the underlying repository consists of 100 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 Zipf distribution[4]. The default parameter for Zipf in our experiments is 0.271 which reflects an empirical access pattern based on theater ticket sales [DSS94a]. Both the algorithms and presented results are governed by the ratio of $I_\$$ and $M_\$$. This ratio is more important than the absolute values of these parameters. For the purposes of this study, we assumed a ratio of 16 ($\frac{I_\$}{M_\$}$) This would correspond to the current costs of $1 per MB of DRAM and $16 per stream supported by a Seagate ST19171WB disk drive [GSZ95]. In the future, these prices are expected to fall. The presented results continue to apply as long as the relative decrease in both $I and $M is identical. Section 3.2 discusses the impact of different ratios between $I_\$$ and $M_\$$.

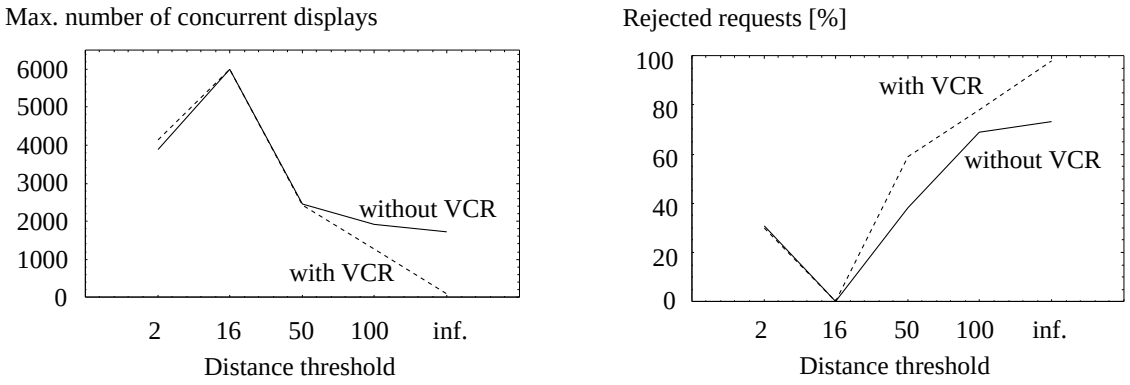
In experiments involving VCR operations, the time duration of a client staying in normal mode and VCR mode is exponentially distributed with the default means of 10 minutes and 20 seconds respectively. The default value of arrival rate in the experimental study is 50 requests per minute. The system utilization factor is set to 90%, i.e., on the average, the system is busy 90% of the time.

3.1 Performance and cost comparison of buffer sharing schemes

This section presents results to demonstrate (a) the importance of d_t and (b) the parameters of BMDT should be set according to those computed by the configuration planner. We configured the system using the planner of Section 2.7 and the default system parameters. Next, we measured run-time performance of the system with BMDT when its distance threshold (d_t) is set to $2, 16, 50, 100$ and infinite (which means no threshold is imposed on BMDT). For each experiment, we measured the maximum number of concurrent displays and the percentage of rejected requests with and without VCR functionality. (Obviously, the configuration of the system with and without VCR functionality is different because of the configuration planner.) The obtained results are shown in Figures 16.a and 16.b. We also measured the average wait time of VCR operations in interactive systems and is reported in Figure 17.a. It is obvious that the best performance is achieved only when the run time threshold is the same as the value used to configure the system (the CBS scheme). In other cases where either no threshold is enforced at run time or the improper threshold value is used, the performance is degraded significantly. The results also demonstrate that enforcing a d_t value at run time is not enough, the system must be configured with the appropriate amount of resources that are in accord with the chosen d_t value. The CBS scheme links the run time buffer management with the system configuration in order to prevent bottlenecks.

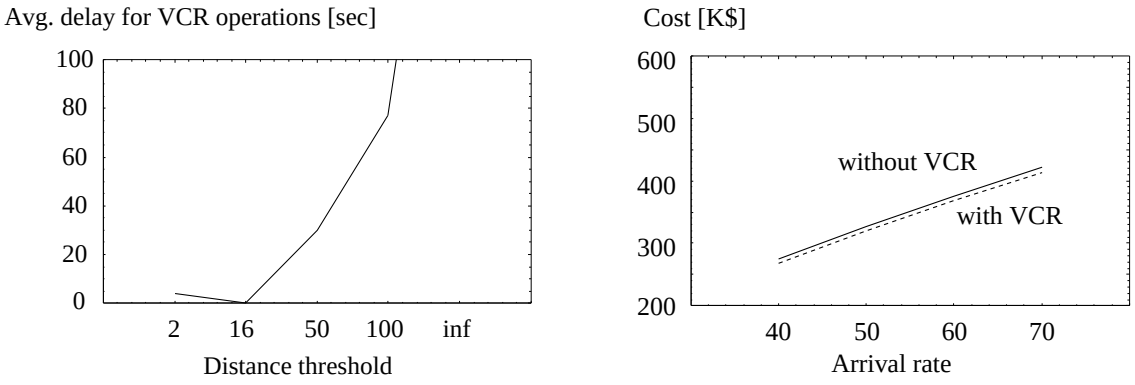
The amount of disk bandwidth required to support VCR functionalities is marginal. To illustrate, Figure 17.b shows the cost of a system configured with and without VCR functionality for different arrival rates. This was done using the configuration planner of CBS. Of course, these results are based on two assumptions. First, the system designer can specify the average duration of VCR functionalities. Second, system cost is

[4] 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.



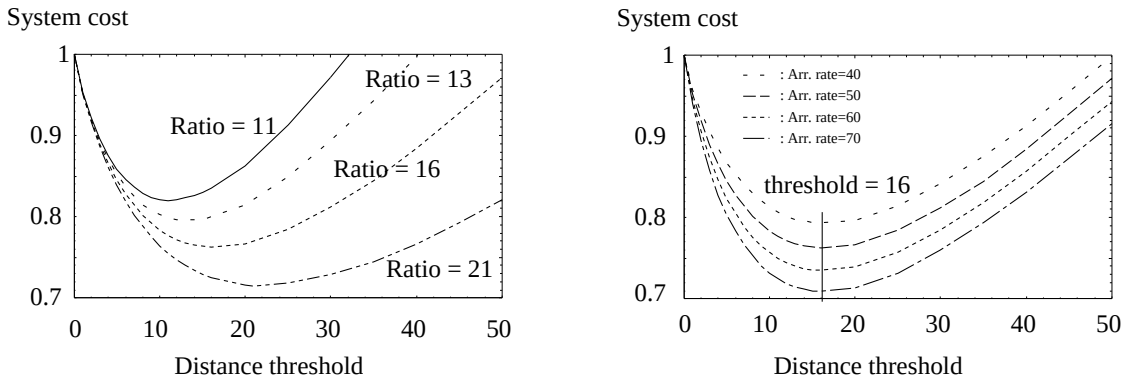
(a) Throughput (b) Percentage of rejected requests

Figure 16: Throughput and percentage of rejected requests as a function of different d_t values



(a) Average Wait Time for VCR Operations (b) Cost with and without VCR Operations

Figure 17: Average VCR wait time for different d_t values, cost with and without VCR operations



(a) Cost as a function of $\frac{I_\$}{M_\$}$, termed Ratio

(b) Cost as a function of λ

Figure 18: System cost

based on disk bandwidth; in particular, we ignore the extra cost of disk space required to store XPRS files. This is reasonable because the number of simultaneous displays supported by a disk is typically a fraction of the number of video clips that it can store.

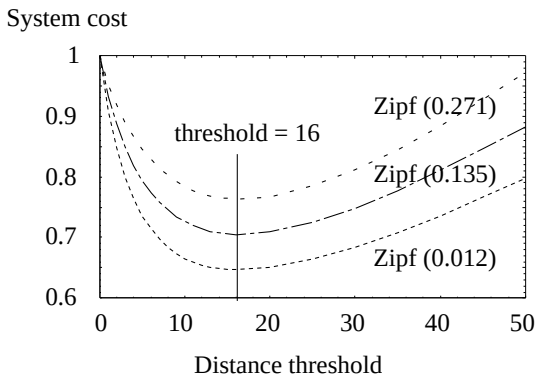
3.2 Characteristic of the optimal d_t

The CBS framework has a higher payoff with larger $I_\$$ and $M_\$$ ratios. To illustrate, in Figure 18.a, we looked at different $I_\$$ and $M_\$$ ratios. For each ratio, we configured BMDT with d_t values ranging from 0 to 50. One of the chosen values was the d_t value that corresponds to the $I_\$$ and $M_\$$ ratio. Next, we measured the system cost per stream for each ratio and normalized it relative to the case where d_t equals to zero (no sharing), marked as 1. This normalization enables us to measure the percentage savings provided by CBS. In Figure 18.a, the lowest system cost is achieved when d_t is 21, 16, 13 and 11. Moreover, the system cost decreases by approximately 30% when d_t equals 21. With smaller d_t values, the reduction in cost attributed to buffer sharing starts to decrease.

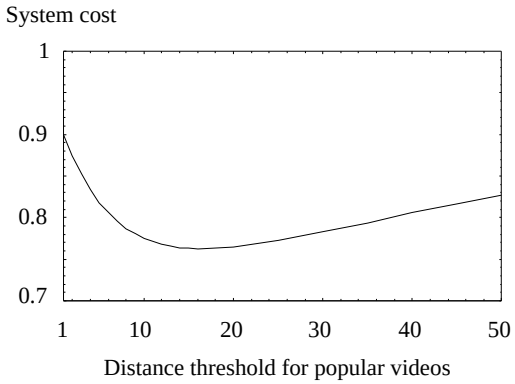
Figure 18.b shows the system cost under different arrival rates and distance threshold values when $\frac{I_\$}{M_\$}$ is 16. We tried four different arrival rates (40, 50, 60 and 70). The result in Figure 18.b shows the lowest system cost is achieved when $d_t = 16$ for every arrival rate, and demonstrates that the optimal d_t value is independent of the arrival rates. Once again, the results are normalized relative to the scenario where buffer sharing is not allowed (d_t equals zero). Note that a higher system load results in a lower cost per stream because buffer sharing amortizes costs across a larger number of concurrent displays.

We also study the relationship between the optimal d_t value and the access frequency distribution ($\frac{I_\$}{M_\$}$ equals 16). We fixed the arrival rate at 50 requests per minute and analyzed three different Zipf distributions with parameters 0.271, 0.135, and 0.012. (Zipf with parameter 0.271 matches well with the empirical data on video rental.) The optimal d_t is independent of the access frequency distribution, see Figure 19.a. With a more skewed distribution, the utilization of buffers increases, resulting in a larger number of simultaneous displays. This amortizes system costs, resulting in a lower cost per stream.

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 system cost. To demonstrate, we conducted the following experiments. First, we identified the 10 most popular videos in the system. For the remaining 90 videos, we fixed d_t at 16 which is

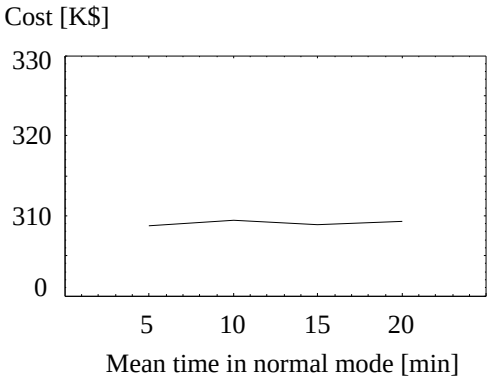


(a) Impact of access frequency distribution on d_t

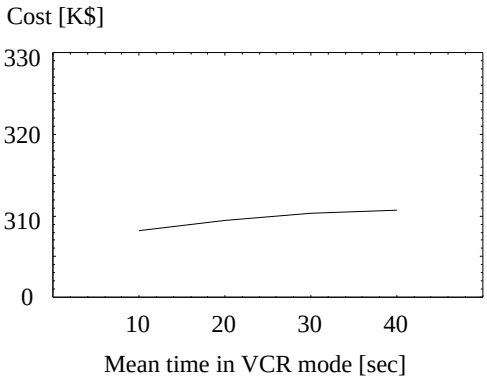


(b) Optimal d_t and video popularity

Figure 19: Value of d_t is independent of both access frequency distribution and video popularity



(a) Cost sensitivity to mean time in normal display



(b) Cost sensitivity to mean time in VCR mode

Figure 20: Mean time in both display and VCR mode has marginal impact on system cost

the optimal value. Then for each number from 1 to 50, we configured 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 19.b shows the system cost vs. the threshold applied on the popular videos. System cost is normalized relative to the no sharing scenario. It is clear that system cost is minimized when d_t equals 16. This result shows the optimal d_t value is independent of the access frequency of the individual video.

3.3 Sensitivity to the frequency of VCR operation

The following experiments are conducted to evaluate the CBS scheme in interactive systems with different frequencies of VCR operations. First we fixed the mean time in VCR mode at 20 seconds, while varying the mean time in normal mode from 5 minutes to 20 minutes. For each experiment, we first configured the system according to the configuration planner, and then run the simulation for 100 hours. We measured the actual number of disk streams and buffer blocks in each experiment and then computed the system cost (shown in Figure 20.a). Similar experiments were conducted with fixed time duration in normal mode (10 minutes) and different time durations in VCR mode (from 10 seconds to 40 seconds). The results are shown in Figure 20.b. In both Figures, the system cost is insensitive to the change in VCR operation frequency. This is because the

CBS scheme absorbs the variation in VCR behavior by applying buffer sharing dynamically when a mode transmission occurs. Therefore, the system will not be degraded to a non-sharing system which costs more.

4 Related Work

There are three main approaches to reduce disk bandwidth requirement in a continuous media server:

- buffer sharing [DDM$^+$95, KRT94, KRT95b]: the idea here is that if one stream for a clip lags another stream for the same clip by only a short time interval, then the system could retain the portion of the clip between the two in buffers. The lagging stream would read from the buffers and not have to read from disk.
- batching of requests [DSST94, DSS94b, OBRS94, WSY95]: in this method, requests are delayed until they can be merged with other requests for the same clip. The merged streams then form one physical stream from the disk and consume only one set of buffers. Only on the network will the stream split at some point for delivery to the individual display station.
- adaptive piggybacking [GLM95]: this approach adjusts the streams for the same video to go either 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.

The focus of this study is on buffering. It ignores the other two approaches in order to maintain both its focus and a manageable length.

The previous studies of buffer sharing can be classified based on how they manage the memory frames. One approach manages memory at the granularity of blocks [KRT95a, ORS95, ORS96] while the other uses bridging [DDM$^+$95, DS96, RZ95]. The main contribution of this paper is a novel framework that subsumes bridging as a component to minimize the cost per simultaneous stream supported by the system. This framework is termed Controlled Buffer Sharing, CBS. It provides an answer to three research questions that were not addressed by previous studies: a) How much memory should be traded for disk bandwidth to realize a cost effective system configuration? b) What is the overhead of supporting VCR operations with buffer sharing? c) what system and application parameters impact a buffer sharing based on bridging.

A difference between how bridging is employed with BMDT as compared to [RZ95, DS96] is their support of VCR functionality. One study [RZ95] suggests that VCR operations can be supported based on the data available in memory. While this requires no additional resources, it can only provide limited VCR support. For example, the time duration of a VCR operation is restricted, and the type of VCR operation could be limited. The other study [DS96] focuses on pause and resume VCR operations and suggests the following approach: the system should reserve a small amount of buffer space on behalf of paused display to cover its distance to the sharing stream. This is different than our proposed design that considers all VCR functionalities.

5 Conclusions and Future Research Directions

This study presents CBS as a framework that meets the performance objectives of an application while minimizing the cost per simultaneous stream supported by a server. Our configuration planner assumes a Poisson distribution for request arrivals. The design of this component must be re-visited when this assumption is

violated. Its run time buffer pool manager, BMDT, employs bridging to merge two displays that reference the same clip and are d_t blocks apart. One insight of this study is that the value of d_t is determined by market forces (cost of memory and disk bandwidth) and is independent of a clip's access frequency. Hence, this framework is independent of the popularity characteristics of the clips that constitute the repository. Our experimental results validate this claim. Moreover, they demonstrate that the framework becomes more cost effective with larger d_t values (increasing ratio of disk bandwidth cost and memory cost, i.e., $\frac{I_\$}{M_\$}$).

In the future, we intend to extend this framework to proxy cache servers in support of both data sharing and system sizing. The basic idea is as follows. With Internet, proxy cache servers are deployed to minimize both the latency incurred by a client and the bursty characteristics of the underlying network. These servers may cache continuous media clips along with other data types, e.g., images, web pages, etc. A proxy server may cache either the entire clip or a portion of it [GS94, SRT99, RHYE99]. Several clients may reference the same clip at the same time and the proxy server must decide what to cache. With partial caching, it may decide to maintain a window (using bridging) to enable several clients to merge and share a single stream from the original server. Moreover, a proxy cache server can decide to make this window disk resident. In essence, it can trade disk bandwidth for memory internally. This minimizes the bandwidth required from the underlying network and the original server. These tradeoffs and design of techniques to manage system resources constitute our future research directions.

References

[And96] D. Andersen. A Proposed Method for Creating VCR Functions Using MPEG Streams. In *Proceedings of the 12th International Conference on Data Engineering*, Feb. 1996.

[CAF+91] S. Christodoulakis, N. Ailamaki, M. Fragonikolakis, Y. Kapetanakis, and L. Koveo. An Object Oriented Architecture for Multimedia Information Systems. In *Proceedings of IEEE Data Engineering*, Sept. 1991.

[CKY94] M. Chen, D.D. Kandlur, and P.S. Yu. Support for Fully Interactive Playout in a Disk-Array-Based Video Server. In *Proceedings of the ACM Multimedia*, Oct. 1994.

[DDM+95] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and Caching in Large-Scale Video Servers. In *Proc. of COMPCON*, 1995.

[DS96] A. Dan and D. Sitaram. Buffer management policy for an on-demand video server. *U.S. Patent No. 5572645*, November 1996.

[DSS94a] 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.

[DSS94b] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In *Proc. of the 2nd ACM International Conference on Multimedia*, October 1994.

[DSSJT94] J.K. Dey-Sircar, J.D. Salehi, J.F.Kurose, and D. Towsley. Providing VCR Capabilities in Large-Scale Video Servers. In *Proceedings of the ACM Multimedia*, Oct. 1994.

[DSST94] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley. Channel Allocation under Batching and VCR Control in Mov ie-On-Demand Servers. Technical Report RC19588, IBM Research Report, Yorktown Height, NY, 1994.

[GG97] J. Gray and G. Graefe. The five-minute rule ten years later and other computer storage rules of thumb. *Sigmod Record*, 26(4), December 1997.

[GLM95] L. Golubchik, J. C.-S Lui, and R. R. Muntz. Reducing i/o demand in video-on-demand storage servers. In *Proceedings of the ACM SIGMETRICS and Performance '95*, pages 25–36, 1995.

[GM98] S. Ghandeharizadeh and R. Muntz. Design and Implementation of Scalable Continuous Media Servers. In *Parallel Computing*, pages 91–122, 1998.

[GS94] S. Ghandeharizadeh and C. Shahabi. On Multimedia Repositories, Personal Computers, and Hierarchical Storage Systems. In *ACM Multimedia*, 1994.

[GS00] Jim Gray and Prashant J. Shenoy. Rules of thumb in data engineering. In *ICDE*, pages 3–12, 2000.

[GSZ95] S. Ghandeharizadeh, J. Store, and R. Zimmermann. Techniques to Quantify SCSI-2 Disk Subsystem Specification for Multimedia. Technical Report 95-610, University of Southern California, 1995.

[GZK+97] S. Ghandeharizadeh, R. Zimmermann, S.H. Kim, W. Shi, and J. Al-Marri. Scalable Video Browsing Techniques for Intranet Video Servers. In *Proceedings of the 7th Annual Workshop on Information Technologies & Systems*, pages 209–218, Atlanta, Georgia, December 13-14, 1997.

[GZS+97] 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.

[Kle75] L. Kleinrock. *Queueing Systems Volume I: Theory*, page 107. Wiley-Interscience, 1975.

[KRT94] M. Kamath, K. Ramamritham, and D. Towsley. Buffer Management for Continuous Media Sharing in Multim edia Database Systems. Technical Report 94-11, University of Massachusetts, Feb. 1994.

[KRT95a] 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.

[KRT95b] M. Kamath, K. Ramamritham, and D. Towsley. Continuous Media Sharing in Multimedia Database Systems. In *Proc. of the 4th Intl. Conference on Database Systems for Advanc ed Applications (DASFAA '95)*, Singapore, April 10-13, 1995.

[LCK+99] D. Lee, J. Choi, J. Kim, S. Min, Y. Cho, C. Kim, and S. Noh. On the Existence of a Spectrum of Policies that Subsumes LRU and LFU Policies. In *Proceedings of ACM-SIGMETRICS*, June 1999.

[MKK95] F. Moser, A. Kraib, and W. Klas. L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flows in a Multimedia DBMS. In *Proceedings of VLDB*, Sept. 1995.

[OBRS94] B. Ozden, A. Biliris, R. Rastogi, and A. Silberschatz. A Low-Cost Storage Server for Movie on Demand Databases. *Proc. of the 20th Intl. Conf. on Very Large Data Bases*, Sept. 1994.

[OOW93] E.J. O'Neil, P.E. O'Neil, and G. Weikum. The LRU-K Page Replacement Algorithm for Database Disk Buffering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 297–306, 1993.

[ORS95] B. Özden, R. Rastogi, and A. Silberschatz. Demand Paging for Video-on-demand Servers. *IEEE International Conference on Multimedia Computing and Systems*, May 1995.

[ORS96] B. Özden, R. Rastogi, and A. Silberschatz. Buffer Replacement Algorithms for Multimedia Databases. *IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[RHYE99] R. Rejaie, M. Handley, H. Yu, and D. Estrin. Proxy Caching Mechansim for Multimedia Playback Streams in the Internet. In *The Fourth International Web Caching Workshop*, March 1999.

[RZ95] 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.

[SG97] W. Shi and S. Ghandeharizadeh. Buffer Sharing in Video-on-Demand Servers. *ACM Sigmetrics Performance Evaluation Review*, 25(2):13–20, September 1997.

[SG98] W. Shi and S. Ghandeharizadeh. Trading Memory for Disk Bandwidth in Video-on-Demand Servers. In *Proceedings of the 13th ACM Symposium on Applied Computing*, Feb. 1998.

[SRT99] S. Sen, J. Rexford, and D. Towsley. Proxy Prefix Caching for Multimedia Streams. In *The Proceedings of the IEEE Infocom*, 1999.

[SV95] P. J. Shenoy and H. M. Vin. Efficient support for scan operations in video servers. In *Proceedings of the ACM Multimedia*, November 1995.

[TPBG93] 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.

[WSY95] 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.

[YCK93] P.S. Yu, M-S. Chen, and D.D. Kandlur. Grouped sweeping scheduling for DASD-based multimedia storage management. *Multimedia Systems*, 1(1):99–109, January 1993.

[ZT99] W. Zhoao and S. Tripathi. Bandwidth-Efficient Continuous Media Streaming Through Optimal Multiplexing. In *Proceedings of ACM-SIGMETRICS*, June 1999.