A Real-Time Prefetching Method for Continuous Media Playback

Sungchae Lim and *Myoung Ho Kim*

Dept. of Computer Science
Korea Advanced Institute of Science and Technology
373-1, Kusung-dong, Yusung-gu, Taejon, 305-701, Korea

{*sclim,mhkim*}*@dbserver.kaist.ac.kr*

FAX: +82-42-869-3510

Abstract

Continuous media(CM) such as digital-coded videos and sounds impose time constraints on multimedia systems to prevent hiccups. In this paper we propose a new CM playback method that is suitable for handling CM streams with various playback rates and rather short duration of playback. Our proposed method is aimed at providing both good response times and high disk utilization. For better disk utilization, the SCAN algorithm is employed for retrieving CM data and then real-time bulk-scan is conducted. Along with this bulk-scan, we propose an EDF-style algorithm with priority-driven property. Owing to flexibility in our disk scanning, our method is able to provide good response time as well as high disk utilization. Through experimental comparisons over our method and the earlier round-style methods, we show the performance advantages.

Key words: *multimedia, continuous media, real-time disk scheduling*

1. Introduction

Recent advances in network technologies and the computing power enable the information system that delivers multimedia data to remote clients in online mode. To establish such a multimedia system, it is required to develop an efficient disk scheduling method that fulfills temporal requirements of continuous media(CM) objects such as digital-coded videos and sounds. The CM object imposes temporal requirements on multimedia systems to prevent undesirable interruptions of playback, called *hiccups*. To fulfill this requirement, it is crucial to develop an efficient CM playback method consisting of real-time disk scheduling and an adequate strategy for admitting CM streams within the limitation of system resource.

To this end, [2, 3, 4, 5] have proposed the round-style method that fixes a constant period and performs disk scheduling based on the constant period. The round-style method prefetches CM data of all served streams during a period and makes the prefetched data be consumed for the next period. As the round-style method, schemes Sweep and Fixed-Stretch are popularly accepted. The two schemes use different disk scheduling policies for data prefetch, i.e., the former uses the elevator-SCAN algorithm and the latter moves the disk head freely retrieving data in a fixed order of streams. Although the round-style methods can ensure timely prefetch in a simple manner, they have some shortcomings from inflexibility in disk usage. Because the round-style methods assign channels to streams in a static fashion from period to period, they cannot efficiently serve CM streams with rather short playback time and low playback rates, e.g., low-quality videos and MPEG-coded sounds. To solve the problem, we make each stream have its own prefetch period and perform channel assignment based on the earliest-deadline-first(EDF) algorithm. Owing to disk scheduling efficiency, our method provides good performance

2. Backgrounds

2.1. CM Data Organization

A simple clustering method for CM data is likely to store all of each CM data continuously on the disk. This clustering tends to cause disk space fragmentations and makes update operations on CM data hard. Instead, it is more reasonable that CM data are clustered in the unit of a disk track as in literature[1, 2, 4]. In this paper we also assume that CM data consist of *segments* that are equal-sized and made up of multiple tracks on the same disk cylinder. Since each segment is continuously stored, its retrieval incurs one time of

0-7695-0281-4/99 $10.00

disk seeking. For easy updates, segments of any CM stream can be freely located on the disk. In addition, A CM stream issues requests for data with respect to its data assumption rate, called *playback rate*. A request for data contains the location information, i.e., *id*s of the beginning and last tracks of an associated segment.

2.2. Real-time Disk Scheduling

To prevent interruptions of CM playback, streams' requests have to be served in time. Thus some real-time disk scheduling schemes have been proposed for timely data retrieval.

A. Scheme Scan-EDF

When requests for data are issued by streams, the earliest-deadline-first(EDF) algorithm can be applied for choosing the next request to be served. Although this algorithm is very useful for scheduling real-time tasks[9], it suffers from excessive seeking overhead, while serving requests according to deadline order. To prevent such a problem, Scan-EDF has been proposed as an integration of the EDF algorithm with the SCAN algorithm. In Scan-EDF, requests with the same deadline are grouped into a set $\mathcal{R}$, and then requests in $\mathcal{R}$ are served through the Elevator-SCAN algorithm. If such $\mathcal{R}$ contains too small numbers of requests, Scan-EDF will produce a disk scheduling result similar to that done by scheme EDF; in contrast, if R contains so many requests, Scan-EDF amounts to scheme Scan. For performance gains in Scan-EDF, deadlines of requests must be appropriately arranged for seek-time optimization.

But, since in most cases deadline arrangement for seek optimization cannot be easily done, performance advantage is not guaranteed if streams with various playback rates have to be served. In addition, since Scan-EDF does not provides an efficient admission strategy preventing playback hiccups, Scan-EDF cannot efficiently handle a situation where CM streams are frequently constructed and disappears over time. Thus, round-style schemes are widely used for timely data retrieval of CM streams[2, 3].

B. Round-Style Schemes

In the round-style methods, all streams request their disk bandwidth based on a given time period, called *round*. For example, suppose the length of round is fixed as 2 seconds. And then, a stream of playback rate b (Kbps) will issue requests for data of size $2b$ (Kbits) with period 2 seconds. Because disk scheduling can be repeatedly performed across rounds and requests are issued with the same period, the round-style methods can easily prevent hiccups of playback. These methods can be categorized into two schemes Fixed-Stretch and Sweep depending on the used disk scheduling mechanism. The former serves requests in a fixed order of streams and has the round length including the worst-case seek time. Meanwhile, the latter employs the Scan algorithm for better seek optimization. A stream in Sweep can be served at the beginning of round i and next served at the end of round $i+1$ depending on request's disk positions. So, the maximum retrieval latency of Sweep is two times longer than that of Fixed-Stretch.

At admission time, each stream requires disk bandwidth based on a constant round period. Note that round length is fixed for performance consideration, rather than varying it over time[3]. This may cause a problem called *bandwidth fragmentation*. To illustrate, suppose that a segment is D in size and a round is L in length. Then, a stream requests disk bandwidth of size nD/L, where n is the number of segments requested in a round. So, the average of bandwidth fragmentations is a half of D/L. Disk wastes from bandwidth fragmentation can be considerable, if many streams are concurrently scheduled. If we make L be large for smaller fragmentation, it will always increase *response time* that is defined as time elapsed between request point and starting point of playback[2, 3].

3. Proposed Method for CM Playback

Although the Scan algorithm is likely to provide good disk utilization owing to its seek optimization feature, this algorithm has difficulty in serving real-time requests. First, time taken to read a given segment depends on its relative position to other segments being served through the same disk scan. Thus, it is difficult to guarantee both of timely retrieval and reasonable seek optimization; such a problem is found in Scan-EDF. Next, the Scan algorithm cannot efficiently serve a specific request with service urgency. This is because service delay can be quite long if requested data is located behind current movement of the disk head. Therefore, in scan-based schemes response time is longer than the average duration of the disk scan; in Sweep, the average response time is longer than a round period. This results in performance degradation if playback of streams are frequently started and finished over time. To resolve these problems of the Scan algorithm, we propose a flexible disk scan method that guarantees high disk utilization and good response time.

3.1. Basic Mechanism

To obtain seek optimization, we choose an integral number k and enable streams to issue requests for data so that k or more segments can be retrieved by a single disk scan while meeting deadlines of requests. We call the disk scan that retrieves multiple segments in batch fashion as *bulk-scan*, and say that bulk-scan has parameter k, if each bulk-scan reads at least k segments in the presence of intensive data requests. For bulk-scan with parameter k, we first make

deadlines of requests arise with time intervals that are sufficient for scanning any k or more segments. Then, the EDF policy is applied to choose requests to be served by each bulk-scan.

As rotational delay can be ignored for reading the segment that is organized in the unit of the disk track[6], the bulk-scan time depends on only the amount and positions of the retrieved segments without effects of rotational delay. Let *seek_time(d)* be the function that gives seek-time for relocating the disk head on a target disk track across d cylinders. According to [5], total seek-time for reading multiple segments by a disk scan is maximized when the target segments are evenly scattered over the disk cylinders. Thus, if one segment consists of X tracks and the total cylinder number of the disk is N_{cyl}, the upper bound of bulk-scan time for reading k segments is as follows:

$$BT(k) = k \cdot X \cdot Revolution\_Time + (k+1) \cdot seek\_time(\lceil \frac{N_{cyl}}{k+1} \rceil) \quad (1)$$

We let $T = BT(k)$ and make request period of each stream be any one of $2T$, $3T$,.., max_pT, where max_p is an integer constant chosen by our proposed CM server. Within this range, request period of a certain stream is determined so that bandwidth fragmentation can be minimized. For example, consider a situation where a stream with playback rate b is newly admitted. At this time, disk bandwidth used by the stream is specified by a pair of request period and the number of requested segments. So the stream will have candidate pairs such as $(i,\ y_i)(2 \leq i \leq max\_p)$ satisfying $\frac{(y_i-1)}{iT} < b \leq \frac{y_iD}{iT}$; here, $\frac{y_iD}{iT}$ is the amount of disk bandwidth used by this stream. Among the pairs, we choose any one with the smallest bandwidth fragmentation. We call this chosen pair as a *data request pattern*(DRP) and let $\langle p_i,\ n_i \rangle$ be DRP of stream S_i from now on.

To illustrate how requests are issued by streams according to their DRPs, suppose that a stream S_i with DRP $\langle 5, 4 \rangle$. We first fetch four segments of the head data of this stream and then begin playback of S_i, where data retrieval performed before playback start is called *initial prefetch*. If initial prefetch is completed between interval $(T_{j-1}, T_j]$, stream S_i can be played back from time T_j in our method and issues four requests before $T_{j+5n} (n = 0, 1, \ldots)$. In this case, four requests in each period have the same deadline $T_{j+5n} (n = 1, 2, \ldots)$.

In turn, we describe how to serve requests through bulk-scan with parameter k. When beginning a bulk-scan at time $t \in (T_{j-1}, T_j]$, our CM server first computes the maximum number of segments able to be scanned within time T_{j+1}. This is given by using (1), i.e., the maximum number is the largest integer k' satisfying $BT(k') \leq T_{j+1} - t$. And then, we choose requests to be served by using the following steps within the maximum number.

(1) Suppose the latest deadline of currently issued requests is T_{j+n}. Let $G1$, $G2$,..., Gn be sets of requests with deadlines $T_{j+1}, T_{j+2}, \ldots, T_{j+n}$, respectively. Choose the largest integer i such that $|G = G1 \cup G2 \cup \ldots \cup Gi| \leq k'$. If $i = n$, return G; otherwise, go to the next step.

(2) Define the cylinder distance between the outermost and innermost segments retrieved by a bulk-scan as *scan distance* of the bulk-scan. Select a request r from G_{i+1} such that bulk-scan for serving $G \cup \{r\}$ has the smallest scan distance, and input this request into G. If $|G| = k'$, return G; otherwise, repeat this step.

To obtain high space locality among retrieved segments, we carefully choose segments through step (2). Here, we refer to a bulk-scan that starts at interval $(T_{j-1}, T_j]$ and ends before T_{j+1} as $SCAN_j$. Since any $SCAN_j$ has at least time T as a duration time, our bulk-scan with parameter k provides disk bandwidth of at least kD/T(Kbps). If we can properly admit new streams, our mechanism consisting of bulk-scan and the EDF policy can guarantee non-hiccuped playback.

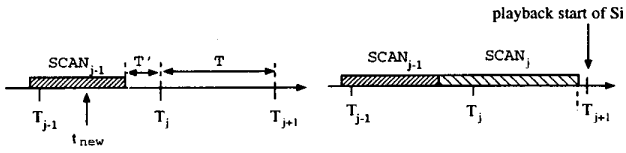
For an admission mechanism, we apply the schedulability test of the preemptive EDF scheduler for periodic tasks[9]. That is, we associate a stream S_i having DRP $\langle p_i,\ n_i \rangle$ with a virtual periodic task having period p_iT and computation time $\frac{T}{k}n_i$. Here, serving of a request for data is considered the occupation of time slot with size $\frac{T}{k}$. In this context, bandwidth reservation of S_i is computed as $\frac{n_i}{kp_i}$, and the total reservation is defined as $R_{used} = \sum_{i=1}^{\ell} \frac{n_i}{kp_i}$, where ℓ is the number of the admitted streams. Thus, we admit a new stream only while current R_{used} plus bandwidth reservation of the newly assigned DRP does not exceed 1. If there is no DRP satisfying such condition, admission of the stream will be refused and its admission is delayed until the end of playback of any other streams. The correctness proof is omitted for the space limitation.

Let Δ^{new} be the set of streams that do not finish the initial prefetch phase, and Δ^{new} be the set of other streams that already begin playback. If we favorably serve requests of initial prefetch, the response time can be reduced. For this, our CM server computes the minimum number of requests has to be served for Δ's smooth playback, and serve only the minimum requests for Δ. Such computation is performed at the beginning of each bulk-scan. Consider a stream S_i that requires r segments at the beginning of $SCAN_j$. If deadline of the requests is T_{j+n}, ratio of disk bandwidth for S_i amounts to $\frac{r}{k \cdot n}$ since those requests can be served from $SCAN_j$ to $SCAN_{n+j-1}$. With this observation, the minimum request number can be computed as $\lceil k \cdot \sum \frac{r_i}{k(d_i - j)} \rceil$. Here, notations r_i and d_i indicate that stream S_i has r_i requests has to be served until T_{d_i}. Therefore, in $SCAN_j$, N_j^{new} requests can be served for fast

prefetch of Δ^{new}:

$$N_j^{new} = k_j^{'} - \lceil k \sum_{S_i \in \Delta} \frac{r_i}{k(d_i - j)} \rceil = k_j^{'} - \lceil \sum_{S_i \in \Delta} \frac{r_i}{d_i - j} \rceil \quad (2)$$

Before beginning $SCAN_j$, we choose up to N_j^{new} requests of Δ^{new} in a FIFO manner, and let them be G_0. By assigning the highest priority to G_0 at the step (1) above, all requests in $G0$ will be served by $SCAN_j$. Figure 1 illustrates how the bulk-scan is performed in our method. Figure 1.(a) show a situation where $SCAN_{j-1}$ ends at time $T_j - T'$ and playback of a stream S_i is newly requested during $SCAN_{j-1}$. If $n_i \geq N_{j-1}^{new}$, initial prefetch for S_i will be completed through $SCAN_j$ as in Figure 1.(b). In our bulk-scan method, duration of a bulk-scan dynamically varies according to workload of issued requests. This property makes our disk scheduling method reclaim efficiently disk times that may be left over without flexible bulk-scans.



(a) playback request of Si at time t_{new} (b) initial prefetch of Si through $SCAN_j$

Figure 1. Bulk-scan for shortening time of initial prefetch

3.2. Implementing the Proposed Method

For implementation, we need to give adequate values to parameters T and max_p. The size of T must be chosen by considering trade-offs between seek optimization and response time. While the greater T provides better seek optimization in general, it may hurt the average response time since waiting time until the beginning of the next disk scan is enlarged. In our research, T is chosen as a value between 2 second and 3 second within which ratio of seek time to the total scan time becomes less than 20%. After choosing T, we determine max_p so that total bandwidth fragmentation is made below a given level. In our method a single stream has bandwidth fragmentation of at maximum $\frac{D}{max\_pT}$. By adjusting max_p, total bandwidth fragmentations is made below 5% of kD/T.

Based on T and max_p, buffer memory is pre-allocated so that our CM server does not reject admission of a stream because of memory space shortage. We make the buffer memory be shared among served steams to minimize memory space requirement. In this case, memory size required by a stream S_i is the same as the amount of data consumed during its one request period, i.e. $D \cdot n_i$[8]. Thus, the minimum memory size for our CM server is equal to $D \cdot k \cdot max\_p$ (Kbits).

The followings are the steps for our proposed CM server.

Step 0: {Initialization phase} $\Delta \leftarrow \Delta^{new} \leftarrow \emptyset$; $R_{res} \leftarrow 0.0$; $serial \leftarrow 1$.

Step 1: {Adjusting parameter R_{used}} For each stream S_i that finishes playback, decrease R_{used} by $\frac{n_i}{kp_i}$; admit new streams while $R_{used} \leq 1$.

Step 2: {Choosing requests to be served} Select up to N_{serial}^{new} number of requests from Δ^{new}; select the remain requests from Δ within $k_{serial}^{'}$; serve selected requests through $SCAN_{serial}$.

Step 3: {Manipulation of streams} Play back streams in Δ^{new} that have completed initial prefetch from $T_{serial+1}$; make streams in Δ whose request period ends at $T_{serial+1}$ issue requests for the next period.

Step 4: {Synchronization phase} Let $serial = serial + 1$; wait until $T_{serial-1}$ if $SCAN_{serial}$ ends before $T_{serial-1}$; go to Step 1.

4. Experimental Results

We perform some experiments to briefly show performance advantages of our proposed method. For this, we analyze efficiencies of disk usage between our proposed method and the earlier schemes such as schemes Sweep and Fixed-Stretch, by using the HP 97560 disk drive in [6]. In the experiment, we assume that playback rates and periods of playback are uniformly distributed over ranges [20 Kbps, 500 Kbps] and [2 minutes, 4 minutes], respectively. The disk track has the size of 36 KBytes, and buffer memory is allocated up to 60 MBytes.

We first examine the response time which is elapsed time from playback request to playback start. In the experiment, playback requests are maintained in the waiting queue until disk bandwidth is available for admission, and served in a FIFO manner. Figure 2 shows how the average response time varies with arrival rate of playback requests; here, we use the segment composed of two disk tracks. We experiment with schemes Fixed-Stretch and Sweep with periods of 3 and 10 seconds. In case of our method, we set parameter T as 2.5 seconds and 3 seconds, which is denoted by EDF*-T in the figure. With these values of T, EDF*-3 and EDF*-2.5 have max_pi of 10 and 12, respectively. Although sweep Fixed-Stretch provides the best response time in a range of low arrival rates of requests, this scheme is so quickly overloaded owing to its heavy losses of disk bandwidth. Meanwhile, scheme Sweep works better for the situation where workload is somewhat high, compared with scheme Fixed-Stretch. But, this scheme gives poor response times. In contrast to these schemes, our method provides

better response time than scheme Sweep, and work well in the range of high arrival rates.

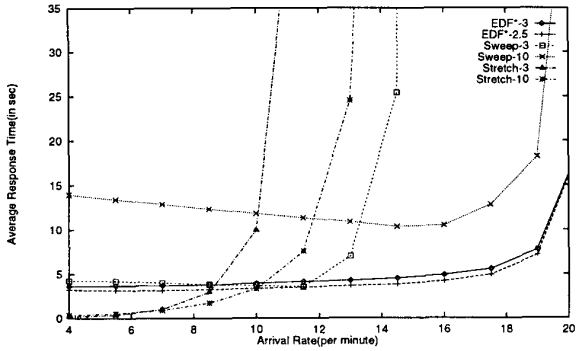


Figure 2. Comparisons of the average response times

To see more clearly throughput of each method, we reject playback requests that cannot be immediately admitted for insufficient disk bandwidth, rather than keeping them in the queue. Then, we calculate ratio of the rejected requests with respect to all of playback requests. In this experiment, the smaller rejection ratio indicates that the better throughput is achieved by the method. To clearly show the results of this experiment, we calculate the rejection ratio for high arrival rates. As known from Figure 3, our proposed method guarantees the best throughput due to its flexible channel usage and low bandwidth fragmentation.

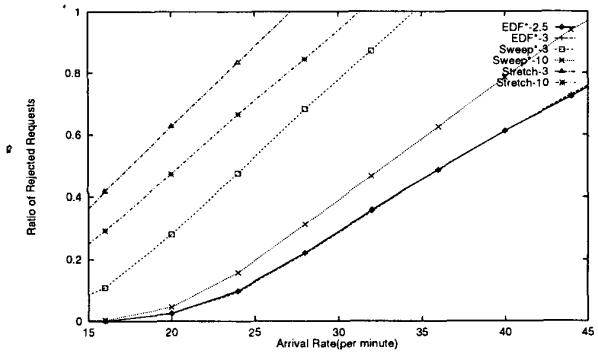


Figure 3. Ratio of the rejected playback requests to overall requests

5. Conclusion

In this paper we have proposed a CM playback method that is based on the bulk-scan and the well-known EDF algorithm. To guarantee reasonable seek optimization, we enable the bulk-scan to retrieve k or greater number of segments while meeting deadlines of requests for data. For this, we developed an admission control strategy using the schedulability test method of the preemptive EDF server. For fast response time, we compute the maximum disk bandwidth that can be used for prefetch of head data of newly admitted streams. Owing to minimized bandwidth fragmentations and flexible disk scans, our proposed method can efficiently support CM streams with a variety of playback rates.

References

[1] Gin-Kou Ma, Chiung-Shien Wu, Mei-Chian Liu and Bao-Shuh P. Lin, "Efficient Real-time Data Retrieval through Scalable Multimedia Storage," *Proceedings of ACM Multimedia,* 1997.

[2] Huang-Jen Chen and Thomas D.C. Little, "Storage Allocation Policies for Time-Dependent Multimedia Data," *IEEE Trans. Knowledge and Data Engineering*, Vol. 8, Oct. 1996.

[3] Edward Chang and Hector Garcia-Molina, "Effective Memory Use in a Media Server," *Proceedings of the VLDB Conference,* 1997.

[4] P. Venkat Rangan and Harrick M. Vin, "Efficient Storage Techniques for Digital Continuous Multimedia," *IEEE Trans. Knowledge and Data Engineering*, Aug. 1993.

[5] Yen-Jen Oyang, et al. , "Design of Multimedia Storage Systems," *IEEE Data Engineering*, pp. 457-465, 1995.

[6] Chris Ruemmler and John Wilkes, "An Introduction to Disk Drive Modeling," *IEEE Computer*, March 1994.

[7] Antoine N. Mourad, "Issues in the Design of a Storage Server for Video-on-Demand," *Multimedia Systems*, pp. 70-86, 1996.

[8] Edward Chang and Yi-Yen Chen, "Minimizing Memory Requirements in Media Servers," *Stanford Technical Report SIDL-WP-1990-0050*, Oct. 1996.

[9] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of ACM*, Vol. 20, No. 1, Jan. 1973.

[10] Houssine Chetto and Maryline Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm," *IEEE Trans. Software Engineering,"* Vol. 15, No. 10, Oct. 1989.