Continuous Media Sharing in Multimedia Database Systems

Mohan Kamath†
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
kamath@cs.umass.edu

Krithi Ramamritham†
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
krithi@cs.umass.edu

Don Towsley‡
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
towsley@cs.umass.edu

Abstract

The timeliness and synchronization requirements of multimedia data demand efficient buffer management and disk access schemes for multimedia database systems. The data rates involved are very high and despite the development of efficient storage and retrieval strategies, disk I/O is a potential bottleneck, which limits the number of concurrent sessions supported by a system. This calls for more efficient use of data that has already been brought into the buffer. We introduce the notion of continuous media caching, which is a simple and novel technique where data that have been played back by a user are preserved in a controlled fashion for use by subsequent users requesting the same data. We present heuristics to determine when continuous media sharing is beneficial and describe the buffer management algorithms. Simulation studies indicate that our technique substantially improves the performance of multimedia database applications where data sharing is possible.

1 Introduction

To support modern applications that use multimedia data, numerous researchers and system developers have been focusing on the development of multimedia database systems. Lately the construction of information superhighways has also received enormous attention and many commercial ventures are focusing on providing a variety of *on-demand* services. Multimedia database systems will also form the backbone for these services. Multimedia data comprises of *static* media, like text and images, that do not vary with time (with respect to the output device) and *dynamic* (or *continuous*) media, like audio and video, which vary with time. Whenever continuous media is involved, in order to ensure smooth and meaningful flow of information to the user, *timeliness* and *synchronization* requirements are imposed. Timeliness requires that consecutive pieces of data from a stream be displayed to the user within a certain duration of time for continuity. Hence, unlike real-time database systems where deadlines are associated with individual transactions, multimedia data has continuous deadlines. Synchronization requires that data from different media be coordinated so that the user is presented with a consistent view.

Thus far, most of the research work in the area of multimedia data management has concentrated on data modeling [3,4,5,12,16,17,25] and storage mangement [1,10,13,20,22,24,26]. Though efficient storage and retrieval strategies have been developed, since the data rates involved are very high (around 2 MB/sec of compressed data per request) disk I/O is a potential bottleneck, *i.e.*, the number of concurrent sessions supported could be limited by the bandwidth of the storage system. Hence, it is important to make better use of data brought from the disk into the database buffer whenever possible. In this paper we explore the potential benefits that can be achieved through *continuous media sharing*, wherein the data fetched into the server's memory is used by multiple requests that arrive over a period of time. Although several buffer management schemes have been studied, thus far there has not been much research to evaluate the potential benefits of continuous media sharing when users are accessing multimedia data concurrently. Though buffer management schemes used in traditional database systems [9] share data, they are unsuitable for multimedia data because of the timeliness and synchronization requirements. Alternatively, schemes developed thus far for multimedia data [1,5] disregard sharing of continuous media data. The Use-And-Toss (UAT) scheme [5] tosses away used data so that the next piece of required data is brought into the same buffer. Thus the UAT scheme is good only if data sharing cannot be exploited. The buffer sharing scheme discussed in [21] only reuses buffer space and does not consider continuous media sharing.

The motivation for our work comes from the fact that once a user has requested a playback, advance knowledge of the multimedia stream to be accessed for this user can be utilized to promote sharing between this user and future users (planned sharing for future accesses). To achieve this, we introduce the notion of *continuous media caching*. It is a simple and novel technique where buffers that have been played back by a user are preserved in a controlled fashion for use by subsequent (or lagging) users requesting the same data. This technique can be used when sufficient buffer space is available at the server to retain data for the required duration as in the case of multimedia database systems that handle video databases. Since the system can avoid fetching the data from the disk again for the lagging user, it is possible to support a larger number of sessions than that permitted by the disk bandwidth. To help the system decide whether a new request is to be shared with an existing one, we have designed suitable heuristics. A new buffer management scheme called SHR has been developed that enhances UAT with continuous media caching. In addition to reusing old data, it also allows sharing of future data. We explore the benefits of continuous media sharing in multimedia database systems by considering the News-On-Demand-Service (NODS) where a customer makes a request by choosing a topic from a set of offered topics. This typifies applications where extensive data sharing is possible.

Batching is another technique for improving the per-

† Supported by NSF grant IRI-9314376
‡ Supported by NSF grant IRI-9114197

Proceedings of the Fourth International Conference on
Database Systems for Advanced Applications (DASFAA'95)
Ed. Tok Wang Ling and Yoshifumi Masunaga
Singapore, April 10-13, 1995

formance of a system by grouping requests that request the same topic [6]. By using continuous media caching in conjunction with batching, the performance of multimedia database systems can be further improved. To examine such techniques, we also study two other schemes BAT-UAT and BAT-SHR, which are enhancements of UAT and SHR respectively with batching. Simulation studies indicate that continuous media caching improves the performance of multimedia database systems when data sharing is possible.

The rest of the paper is organized as follows. Section 2 discusses the data characteristics of multimedia data and NODS in particular. Issues related to continuous media sharing are discussed in section 3. In section 4 we briefly discuss heuristics that can help the system decide whether a new request is to be shared with an existing one. Section 5 discusses the UAT scheme and its enhancement the SHR scheme (through the use of continuous media caching). Continuous media caching schemes enhanced with batching techniques are discussed in section 6. Details of the performance tests are presented in section 7. Section 8 interprets the results of the performance tests and section 9 concludes with a summary of this work.

2 Data Characteristics

Multimedia objects are captured using appropriate devices, digitized and edited, until they are in a form that can be presented to the user. These independently created objects are then linked using spatial and temporal information, ready to be played back by the user. At playback time, to ensure smooth and meaningful flow of information to the user, there are timeliness and synchronization requirements which vary from application to application. When multiple concurrent sessions are to be served, ensuring timeliness is a complex goal in a bandwidth limited system. We do not discuss synchronization requirements here since detailed discussions can be found in [19,23]. The rest of this section describes the data characteristics of NODS.

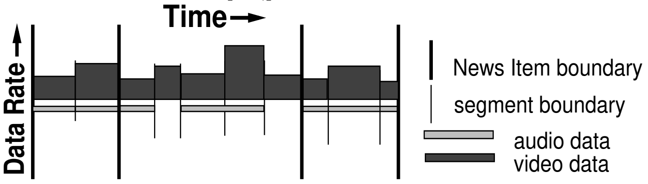


Figure 1: Multimedia Data Stream of a request

In NODS, given that the number of topics in high demand is likely to be limited at any particular time, given the higher level of interest in certain topics like sports or business, the probability of one or more users requesting the same item for playback is very high. To request a service, the user chooses a topic and a language from the variety of options available, *e.g. topic* from one of Politics, International, Business, Sports, Entertainment, Health, Technology etc. and *language* from one of English, Spanish, Italian, French, German, Hindi etc. Each topic consists of a list of news items. News items are composed of video and audio segments and possibly some images. All news items to be displayed when a topic is chosen are compiled in advance and the bandwidth required at every instance of playback is known in advance. *Figure 1* shows the data stream of a request. Because video is often compressed, we assume that the continuous media data is played out at a variable bit rate. The data rate actually depends on the amount of action that is packed in the video. For example a video stream of a news reader reading news will require a lower data rate than the video stream from a moving camera (e.g. sports clippings). We assume that a data stream is divided into *segments* where each segment is characterized by a uniform playout data rate. For simplicity, we also assume that segments are of equal length (of 1 second duration). The data rate can vary from segment to segment and hence the amount of buffer space required for a segment varies.

3 Continuous Media Sharing

In this section we introduce the concept of *continuous media caching* which promotes planned sharing. It is a simple and novel technique where buffers that have been played back by a user are preserved in a controlled fashion for use by subsequent users requesting the same data. We try to efficiently utilize the high memory available in large scale multimedia database servers.

Sharing of data can occur at various levels in NODS as we discuss below. Sharing can occur when there are multiple requests for the same topic. Sharing is also possible if some news items are shared between topics, *e.g.*, news on an international trade agreement can be included in a variety of topics like Politics, International and Business because of its relevance to all of them. In addition, sharing of continuous media can also occur at the media level, *e.g.*, for news items in different languages, additional storage is necessary only for the audio segments as the video segments can be shared. This clearly shows that sharing at different levels can lead to compound sharing. We focus exclusively on the first type of sharing since this is likely to be more frequent.

Since the buffer requirements for images and audio are less compared to video, we consider only the sharing requirements of video data. We mainly focus on sharing that occurs when multiple users request service on the same topic (though they may have chosen a different language). Several possibilities exist for sharing, depending upon the arrival time of the customers. When the same service is being requested by two (or more) users simultaneously, one of the following could be true:

1. they arrive at the same time (start at the same time).
2. they arrive with a gap of few time units between them.

Depending upon the order in which users share an application, they may be classified as leading (the first one) or lagging (the subsequent ones that can share the buffers). Using the same buffers for both users can definitely improve performance in case 1 since the same data is used simultaneously. In case 2, continuous media caching can be used to promote sharing provided the length of the gap is short. If the gap is too long, it is better to obtain the data for the lagging user directly from the disk and reuse the data buffers from the leading user, if possible, just immediately after the contents have been played out.

The disk and buffer requirements are compared as a function of time for the non-shared case and the shared case (continuous media caching) in *figure 2*. Two requests for the same topic (containing 5 segments) that arrive with a time difference corresponding to two segments is shown in the figure. In the non-shared case, disk and buffer bandwidths are allocated independently for each request. However for the shared case, since it is known that segments three through five will be available in the buffer (they are loaded into the buffer for the first request) when the second request needs it, the buffers are reused, *i.e.*, the buffers are preserved until the corresponding playback time for the second user. Since the segments need not be fetched again from the disk for the second request, there is a decrease in the number of disk I/Os while the buffer requirements may increase.

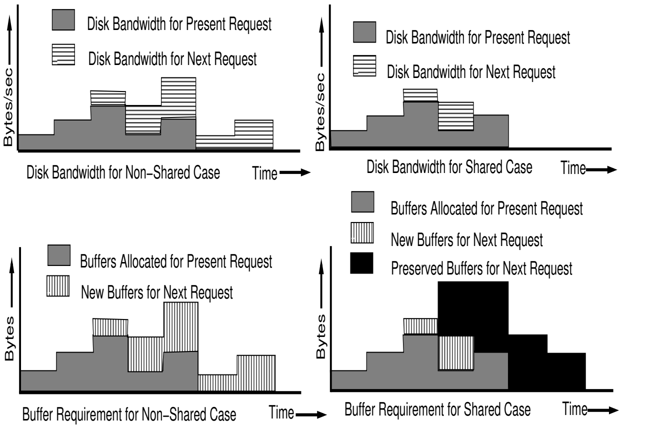


Figure 2: Buffer Sharing between two requests

Details of the segments that are actually present in the buffer at each time instant in the shared case (bottom right case of *figure 2*) are shown in *figure 3*. It differentiates the segments as follows — segments that are loaded for the first user, segments fetched from the disk for the second user and segments that have been used by the first user but preserved for the second user. It illustrates how the continuous media caching technique preserves only the necessary segments for just the required amount of time. By sharing the continuous media data buffers effectively, we illustrated how more number of sessions can be supported than that supported by the disk bandwidth. The buffer and disk bandwidth for the schemes we propose are determined using this technique.

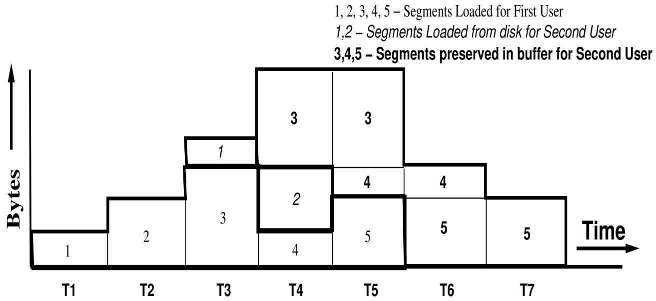


Figure 3: Segments available in buffer at each time slot (zooming into bottom right of figure 2)

4 Heuristics to determine when sharing in beneficial

The criteria used to determine whether a request is to be shared is critical to the success of systems that use continuous media caching. Several factors determine this and there are tradeoffs. As we discussed earlier, this technique pays off when the difference in the arrival times of two requests to be shared is small but not when it is large. We denote the threshold value for the maximum tolerable inter arrival time to share a new request with an existing one by *max-iat*. Hence the inter-arrival time (*iat*) has to be less than this value for sharing to occur. Determining this threshold value beyond which there is no payoff is a complex task. This threshold value depends on a number of parameters — mean inter arrival time between requests (sec), total memory available in the system (MB) and the data rate of the topic (MB/s). If more memory is available, then segments can be cached for a longer duration of time; hence, *max-iat* is directly proportional to memory size. On the other hand, if the data rate is high, it is difficult to cache segments for a longer duration since the buffer space needed will be too high. Hence *max-iat* is inversely proportional to the data rate. Based on the above facts, we arrive at the following set of equations:

$$max\_iat = \mathcal{C} \cdot \frac{Mean\_Arrival\_Time \cdot Memory\_Size}{Data\_Rate} \quad (1)$$

$$iat \leq max\_iat \quad (2)$$

The constant $\mathcal{C}$ can be determined via empirical tests as we do later in section 8.1. Thus, for the SHR scheme, we have used inequality (2) as a basis for deciding when the requests are to be shared. If *iat* is larger than *max-iat* then continuous media caching is not used and the request has to be satisfied using the UAT scheme. It should be noted that in all these cases, a request is considered for scheduling as soon as it arrives (zero response time).

5 Buffer Management Schemes

In this section we first review the UAT buffer management scheme for handling multimedia data and then describe the SHR scheme. While UAT tries to reuse past data alone, SHR is an improvement on UAT that allows future data to be shared as well. Although numerous buffer management schemes (FIFO, LRU, etc.) [9,14] have been studied in the context of traditional and real-time database systems, we will not discuss them since they are not quite suitable in the context of multimedia data which has continuous deadlines.

In traditional database systems, once a page is brought from the disk into the buffer, in virtual memory operating system environments a *fix* operation is performed to ensure that it is not paged out. Once it is determined that the page is no longer necessary or is a candidate for replacement an *unfix* is performed on that page. Also, buffer allocation and page replacement are considered orthogonal issues. Essentially these techniques cannot handle continuous deadlines and hence are unsuitable for multimedia data. Part of the motivation for the continuous media caching scheme came from the way code pages (which are read only) are shared in virtual memory operating systems.

Before we discuss the UAT and SHR schemes, we provide details of our memory model, admission control, and bandwidth reservation scheme. In multimedia database systems, advance knowledge of the data stream to be accessed is usually available and can be utilized for efficient buffer management. Since we are assuming a large main buffer, this information can be stored in main buffer (*i.e.*, the number of segments that make up a topic and the segment data rates for each segment). Unused buffer space exists in a global *free-pool*. Buffers needed for the segments of a request are acquired (analogous to fix) from the free-pool and returned (analogous to unfix) to the free-pool when they have been used. When buffers are requested from the free-pool, the oldest buffer in the free-pool is granted to the requester.

The admission control and reservation scheme is dependent upon whether a request can be shared. Given $(n-1)$ sessions already in progress, the n^{th} request is accepted iff the following inequalities hold.

$$\sum_{i=1}^{n} l \cdot d_{i,t} \leq M \qquad \forall t = 1,..,s \tag{3}$$

$$\sum_{i=1}^{n} d_{i,t} \leq R \qquad \forall t = 1,..,s \tag{4}$$

where $d_{i,t}$ represents the continuous media rate (audio and video) for request i at time slot t, l represents the length of each segment, s represents the total number of segments in the request (called the length of the request). The first inequality corresponds to the constraints imposed by the buffer bandwidth (buffer size $= M$) and the second by the disk bandwidth (data rate supported $= R$). If the inequalities hold for a request, the required buffer and disk bandwidth is reserved and the request is guaranteed to succeed. We mentioned earlier that since the video rate dominates the audio rates, only the requirements of video are considered for sharing. The availability of the necessary bandwidth for audio however is also to be checked by the admission control policy to determine whether the synchronization requirements of audio and video streams can be met.

5.1 Traditional Scheme - Use And Toss (UAT)

Our UAT scheme replaces data on a segment basis and performs admission control and bandwidth reservation to anticipate future demand. Using a timestamp mechanism, a FIFO policy is used to decide which of the segments from the free-pool is to be allocated to a new request, *i.e.*, when a new request is to be allocated a segment from the free-pool, the oldest segment in the free-pool is allocated first. UAT reuses segments that may already exist in the buffer, *i.e.*, if any of the required segments happen to exist in the free-pool then they are grabbed from the free-pool (unplanned sharing), thereby avoiding fetching those segments again from the disk. UAT does not consider sharing explicitly (planned sharing), *i.e.*, when the segments of a topic have been played back, they are not preserved in the buffer for a lagging user who might have requested the same topic. Hence admission control and bandwidth reservation are performed independently (using inequalities 3 and 4) for requests from different users. If the necessary bandwidth is available, then the required disk and buffer bandwidths are reserved; else the request is rejected.

5.2 Proposed Scheme - Sharing Data (SHR)

The UAT scheme has been enhanced using the continuous media caching technique to form the SHR scheme. We focus only on buffer management and do not discuss details of disk scheduling here. Segments required by the different streams can be obtained from the disk in a timely manner by proper layout of the data on disk using schemes like striping, staggered striping or other techniques [1,13,20,22,24,26].

Segments that are explicitly preserved begin with the segment that is being played back by the leading user when the lagging user enters the system. All segments that follow this are also preserved in the buffer until the lagging user needs it for playback (segments 3,4 & 5 in *figure 3*). If it happens that some of the segments are in the free-pool (segments 1 & 2 after playback in *figure 1*), they can be reacquired from the free-pool if the buffer space corresponding to those segments has not already been granted to other sessions. If all the segments (loaded for the first user prior to this user's arrival) are available, then no data is to be fetched from the disk for the second user (only buffer bandwidth is checked/reserved), else data is to be fetched from the disk for those segments that do not exist in the free-pool(both buffer and disk bandwidth checked/reserved).

```
do while there are requests in the queue
    select next request ;
    admitted = false;
    For each segment in the topic, check if the segment is already
    in the free-pool and grab it if it is available ;
    For each segment grabbed from the free-pool, modify the
    total profile to reflect preserving of these segments and
    check for buffer bandwidth violation ;
    For segments that have to be fetched from the disk, modify
    the total profile for disk and buffer accordingly and check
    for buffer/disk bandwidth violation ;
    if request is shared
       For segments that exist in the buffer and are shareable, modify
       the total profile for buffer and check for bandwidth violation ;
    endif
    if no violation at any stage
        admitted = true
        /* total profile contains reservation for this request */
    else
        restore old total profile
        reject request ;
    endif
end do
```

Figure 4: Admission Control and Bandwidth Reservation

The admission control and bandwidth reservation scheme for SHR is shown in *figure 4*. The total profile refers to the total requirements of buffer/disk bandwidth. It also describes the sequence in which the availability of segments in the buffer is checked along with admission control and buffer/disk bandwidth reservation. The admission control policy checks if there is sufficient disk bandwidth available for segments to be loaded from the disk and sufficient buffer bandwidth for loading these segments in the buffer and preserving the other segments (*i.e.*, loading segments 1 & 2 and preserving segments 3, 4 and 5 in *figure 3*). If the required bandwidth is available then appropriate disk and buffer bandwidths are reserved. If sharing is not possible, the SHR scheme behaves like the UAT scheme and an attempt is made to reserve the necessary buffer and disk bandwidth for each request independently. If this also fails, then the new request is rejected.

```
do forever
  do for all active sessions
    if buffers are shared with another session
        if leading user
            preserve the used buffers for lagging users ;
            acquire new buffers from the free pool ;
            fetch data from disk into the buffer
        else if lagging user
            if there are more lagging users
                preserve the used buffers for lagging users ;
                read from preserved buffers ;
            else
                return the used buffers to the free pool ;
                read from preserved buffers ;
            endif
        endif
    else
        when a new segment is encountered
            return the used buffers to the free pool ;
            acquire new buffers from the free pool ;
            fetch segment from disk;
    endif
  end do
end do
```

Figure 5: SHR Buffer Allocation & Replacement

Buffer allocation and replacement is done in an integrated manner as shown in *figure 5*. Buffer allocation is done when a inter segment synchronization point is encountered. The figure describes how buffers are allocated for playback and how they are freed/preserved after playback depending on whether there are lagging users for a request. Information about allocated buffers can be maintained using a hash table.

6 Batching Schemes

Batching tries to support more number of concurrent users by grouping requests that arrive within a short duration of time for the same topic. This type of sharing corresponds to case 1 discussed in section 3 since the requests are satisfied simultaneously (even thought they arrive at different times). Our objective is to improve the system performance by using continuous media caching in conjunction with batching schemes.

The response time of a request is the time interval between the instant it enters the system and the instant at which it is served (*i.e.*, the first segment of the request is played back). Since customers will be inclined to choose a service with a faster response time, it is important to study the effect of batching and sharing schemes on response time. The factors that determine the actual response time of a request include the *scheduling policy* and the *scheduling period*. The scheduling period is the time interval between two consecutive invocations of the scheduler. At each invocation of the scheduler (scheduling point), requests for the same topic are batched and scheduled according to the scheduling policy. If the *tolerable response time* is small, then it is necessary to have smaller scheduling periods else the percentage of successful requests will be low. However in order to increase sharing through "batching", the scheduling period has to be large. Hence, as a compromise, some intermediate value has to be chosen such that batching can be achieved with smaller response times. The scheduling policy determines the order in which the requests are to be considered at each scheduling point. The First-Come-First-Serve (FCFS) policy, selects the topic which has the oldest request and batches all requests waiting for that topic. In the Maximum-Group-Scheme (MGS) policy, requests for the same topic are batched and the topic which has the maximum number of requests is considered first. Other variations are possible but we consider only the FCFS policy which is recommended as a fair policy in [6].

At each scheduling point, in the FCFS scheme, the grouped requests are considered one by one based on the earliest arrival time of a request in the group. A request first goes through the admission control and bandwidth reservation phase. If the required resources are available and allocated, then all the requests for that topic are successful. If this request cannot be satisfied, it will be tried again at the next scheduling point and so on as long as the response time requirements are met. In this case, the remaining requests are considered at the present scheduling point. Hence a request that could not be satisfied at a particular scheduling point may be successful at the next scheduling point. If the actual response time exceeds the maximum tolerable response time, then the request is no longer considered for scheduling and is rejected.

6.1 BAT-UAT and BAT-SHR Schemes

BAT-UAT scheme is the FCFS batching scheme that uses the UAT buffer management scheme. Since the UAT scheme does not use continuous media caching, data sharing is achieved purely through the effect of batching in BAT-UAT. The BAT-SHR scheme is the FCFS batching scheme that uses the SHR buffer management scheme. Here the effect of sharing is achieved in two different ways, through batching & continuous media caching and hence the extra data sharing that could not be achieved through batching is achieved via continuous media caching *i.e*, requests for the same topic may have been scheduled in different neighboring scheduling points, but sharing is achieved through continuous media caching when the neighboring points are separated by a time less than *max-iat*.

7 Performance Tests

In this section we discuss our experimental setting, a few of our assumptions and then describe the various parameters and metrics used for the tests.

We use a client-server architecture for the tests. A DMA transfer is assumed between the disk and the buffer at the server. Thus the CPU cannot be a bottleneck and hence is not included in our model. It is assumed that the network between the client and the server provides certain quality-of-service (QOS) guarantees with respect to network delay and bandwidth (detailed discussion of reliability and networking issues for continuous media can be found in [2]). We will assume a two level storage system: primary storage (main memory buffer) and secondary storage (disk). Knowing the average seek time, rotational latency and data read rate, the effective transfer rate, R can be determined for a storage system. This is used in performing bandwidth allocations for the disk.

The parameters to be considered fall in four different categories — *customer, system, topic* and *batching*. They are listed below in the table with their default values. Topic parameters (length, data rate, etc.) are generated using a uniform distribution. Customer arrivals are modeled using a Poisson process. The topic chosen by a customer is modeled using Zipf's law [27]. According to this, if the topics $1, 2, ..., n$ are ordered such that $p_1 \geq p_2 \geq ... \geq p_n$ then $p_i = c/i$ where c is a normalizing constant and p_i represents the probability that a customer chooses topic i. This distribution has been shown to closely approximate real user behavior in videotex systems [11] which are similar to on-demand services. The system and batching parameters were selected based on practical considerations.

Category	Parameter	Setting (Default Values)
Customer	Inter-Arrival Time	Poisson Distrib. mean = 10 sec
	Topic Chosen	Zipf's Distribution (1-10)
System	Buffer Size	1.28 GB
	Disk Rate	< determined using tests>
Topic	Total Number	10
	Length	Uniform Distrib. (500-700) sec
	Data Rate of Segment	Uniform Distrib. (1-2) MB/sec
Batching	Max. Response Time	40 sec
	Scheduling Period	10 sec

While the percentage of requests that are successful seems to be the right metric for these systems, it is not sufficient for system designers since they are more interested in metrics that are closely tied to QOS guarantees. An example of a metric that provides a QOS guarantee is the maximum arrival rate that can be sustained if 95% of the requests are to succeed given certain resources. A better metric would be the resource requirements to ensure a 95% success rate given a specific workload and this is the metric we use for our tests. Evaluating metrics that are based on QOS guarantees is a time consuming process since it is iterative in nature. For example, to determine the disk bandwidth that ensures a 95 % success rate when the arrival rate is 6/min (inter-arrival time is

10 sec), starting from a small value of disk bandwidth (for which the success rate may be lower than 95%), tests are to be performed by incrementing the disk bandwidth, until the success rate reaches 95%. Response time is more relevant for the batching schemes, since the scheduling is done at periodic intervals. A tolerable response time is specified for these tests and 95% of the requests should have response times smaller than the tolerable response time (requests that cannot meet the response time requirements are rejected).

8 Results

Each simulation consisted of 25 iterations and each iteration simulates 2000 customers. The confidence interval tests of the results indicate that 95 % of the results (percentage of successful requests) are within 3 % of the mean in almost all cases. All parameters are set to their default values as specified earlier unless explicitly indicated in the graphs. Topic details (the database accessed by users) were generated for every iteration. We first discuss the tests performed to determine *max_iat*, the sharing threshold. Then we compare the results of the buffer management schemes (UAT and SHR) and, finally we compare the batching schemes (BAT-UAT and BAT-SHR). The area below the curves are filled in light gray for schemes that do not use continuous media caching and in dark gray for schemes that use continuous media caching. Hence in all of the graphs, the light gray region shows the benefit of continuous media caching.

8.1 Determination of best value for *max_iat*

Based on the parameters described in the experimental settings, we performed some tests by varying the value of *max_iat*. As seen in *figure 6(a)*, we observe that the disk bandwidth requirement curve is trough shaped, with the curve bottoming out at 60 sec. Below this threshold value, we permit fewer number of sessions to be shared and hence more accesses are directed to the disk. Above this value, we try to share more sessions, but since more buffers are occupied by the data that is to be preserved over longer periods of time, less memory is available for sharing of new requests and hence frequent trips are required to the disk. It is a very important trade-off in this technique. Having determined the value of *max_iat* as 60 for the parameter values mentioned, we modify equation (1) as follows:

$$max\_iat = 60\left(\frac{Arrival\_Mean}{10}\right)\left(\frac{Buffer\_Size}{1280}\right)\left(\frac{2}{Data\_Rate}\right) \quad (5)$$

The constants in each of the paranthesis corresponds to the default value of that parameter.

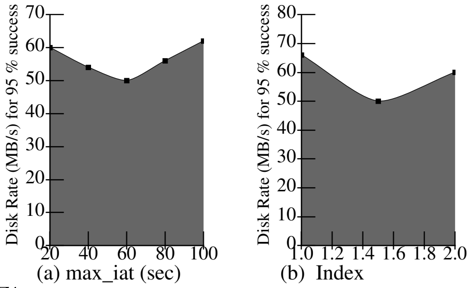


Figure 6: Variation in disk bandwidth requirements

However as we performed more tests, we noted that the performance of SHR scheme was still not satisfactory. Closer investigation revealed that the effect of buffer size is not linear but is complex. Hence we varied the *index* (exponent) of the buffer size component in equation (5) and plotted the required bandwidth as shown in *figure 6(b)*. From this we determined that the optimal value for the index is close to 1.5 and we arrived at the following equation. A linear function of the other components showed good behaviour.

$$max\_iat = 60\left(\frac{Arrival\_Mean}{10}\right)\left(\frac{Buffer\_Size}{1280}\right)^{1.5}\left(\frac{2}{Data\_Rate}\right) \quad (6)$$

We found this heuristic to be satisfactory and have used it in the tests discussed in the next few subsections.

8.2 Comparison of UAT and SHR

Here we analyze the behavior of UAT and SHR and discuss how continuous media caching helps to improve the performance of the system. *Figure 7(a)* shows the effect of the mean inter-arrival time on the disk bandwidth requirement. For smaller values of mean inter-arrival time, higher disk bandwidths are needed by UAT compare to SHR to satisfy the requests. The disk bandwidth requirement drops gradually as the mean inter-arrival time increases. This shows the ability of continuous media caching schemes to improve the number of concurrent requests that can be handled by a system at high arrival rates.

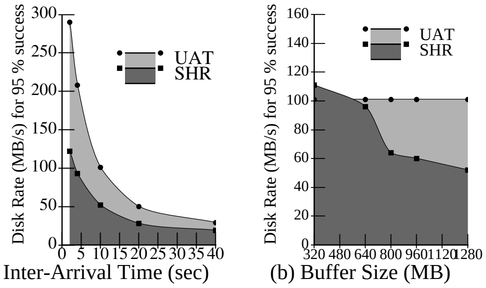


Figure 7: Effect of Inter Arrival Time and Buffer Size

The effect of buffer size on the performance of the schemes is studied in *figure 7(b)*. In general if the buffer space available is low, buffer management schemes experience poor performance. It can be observed that at lower buffer sizes SHR is not able to exploit sharing but as the buffer size available increases, SHR uses the extra buffer to exploit sharing and hence begins to outperform UAT. It can be noted that UAT does not utilize the available buffer resources to the maximum whereas SHR does. Hence UAT requires a constant disk bandwidth while the disk bandwidth for SHR reduces steadily with an increase in buffer size. Our heuristics have been carefully designed such that the SHR scheme behaves like the UAT scheme when the available buffer size is small and uses sharing only if more buffer is available.

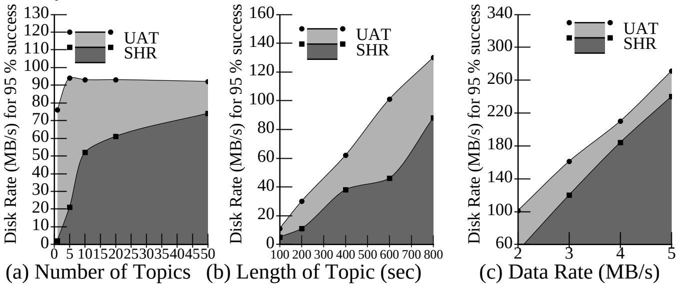
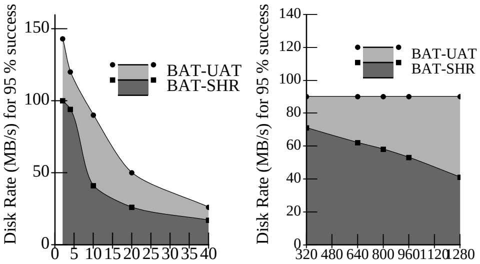


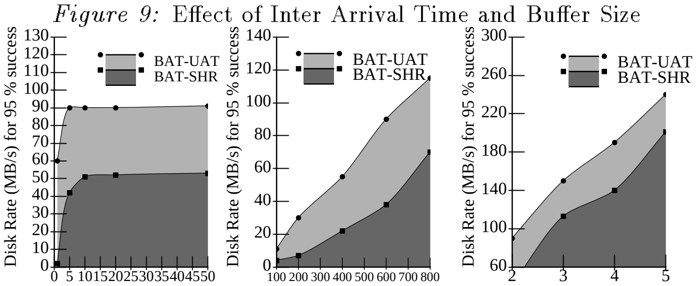
Figure 8: Effect of Topic Parameters

Next we compare the effect that the number of topics have on the performance of UAT and SHR. This test is important since the probability of locating a required segment in memory with or without planning depends upon the number of topics, *i.e.*, the lesser the number of topics, the higher the probability of finding a required segment. It can be observed in *figure 8(a)* that the increase in bandwidth is initially steep and then becomes more gradual. It can also be noticed that SHR dominates the performance (with a lesser disk bandwidth requirement) even as the number of topics increases. *Figure 8(b)* explores the effect of the length of the topic on the performance of the different schemes. This is important due to the fact that increasing the length of the topic results in a request taking longer to complete and hence the number of concurrent customers in the system increases, which in turn demands increased buffer and disk bandwidth. Thus the disk bandwidth requirements of the two policies increases with an increase in topic length. However if SHR is used, some of the concurrent sessions are shared and hence the disk bandwidth requirement for SHR is less than that of UAT. Another important parameter to be studied is the average data rate of the segments. In the MPEG-2 standards, the average data rate for a compressed high quality video is around 2MB/s. Hence we have performed tests for an average data rate ranging from 2-5MB/s. As expected, we note from *figure 8(c)* that with increasing data rates, the disk bandwidth requirement increases. However we observe that SHR requires less disk bandwidth than UAT. As the data rate increases, the gap between UAT and SHR decreases. To prevent larger buffer space being occupied by caching huge amounts of data, the number of data accesses directed to the disk increases.



(a) Inter-Arrival Time (sec) (b) Buffer Size (MB)

Figure 9: Effect of Inter Arrival Time and Buffer Size



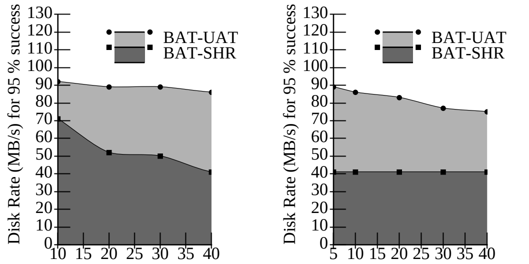
(a) Number of Topics (b) Length of Topic (sec) (c) Data Rate (MB/s)

Figure 10: Effect of Topic Parameters

8.3 Comparison of BAT-UAT and BAT-SHR

The batching schemes also behave similar to the two schemes discussed earlier, the main difference being that requests for the same topic are grouped and hence the overall disk bandwidth requirements is less in all cases. The results of these tests are shown in *figures 9 & 10*. It can be seen that the gray regions grow wider indicating that the benefits from continuous media caching is even larger if used in conjunction with batching.

We now discuss results of some additional tests which are based on varying the batching parameters, *i.e.*, tolerable response time and scheduling period. The effect of the maximum tolerable response time on the two batching schemes is shown in *figure 11(a)*. Here we observe that the drop in disk bandwidth requirement for BAT-SHR is steeper compared to BAT-UAT. This clearly indicates that continuous media caching in conjunction with batching takes advantage of relaxed response times. *Figure 11(b)* indicates more precisely how continuous media caching increases the effect of sharing. While it can be observed that, at low scheduling intervals, the effect of sharing is not completely achieved by BAT-UAT through batching, BAT-SHR achieves this by using continuous media caching in conjunction with batching. Hence the disk bandwidth required for BAT-SHR is almost the same irrespective of the scheduling period while BAT-UAT requires relatively higher disk bandwidth at smaller scheduling periods. As discussed earlier, in order to satisfy the response time requirements, it might be necessary to have smaller scheduling periods. To have better performance (lesser disk bandwidth) at smaller scheduling periods, BAT-SHR scores over BAT-UAT.



(a) Tolerable Response Time (sec) (b) Scheduling Period (sec)

Figure 11: Effect of Batching Parameters

Through all the above experiments, we have been able to demonstrate that information about past and future access can be used to exploit continuous media sharing, thereby substantially enhancing the performance of multimedia database systems even under a variety of conditions.

9 Conclusion

Since the data rates in multimedia database systems are very high, the I/O system could potentially limit the number of concurrent sessions supported. In this paper, we addressed this problem by exploring the potential benefits of continuous media sharing. For our study, we considered the application of News-On-Demand-Service. By effectively utilizing the advance knowledge of the multimedia stream to be accessed, our continuous media caching technique preserves buffers in a controlled fashion for use by subsequent users requesting the same data, thereby reducing disk accesses. We also presented heuristics that help the system decide whether a new request is to be shared with an existing one. Using continuous media caching we improved the UAT scheme to develop the SHR scheme and, in conjunction with a batching scheme BAT-UAT, created BAT-SHR, a new scheme that compounds the effect of sharing. We have also done a preliminary analysis of our schemes to determine the enhancements necessary to support fast-forward and rewind but have not included it here to keep the discussion short (details can be found in [15]).

We studied the impact of continuous media sharing by measuring the disk bandwidth required for 95% of the requests to succeed. We observed that the disk bandwidth required for SHR is low in all cases. This verifies the fact that SHR reduces the number of disk I/Os by promoting sharing and can give good performance even at low disk bandwidth. Similar observations were also made about BAT-SHR, substantiating the fact that continuous media caching can provide significant performance gains when used with batching schemes. In particular, tests to study the effect of scheduling period indicated that the disk bandwidth requirement of BAT-SHR is constant since sharing that could not be achieved through batching is obtained via continuous media caching. SHR schemes always performs better than UAT schemes and are as good in the worst case.

Our sharing model is quite general and this technique is useful when the pattern of data accesses is known in advance and can be utilized by many other multimedia applications like Video-On-Demand. It can also be used in video database environments where there are a large number of requests for some hot topics (*e.g.* home shopping videos of newly released products). Currently the data fetch/transfer time between a tertiary and a secondary device is high and this is one of the areas in multimedia that needs immediate attention. Our technique can be modified and used for efficient data management on the secondary device in such an environment.

References

[1] D. P. Anderson, Y. Osawa and R. Govindan, "A File System for Continuous Media", in *ACM Transactions on Computer Systems*, Vol. 10, No. 4, November 1992, pp 311-337.

[2] D. P. Anderson, "Metascheduling for Continuous Media", in *ACM Transactions on Computer Systems*, Vol. 11, No. 3, August 1993, pp 226-252.

[3] P. B. Berra et al, "Issues in Networking and Data Management of Distributed Multimedia Systems", in *Proceedings of Symposium on High Performance Distributed Computing*, September 1992.

[4] S.Christodoulakis, "Multimedia Data Base Management: Applications and Problems - A Position Paper", in *Proc. of ACM SIGMOD*, 1985, pp 304-305.

[5] S.Christodoulakis et al, "An Object-Oriented Architecture for Multimedia Information Systems", in *The Quarterly Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, Vol. 14, No. 3, September 1991, pp 4-15.

[6] A. Dan, D. Sitaram and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching", *IBM Research Report RC 19381*, 1994.

[7] J. K. Dey, James D. Salehi, James F. Kurose, Don Towsley, "Providing VCR Capabilities in Large-Scale Video Servers", in Proc. of *ACM Multimedia*, San Francisco, October 1994.

[8] J.K. Dey, C.H. Shih and M. Kumar, "Storage Subsystem Design in a Large Multimedia Server for High-Speed Network Environments", in *IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, San Jose, February 1994.

[9] W. Effelsberg and T. Haerder, "Principles of Database Buffer Management", in *ACM Transactions on Database Systems*, Vol. 9, No. 4, December 1984, pp 560-595.

[10] L. Felician, "Simulative and Analytical Studies on Performances in Large Multimedia Databases", in *Information Systems*, Vol. 15, No. 4, pp 417-427, 1990.

[11] J. Gecsei, *The Architecture of Videotex Systems*, Prentice-Hall, Englewood Cliffs NJ, 1983.

[12] S. Gibbs, C. Breiteneder and D. Tsichritzis, "Audio/Video Databases: An Object Oriented Approach", in *Proceedings of 9th International Conference on Data Engineering*, Vienna, April 1993, pp 381-390.

[13] J. Gemmel and S. Christodoulakis, "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval", in *ACM Transactions on Information Systems*, Vol. 10, No. 1, January 1992.

[14] J. Huang and J. A. Stankovic, "Buffer Management in Real-Time Databases", *Technical Report 90-65*, Department of Computer Science, University of Massachusetts, July 1990.

[15] M. Kamath, K. Ramamritham and D. Towsley, "Buffer Management for Continuous Media Sharing in Multimedia Database Systems", *Technical Report 94-11*, Department of Computer Science, University of Massachusetts, February 1994.

[16] K. Kawagoe, "Continuous Media Data Management", in *SIGMOD Record*, Vol. 20, No. 3, September 1991, pp 74-75

[17] W. Klas, E.J. Neuhold and M. Shrefl, "Using an Object-Oriented Approach to Model Multimedia Data", in *Computer Communications*, Vol. 13, No. 4, pp 204-216, 1990.

[18] D. L. Gall, "MPEG: A Video Standard for Multimedia Applications", in *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp 46-58.

[19] T.D.C.Little, A.Gafoor et al., "Multimedia Synchronization", in *The Quarterly Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, Vol. 14, No. 3, September 1991, pp 26-35.

[20] P. Lougher and D. Shepherd, "The Design of a Storage Server for Continuous Media", in *The Computer Journal*, Vol. 36, No. 1, 1993.

[21] R. Ng and J. Yang, "Maximizing Buffer and Disk Utilization for News-on-Demand", in *Proc. of the 20th VLDB Conference*, Santiago, Chile, 1994, pp 451-462.

[22] R. Staehli and J. Walpole, "Constrained-Latency Storage Access", in *Computer*, March 1993, pp 44-53.

[23] R.Steinmetz, "Synchronization Properties in Multimedia Systems", in *IEEE Journal on Selected Areas of Communication*, Vol.8, No.3, April 1990, pp 401-412.

[24] P. Venkat Rangan and Harrick M. Vin, "Efficient Storage Techniques for Digital Continuous Multimedia", in *IEEE Transactions on Knowledge and Data Engineering*, August 1993.

[25] D.Woeld and W.Kim, "Multimedia Information Management in an Object-Oriented System", in *Proc. of the 13th VLDB Conference*, Brighton, 1987, pp 319-329.

[26] C. Yu et al, "Efficient Placement of Audio Data on Optical Disks for Real-Time Applications", *Communications of the ACM*, Vol. 32, No. 7, July 1989, pp 862-871.

[27] G.K. Zipf, *Human Behaviour and the Principles of Least Effort*, Addison-Wesley, Reading MA, 1949.