Reducing Initial Latency in a Multimedia Storage System

Edward Chang and Hector Garcia-Molina
Stanford University
Electrical Engineering Department
Stanford, CA 94305
{echang, hector}@cs.stanford.edu

Abstract

A multimedia server delivers presentations (e.g., videos, movies, games), providing high bandwidth and continuous real-time delivery. In this paper we present techniques for reducing the initial latency of presentations, i.e., for reducing the time between the arrival of a request and the start of the presentation. Traditionally, initial latency has not received much attention. This is because one major application of multimedia servers is "movies on demand" where a delay of a few minutes before a new multi-hour movie starts is acceptable. However, latency reduction is important in interactive applications such as video games and browsing of multimedia documents. Various latency reduction schemes are proposed and analyzed, and their performance compared. We show that our techniques can significantly reduce (almost eliminate in some cases) initial latency without adversely affecting throughput. Moreover, a novel on-disk partial data replication scheme that we propose proves to be far more cost effective than any other previous attempts at reducing initial latency.

Keywords: multimedia, data placement, data replication.

1 Introduction

The delivery of multimedia presentations poses many challenges, from data retrieval [6] [9], to real-time network transmission [10] [11] [12], to user interface design. In this paper we focus on the data retrieval aspect.

An effective multimedia storage system must retrieve data at very high rates and in a *continuous* fashion. That is, each multimedia presentation (e.g., movie, video, game) can be viewed as a sequence of data segments, where each segment must arrive at the display device at or before the time of display. To achieve this, the storage system must make each segment available to the network at the appropriate times, even if the disk arms are busy servicing other requests.

In addition to high bandwidth and continuous delivery, our goal is to also minimize the *initial latency*. The initial latency is the time between the arrival of a new request and the time when its first data segment becomes available in the server's memory. Traditionally, initial latency has not received much attention. We believe that this is because one major application of multimedia servers is "movies on demand" where a delay of a few minutes before a new multi-hour movie starts is acceptable. Indeed, some of the proposed servers either ignore the latency issue entirely [5] [9] or acknowledge having initial latencies on the order of minutes [7].

However, we believe that there are important applications where high latencies are not acceptable. For example, consider a video game where at each step the player's actions determine what short video to play next. Here we clearly do not want the player to wait a significant amount of time before each video scene starts. We could try to pre-fetch all the possible videos that might be selected, but this would significantly increase the server load and memory requirements. Instead, we will reduce the initial latencies by intelligently placing the data on the disks. Another application that requires low latency is hypermedia documents, as found in the World Wide Web or a Digital Library. Here a user may examine a "page" that contains links to a variety of other pages, some of which may be multimedia presentations. Again, it is inefficient to pre-fetch all linked options, and the user does not want a significant delay between the time he clicks on a link and the time the presentation starts.

As mentioned above, we reduce initial latencies simply by placing data segments intelligently on disk. However, it is important not to sacrifice throughput (i.e., reduce the number of concurrent requests that can be serviced) and not to violate the continuous delivery constraint. Our investigation into data allocation begins with the techniques of [7] [14] [15], which already achieve high bandwidth and continuous delivery for supporting multiple concurrent data streams. We show by changing the data placement and disk scheduling policies, initial latency is drastically reduced without adversely affecting throughput. Next, we propose a novel on-disk partial data replication scheme that is able to eliminate initial latency with a small percentage of disk space overhead. Initial segments can be replicated at key positions on the disk without increasing memory requirement or decreasing throughput. This on-disk replication approach proves to be far more cost effective than the in-memory replication scheme commonly used to reduce initial latency.

Trying to reduce initial latencies introduces interesting tradeoffs among latency, throughput, disk speeds and main memory available. For example, adding memory to a server can be used to increase throughput or to reduce latencies, so it is important to understand the interrelationships between all these factors. A second contribution of this paper is to analyze these tradeoffs through an analytic model, and to present some guidelines for the choice of appropriate parameters.

In this paper we consider the data allocation and replication strategies for a *single* disk that contains a set of multimedia presentations or videos. The results can be easily generalized to a system with M disks in two ways. The first is to assume that the multi-disk system partitions videos into M sets, and each is allocated to one disk using our techniques. The second way is to assume that the M units represent a disk array that can be viewed as a single "virtual" disk with high bandwidth and striped segments. Here again, we can use our techniques to allocated the segments to the virtual disk. (We refer interested readers to [1] [4] [5] for a discussion of other issues related to multiple disks.)

The rest of this paper is organized as follows. Section 2 describes the previously-studied allocation strategies and performance model that we use as a starting point. Section 3 describes our allocation techniques for reducing initial latency, and which Section 4 evaluates through a case study.

2 Constrained and Partitioned Allocation

In this section we summarize the existing storage allocation policies that we use as our starting point. To assist the reader, Table 1 gives the main parameters used to describe these and the following policies. The first portion of Table 1 lists the performance-related parameters, including throughput and initial latency. The second portion describes the physical and derived characteristics of the hardware, including memory and disks. The last part summarizes the notation we use to refer to presentations (e.g., movies, videos, games) and their segments.

Seek time is the most critical factor over which we have some control (through segment allocation). To reduce this factor, early studies [2] [6] [14] take advantage of the sequential access nature of a presentation and propose to limit the distance between their segments on disks. This placement policy is referred to as *constrained allocation*. For example, say a presentation X_i is divided into n segments, ($X_{i,1}$, $X_{i,2}$, $X_{i,3}$..., $X_{i,n}$), for storage on disk. Constrained allocation places each pair of consecutive segments within D tracks. This reduces the seek distance from what in the worst case can be the radius of the disk, to a maximum of D tracks.

Parameter	*Description*
N	Throughput
$T_{Latency}$	Initial latency to access a media
Mem	Total size of memory, bytes
DR	Data display rate, bytes/s
TR	Disk transfer rate, bytes/s
S	Segment size, bytes
R	Number of disk partitions, or regions
CYL	Number of cylinders, or tracks per surface
α	Seek time parameter, constant part
β	Seek time parameter, distance dependent part
T_{Seek}	Inter-segment seek time, milliseconds
$T_{RegionSeek}$	Total seek time in a disk region, milliseconds
$T_{RegionTR}$	Total transfer time in a disk region, seconds
m	Number of replicas for first segments
X_i	i^{th} presentation
$X_{i,j}$	Segment j of presentation i
K	Number of presentations

Table 1. Parameters

In principle, the smaller D is, the lower the overhead of the disk arm, and therefore the higher the throughput of the storage system. To achieve a small D, [7] introduces a combined allocation scheme that employs both constrained and unconstrained allocation policies. The scheme partitions a disk into R equal regions. Segments of all presentations are assigned in a zigzag manner that follows the disk head movement as the head sweeps the disk. Since the regions divide the disk evenly, the distance between two contiguous segments is constrained by a maximum of two regions (assuming that at worst one segment is at the head of one region and another is at the tail of the next region), or $D = 2/R$ of the disk radius. Within a region, segments of different presentations are placed with no constraints.

Table 2 illustrates this scheme with a disk with 8 regions and three videos. The first segment of each video is placed in region 1. Subsequent segments are placed in regions 2-8, and then in the reverse order from 8 to 1. Additional segments (not shown) would repeat the pattern.

To display objects, the disk head moves like an elevator from the innermost region (region 1) to the outermost (region 8). After reaching the outermost region, it swings inward to the innermost region. This procedure repeats itself until there are no more segments to retrieve. At each region, the

Rgn	Video	1	Video	2	Video	3
1	$X_{1,1}$	$X_{1,16}$	$X_{2,1}$	$X_{2,16}$	$X_{3,1}$	$X_{3,16}$
2	$X_{1,2}$	$X_{1,15}$	$X_{2,2}$	$X_{2,15}$	$X_{3,2}$	$X_{3,15}$
3	$X_{1,3}$	$X_{1,14}$	$X_{2,3}$	$X_{2,14}$	$X_{3,3}$	$X_{3,14}$
4	$X_{1,4}$	$X_{1,13}$	$X_{2,4}$	$X_{2,13}$	$X_{3,4}$	$X_{3,13}$
5	$X_{1,5}$	$X_{1,12}$	$X_{2,5}$	$X_{2,12}$	$X_{3,5}$	$X_{3,12}$
6	$X_{1,6}$	$X_{1,11}$	$X_{2,6}$	$X_{2,11}$	$X_{3,6}$	$X_{3,11}$
7	$X_{1,7}$	$X_{1,10}$	$X_{2,7}$	$X_{2,10}$	$X_{3,7}$	$X_{3,10}$
8	$X_{1,8}$	$X_{1,9}$	$X_{2,8}$	$X_{2,9}$	$X_{3,8}$	$X_{3,9}$

Table 2. Two Direction Data Placement

disk head picks up all requested segments in the region. For instance, if presentation X_1 is requested, the disk head picks up segment $X_{1,1}$ as it services region 1. If X_1 and X_2 are both requested at the same time, then two segments, $X_{1,1}$ and $X_{2,1}$ are retrieved from region 1; and $X_{1,i}$ and $X_{2,i}$, i = 2 to 16, are retrieved from the subsequent regions. At both ends of the disk, the regions are serviced twice before the disk head moves inward or outward. For instance, the disk head when on region 8 the first time retrieves segment $X_{i,8}$, the next time period it stays in the same region to retrieve segment $X_{i,9}$.

A new request for a currently displayed or new presentation can arrive any time. For example, say a request for presentation X_3 arrives while the disk head is servicing the second region and moving toward the third region. The new request can be serviced provided the disk bandwidth is adequate. However, the new request must wait for the disk head to reach region 8, swing back to region 1, and reverse its direction to pick up segment $X_{3,1}$.

This scheme guarantees each concurrent presentation a fraction of the disk bandwidth while at the same time limiting the length of seeks D to $2/R$ of the disk radius. However, new presentations must wait until the disk head sweeps to the starting position. In our example above, a new request could wait for the disk head to travel up to 16 regions. In general, the maximum initial latency is the time for the disk head to service 2 $\times$ R regions[1]. The simulation reported in [7] shows that this delay can be large, especially when the system is busy. For example, with 33 concurrent presentations and 24 regions, the initial latency can be up to 63.2 seconds. As we argued in Section 1, such delays may be unacceptable in many applications with unpredictable requests for short presentations.

In Section 3 we study a variety of schemes to reduce this latency without adversely affecting the number of concurrent presentations. However, before that, we briefly derive expressions for the latency and throughput under the initial strategy of this section. These expression will then be contrasted to those derived for the later schemes. The derivations that follow are equivalent to those of [7] [9] [14], except that we derive closed-form expressions for both $T_{Latency}$ and the maximum throughput N, instead of getting their values by solving a set of equations numerically. The analytical expressions are useful for explaining and contrasting the various approaches.

2.1 Performance Expressions

As mentioned earlier, the worst case initial latency for the R region disk partitioning scheme is the time to service 2 $\times$ R regions. This can be written as ($R \geq 2$)

$$T_{Latency} = 2 \times R \times (T_{RegionTR} + T_{RegionSeek}) \quad (1)$$

where $T_{RegionTR}$ is the time to transfer all requested segments in a region, and $T_{RegionSeek}$ is the total seek time for segments fetched in a region. With N simultaneous displays, $T_{RegionTR}$ is N times the time to transfer a segment, or

$$T_{RegionTR} = N \times (S/TR).$$

The worst case seek distance in a region, with non-constrained allocation in the region, is the width of the region or CYL/R tracks. Using the seek time model of [13] the seek time is $\alpha + (\beta \times \sqrt{\#tracks})$, where α and β are disk parameters. Hence, the worst individual seek time can be written as

$$T_{Seek} = \alpha + (\beta \times \sqrt{\frac{CYL}{R}}). \quad (2)$$

The total seek time for N segments of a region is[2]

$$T_{RegionSeek} = N \times T_{Seek}.$$

Substituting $T_{RegionTR}$ and $T_{RegionSeek}$ into Equation 1 yields

$$T_{Latency} = 2 \times R \times N \times (\frac{S}{TR} + \alpha + (\beta \times \sqrt{\frac{CYL}{R}})). \quad (3)$$

To derive the maximum throughput, N, we need to select the segment size S and the number of regions R that maximize N without violating two constraints: memory availability and display continuity. The former specifies that the memory required to display N concurrent presentations be less than or equal to the available memory, Mem. At first glance, it may appear that $N \times S$ bytes are needed for N presentations. However, as each segment is played, the memory blocks that make it up can be reused for other segments. So on the average over time only $S/2$ bytes are

[1] In the degenerate case $R = 1$, the delay need not be multiplied by 2. The evaluation in Section 4 takes this into account.

[2] Actually, one of the N seeks can be for up to $\frac{2 \times CYL}{R}$ but we ignore this here.

needed per presentation. Thus, the memory constraint is: $SN/2 \leq Mem$. This inequality can be rewritten as

$$S \leq 2 \times Mem/N. \tag{4}$$

To satisfy the second constraint, we need to use buffering and read-ahead (or pre-fetching) to ensure that for each presentation there is always a segment that is ready to be transmitted as soon as the last one is consumed. This constraint can be written as

$$N \times (\frac{S}{TR} + T_{Seek}) \leq \frac{S}{DR}. \tag{5}$$

The right side of Inequality 5 is the time to consume a segment from memory. To maintain a continuous display, the buffer must be replenished before it runs out. In other words, in that amount of time we must be able to read the next segment, as well as the other segments (for other presentations) that are read with the disk head visits a region. The time to read N segments is thus on the left of Inequality 5. We can rewrite Inequality 5 as

$$N \times T_{Seek} \leq S \times (\frac{1}{DR} - \frac{N}{TR}) \tag{6}$$

and then substitute S by $\frac{2 \times Mem}{N}$ in the right hand side, to obtain

$$N \times T_{Seek} \leq (\frac{2 \times Mem}{N}) \times (\frac{1}{DR} - \frac{N}{TR}). \tag{7}$$

Multiplying N and moving all terms to one side, we obtain the expression

$$T_{Seek} \times N^2 + \frac{2 \times Mem}{TR} \times N - \frac{2 \times Mem}{DR} \leq 0.$$

This second order polynomial can be solved for equality. Selecting the only positive root, and selecting the closest smaller integer value we obtain an expression for the maximum throughput:

$$N = \lfloor \frac{\sqrt{(\frac{Mem}{TR})^2 + \frac{2 \times Tseek \times Mem}{DR}} - \frac{Mem}{TR}}{T_{Seek}} \rfloor. \tag{8}$$

Once N is determined, we can compute the minimum segment size S that satisfies both Inequalities 4 and 5, and then use this value to compute $T_{Latency}$ using Equation 3.

3 Schemes for Reducing Initial Latency

This section proposes data placement and disk scheduling policies that attain low initial latency while maintaining high throughput. We study four policies, each given a short name for future reference. Each policy uses the previously listed policies, plus a new technique that reduces latency further or reduces memory requirements.

Rgn	*Video*	*1*	*Video*	*2*	*Video*	*3*
1	$X_{1,1}$	$X_{1,9}$	$X_{2,1}$	$X_{2,9}$	$X_{3,1}$	$X_{3,9}$
2	$X_{1,2}$	$X_{1,10}$	$X_{2,2}$	$X_{2,10}$	$X_{3,2}$	$X_{3,10}$
3	$X_{1,3}$	$X_{1,11}$	$X_{2,3}$	$X_{2,11}$	$X_{3,3}$	$X_{3,11}$
4	$X_{1,4}$	$X_{1,12}$	$X_{2,4}$	$X_{2,12}$	$X_{3,4}$	$X_{3,12}$
5	$X_{1,5}$	$X_{1,13}$	$X_{2,5}$	$X_{2,13}$	$X_{3,5}$	$X_{3,13}$
6	$X_{1,6}$	$X_{1,14}$	$X_{2,6}$	$X_{2,14}$	$X_{3,6}$	$X_{3,14}$
7	$X_{1,7}$	$X_{1,15}$	$X_{2,7}$	$X_{2,15}$	$X_{3,7}$	$X_{3,15}$
8	$X_{1,8}$	$X_{1,16}$	$X_{2,8}$	$X_{2,16}$	$X_{3,8}$	$X_{3,16}$

Table 3. Unidirectional Data Placement

1. Unidirectional data layout policy (Scheme *Unidirectional*);

2. Unidirectional, plus sequential access of segments within a region (Scheme *Sequential*);

3. Sequential, plus constrained allocation of segments within a region (Scheme *Constrained*);

4. Constrained, plus replication of first segments of presentations (Scheme *Replicated*).

In reality, the schemes we propose can be used independently. For example, replication can be used with unidirectional layout only, or directly on the scheme of Section 2. However, to reduce the number of combinations that have to be analyzed, here we study them in the sequence defined above. After presenting each scheme, we will derive expressions for $T_{Latency}$ and maximum throughput N.

3.1 Scheme Unidirectional

Instead of placing the segments in an elevator-like zigzag manner as proposed in [7] (see Section 2), with unidirectional layout we place the data segments in one direction [8]. Table 3 illustrates this layout on our three video, 8 region example. As shown, the segments of a given presentation are placed (in presentation order) from the innermost region to the outermost. When the outermost region is reached, the next segment is placed in the innermost region again. This scheme modifies the disk scheduling policy slightly: The disk head retrieves data only in the same direction as the data placement. After a unidirectional sweep, the disk head returns to the other end to start the next round of retrieval.

The maximum initial latency is reduced from the time to service $2 \times R$ regions, to the time to service R regions plus the cost of resetting the disk head to the first region. For modern disks, this disk head reset from one end of the disk to another is typically less than 20 milliseconds. Compared with the magnitude of the initial latency that we aim to improve, this 20 millisecond cost is negligible.

Since the maximum initial latency is roughly half of that for our first scheme, our latency expression is simply Equation 3 divided by 2, or

$$T_{Latency} = R \times N \times (\frac{S}{TR} + \alpha + (\beta \times \sqrt{\frac{CYL}{R}})). \quad (9)$$

This represents a reduction of 50% in initial latency, with essentially no change to the cost and the maximum throughput achievable.

3.2 Scheme Sequential

With non-constrained allocation of segments within a region (our assumption so far), each seek distance can be as large as the width of the region. However, forcing the disk head to pick up segments in their on-disk sequential order (as opposed to some fixed presentation order) cuts down the expected seek distance to $\frac{CYL}{R \times N}$, where N is the number of concurrent presentations. Thus, with this scheme the disk head scans through a region once, picking up segments for the N presentations. This scheme is similar to the *grouped sweeping scheduling* policy proposed in [15].

However, the sequential access policy suffers a potential drawback. When segments within a region are read in a fixed presentation order (Section 2), we know that between the time segment $X_{i,j}$ is read, and its continuation segment $X_{i,j+1}$ is read, we will do at most $N-1$ accesses to other segments. With sequential access, on the other hand, segments are read in different order within each region. In the worst case, segment $X_{i,j}$ could be read first in one region, while the next segment $X_{i,j+1}$ could be read last in its region. This means that we could have $2 \times (N-1)$ other accesses in between. However, if $X_{i,j}$ is physically last in its region, and $X_{i,j+1}$ is first, then there could be no other accesses between these two X_i reads.

To insulate the playback process from this variability in inter-segment times, we proceed as follows. Say a new presentation X_i needs to be started and its first segment $X_{i,1}$ is in region k. When region k is scanned, we read $X_{i,1}$ into a first memory buffer (size S) but do not immediately start playback. When region k is fully scanned, we start playback of $X_{i,1}$. If $X_{i,2}$ occurs at the beginning of region $k+1$, then we need to read it into a *cushion* buffer because the first buffer is still in use. At the other extreme, if $X_{i,2}$ appears at the end of $k+1$, then we do not need the cushion buffer at all, since the new segment arrives in memory as playback of the first one completes. In any case, by the time the scan of region $k+1$ completes, we have a full buffer of X_i ready for transmission, and we are ready to repeat the process.

Thus, playback takes place as in the earlier scenarios, with the exception of the cushion buffers that are needed to handle the variable time between accesses to consecutive segments. Under the assumption that on the average buffers are half full, we need $\frac{N \times S}{2}$ memory for the main buffers of the N presentations, plus $\frac{N \times S}{2}$ for the cushion buffers, for a total of $N \times S$. This means that our memory constraint is $N \times S \leq Mem$.

The reduced seek times for the sequential policy leads to a smaller initial latency. However, we also need to take into account the start up delay to fill up a buffer with $X_{i,1}$ and wait for the region scan to complete. In the worst case, the request for presentation X_i arrives just after the disk head has passed over $X_{i,1}$ and while this segment is at the very beginning of region k. In this case, we need to wait for R region scans to return to the beginning of region k, plus the full scan of region k before we start playback. Hence, the equation for latency is similar to Equation 9 where the R factor is replaced by $R+1$, and the inter-segment seek distance (in the radical) is divided by N to reflect the shorter seeks:[3]

$$T_{Latency} = (R+1) \times N \times (\frac{S}{TR} + \alpha + (\beta \sqrt{\frac{CYL}{R \times N}})). \quad (10)$$

To compute the maximum throughput achievable with sequential access, we proceed as in Section 2. We start with Inequality 5, replacing S by Mem/N (from our new memory constraint), and using our reduced seek time expression. We omit the rest of the details as the derivation is very similar to that for the continuous policy that we describe next.

3.3 Scheme Constrained

With the sequential access policy, we need additional buffers to cushion the variability in times between segments of the same presentation. This variability can be eliminated by placing segments in the same order within all regions. Thus, segment $X_{i,j}$ is constrained at both levels: it must go into the region that follows $X_{i,j-1}$, and within that region it must be placed in a fixed order with respect to the segments of other presentations.

To illustrate, let us return to our three video, 8 region example of Table 3. Assume we decide to order presentations within a region in the order X_1, X_2, X_3. The segments in region 1 could be placed in the order $X_{1,1}$, $X_{2,1}$, $X_{3,1}$, $X_{1,9}$, $X_{2,9}$, $X_{3,9}$, $X_{1,17}$, $X_{2,17}$, $X_{3,17}$, ..., $X_{1,1+(R \times j)}$, $X_{2,1+(R \times j)}$, $X_{3,1+(R \times j)}$, ... In this case, the segments of presentations are interleaved with each other. An alternative is to order the segments as $X_{1,1}$, $X_{1,9}$, $X_{1,17}$, ..., $X_{1,1+(R \times j)}$, ..., $X_{2,1}$, $X_{2,9}$, $X_{2,17}$, ..., $X_{2,1+(R \times j)}$, ..., $X_{3,1}$, $X_{3,9}$, $X_{3,17}$, ..., $X_{3,1+(R \times j)}$, ... In this case, all the X_1 segments in the first

[3] Within a region, some seeks could cover more than $CYL/(R \times N)$ cylinders, but then some of the other seeks would have to be shorter. Given the form of our seek time expression, the worst case occurs if the total seek distance that must be covered within the region, CYL/R, is uniformly split among the N seeks. Please also see [13] for explanation.

region are next to each other, followed by the X_2 segments, and so on.

In either case, when the disk head scans this region, the segments will be read according to their order on disk rather than the arrival order of the requests. Thus, the number of cylinders between consecutive segments of a presentation is constrained by $|Track(X_{i,j+1}) - Track(X_{i,j})| \leq CYL/R$. Since the variability in access times is eliminated, we do not need the cushion buffers, so the memory requirement is as with the unidirectional policy, or $(N \times S)/2 \leq Mem$. The worst case initial latency is as with sequential access except that only R regions (not $R+1$) are involved:

$$T_{Latency} = R \times N \times (\frac{S}{TR} + \alpha + (\beta \times \sqrt{\frac{CYL}{R \times N}})). \quad (11)$$

It can be shown [3] that N must satisfy the following quadric inequality:

$$N^4 + (4M_2 - M_3) \times N^3 + 4(M_2^2 - M_1) \times N^2$$
$$- 8M_1M_2 \times N + 4M_1^2 \leq 0 \quad (12)$$

where $M_1 = \frac{Mem}{\alpha \times DR}$, $M_2 = \frac{Mem}{\alpha \times TR}$, and $M_3 = \frac{\beta^2 \times CYL}{\alpha^2 \times R}$. All parameters except R are determined by the hardware configuration. For each R value, the quadric equation can be numerically solved, and we can obtain N by rounding down the smallest positive real solution.

While constrained allocation can eliminate access variability and reduce seek times, it does limit the way disk space can be allocated. This may not be acceptable in a dynamic environment where presentations are created and deleted, and are of varying lengths. On the other hand, in a static application it may be possible to initially write all the presentations to disk in the proper order. In Section 4 we will compare the performance of constrained allocation to others, but the reader should keep in mind that this policy may not be applicable in all environments.

3.4 Scheme Replicated

Initial latency can be entirely eliminated by replicating an initial portion of a presentation in memory. However, this approach is too expensive due to high memory cost. In this section we show the cost of the im-memory caching approach, and propose our much more cost effective on-disk replication scheme.

When a request comes in, the conventional approach plays the replicated copy in memory until the disk head reaches the first non-replicated segment on disk. At that point, playback continues from disk. The time to reach the first disk segment is in the worst case $T^{'}_{Latency}$, so the amount of memory needed to hold the replicas is the number of presentations K times the data needed for playback during $T^{'}_{Latency}$ seconds. Notice that K is the total number of presentations stored, not just those that are currently being played. Also notice that we use a "prime" symbol to refer to the latency of the original storage system without replication; the new latency will be zero. Thus, the memory needed for replicas is (in addition to the memory needed for usual playback):

$$K \times DR \times T^{'}_{Latency}. \quad (13)$$

Our latency reducing techniques proposed so far can cut down $T^{'}_{Latency}$ and hence reduce the required memory. Still, the amount of memory can be large. For instance, if DR is 1.5 Mbps (a typical value), our system stores 100 presentations, and $T^{'}_{Latency} = 5$ seconds, we need nearly 100 MBytes (750 Mbits) to reduce the latency to zero.

To reduce the memory cost, we propose an on-disk segment replication scheme. The on-disk segment replication scheme consists of the following steps:

1. Determine a tolerable initial latency. For example, a delay that is smaller than the video frame update rate (25 frames/seconds) is unnoticeable by human eyes. In this case, the maximum initial latency that is acceptable is $\frac{1}{25} = 0.04$ seconds.

2. Compute the number of replicas, m, required to support the target latency. The value of m can be obtained by dividing the initial latency of the storage system $T^{'}_{Latency}$ by the target initial latency, and subtracting one (we do not need a replica where the presentation starts). For example, if the initial latency of the storage system is 1 second and our target latency is 0.04, then we need $\frac{1}{0.04} - 1 = 24$ replicas on the disk.

3. For each presentation, its m replicas contain its beginning segment(s). The size of each replica is $DR \times T^{'}_{Latency}$, i.e., enough data so that in the worst case we can sustain playback until the disk head reaches the first non-replicated segment. (We discuss how to reduce this size later on in this section.)

4. Place the replicas on disk so that, together with the starting point of the presentation, they are separated by a distance of $\frac{CYL}{m+1}$.

What is the memory requirement for this on-disk segment replication scheme? Without any optimization, the extra memory required is equal to the unconsumed read-ahead data in memory when the disk head reaches the first non-replicated segment. This excess memory will remain tied up for the duration of the presentation. To illustrate, consider three replicas that are separated from each other and the initial segment by $\frac{CYL}{4}$ tracks. The distance from the replicas to the original copy is $\frac{CYL}{4}$, $\frac{CYL}{2}$, and $\frac{3 \times CYL}{4}$

tracks respectively. Assume that the first non-replicated segment is in the cylinder that precedes the one with the first replica. In this case, if we happen to read this first replica to start the presentation, all data will be consumed by the time we reach the first non-replicated segment, so we will not need any extra memory. However, if the first non-replicated segment is in the cylinder that follows the first replica, we will have a lot of unplayed data when we reach the non-replicated segment. At that point, we start reading the non-replicated segments of the presentation in a normal fashion, but we will not be able to release the extra memory. That is, at that point, data starts coming in from disk at the same rate we are playing it back, so we cannot "wither down" the read-ahead buffer.

On the average, we expect that when we reach the first non-replicated segment we will have consumed half of the replica. Thus, we will have to maintain in memory $(DR \times T^{'}_{Latency})/2$ bytes for each active presentation, for a total extra memory requirement of

$$N \times DR \times T^{'}_{Latency}/2. \tag{14}$$

The extra memory requirement for the on-disk replication scheme is proportional to N (Equation 14), as opposed to the in-memory scheme where the requirement is proportional to K (Equation 13). For the latter, the memory requirement is fixed, and cannot be improved. For the former, we can actually eliminate the extra memory requirement by using variable size replicas, as we discuss next.

To remove the memory requirement, we need to ensure that no data is left in the buffer when the disk head reaches the first non-replicated segment. To accomplish this, the size of each replica should depend on the distance between the replica and the first non-replicated segment. The closer the replica is, the smaller the replica needs to be to cover playback until the disk head catches up with the presentation. For example, when a replica is a full radius away from the first non-replicated segment, the size of the replica should be $DR \times T^{'}_{Latency}$. When the replica is one-half disk radius away from the medium, a replica of half that size is sufficient. Replicating just the necessary amount of data to cover the time for the disk head to "catch up" with the presentation on disk leaves no data in the read-ahead buffer. Therefore, this scheme eliminates the buffering requirement entirely.

An important point to make is that disk replication may involve reading at startup more data than sitting in a normal segment. If we are not careful, this can disrupt the continuous display constraint. For example, suppose that we are running the system with $N-1$ presentations, i.e., one less than capacity. At this point the system can take on an additional presentation, so assume that we read a replica as we process a given region k. Say the size of the replica is $3S$, where S is the size of a regular segment. If we do the read, we violate the continuity constraint, since we can at most do N segment reads (of size S, not $3S$) as we process k.

The solution is to split the size $3S$ replica across three contiguous regions. Thus, region k contains the first segment of the replica, region $k+1$ contains the second segment, and so on. As we read region k we read the first segment. This provides enough data to get to the second replica segment, and so on. Finally, we get to the first-non replicated segment where the process continues as usual. Notice, incidentally, that say the third segment of one replica may end up in the same region as the first segment of the next replica. This causes no problems. In summary, by spreading out the segments that make up a replica, we ensure that the number of segments read in each region is at or below the allowable limit. Thus, the maximum throughput with data replication is the same as in the original system with no replication.

The extra disk space required to support on-disk replication is the average size of a replica times the number of replicas. For K presentations, the total disk space requirement is

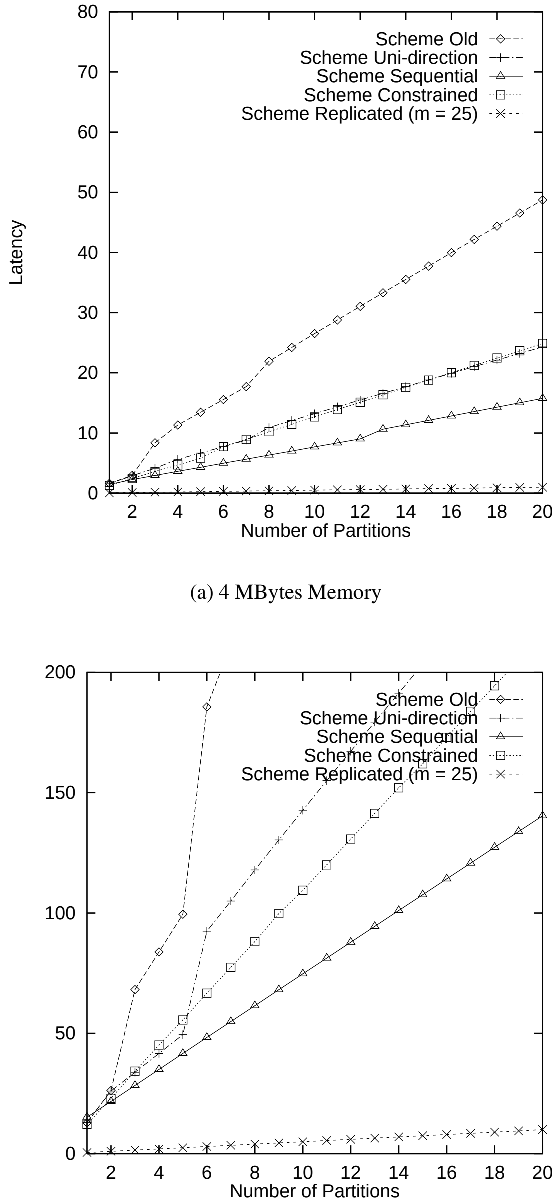
$$K \times DR \times T^{'}_{Latency} \times m/2. \tag{15}$$

The disk space required is $\frac{m}{2}$ times the requirement for the memory for the in-memory replication scheme (Equation 13). However, disk storage is much cheaper than memory by a factor of 100 (disk prices are dropping more rapidly than memory prices these days), so we believe that disk replication is significantly more cost effective. For example, we argued earlier that initial latencies of 0.04 seconds are negligible. To support 0.04 seconds of delay, m is 24. The disk space required by the on-disk replication scheme is 12 times the memory space required by the in-memory replication scheme. Thus the cost of the on-disk replication scheme is $\frac{12}{100}$, or 12%, of the in-memory replication scheme.

4 Case Study

To compare the proposed schemes we use the same Seagate Barracuda 2 disk parameters (listed in Table 4) as used in [7]. To confirm the latency and throughput results, we also implemented the resource planner documented in [7]. The numbers obtained with the planner and via our analytic formulas agree very closely. Figure 1 and 2 present initial latency and maximum throughput results for five different data placement and disk head scheduling policies: Old (Section 2), Unidirectional, Sequential, Constrained, and Replicated. For scheme Replicated, we use $m = 24$ replicas. Recall that our latency reducing schemes build upon each other. Thus, for example, Replicated uses Constrained allocation.

Our evaluations start with a 4 MByte main memory configuration that grows to 64 MBytes. We believe that a 4 MByte memory configuration is too small for today's computers; however, we study this size merely to compare the



(a) 4 MBytes Memory

(b) 64 MBytes Memory

Figure 1. $T_{Latency}$ versus R

Parameter Name	*Value*
Disk Capacity	*2.08 GBytes*
# of cylinders, CYL	*2,710*
Min. Transfer Rate TR	*68.6 Mbps*
Display Rate DR	*1.5 Mbps*
Max. Rotational Delay	*8.33 ms (milliseconds)*
α	10.63 ms if #cyl $\geq$ 400, or 8.73
β	0.0052 ms if #cyl $\geq$ 400, or 0.2

Table 4. Seagate Barracuda 2 Parameters

results with those of [7]. This paper shows only the study results for memory sizes 4 and 64 MBytes. The extended version of this paper [3] has additional graphs.

Figure 1 shows the relationship between $T_{Latency}$ and R under different memory configurations. As expected, scheme Unidirectional cuts the latency of Old by half. It is reduced further by the other schemes, until Replicated (with $m = 24$) makes it negligible. Notice that the discontinuities in latency (e.g., in scheme Sequential with 64 MBytes, as R goes from 5 to 6) occur when the throughput N increases by 1 or more.

Two counter-intuitive observation appear from Figure 1. First, maximum initial latency grows as memory expands. Intuitively, it would seem that adding resources (i.e., memory) should help in achieving our objectives, i.e., in reducing latency. Actually, the added resources do help our other objective, maximum throughput. Having more memory let us increase the segment size (see Inequality 4). This in turn let us playback a segment for a longer period, allowing us to process more concurrent presentations. Unfortunately, a larger S also increases $T_{Latency}$ linearly (see Equation 10). A second, related counter-intuitive observation is that scheme Constrained has a higher initial latency than scheme Sequential. Again, scheme Constrained has less variability than Sequential; this improves throughput, but degrades initial latency.

Figure 2 shows the relationship between the maximum throughput N and number of partitions R. Scheme Unidirectional enjoys the same throughput as scheme Old, as it improves only initial latency. Scheme Replicated achieves the same throughput as Constrained. Scheme Constrained achieves a significant improvement over Old when memory size is small. When memory size is large, all schemes achieve about the same level of throughput.

Notice that scheme Sequential has the worst throughput. Even though its seeks are significantly shorter than for schemes Old and Unidirectional, it still performs worse due to its high buffer requirements. Each presentation requires twice as much memory (to cushion access variability), and overall this causes maximum throughput to drop. Only when there is ample memory (e.g., 64 MByte case) does Sequential appear to be a viable alternative over Old and Unidirectional: in that case it can reduce initial latency significantly while only reducing throughput very slightly.

Figure 1 shows only the relationship between $T_{Latency}$ and R, but not N. To further understand the interplay between these parameters, let us investigate the initial latency incurred when achieving the *same* maximum N. In Table 5 we compare Schemes Constrained and Old when they achieve the same N. With a 4 MByte main memory, scheme Constrained supports 33 simultaneous displays with a latency of 1.28 seconds ($R = 1$). To accomplish the same level of throughput requires scheme Old to partition the disk

Memory	*N*	*Constrained*		*Old*	
		Latency	*R's*	*Latency*	*R's*
4 M	*33*	*1.28*	*1*	*62*	*24*
16 M	*41*	*4.19*	*1*	*291*	*35*
32 M	*43*	*7.58*	*1*	*333*	*21*
64 M	*44*	*12.21*	*1*	*182*	*6*

Table 5. $T_{Latency}$ Comparison Given N

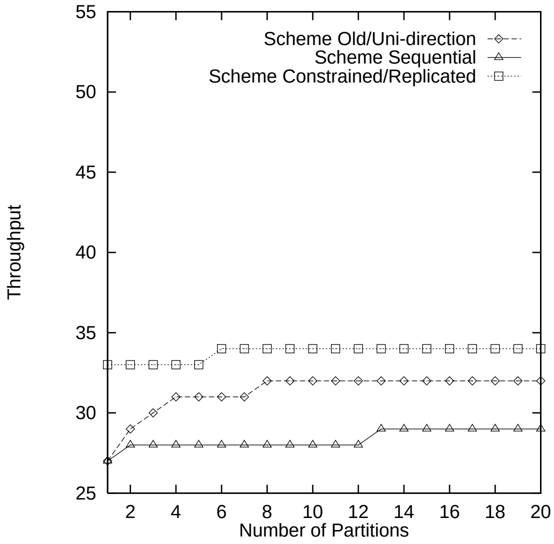
into 24 regions, incurring a 62 second delay. This and the other comparisons of Table 5 illustrate that the real magnitude of initial latency improvement is much larger than is shown in Figure 1.

4.1 Additional Observations and Analysis

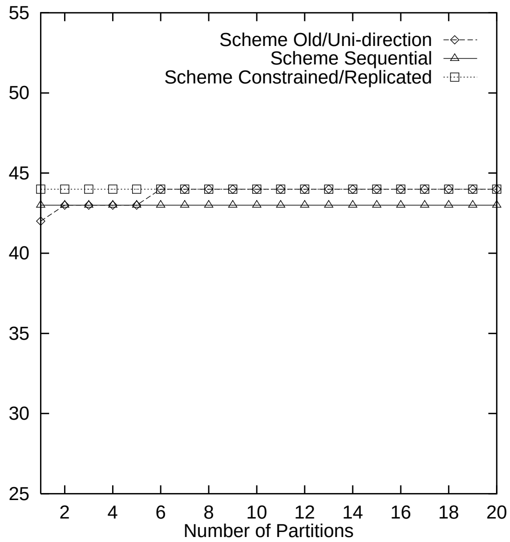
The case study reveals a few important facts. First, a small number of regions works well under the Unidirectional and Sequential policies. Increasing the number of disk partitions accomplishes a marginal improvement in throughput, but increases initial latency dramatically. To verify this observation let us examine Inequality 5. Leaving N on the left side of the inequality, we rewrite it as

$$N \leq \frac{TR}{DR} \times \frac{S}{S + (T_{Seek} \times TR)}. \tag{16}$$

Notice that in the inequality TR/DR is the throughput cap. The major factor that influences N is T_{Seek}. In other words, the best way to improve N is to reduce T_{Seek}. Now recall that T_{Seek} is $\alpha + (\beta \times \sqrt{CYL/(R \times N)})$. CYL is a constant, so the T_{Seek} reduction can be achieved by increasing N or R. Under scheme Old, the seek time is not reduced by the N factor, so throughput can only be improved by increasing R. With the Sequential or Constrained policies, seek times are also reduced by the N factor (within the radical in T_{seek}), and this eliminates the need for a large R. Thus, with a large N, the magnitude left to be improved by a large R is insignificant. In short, with all policies except Old and Unidirectional, a large R does not increase throughput significantly, but it does increase latency linearly. Thus, if low initial latency is a goal, $R = 1$ is a good choice for all policies except Old and Unidirectional. The next observation concerns the memory requirement. Intuitively, with a large R, the seek distance is shorter in a region, therefore the memory required to buffer data is smaller to support the same N. In other words, a large R value reduces memory required per presentation. This argument is accurate, but the question remains: how much memory can disk partitioning save? To illustrate, Table 6 lists the memory requirement and R values needed for scheme Constrained to achieve the same throughput $N = 43$. The table also shows the initial latency for each scenario. We can see that the memory savings are not significant as R grows, but the latency growth



(a) 4 MBytes Memory



(b) 64 MBytes Memory

Figure 2. N versus R

Regions	*N*	*Memory Required*	*Initial Latency*
1	*43*	29.92 MBytes	7.42 seconds
2	*43*	28.57 MBytes	14.18 seconds
4	*43*	27.62 MBytes	27.41 seconds
8	*43*	26.95 MBytes	53.48 seconds
16	*43*	26.47 MBytes	105.06 seconds
32	*43*	26.13 MBytes	207.45 seconds
64	*43*	25.90 MBytes	411.12 seconds

Table 6. Memory Requirement by R

is significant. To see why, we rewrite Inequality 5 as

$$S \geq T_{Seek} \times \frac{N \times DR \times TR}{TR - N \times DR}. \quad (17)$$

If N, DR and TR are fixed, then T_{Seek} is the only factor that can save memory. As with throughput, a large R value does not save much on memory.

4.2 Storage System Design Guidelines

In designing a storage system for multimedia applications, the goal is high throughput and low initial delay with minimum cost. The cost includes not only the hardware cost, but also the implementation and maintenance costs. For the latter, a simple design is critical.

Partitioning the disk into regions ($R \geq 2$) is beneficial mainly if memory conservation is important and the initial latency incurred is not critical. Interactive applications with unpredictable access patterns are not in this category.

With a single partition it is quite simple to use the Constrained policy and get its high throughput and low latency. In particular, we propose to place each presentation on contiguous tracks and cylinders. On an 8.3 GByte IBM 3.5-inch Ultrastar 2 XP disk, for instance, six 1.3 GByte movies can be laid out to form six concentric circles. With $R = 1$ the distance between retrieving two segments of a presentation is bounded by CYL tracks. This CYL distance may look large at first glance, but because segments are retrieved in physical order, each seek distance is cut on the average by a factor of N. Furthermore, because the segments are retrieved in the same order in each disk sweep, the memory requirements are low.

If the above scheme still does not provide low enough initial latency, it can be further reduced by replicating data. The in-memory and on-disk replication schemes can lower the latency to essentially zero, but as we discussed, the on-disk approach is more cost effective.

5 Conclusion

Initial latency must be reduced to make the performance of interactive multimedia applications acceptable to the users. In this study we have presented several techniques for reducing initial latency in multimedia presentations. The first technique, Unidirectional, reduces latency by laying down segments on disk in a single direction. The Sequential and Constrained techniques cut latency by reading segments within a region in physical order. For Constrained, this order is the same for all regions, reducing the variability in time between accesses to consecutive segments of a presentation. We have also proposed a novel on-disk replication scheme, which replicates only initial segments of presentations. Each replica is strategically placed on disk, and its size varies. We show the on-disk replication scheme is much more cost effective than the in-memory replication scheme.

Our results show that these techniques can significantly cut initial latencies while keeping throughput high. Each technique reflects different costs, e.g., the cost of disk replicas, or the cost of constrained allocation on disk. Thus, the proper choice of technique will depend on the target latency, throughput, and costs for the application.

References

[1] S. Berson and S. Ghandeharizadeh. Staggered striping: A flexible technique to display continuous media. *MULTIMEDIA TOOLS AND APPL.*, 1(2):127–148, June 1995.

[2] P. Bocheck and H. Meadows. Disk partitioning technique for reducing multimedia access delay. *ISMM Distributed Systems and Multimedia Applications*, August 1994.

[3] E. Chang and H. Garcia-Monila. Reducing initial latency in a multimedia storage system. *Stanford University Technical Report SIDL-WP-1996-0031 http://www-diglib.stanford.edu*, February 1996.

[4] P. M. Chen and E. K. Lee. Raid: High-performance, reliable secondary storage. *ACM Comp. Surveys*, 26(2), 1994.

[5] A. L. Chervenak and D. A. Patterson. Choosing the best storage system for video service. *Proceedings of ACM Multimedia 95*, pages 109–118, November 1995.

[6] S. Ghandeharizadeh. Continuous retrieval of multimedia data using parallelism. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 5(4):658–69, 1993.

[7] S. Ghandeharizadeh, S. Kim, and C. Shahabi. On configuring a single disk continuous media server. *SIGMETRICS PERFORMANCE EVALUATION*, 23(1):37–46, May 1995.

[8] S. Ghandeharizadeh, S. Kim, and C. Shahabi. On disk scheduling and data placement for viedo servers. *USC TR*, December 1995.

[9] T. Kunii. Issues in storage and retrieval of multimedia data. *MULTIMEDIA SYSTEMS*, 3(5-6):198–304, 1995.

[10] J. Nussbaumer. Networking requirements for interactive video on demand. *IEEE Journal On Selected Areas In Communications*, 13(5):779–787, June 1995.

[11] S. Ramanathan and P. V. Rangan. Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks. *IEEE/ACM Trans. on Networking*, 1(2), 1993.

[12] S. Ramanathan, P. V. Rangan, and H. M. Vin. Optimal communication architectures for multimedia conferencing in distributed systems. *Proc. of 12th Conference on Distributed Computing Systems*, June 1992.

[13] F. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming raid-a disk array management system for video files. *First ACM Conference on Multimedia*, August 1993.

[14] H. M. Vin. and P. V. Rangan. Designing a multi-user hdtv storage server. *IEEE Journal on Selected Areas in Communication, Special Issue On HDTV and Digital Video Communication*, 11(1), January 1993.

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