Multimedia Systems 8: 105–117 (2000)

Multimedia Systems

$\mathcal{SMDP}$: minimizing buffer requirements for continuous media servers

Youjip Won[1,⋆], **Jaideep Srivastava**[2]

[1] Division of Electrical and Computer Engineering, Hanyang University, Seoul, Korea; e-mail: yjwon@email.hanyang.ac.kr
[2] Department of Computer Science and Engineering, University of Minnesota, Minneapolis, Minn, USA; e-mail: srivasta@cs.umn.edu

Abstract. Excessive buffer requirement to handle continuous-media playbacks is an impediment to cost-effective provisioning for on-line video retrieval. Given the skewed distribution of video popularity, it is expected that often there are concurrent playbacks of the same video file within a short time interval. This creates an opportunity to batch multiple requests and to service them with a single stream from the disk without violating the on-demand constraint. However, there is a need to keep data in memory between successive uses to do this. This leads to a buffer space trade-off between servicing a request in *memory mode* vs. servicing it in *disk-mode*. In this work, we develop a novel algorithm to minimize the buffer requirement to support a set of concurrent playbacks. One of the beauties of the proposed scheme is that it enables the server to dynamically adapt to the changing workload while minimizing the total buffer space requirement. Our algorithm makes a significant contribution in decreasing the total buffer requirement, especially when the user access pattern is biased in favor of a small set of files. The idea of the proposed scheme is modeled in detail using an analytical formulation, and optimality of the algorithm is proved. An analytical framework is developed so that the proposed scheme can be used in combination with various existing disk-scheduling strategies. Our simulation results confirm that under certain circumstances, it is much more resource efficient to support some of the playbacks in memory mode and subsequently the proposed scheme enables the server to minimize the overall buffer space requirement.

Key words: Multimedia – Disk scheduling – Data retrieval – Buffer management – Synchronization

1 Introduction

1.1 Motivation

A principal goal in designing a *VOD* server is to support as many concurrent streams as possible with minimum amount of resources, e.g., disk bandwidth and memory buffer. In supporting *jitter-free* playback of continuous-media streams, a certain amount of memory buffer needs to be allocated to synchronize the asynchronous disk operations, *retrieval*, with synchronous playback oepration. An important factor governing synchronization buffer size is the disk-scheduling strategy. A common characteristic of disk-scheduling in continuous-media playback is that the synchronization buffer size increases extremely rapidly as disk bandwidth utilization reaches 100%.

It is not unreasonable to assume that the user access pattern shows some degree of *skewness*, i.e., most of the requests are for a small number of *hot* files, while the rest of the files are rarely accessed. This characteristic was studied in a user survey on commercial video rental [vid92]. In a *VOD* environment, individual playbacks are interleaved according to their respective arrival times. Further, some of the playbacks are for the same file (or video title). Thus, in a certain situation, it may be better for a playback to use the data blocks which have been loaded by the preceding playback, instead of reading the blocks directly from the disk, which is called *memory-mode* playback. In memory-mode playback, playback does not require memory buffer for synchronization, but an appropriate amount of buffer space needs to be allocated to retain the data blocks loaded by the preceding stream in memory. The amount of buffer space for memory-mode service is linearly proportional to the interval length between the playbacks, and the playback bandwidth. Further, the buffer space requirement for memory-mode playback *is not* affected by aggregate disk bandwidth utilization.

Per playback buffer space requirement for *disk-mode* service, where playback reads the data blocks from the disk, increases with the aggregate disk bandwidth utilization, even though individual playback bandwidth requirement remains unchanged. Further, as the disk bandwidth utilization reaches 100%, the rate of increase becomes extremely high, whose asymptotic behavior follows $O(\frac{1}{1-utilization})$. There are a

⋆ This work was done while the author was at University of Minnesota, Minneapolis and was in part funded by grant No. 1999-1-303-001-3 from the interdisciplinar Research program of the KOSEF.
Correspondence to: Y. Won

number of ways to service a given set of playbacks. One extreme is to support all the playbacks from the disk retrieval and the other extreme is to pre-load all the required files into the main memory to avoid any disk operation. We exploit the characteristics of the two ways of supplying data blocks to a playback, and propose a scheme to compromise these two extremes, providing the best (or minimum) total buffer requirement.

1.2 Relation to the previous work

Recent years have seen a number of studies discussing the issues in multimedia server design [AOG92, CKY93, KHS94, RVR92, RW92, GKS96, TF98]. Anderson et al. [AOG92] modeled continuous-media playback from the view point of a consumer/producer relationship and established a mathematical model for its playback requirement. Initial focus in this area was on how to provide *continuity* in playback operation during frame retrieval from the disk, whose behavior is asynchronous in nature. Rangan et al. [RVR92] modeled the *FIFO* disk-scheduling algorithm to preserve the continuity requirement. Kenchammmna-Hosekote et al. [KHS94] used the *SCAN* scheduling algorithm in supporting continuous-media playback and established the formula for the continuity requirement. Chen et al. [CKY93] proposed the *group sweep scheduling(GSS)* algorithm which is a hybrid between *FIFO* and *SCAN* scheduling. Each scheduling policy has its own advantages and disadvantages, which are later discussed in detail. Reddy et al. [RW92] approached the problem of supporting continuity from the viewpoint of *real-time* systems. In addition to head-scheduling issues in a single disk, there have been a number of studies on how to design a disk subsystem for a continuous-media file server [GC92, Gem93, VGGG94a, VGGG94b, OBRS94]. A multimedia stream requires guaranteed disk bandwidth to produce a timely flow of data. The disk-scheduling algorithm focuses on supporting multiple streams without violating the timing constraints. [PD93, Ant96] expand the disk-scheduling algorithm to array of disks. It is possible that a playback uses the data blocks that have has been loaded by a preceding playback. In this case, memory buffer for disk I/O is not required for the latter stream. Dan and Sitaram investigated the idea of maintaining in memory the data blocks used by one playback for a certain period of time, to be utilized by the subsequent playback request, called *interval caching* [DS93]. They used a Markovian model to obtain the probability of future utilization of in-memory data blocks. Recent efforts have incorporated the concept of *zoning* in disk layout for continuous-media file systems [Gha95, Ren96].

1.3 Contributions

In this work, we first build an analytical framework for memory buffer requirement as a function of disk bandwidth utilization. We analyze the asymptotic behavior of the memory buffer requirement which is independent of individual disk-scheduling algorithms.

Due to the nature of an *on-demand* service environment, playback requests arrive at the server in a non-deterministic fashion and the overall disk bandwidth utilization changes dynamically. Subsequently, buffer requirements for individual disk-mode playbacks change dynamically as well. We develop an analytical framework which computes the buffer requirement of the disk-mode playback. We develop an algorithm, called $\mathcal{SMDP}$, which efficiently determines the service mode of the playback minimizing the overall memory buffer requirement. When there is a change in disk bandwidth utilization, i.e., *arrival of a playback* or *termination of a playback*, $\mathcal{SMDP}$ algorithm is dynamically executed to re-examine the service modes of the individual playbacks.

The rest of the paper is organized as follows: Sects. 2 and 3 describe the buffer requirement to maintain a set of playbacks from disk and memory, respectively. In Sect. 4, we explain the characteristics of each service mode. Section 5 describes the service-mode determination problem and the proposed approach. In Sect. 6, we generalize the proposed idea to a heterogeneous playback environment and explain the overhead of service mode transition. In Sects. 7 and 8, we provide simulation results for the proposed algorithm and the conclusion, respectively. The appendices contain a detailed analysis of the proposed scheme and the proofs of the lemmas.

2 Buffer requirements for continuous-media disk retrieval

In retrieving data blocks from a disk, a certain amount of buffer space needs to be dedicated to each session to synchronize the asynchronous disk retrievals and synchronous playback. We call the memory buffer for this purpose a synchronization buffer. To support multiple playback sessions concurrently, a set of disk retrieval (or read) operations need to be executed periodically. Individual disk-read operations have to be completed by their respective firm deadlines. To guarantee the completion of a disk-read operation by its deadline, a certain amount of system resources have to be reserved. Playback of a continuous-media file requires various types of resources, i.e., *disk bandwidth* and *main memory*[1]

2.1 The schedulability condition

We formulate the general constraints in disk-scheduling for continuous-media playback. Let $\mathbf{s} = \{s_1, \ldots, s_n\}$ be a set of n streams, and let r_i be the playback rate for stream s_i. n_i and b are the number of disk blocks to be fetched for s_i in a cycle and the size of a block, respectively. A *cycle* is the length of the time interval between successive bursts of disk-read operations. Let $T(\mathbf{s})$ be the length of the cycle for playbacks $\mathbf{s}$. To ensure jitter-free continuous-media playback, the amount of data blocks fetched from the disk within a cycle should be no less than the amount of data consumed by playback during the same amount of time. This constraint can be formulated as Eq. 1.

$$\forall i \; bn_i \geq r_i T(\mathbf{s}) \,. \tag{1}$$

[1] We assume that system bus bandwidth, CPU, etc. are at a lesser premium compared with *disk bandwidth* or *main memory*, and hence do not consider them in this context.

On the other hand, $T(\mathbf{s})$ should be no less than the total time required to read the blocks for all playbacks in a cycle. Each stream requires bn_i bytes to be loaded from the disk. Thus, $T(\mathbf{s})$ can be expressed as follows:

$$T(\mathbf{s}) \geq \sum_{i=1}^{n} \left(\frac{bn_i}{B_{max}} + \delta_i \right) \geq \sum_{i=1}^{n} \frac{bn_i}{B_{max}} + \sum_{i=1}^{n} \delta_i \,. \quad (2)$$

Total buffer size is proportional to the length of $T(\mathbf{s})$(Eq. 1). Thus, in our effort to find minimum buffer requirement in supporting a set of playbacks, we replace $\geq$ with = in further derivation step. δ_i in Eq. 2 is the head re-positioning overhead in switching the disk-read operation from one playback to another. Detailed modeling and analysis of operational overhead requires rather sophisticated techniques and an analytical formulation which incorporates all the factors, e.g., *seek*, *latency*, *cruise time*, *settle-down time*, *head change*, etc., is virtually impractical for our purpose. [RW94, WGP94] provide good summaries of disk internals. B_{max} is the maximum data transfer rate, which is governed by the rotational speed and magnetic density of the disk plate. In disk I/O operations, head repositioning overhead is non-trivial. Consider a disk spinning at 7200 RPM with 5 ms average seek time. It can transfer data at 5 MBps, i.e., it takes 0.4 ms to read a 2048-byte block. Considering a 5-ms average seek time and 4.2-ms average rotational latency($\frac{60}{7200 \cdot 2}$ s), it actually takes 9.6 ms to read a 2-KB block. In our example, 95% of the time in retrieving 2048-Byte block is spent on head positioning. To minimize the operational overhead in disk I/O, making the block size larger is often proposed as a solution [BR96], especially in bandwidth-intensive continuous-media applications.

We can partition Eq. 2 into two parts, namely, the *data transfer time*, which is $\sum_{i=1}^{n} \frac{bn_i}{B_{max}}$, and the *positioning overhead*, $\mathcal{O}(\mathbf{s}) = \sum_{i=1}^{n} \delta_i$. *Positioning overhead* is governed by the disk-scheduling policy, and the *data transfer* time depends on the total number of blocks to be transferred. The latter is independent of the scheduling policy. Let **n** be the vector $\langle n_1, \ldots, n_n \rangle$, which satisfies both Eqs. 1 and 2. With **s** and $\mathcal{O}(\mathbf{s})$, we can reformulate the continuity constraints of Eq. 1 as follows [KHS94]:

$$\|\mathbf{n}\| \geq \alpha_* \frac{\mathcal{O}(\mathbf{s}) \sum_{i=1}^{n} r_i}{\frac{b}{B_{max}} \left(B_{max} - \sum_{i=1}^{n} r_i \right)} \,. \quad (3)$$

α_* denotes the coefficient which is determined by the disk-scheduling strategy. n_i in **n** represents the number of blocks to be fetched in a cycle for playback i, and $\sum_{i=1}^{n} n_i$ is the total number of blocks to be fetched in a cycle. Thus, **n** in Eq. 3 is an asymptotic measure of the buffer space requirement in supporting a set of playbacks, **s**. Observing Eq. 3, we find that, as $\sum_{i=1}^{n} r_i$ approaches B_{max}, **n** increases *very* rapidly.

2.2 Aggregate positioning overhead for different scheduling policies

Another important concern is the analysis of aggregate operational overhead for a set of disk retrieval operations, $\mathcal{O}(\mathbf{s})$. We first model the seek time behavior of the disk head movement. A number of experimental measurements have resulted in the following model of seek time behavior [RW94, WGP94].

$$T_{seek} = \begin{cases} a_1 + b_1\sqrt{x} \; if \; x \leq c \\ a_2 + b_2 x \quad Otherwise \end{cases} \quad (4)$$

Here, x and c denote the seek distance and threshold value, respectively, in terms of the number of cylinders. In the HP 97560 disk, a_1, b_1, a_2 and b_2 are 3.24 ms, 0.4 ms, 8.0 ms and 0.008 ms, and c is 383 cylinders [RW94]. Let $\mathcal{C}$ be the number of cylinders in the disk. We establish an upper bound for head-positioning overhead in supporting a set of playbacks $\mathbf{s} = s_1, \cdots, s_n$.

In the *SCAN* scheduling algorithm, the disk head scans the cylinder from the center of the platter outwards (or vice versa), and reads the data blocks in cylindrical order. Thus, the respective cylinders are visited once in each cycle. The upper bound on the overhead in *SCAN* scheduling, $\mathcal{O}_{SCAN}(\mathbf{s})$, is expressed as in Eq. 5, given that $\frac{\mathcal{C}}{n-1} \leq c$ [KHS94]. This formulation is due to the fact that when the inter-cylinder distance is shorter than a certain threshold, seek time is proportional to the square root of the distance:

$$\mathcal{O}_{SCAN}(\mathbf{s}) = (n-1) \left(a_1 + b_1 \sqrt{\frac{\mathcal{C}}{n-1}} \right) \,. \quad (5)$$

In *FIFO* scheduling [RVR92], the disk head reads the data blocks in a fixed order independent of the location of the data blocks for each stream. Due to this property of *FIFO* scheduling, the disk head makes a full sweep of the disk platter $n-1$ times in the worst case. The corresponding maximum overhead in *FIFO* scheduling, $\mathcal{O}_{FIFO}(\mathbf{s})$, is expressed as in Eq. 6:

$$\mathcal{O}_{FIFO}(\mathbf{s}) = (n-1)(a_2 + b_2\mathcal{C}) \,. \quad (6)$$

In group sweep scheduling, namely *GSS* [CKY93], the set of playback streams are partitioned into a certain number of groups, say g. *SCAN* scheduling is used within a group and *FIFO* scheduling is used between the groups. Since there are g groups to be scanned, the disk head makes $g-1$ sweeps of the disk platter. The size of the group is as large as $\left\lceil \frac{n}{g} \right\rceil$, where n is the number of playback sessions. With this figure, the positioning overhead in *GSS*, $\mathcal{O}(\mathbf{s})$, can be formulated as in Eq. 7. In Eq. 7, when g is equal to one, $\mathcal{O}_{GSS}(\mathbf{s})$ becomes the same as $\mathcal{O}_{SCAN}(\mathbf{s})$; and when g is equal to n, becomes the same as $\mathcal{O}_{FIFO}(\mathbf{s})$:

$$\mathcal{O}_{GSS}(\mathbf{s}) = \left(\left\lceil \frac{n}{g} \right\rceil - 1 \right) \left(a_1 + b_1 \sqrt{\frac{\mathcal{C}}{\left\lceil \frac{n}{g} \right\rceil - 1}} \right) + (g-1)(a_2 + b_2\mathcal{C}) \,. \quad (7)$$

Comparing head-positioning overheads in Eqs. 5, 6, and 7, we observe that the *SCAN* and *FIFO* disk-scheduling strategies have the smallest and largest overhead, respectively, in supporting a set of playback sessions, **s**, when viewed from their asymptotic behaviors. Considering that *SCAN* makes a full sweep each cycle independent of the number of playbacks, when there are a relatively small number of playback

sessions, the *FIFO* strategy performs better than the *SCAN* strategy.

2.3 Buffer requirements for different scheduling policies

If we substitute $\mathcal{O}(\mathbf{s})$ in Eq. 3 with the actual overhead incorporating disk head movement characteristics under different scheduling strategies, namely, $\mathcal{O}_{SCAN}(\mathbf{s})$, $\mathcal{O}_{FIFO}(\mathbf{s})$, and $\mathcal{O}_{GSS}(\mathbf{s})$, we can obtain the buffer requirement vector $\mathbf{n}$ for each of these policies.

The *SCAN* strategy reads data blocks in cylindrical order in one direction, i.e., either inbound or outbound. For the individual playback, its disk-read operations in consecutive cycles can be adjacent or as much as $2T(\mathbf{s})$ apart. Thus, to compensate for interval variance between consecutive disk operations, a double-buffering scheme is used. In the *SCAN* disk-scheduling policy, the total buffer requirement $\mathbf{n}_{SCAN}$ can be expressed as in Eq. 8:

$$\mathbf{n}_{SCAN} = 2\left(\frac{\mathcal{O}_{SCAN}(\mathbf{s})\sum_{i=1}^{n} r_i}{\frac{b}{B_{max}}\left(B_{max}-\sum_{i=1}^{n} r_i\right)}\right). \tag{8}$$

In *FIFO* scheduling, the disk operations for each stream are exactly $T(\mathbf{s})$ apart. Thus, an application can consume a data block immediately after it has been loaded into the memory without introducing additional memory buffer. Thus, the buffer requirement for *FIFO* scheduling, $\mathbf{n}_{FIFO}$, can be expressed as in Eq. 9:

$$\mathbf{n}_{FIFO} = \frac{\mathcal{O}_{FIFO}(\mathbf{s})\sum_{i=1}^{n} r_i}{\frac{b}{B_{max}}\left(B_{max}-\sum_{i=1}^{n} r_i\right)}. \tag{9}$$

GSS is a hybrid of *FIFO* scheduling and *SCAN* scheduling. Let us partition the set of playback streams $\mathbf{s}$ into g groups, namely $g_1, \ldots, g_g$. The g_is in consecutive cycles are exactly $T(\mathbf{s})$ apart. Within the group g_i, the disk operations will be performed based on the cylindrical position. If we assume that each group has the same amount of data blocks to read, then the maximum variance of the interval between disk operations for a stream in consecutive cycles is $\frac{T(\mathbf{s})}{g}$, and thus an additional amount of buffer space needs to be introduced to compensate for this interval variance. The buffer requirement for *GSS* scheduling algorithm, $\mathbf{n}_{GSS}$, can be formulated as in Eq. 10:

$$\mathbf{n}_{GSS} = \left(1+\frac{1}{g}\right)\left(\frac{\mathcal{O}_{GSS}(\mathbf{s})\sum_{i=1}^{n} r_i}{\frac{b}{B_{max}}\left(B_{max}-\sum_{i=1}^{n} r_i\right)}\right). \tag{10}$$

2.4 Generalized schedulability condition

Equations 8, 9, and 10 enable us to extract the common characteristics (or asymptotic behavior) of the buffer requirements and schedulability conditions for different disk-scheduling strategies. The buffer requirement formulation consists of *multiplication constant*, α_*, *overhead factor*, κ_*, and *bandwidth utilization*. We develop a general formulation of buffer requirement for continuous-media playback. Let $\mathcal{M}(\mathbf{s})$ be the buffer requirement for supporting a set of continuous-media playbacks. $\mathcal{M}(\mathbf{s})$ can be formulated as in Eq. 11.

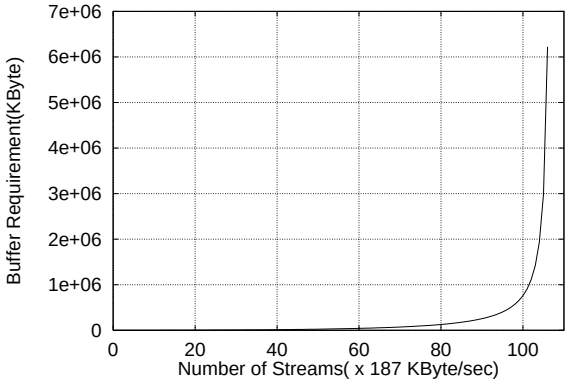

Fig. 1. Buffer space requirement

$$\mathcal{M}(\mathbf{s}) = \alpha_* \cdot \kappa_* \frac{\sum_{i=1}^{n} r_i}{B_{max}-\sum_{i=1}^{n} r_i}. \tag{11}$$

In Eq. 11, α_* and κ_* correspond to the multiplication constant and $\frac{\mathcal{O}(\mathbf{s})B_{max}}{b}$, respectively. α_*s correspond to 2, 1 and $(1+\frac{1}{g})$ for the *SCAN*, *FIFO*, and *GSS* scheduling policies, respectively. Observing the formulations provided so far, we can deduce that a new playback from the disk can be initiated as long as the aggregate playback bandwidth does not exceed the maximum disk bandwidth regardless of the disk-scheduling policy. Details of this call admission property are provided in Theorem 3 in Appendix B. It is interesting to observe that in Eqs. 8, 9 and 10, the multiplication constants, α_*s for *SCAN*, *FIFO*, and *GSS* scheduling strategies, respectively, are inversely proportional to their aggregate operational overheads in Eqs. 5, 6, and 7.

In Fig. 1, we plot the relationship between the number of streams and the total amount of required buffer space. We consider playback of MPEG-1 streams at 1.5 Mbps (187 KBps). The disk spins at 7200 RPM, and one revolution takes 8.3 ms. It has 1964 cylinders and its seek time is modeled after the HP97560 disk [RW94]. The disk is formatted with 4096 byte blocks and takes 0.2 ms to read one block, which is equivalent to a rate of 20 MBps. Theoretically, the disk can support up to 106, i.e., $\lfloor\frac{20000}{187}\rfloor$, streams. As shown, the buffer requirement increases extremely rapidly as it approaches the upper bound.

3 Buffer requirement for memory-mode playback

Consider a sequence of concurrent playbacks for file i, $\mathbf{s}_i = s_i^1, s_i^2, \ldots, s_i^n$, which are chronogically ordered according to their starting time. Let $t(s_i^k)$ and Δ_i^k be the starting time of the playback and the interval between consecutive playbacks, respectively. We define Δ_i^k as $t(s_i^{k+1}) - t(s_i^k)$.

Assume that the playback s_i^k is supplied data blocks from the disk, i.e., in *disk mode*, and a certain amount of buffer memory has been allocated for synchronization purpose. The immediate successive playback, s_i^{k+1}, can either be supplied the data blocks from the disk or can use the data blocks previously loaded by s_i^k. In the latter situation, the playback s_i^{k+1} does not require buffer memory for synchronization purposes, but because it is Δ_i^k behind s_i^k, a certain amount of buffer memory needs to be allocated to retain the data blocks which have been used by s_i^k. With a playback rate

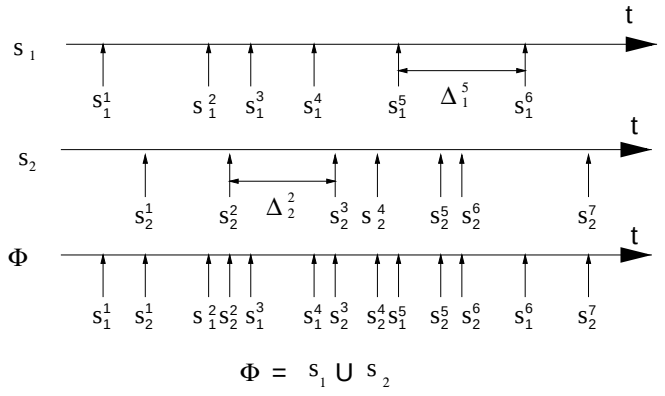

Fig. 2. Sequence of playbacks

r_i(bytes/s), the buffer size for maintaining the data blocks for interval $[s_i^k, s_i^{k+1}]$ is $r_i(t(s_i^{k+1}) - t(s_i^k))$. Figure 2 illustrates the request arrivals for playbacks. s_1 and s_2 denote the sequence of request arrivals for file 1 and file 2, respectively.

In Fig. 1, we observed that the total buffer size increases *very* fast as disk utilization reaches 100%. Consecutive playbacks for the same file may be sufficiently close that the total amount of buffer memory required becomes smaller by supporting some of the playbacks in memory mode. We analyze this idea to show how to achieve the minimum buffer requirement.

4 Determining service mode of a playback

4.1 Problem formulation

We define two *service modes* of playbacks with respect to the source of data, namely *disk mode*, and *memory mode*. In disk-mode service, the playback reads the data blocks from the disk. In memory-mode service, the playback uses the data blocks which have been loaded by the preceding playback of the same file. Each service mode requires different amounts of buffer space. Our objective is *to support a set of multimedia playbacks with minimum total buffer space requirement*. Towards achieving this objective, we investigate the trade-offs for the two service modes and develop an algorithm to determine the optimal service modes for individual playbacks.

Before discussing the solution, we describe a number of variables that are used to formulate the problem. The set of playbacks $\mathbf{s}$ is partitioned into subsets $\mathbf{s}_1, \mathbf{s}_2, \ldots$. $\mathbf{s}_i = s_i^1, s_i^2, \ldots$ is a sequence of playbacks for file i ordered according to playback starting time. We partition $\mathbf{s}$ into two sets with respect to service modes, namely $\mathbf{s}_D$ for disk mode and $\mathbf{s}_M$ for memory mode. The total buffer requirements for $\mathbf{s}$, $\mathcal{M}(\mathbf{s})$, can be expressed as $\mathcal{M}(\mathbf{s}_D) + \mathcal{M}(\mathbf{s}_M)$. $\mathcal{M}(s_i^k)$, $s_i^k \in \mathbf{s}$, denotes the buffer size allocated for s_i^k. If s_i^k is in disk mode, $\mathcal{M}(s_i^k)$ corresponds to synchronization buffer size. If it is in memory mode, $\mathcal{M}(s_i^k)$ corresponds to the buffer size to keep the data blocks in memory for a certain time interval. $\mathcal{M}(\mathbf{s}_M)$ can be expressed as $\sum_{s_i^k \in \mathbf{s}_M} \Delta_i^{k-1} r_i$. $\mathcal{M}(\mathbf{s}_D)$ can be obtained using Eq. 3, and one of Eqs. 5, 6, or 7, depending on the disk-scheduling strategy. We use $\check{s}_i^k$ to denote memory-mode playback when it is necessary to distinguish it from disk-mode playback, s_i^k.

With a given set of playbacks, $\mathbf{s}$, the algorithm determines the service mode of each playback to minimize the total buffer requirement. The **s***ervice-***m***ode* **d***etermination* **p***roblem*, $\mathcal{SMDP}$ is defined as follows.

Definition 1. $\mathcal{SMDP}$. *Given a set of playback requests* $\mathbf{s}$*, partition it into* $\mathbf{s}_M$ *and* $\mathbf{s}_D$ *such that the total buffer space requirement, i.e.,* $\mathcal{M}(\mathbf{s}) = \mathcal{M}(\mathbf{s}_D) + \mathcal{M}(\mathbf{s}_M)$ *is minimized.*

4.2 Profitability of service-mode conversion

We introduce the notation $\mapsto$ to represent a conversion of service mode. When s_i^k is currently in disk mode, converting it to memory mode, i.e., $s_i^k \mapsto \check{s}_i^k$, changes the buffer requirement. $s_i^k \mapsto \check{s}_i^k$ reduces the total synchronization buffer requirement, since playback s_i^k no longer requires disk bandwidth. The amount of reduction can be formulated as $\mathcal{M}(\mathbf{s}_D) - \mathcal{M}(\mathbf{s}_D \setminus \{s_i^k\})$. Since s_i^k is now serviced in memory-mode, a certain amount of memory buffer is required for memory-mode service of s_i^k. We introduce the term *profitability* to denote whether a given *service-mode conversion* actually decreases the total buffer requirement.

Definition 2. *Converting the service mode of playback,* s_i^k*, from disk mode to memory-mode,* $s_i^k \mapsto \check{s}_i^k$*, is called profitable if the total amount of buffer memory decreases as a result of the conversion.*

After converting s_i^k to memory mode, the set of playbacks in disk mode corresponds to $\mathbf{s}_D \setminus \{s_i^k\}$. The conversion, $s_i^k \mapsto \check{s}_i^k$, is said to be *profitable* if Eq. 12 holds.

$$\underbrace{\mathcal{M}(\mathbf{s}_D) - \mathcal{M}(\mathbf{s}_D \setminus \{s_i^k\})}_{\text{Reduction in synchronization buffer}} \geq \underbrace{\Delta_i^{k-1} s_i^k}_{\text{Memory buffer increase}} . \qquad (12)$$

The respective decrease in the buffer requirement is called *profit* and is denoted by $\mathcal{P}_{\mathbf{s}}(s_i^k \mapsto \check{s}_i^k)$. It is formulated as in Eq. 13. Equation 12 is equivalent to $\mathcal{P}_{\mathbf{s}}(s_i^k \mapsto \check{s}_i^k) > 0$.

$$\mathcal{P}_{\mathbf{s}}(s_i^k \mapsto \check{s}_i^k) = \mathcal{M}(\mathbf{s}_D) - \mathcal{M}(\mathbf{s}_D \setminus \{s_i^k\}) - \Delta_i^{k-1} r_i . \qquad (13)$$

As the aggregate bandwidth of the disk-mode playbacks increases, per-playback buffer requirement (in disk mode) increases as well. In contrast, the buffer size for a memory-mode playback is not affected by the aggregate disk bandwidth. Service modes of ongoing playbacks need to be examined upon the arrival or departure of a playback, and need to be updated dynamically towards minimizing the buffer memory requirements. When a new playback request arrives, the buffer requirements for the individual disk-mode playbacks are re-computed. We call converting a service mode from disk to memory as *merging* of playbacks, since the playback then re-uses the data loaded by the preceding playback for the same file. When playback finishes, we check if there is any memory-mode playback whose service-mode conversion results in the decrease of the total buffer requirement. When

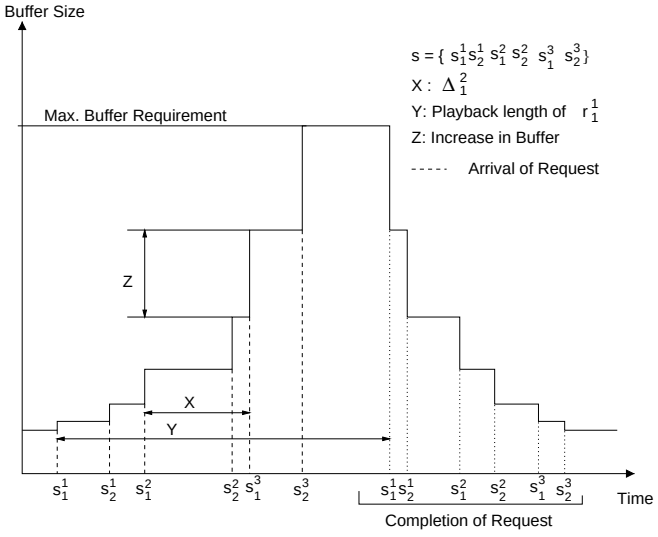

Fig. 3. Buffer size upon start and completion of playbacks

service mode is converted from memory mode to disk mode, the playback is supplied the data from the disk rather than using the loaded data. Converting a playback mode from memory to disk is called *splitting*.

5 Adaptive service-mode conversion

5.1 Properties of service-mode conversion

Initially, we consider that all playbacks have the same playback rates, i.e., they are *homogeneous* from the perspective of disk bandwidth requirement. Later in this article, we remove this constraint and prove that the proposed scheme works for the generic environment where individual playbacks have different playback rates.

The total amount of buffer increases in a stepwise fashion as new playbacks are added to the set of ongoing playbacks **s**, as shown in Fig. 3. The amount of buffer decreases in a similar fashion as each playback completes. As shown in Fig. 3, even though the playbacks have the same playback rates, the amount of increase in total buffer size resulting from adding a new playback gets larger as there are more concurrent playbacks. One of the important properties is that the profit of $s_i^k \mapsto \check{s}_i^k$ depends on the disk bandwidth utilization. More specifically, the profit of disk-to-memory-service mode conversion for the same playback is *larger* under higher disk utilization. The formal description of this property is provided in Lemma 1 in Appendix B. It implies that service modes need to be re-examined when there is a change in ongoing playback sessions, i.e., arrival or departure of a playback.

We provide a formulation to compute the amount of decrease in the buffer requirement resulting from converting a set of playbacks from disk mode to memory mode. Let **v** be the set of playbacks of **s** to be converted from disk mode to memory mode. Then, the amount of buffer space reduction resulting from converting the service modes of **v** is formulated as in Eq. 14. Detailed derivation steps of Eq. 14 are provided in Lemma 2 in Appendix B. In Sect. 2, we showed that the individual disk-scheduling algorithms, e.g., *SCAN*, *FIFO*, *GSS*, etc. for continuous-media playback share the same asymptotic behavior in buffer space requirement. We use the asymptotic bound Θ [CaRLR90] to denote the buffer size requirement in our formulation instead of providing the formula for individual scheduling algorithms.

$$\Theta(\mathcal{M}(\mathbf{s}_D)) - \Theta(\mathcal{M}(\mathbf{s}_D \setminus \mathbf{v})) = \frac{B_{max} \sum_{s_i \in \mathbf{v}} r_i}{(B_{max} - \sum_{i=1}^{n} r_i)(B_{max} - (\sum_{i=1}^{n} r_i - \sum_{s_i \in \mathbf{v}} r_i))} . \quad (14)$$

Equation 14 is used as the basis for finding the optimal set of playbacks for service-mode conversion.

5.2 Adaptive merging of playbacks

We introduce two algorithms, called $\mathcal{SMDP}_{arr}$ and $\mathcal{SMDP}_{dep}$, that are executed in the event of arrival and departure of playbacks, respectively. The fundamental idea of these two algorithms is the same. Since the former is executed with the arrival of playback request and the latter is excuted with the completion of playback, the comparison and selection conditions are symmetric to each other.

When a new request arrives, it is assumed to be supported in disk mode and we re-compute the synchronization buffer requirement for all the disk-mode playbacks including the new one. Then, the algorithm computes the profits of service-mode conversions for the individual disk-mode playbacks, and sorts them in decreasing order of profit. Computation of the profit is based on Eq. 14. Once the disk-mode playbacks are sorted according to their profit, the algorithm converts the playback to memory mode one by one. Each conversion is followed by a re-computation of the profits for the remaining disk-mode playbacks. It is proved in Lemma 4 that once the playbacks are sorted according to the profit, the relative order is not affected by the removal of a playback from the list. As there is a smaller number of disk-mode playbacks, per-playback synchronization buffer size decreases, and hence the profit of service-mode conversion decreases. The algorithm continues converting the service modes from disk to memory until there is no more *profitable* conversion.

Table 1 illustrates the algorithm $\mathcal{SMDP}_{arr}$. $\mathcal{SMDP}_{arr}$ is executed upon the arrival of a new playback. The algorithm examines only the disk-mode playbacks for profitability. $\mathcal{SMDP}_{arr}$ computes the required buffer space for each interval $[t(s_i^k), t(s_i^{k+1})]$ and determines if $s_i^{k+1} \mapsto \check{s}_i^{k+1}$ is *profitable*.

The **for** loop in lines 9–13 initializes the set $I2$ with the *profitable* intervals. Since all playbacks require the same amount of buffer, performing $s_i^k \mapsto \check{s}_i^k$ with the *shortest-interval-first* policy will result in maximizing the total profit. For each iteration of the **while** loop in lines 14–24, the algorithm selects the shortest time interval Δ_i^k. If $\mathcal{P}_{\mathbf{s}'}(s_i^{k+1} \mapsto \check{s}_i^{k+1})$ is greater than 0, the service mode of s_i^{k+1} is converted to memory mode. The *profit* $\mathcal{P}_{\mathbf{s}'}(s_i^{k+1} \mapsto \check{r}_i^{k+1})$ varies depending on $\mathbf{s}'$. As the service modes of the playbacks are converted to memory mode (lines 21 and 22), the effect of $\mapsto$ becomes less significant. Eventually, the reduction in buffer requirement for disk-mode playbacks, which results

from $s_i^k \mapsto \check{s}_i^k$, becomes smaller than $\Delta_i^{k-1} r_i$; and thus the profit of $s_i^k \mapsto \check{s}_i^k$ eventually becomes negative. If the profit of a service-mode change in the shortest interval is less than 0, this implies that there are no more *profitable* playbacks, and hence the algorithm terminates.

5.3 Optimality of the $\mathcal{SMDP}$ algorithm

Let **v** be a set of disk-mode playbacks to be converted to memory mode. Theorem 1 states that given a set of disk-mode playbacks, $\mathbf{s}_D$, algorithm $\mathcal{SMDP}_{arr}$ finds **v**, $\mathbf{v} \in \mathbf{s}_D$, which minimizes the total buffer requirement.

Theorem 1. (Proof of optimality) *Let $\mathbf{s}_D$ be a set of disk-mode playbacks and $\mathbf{v} \in \mathbf{s}_D$ be a set of playbacks converted to memory mode.* **v***, which minimizes the total buffer requirement, is found by converting the service modes with the shortest-interval-first policy.*

Proof. We prove the theorem by contradiction. $\mathcal{SMDP}_{arr}$ converts the service mode of the playbacks with the *shortest-interval-first policy*. The algorithm selects the playback for the service mode conversion based on the interval with its preceeding playback for the same file. The playback with shorter interval is preferred. Let us assume that **v** is the optimal set, which maximizes the profit of conversions, and there exists a playback $s_i^k \notin \mathbf{v}$ in disk mode such that Δ_i^{k-1} is shorter than the window Δ' of a certain playback $s' \in \mathbf{v}$. This means that **v** is not based on the *shortest-interval-first* policy. The profit of converting the playbacks in **v** to memory mode can be formulated as follows (refer to Eq. 14).

$$\mathcal{P}_{\mathbf{s}}(\mathbf{v} \mapsto \check{\mathbf{v}}) = \frac{B_{max} \sum_{s_i^k \in \mathbf{v}} r_i}{(B_{max} - \sum_{s_i^k \in \mathbf{s}_D} r_i)(B_{max} - \sum_{s_i^k \in \mathbf{s}_D} r_i - \sum_{s_i^k \in \mathbf{v}} r_i)} - \sum_{s_i^k \in \mathbf{v}} \Delta_i^{k-1} r_i \,.$$

Let **u** be $\mathbf{v} \setminus \{s_i^k\} \cup \{s'\}$. All playbacks require the same amount of bandwidth and $|\mathbf{v}| = |\mathbf{u}|$. Thus, the synchronization buffer space reduction resulting from service-mode conversions in **v** and in **u** is the same. It can be expressed as in Eq. 15:

$$\mathcal{M}(\mathbf{s}_D) - \mathcal{M}(\mathbf{s}_D \setminus \mathbf{v}) = \mathcal{M}(\mathbf{s}_D) - \mathcal{M}(\mathbf{s}_D \setminus \mathbf{u}) \,. \quad (15)$$

Thus,

$$\mathcal{P}_{\mathbf{s}}(\mathbf{v} \mapsto \check{\mathbf{v}}) - \mathcal{P}_{\mathbf{s}}(\mathbf{u} \mapsto \check{\mathbf{u}}) = r(\Delta_i^{k-1} - \Delta') \,. \quad (16)$$

From our assumption, $\Delta_i^{k-1} > \Delta'$. This leads to the fact that $\mathcal{P}_{\mathbf{s}}(\mathbf{v} \mapsto \check{\mathbf{v}})$ is less than $\mathcal{P}_{\mathbf{s}}(\mathbf{u} \mapsto \check{\mathbf{u}})$ in Eq. 16, and **v** cannot be optimal. This is a contradiction, and hence the claim.

5.4 Adaptive splitting of playbacks

When a memory-mode playback terminates, disk bandwidth utilization remains the same, and thus no further action is required. Termination of a disk-mode playback implies that the aggregate disk bandwidth utilization decreases and subsequently per-playback buffer requirements for disk-mode playbacks become smaller. Hence, when the disk-mode playback terminates, individual memory-mode playbacks are examined for possible service-mode conversion. Upon the completion of the playback, the server examines the profitability of the remaining memory-mode playbacks. For each memory-mode playback s_i^k, if $\Delta_i^{k-1} r_i$ is greater than $\mathcal{M}(\mathbf{s}_D \cup \{s_i^k\}) - \mathcal{M}(\mathbf{s}_D)$, i.e., it requires less buffer space to service s_i^k in disk mode, then the service mode of $\check{s}_i^k$ needs to be changed to disk mode. The profit of $\check{s}_i^k \mapsto s_i^k$ is formulated in Eq. 17:

$$\mathcal{P}_{\mathbf{s}}(\check{s}_i^k \mapsto s_i^k) = \Delta_i^{k-1} r_i - \left\{ \mathcal{M}(\mathbf{s}_D \cup \{s_i^k\}) - \mathcal{M}(\mathbf{s}_D) \right\} . \quad (17)$$

Algorithm $\mathcal{SMDP}_{dep}$(**s**) in Table 2 shares the same basic idea and formulation with $\mathcal{SMDP}_{arr}$ in Table 1, and we do not provide its detailed description. $\mathcal{SMDP}_{dep}$(**s**) examines the requests in memory mode, in decreasing order of interval length, since the profit in Eq. 17 is maximized by the longest interval length.

Table 1. Algorithm for $SMDP_{arr}$

Algorithm 1. $\mathcal{SMDP}_{arr}$

```
▷ Minimizing Buffer Requirement with Arrival of the requests
SMDP_arr(s){
  Set of Intervals: I1, I2 ;
  Set of Requests: s' ;
  s' ← {s_i^k ∈ s such that s_i^k is in diskmode} ;
  Boolean: DONE ;
  I1 ← intervals in s' ;
  I2 ← φ ;
  DONE ← False ;
  ▷ Initializing profitable interval sets
  for all Δ_i^k ∈ I1
    if (P_s'(s_i^{k+1} ↦ ř_i^{k+1}) ≥ 0) then
      I2 ← I2 ∪ {Δ_i^k}
    end if
  end for
  I2 ← sort(I2) ;
  ▷ with the increasing order of interval length
  while ( not DONE )
    Δ_i^k ← ShortestInterval(I2) ;
    if P_s'(s_i^{k+1} ↦ š_i^{k+1}) ≤ 0 then
      DONE ← TRUE ;
    else
      I2 ← I2 − {Δ_i^k} ;
      s_i^{k+1} ↦ š_i^{k+1} ;
      s' ← s' - {s_i^{k+1}} ;
      s ← s - {s_i^{k+1}} ∪{š_i^{k+1}} ;
    end if
  end while
  return(s) ;
}
```

6 Generalization

6.1 Heterogeneous playbacks

In a large-scale VOD server environment, it is natural that individual users make playback requests with different QoS requirements, which result in different disk bandwidth requirements. We relax the assumption of homogeneous playback rates and enhance the proposed algorithm to find the

Table 2. Algorithm for $SMDP_{dep}$(**s**)

Algorithm 2. $\mathcal{SMDP}_{dep}$
▷ *Minimizing Buffer Requirement with the completion of the playback*
$\mathcal{SMDP}_{dep}$(**s**){
1 *Set of Intervals: I1, I2 ;*
2 *Set of Requests:* **s'** *;*
3 *Boolean: DONE ;*
4 $I1 \leftarrow$ *intervals in* **s** *;*
5 $I2 \leftarrow$ **s** *;*
6 **s'** $\leftarrow$ **s** *;*
7 *DONE* $\leftarrow$ *False ;*
▷ *Initializing profitable interval sets*
8 **for** *all* $\Delta_i^k \in I1$
9 **if** $(\mathcal{P}_{\mathbf{s}'}(\check{r}_i^{k+1} \mapsto s_i^{k+1}) \geq 0)$ **then**
10 $I2 \leftarrow I2 \cup \{\Delta_i^k\}$
11 **end if**
12 **end for**
13 $I2 \leftarrow sort(I2)$ *;*
▷ *with the decreasing order of interval length*
14 **while** *(not DONE)*
15 $\Delta_i^k \leftarrow LongestInterval(I2)$ *;*
16 **if** $\mathcal{P}_{\mathbf{s}'}(\check{r}_i^{k+1} \mapsto s_i^{k+1}) \leq 0$ **then**
17 *DONE* $\leftarrow$ *TRUE ;*
18 **else**
19 $I2 \leftarrow I2 - \{\Delta_i^k\}$ *;*
20 $\check{s}_i^{k+1} \mapsto s_i^{k+1}$ *;*
21 **s'** $\leftarrow$ **s'** - $\{\check{s}_i^{k+1}\}$ *;*
22 **s** $\leftarrow$ **s** - $\{s_i^{k+1}\} \cup \{\check{s}_i^{k+1}\}$ *;*
23 **end if**
24 **end while**
25 *return*(**s**) *;*
}

optimal service modes for a set of *heterogeneous* playbacks. Heterogenity of playback bandwidths introduces another dimension of complexity, since it is not possible to determine the relative profit by comparing the interval lengths alone. An important characteristic in heterogeneous playback is that the increase in the synchronization buffer due to the addition of a new playback is hyper-linearly proportional to the playback rate of the new request. Consider the following example. Let the playback rates for s_x and s_y be 2 Mbps and 4 Mbps, respectively. If the addition of s_x via disk mode requires a 3-MB increase in total buffer size, then addition of s_y in disk mode entails *more than* a 6-MB increase of buffer requirement. Detailed description of this property is described in detail in Lemma 3 in Appendix B.

6.2 Optimality of generalized $\mathcal{SMDP}_{arr}$

In heterogeneous playbacks, the generalized $\mathcal{SMDP}_{arr}$, namely $g\mathcal{SMDP}_{arr}$, converts the service modes of the playbacks based on *LPF* (*largest-profit-first*) policy. In the generalized version of $\mathcal{SMDP}$, the playbacks are sorted with respect to the profit (line 13 of Table 1) instead of the interval length. In a homogeneous playback environment, profit is proportional to interval length.

The following important property makes *LPF*-based conversion valid, and makes the algorithm very efficient ($O(\log n)$ for each arrival of new playback). *Once the disk-mode playbacks are sorted according to their profits, the relative ranks of the playbacks do not change, though some of the playbacks are removed from the sorted list.* This property serves as the basis for proving the *optimality* of the $g\mathcal{SMDP}_{arr}$ algorithm. The details are shown formally in Lemma 4 in Appendix B.

Theorem 2. *Given a set of playbacks* **s**, *and disk-mode playbacks* $\mathbf{s}_D \subset \mathbf{s}$, $g\mathcal{SMDP}_{arr}$ *finds an optimal set of playbacks,* $\mathbf{v} \subset \mathbf{s}_D$, *to be converted to memory mode, whose conversion results in minimzation of total buffer requirement.*

Proof. Let **v** be the optimal solution which minimizes the overall buffer requirement and let us assume that there exists a playback s_i^k such that $s_i^k \notin \mathbf{v}$, and

$$\mathcal{P}_{\mathbf{s}_D}(s_i^k \mapsto \check{s_i^k}) > \mathcal{P}_{\mathbf{s}_D}(s' \mapsto \check{s'}), \exists s' \in \mathbf{v}\,.$$

Let $\mathbf{u} = \mathbf{v} \setminus \{s_i^k\} \cup \{s'\}$.

$$\mathcal{P}_{\mathbf{s}_D}(\mathbf{v} \mapsto \check{\mathbf{v}}) = \mathcal{M}(\mathbf{s}_D) - \mathcal{M}(\mathbf{s}_D - \mathbf{v}) - \sum_{s_j^l \in \mathbf{v}} \Delta_j^{l-1} r_j\,. \quad (18)$$

From Lemma 2, the aggregate reduction of buffer space resulting from converting the disk-mode playbacks is the sum of the incremental reductions resulting from changing the service modes one by one. Thus, we obtain the following from Eq. 18. Let $\mathbf{w} = \mathbf{v} \setminus \{s'\} = \mathbf{u} \setminus \{s_i^k\}$.

$$\mathcal{P}_{\mathbf{s}_D}(\mathbf{v} \mapsto \check{\mathbf{v}}) = \mathcal{P}_{\mathbf{s}_D}(\mathbf{w} \mapsto \check{\mathbf{w}}) + \mathcal{P}_{\mathbf{s}_D \setminus \mathbf{w}}(s_i^k \mapsto \check{s_i^k})\,.$$

Thus,

$$\mathcal{P}_{\mathbf{s}_D}(\mathbf{v} \mapsto \check{\mathbf{v}}) - \mathcal{P}_{\mathbf{s}_D}(\mathbf{u} \mapsto \check{\mathbf{u}}) = \mathcal{P}_{\mathbf{s}_D \setminus \mathbf{w}}(s' \mapsto \check{s'}) - \mathcal{P}_{\mathbf{s}_D \setminus \mathbf{w}}(s_i^k \mapsto \check{s_i^k})\,. \quad (19)$$

From Lemma 4, and $\mathbf{s}_D \setminus \mathbf{w} \subset \mathbf{s}_D$, the right-hand side of Eq. 19 is less than 0. Thus, $\mathcal{P}_{\mathbf{s}_D}(\mathbf{v} \mapsto \check{\mathbf{v}})$ is less than $\mathcal{P}_{\mathbf{s}_D}(\mathbf{u} \mapsto \check{\mathbf{u}})$, and **v** is not optimal. This leads to contradiction, and thus the optimal solution **v** should consist of playbacks chosen by the *LPF* policy. Hence, the claim.

6.3 Overhead in transition phase

Converting a service mode from the disk to memory entails a *transition phase* during which the playback requires buffer space for disk-mode service as well as for memory-mode service. The transition phase lasts until the playback mode is completely converted to memory mode. When the system decides to convert the service mode of a certain playback from disk to memory, the data blocks required by to-be-converted playback are not immediately available in memory since the memory buffer does not retain the consumed data blocks for possible later usage. For this reason, the playback needs to be serviced from the disk for a certain interval. Let t_0 be the arrival time of s_i^k. When $\mathcal{SMDP}_{arr}$ decides to perform $s_i^k \mapsto \check{s}_i^k$, the immediate predecessor of s_i^k, i.e., s_i^{k-1}, starts to retain data blocks in memory from t_0. Without loss of generality, we assume that s_i^{k-1} is being serviced in disk mode. s_i^k can be serviced in memory mode only when the appropriate data blocks are available in memory. At time t_0, playback s_i^{k-1} has been playing for Δ_i^{k-1} time units. s_i^k requires the data blocks from the beginning of the file, which are not available in memory at t_0. Let l_0

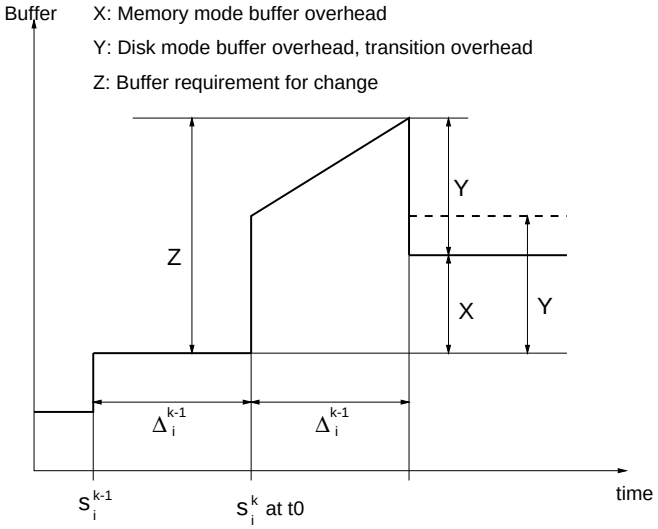

Fig. 4. Transition phase of $s_i^k \mapsto \check{s}_i^k$

be the index of the data block that s_i^{k-1} starts to keep in memory at t_0. s_i^k will require l_0 at time $t_0 + \Delta_i^{k-1}$. Thus, at $t_0 + \Delta_i^{k-1}$, s_i^k can be serviced in memory mode. The time interval between the start of conversion and the completion of conversion, i.e., $[t_0, t_0 + \Delta_i^{k-1}]$, is called the *transition phase*. The length of the transition phase for $s_i^k \mapsto \check{s}_i^k$ corresponds to the interval length to its immediate predecessor, Δ_i^{k-1}. The transition phase introduces an additional buffer overhead. In a *video-on-reservation (VOR)* environment, where the playback starting time is known a priori, transition overhead can be eliminated. Figure 4 illustrates how buffer size changes during the transition phase.

It is possible that the service modes of more than one playback need to be changed when a new playback arrives, i.e., there is an *overlap of transition phases*. Converting the service modes of several playbacks simultaneously causes the concurrent existence of multiple transition phases, which can cause a bursty increase in buffer usage. To minimize the additional buffer space overhead for multiple concurrent transitions, we propose *interleaving* of transition phases. The order in which a set of conversion is performed does not affect the resulting buffer size, as shown in Lemma 2 in Appendix B. To minimize the peak demand of buffer requirement during transition, i.e., *transition overhead*, the transition phases of the requests are interleaved, in *increasing order* of *profit*.

7 Simulation-based evaluation

We present the results of a detailed simulation study to evaluate the effectiveness of the proposed scheme. In our simulation, there are 100 different video titles available to users. The user access pattern is modeled using the *Zipf* distribution[2]. We adjust the skew factor α to model different user access skewness. The disk model used in the simulation is the same as the one introduced in Sect. 2.4. Theoretically, it can support up to $\lfloor \frac{20000}{187} \rfloor$ = 106 MPEG-1 streams.

[2] $P_j = \sum_{i=1}^{j} \frac{C}{i^{1-\alpha}}$, $C = \frac{1}{\sum_{i=1}^{n} \frac{1}{i^{1-\alpha}}}$

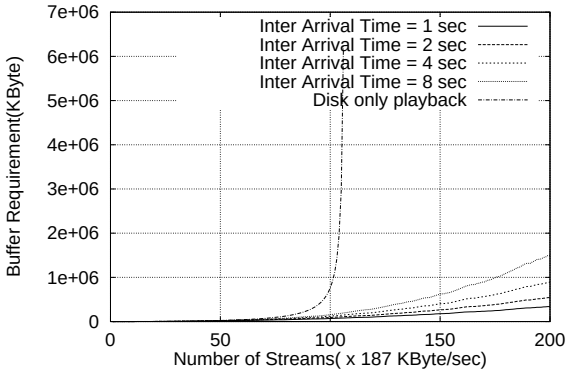

Fig. 5. Buffer requirement under different average inter-arrival time

7.1 Effectiveness of $\mathcal{SMDP}$ algorithm

In our simulation environment, each user requests an MPEG-1 stream, at 187 KB/s. As can be seen in Fig. 1, it requires 760 MB of buffer space to support 100 concurrent playbacks from the disk. We compare this buffer size with the buffer size with $\mathcal{SMDP}$ algorithm. By carefully determining the service mode of each playback, the maximum buffer requirement decreases substantially. When we adopt $\mathcal{SMDP}$ algorithm in determining the service mode of the playbacks, the buffer requirement reaches 68 MB when the server accommodates the 100th playback with mean inter-arrival time of 1 s. It is only 8.9% of the buffer requirement without the $\mathcal{SMDP}$ algorithm. The $\mathcal{SMDP}$ algorithm enables the server to reduce a significant amount of buffer space in supporting a set of continuous-media playbacks.

We investigate the effectiveness of the proposed scheme under different server loads. Simulation is performed under different average inter-arrival times to mimick the different levels of server loads.

Figures 5 and 6 illustrate the buffer requirement under different request inter-arrival times. We consider five different average inter-arrival times: 1 s, 2 s, 4 s, 6 s, and 8 s, and the skew factor α is 0.15. There are two major observations which deserve some attention. With the same number of concurrent streams, the buffer space requirement becomes smaller as the average inter-arrival time of the requests gets smaller. This is because the time interval between the adjacent playbacks for the same file is smaller under heavier workload, and thus the profit of service-mode conversion from the disk mode to memory mode becomes more significant. Subsequently, more streams tend to be supported in *memory mode*. These results confirm that *the proposed scheme is more effective especially under heavy-load circumstances*. The second issue is related to the limitation of disk bandwidth capacity. In disk-only mode, we can theoretically support up to 106 streams. In contrast, the proposed scheme enables the server to support more streams with less amount of buffer space. Further, the number of concurrent playbacks is no longer governed by the maximum disk transfer rate and can be larger than the number of playbacks supported from the disk.

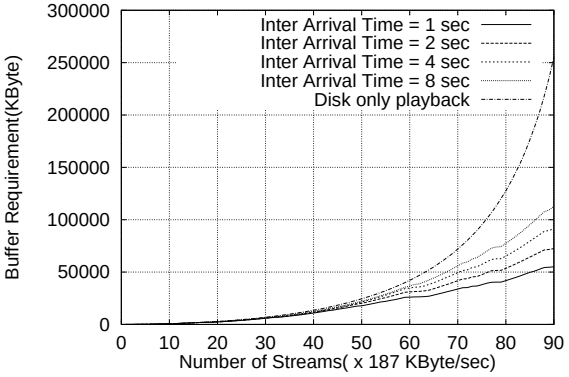

Fig. 6. Buffer requirement under different average inter-arrival time

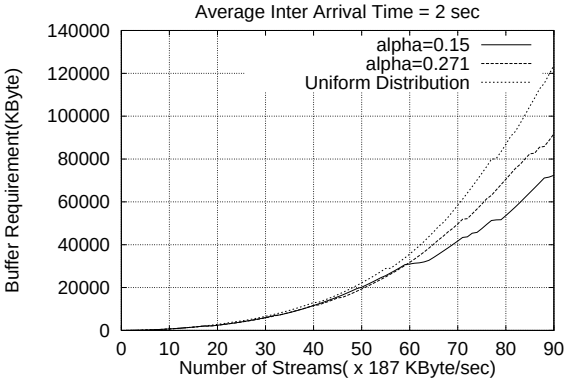

Fig. 7. Effect of different access skewness

7.2 Effect of user access pattern

If Fig. 7, we visualize the effect of access skewness on the buffer space requirement. Depending on the type of contents in the server, the user access pattern may vary. It is widely known that in movie on demand, user accesses are focused on a small set of hot files, while most of the files are rarely accessed [vid92]. Meanwhile, it is possible that the user accesses the files in relatively uniform fashion. One of the advantages of the proposed scheme is the fact that it can exploit the user access skewness. We verify this characteristics via simulation study. The average inter-arrival time in Fig. 7 is 2 s. We run the simulation under three different access skewnesses, with $\alpha = 0.15, 0.271$ and 0.99. Skewed factor, $\alpha = 0.99$, is used to generate the uniform access pattern. As illustrated, when the user access pattern is more skewed, i.e., user accesses are towards a smaller subset of the files, the buffer space requirement becomes smaller.

8 Conclusion

An excessive buffer space requirement to handle continuous-media playbacks is a serious impediment, to cost-effective provisioning for on-line video retrieval, especially when required I/O bandwidth reaches its full capacity. Given the skewed distribution of video popularity, it is expected that often there will be concurrent playbacks for the same video within a short time interval. This creates an opportunity to batch multiple requests and to service them with a single stream from the disk. At the exchange, there is a need to keep the data blocks in memory buffer between successive uses. This leads to a buffer space trade-off between servicing a request in memory mode vs. servicing it in disk mode. In this work, we develop a novel algorithm to minimize the buffer requirement to support a set of concurrent playbacks. Our algorithm makes a significant contribution in decreasing the total buffer requirement, especially when the user access pattern is biased in favor of a small set of files. The idea of the proposed scheme is elaborately modeled using an analytical formulation, and the optimality of the algorithm is proven. The proposed scheme can be used in combination with any existing disk-scheduling strategy without significant overhead. Our detailed simulation results confirm that beyond a certain threshold of the number of concurrent streams, less buffer space is required in supporting a playback by memory mode, and subsequently the proposed strategy effectively minimizes the overall buffer space under heavy-load conditions.

References

[Ant96] Mourad A (1996) Issues in the design of a storage server for video-on-demand. Multimedia Syst 4: 70–86

[AOG92] Anderson DP, Osawa Y, Govindan R (1992) A File System for Continuous Media. ACM Trans Comput Syst 10(4): 311–337

[BR96] Brubeck DW, Rowe LA (1996) Hierarchical storage management in a distributed VOD system. IEEE Multimedia Mag 3(3): 37–47

[CaRLR90] Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to Algorithms, 1st edition, Chapter 2. MIT Press, Cambridge, Mass.

[CKY93] Chen M-S, Kandlur DD, Yu PS (1993) Optimization of the grouped sweeping scheduling (GSS) with heterogeneous multimedia streams. ACM Multimedia, 1993, pp 235–242

[DS93] Dan A, Sitaram D (1993) Buffer Management Policy for an On-Demand Video Server. Technical Report IBM Research Report RC 19347. IBM Research Division, T. J, Watson Research Center, Yorktown Heights, N.Y.

[GC92] Gemmell J, Christodoulakis S (1992) Principles of Delay-Sensitive Multimedia Data Storage and Retrieval. ACM Trans Inf Syst 10(1): 51–90

[Gem93] Gemmell J (1993) Multimedia Network File Servers: Multi-Channel Delay-Sensitive Data Retrieval. In: Proc. of 1st ACM Multimedia Conf, October 1993. ACM Press, New York

[Gha95] Ghandeharizadeh S, Kim S, Shahabi C (1995) Continuous Display of Video Objects Using Multi-Zoned Disks. Technical report. University of Southern California

[GKS96] Ghandeharizadeh S, Kim SH, Shahabi C (1996) On disk scheduling and data placement for video servers. In: Proceedings of ACM Multimedia Systems, 1996

[KHS94] Kenchammana-Hosekote DR, Srivastava J (1994) Scheduling Continuous Media on a Video-On-Demand Server. In: Proc. of International Conference on Multimedia Computing and Systems, May 1994, Boston, Mass.

[OBRS94] Özden B, Biliris A, Rastogi R, Silberschatz A (1994) A Low-Cost Storage Server for Movie on Demand Databases. In: Proc. of VLDB, 1994

[PD93] Lougher P, Shepherd D (1993) The design of a storage server for continuous media. Comput J 36(1): 32–42

[Ren96] Tewari R, King R, Kandlur D, Dias DM (1996) Placement of Multimedia Blocks on Zoned Disks. In: Proceedings of SPIE West, 1996

[RVR92] Rangan P, Vin H, Ramanathan S (1992) Designing an on-demand multimedia service. IEEE Commun Mag 30(7): 56–65

[RW92] Reddy ALN, Wyllie J (1992) Disk Scheduling in a Multimedia I/O system. In: Proc. ACM Multimedia Conf, 1992. ACM Press, New York, pp 225–233
[RW94] Ruemmler C, Wilkes J (1994) An Introduction to Disk Drive Modeling. IEEE Comput 27(3)
[TF98] Triantafillou P, Faloutsos C (1998) Overlay striping and optimal parallel I/O in modern applications. Parallel Comput 24: 21–43
[VGGG94a] Vin HM, Goyal P, Goyal A, Goyal A (1994) A Statistical Admission Control Algorithm for Multimedia Servers. In: Proc. of ACM Multimedia Conf, October 1994, San Francisco, Calif., pp 33–40
[VGGG94b] Vin HM, Goyal A, Goyal A, Goyal P (1994) An Observation-Based Admission Control Algorithm for Multimedia Servers. In: Proc. of 1st IEEE International Conf. on Multimedia Computing and Systems, May 1994, Boston, Mass., pp 234–243
[vid92] Video StoreMagazine, 13 December 1992
[WGP94] Worthington B, Ganger G, Patt Y (1994) Scheduling Algorithms for Modern Disk Drives. In: Proc. of ACM SIGMetrics, May 1994

Appendix A
Time complexity

In addition to the optimality of the solution, the time complexity of the proposed algorithm is another important concern which determines its practicality. Given a set of disk-mode playbacks, $\mathbf{s}_D$, the algorithm sorts the playbacks with respect to their profit, whose time complexity is $\Theta(n \log n)$, where n is the number of playbacks.

In a practical situation, the profit of a service-mode change needs to be re-examined upon the arrival or departure of the playback. Since $g\mathcal{SMDP}_{arr}$ and $g\mathcal{SMDP}_{dep}$ are identical from the aspect of time complexity, we provide the analysis only on $g\mathcal{SMDP}_{arr}$ algorithm.

When a new request arrives, the algorithm sorts the disk-mode playbacks with respect to their profits, incurring an $\mathcal{O}(n \log n)$ time complexity. However, the property revealed in Lemma 4 allows reducing the complexity further. Lemma 4 states that even though there is an arrival or a departure of a playback, the relative ranking of the remaining playbacks in the sorted list does not change. Thus, when a new playback arrives, the existing disk-mode playbacks, $\mathbf{s}_D$, remain sorted properly with respect to their profits. The major task is finding a proper ranking for the newly arrived playback. The profit of change for each playback in $\mathbf{s}_D$ needs to be re-computed. The synergy can be found in insertion sort [CaRLR90]. This *insertion* and *re-evaluation* takes $\Theta(\log n)$ time.

Appendix B
Proofs

Theorem 3. *Given a set of streams,* $\mathbf{s}$*, their playbacks can be supported from the disk if and only if* $\sum_{i=1}^{n} r_i < B_{max}$*, regardless of the disk-scheduling algorithm.*

Proof. Intuitively, if the aggregate bandwidth required to support $\mathbf{s}$ exceeds the maximum bandwidth of the disk, the continuity requirement cannot be satisfied. We look at this condition and its proof via an analytical formulation.

(⇒) Let us assume that $\sum_{i=1}^{n} r_i > B_{max}$ with the continuity requirement satisfied. Continuity requirement in Eq. 1 can be re-written as follows:

$$r_i \left(\sum_{i=1}^{n} n_j b + B_{max} \mathcal{O}(\mathbf{s}) \right) < B_{max} n_i b \,. \tag{20}$$

Equation 20 holds for all $i = 1, \ldots, n$. Thus,

$$\left(\sum_{i=1}^{n} r_i \right) \left(\sum_{i=1}^{n} n_j b + B_{max} \mathcal{O}(\mathbf{s}) \right) < B_{max} b \left(\sum_{i=1}^{n} n_i \right) . \tag{21}$$

Equation 21 can be re-written as follows:

$$\left(\sum_{i=1}^{n} r_i \right) \mathcal{O}(\mathbf{s}) B_{max} < b \left(B_{max} - \sum_{i=1}^{n} r_i \right) \sum_{i=1}^{n} n_i \,. \tag{22}$$

The left-hand side of Eq. 22 should be greater than 0 regardless of the scheduling algorithm. However, since $B_{max} < \sum_{i=1}^{n} r_i$, the right-hand side of Eq. 22 is less than 0. Contradiction, and hence (⇒) holds.

⇐) If $\sum_{i=1}^{n} r_i < B_{max}$, then the bottom part of Eq. 11 is greater than 0, and thus the buffer requirement to support $\mathbf{s}$ with the respective disk-scheduling policy can be obtained from Eqs. 5, 6, 7 and 11. Hence, the claim.

We formulate the amount of reduction in buffer requirement resulting from removing a playback from the disk, i.e., converting it to memory-mode playback. For notational simplicity, let R be the aggregate bandwidth requirement for disk-mode playbacks, i.e., $R = \sum_{s_i \in \mathbf{s}_D} r_i$. Removing a playback $s_i \in \mathbf{s}_D$ will result in reduction of the buffer requirement. We can compute the amount of *reduction* as in Eq. 23:

$$\begin{aligned} &\Theta(\mathcal{M}(\mathbf{s}_D)) - \Theta(\mathcal{M}(\mathbf{s}_D \setminus \{s_i^k\})) \\ &= \frac{R}{B_{max} - R} - \frac{R - r_i}{B_{max} - (R - r_i)} \\ &= \frac{r_i B_{max}}{(B_{max} - R)(B_{max} - (R - r_i))} \,. \end{aligned} \tag{23}$$

Lemma 1. *Assume that* $\mathbf{s}$ *and* $\mathbf{s}'$ *consist of only disk-mode playbacks. If* $\mathbf{s}' \subset \mathbf{s}$ *and* $s_i^k \in \mathbf{s}'$*, then* $\mathcal{P}_{\mathbf{s}'}(s_i^k \mapsto \check{s}_i^k) \leq \mathcal{P}_{\mathbf{s}}(s_i^k \mapsto \check{s}_i^k)$.

Proof. $\mathcal{P}_{\mathbf{s}}(s_i^k \mapsto \check{s}_i^k) = \mathcal{M}(\mathbf{s}_D) - \mathcal{M}(\mathbf{s}_D - \{s_i^k\}) - \Delta_i^{k-1} r_i$. We have $R = \sum_{r_i \in \mathbf{s}} r_i$ and $R' = \sum_{r_i \in \mathbf{s}'} r_i$. Now,

$$\begin{aligned} &\mathcal{P}_{\mathbf{s}}(s_i^k \mapsto \check{s}_i^k) - \mathcal{P}_{\mathbf{s}'}(s_i^k \mapsto \check{s}_i^k) \\ &= \underbrace{\left\{ \mathcal{M}(\mathbf{s}_D) - \mathcal{M}(\mathbf{s}_D - \{s_i^k\}) \right\}}_{A} \\ &\quad - \underbrace{\left\{ \mathcal{M}(\mathbf{s}') - \mathcal{M}(\mathbf{s}'_D - \{s_i^k\}) \right\}}_{B} . \end{aligned} \tag{24}$$

By applying Eq. 23 to A and B in Eq. 24, we obtain the following:

$$\begin{aligned} &\mathcal{P}_{\mathbf{s}}(s_i^k \mapsto \check{s}_i^k) - \mathcal{P}_{\mathbf{s}'}(s_i^k \mapsto \check{s}_i^k) \\ &= \frac{B_{max} r_i}{(B_{max} - R)(B_{max} - (R - r_i))} \\ &\quad - \frac{B_{max} r_i}{(B_{max} - R')(B_{max} - (R' - r_i))} \,. \end{aligned}$$

Since R is greater than R', $\frac{B_{max}r_i}{(B_{max}-R)(B_{max}-(R-r_i))} - \frac{B_{max}r_i}{(B_{max}-R')(B_{max}-(R'-r_i))}$ is greater than 0. Hence, the claim.

Lemma 2. *Let* $\mathbf{s} = \{s_1, \ldots, s_n\}$ *and* $\mathbf{v}$ *be the set of disk-mode playbacks and the set of playbacks to be changed from disk mode to memory mode, respectively. Then, the decrease in buffer memory resulting from the changes is*

$$\Theta(\mathcal{M}(\mathbf{s}_D)) - \Theta(\mathcal{M}(\mathbf{s}_D \setminus \mathbf{v})) = \frac{B_{max}\sum_{s_i\in\mathbf{v}} r_i}{(B_{max} - \sum_{i=1}^{n} r_i)(B_{max} - (\sum_{i=1}^{n} r_i - \sum_{s_i\in\mathbf{v}} r_i))}.$$

Proof. We use induction to prove this lemma. Let R be the sum of disk-mode playbacks in $\mathbf{s}$, $\sum_{i=1}^{n} r_i$.

In case $|\mathbf{v}| = 1$, the lemma is simplified into Eq. 23. Hence, the lemma holds.

In case $|\mathbf{v}| = 2$, let $\mathbf{v} = \{s_i, s_k\}$. We consider that service-mode changes one by one. Change $s_i \mapsto \check{s}_i$ will decrease the buffer requirement as follows:

$$\Theta(\mathcal{M}(\mathbf{s})) - \Theta(\mathcal{M}(\mathbf{s} \setminus \{s_i\})) = \frac{r_i B_{max}}{(B_{max} - R)(B_{max} - (R - r_i))}. \quad (25)$$

As a result of $s_i \mapsto \check{s}_i$, the sum of the playback rates from the disk mode decreases to $R - r_i$. Thus, a decrease in the buffer requirement resulting from $s_k \mapsto \check{s}_k$ is as follows:

$$\Theta(\mathcal{M}(\mathbf{s} \setminus \{s_i\})) - \Theta(\mathcal{M}(\mathbf{s} \setminus \{s_i\} \setminus \{s_j\})) = \frac{r_j B_{max}}{(B_{max} - (R - r_i))(B_{max} - (R - r_i - r_j))}. \quad (26)$$

The sum of Eq. 25 and Eq. 26 results in Eq. 27:

$$\Theta(\mathcal{M}(\mathbf{s})) - \Theta(\mathcal{M}(\mathbf{s} \setminus \{s_i, s_j\})) = \frac{(r_i + r_j)B_{max}}{(B_{max} - R)(B_{max} - (R - r_i - r_j))}. \quad (27)$$

The result in Eq. 27 is not affected by the order of the changes. Hence, the lemma holds in the case $|\mathbf{v}| = 2$.

Now, let us assume that the lemma holds for $|\mathbf{v}| = k-1$. When $|\mathbf{v}| = k$, the decreases in the buffer requirement is the sum of the decrease obtained via k changes. For notational simplicity, let us number the playback rates in $\mathbf{v}$ as $r'_1, \ldots, r'_k, k < n$. The decrease from the first $k-1$ changes is obtained by Eq. 28, since we assumed that the lemma holds when $|\mathbf{v}| = k - 1$:

$$\Theta(\mathcal{M}(\mathbf{s})) - \Theta(\mathcal{M}(\mathbf{s} \setminus \{s'_1, \ldots, s'_{k-1}\})) = \frac{\sum_{i=1}^{k-1} r'_i B_{max}}{(B_{max} - R)(B_{max} - (R - \sum_{i=1}^{k-1} r'_i))}. \quad (28)$$

Now, changing the service mode of r_k brings the following decrease:

$$\Theta(\mathcal{M}(\mathbf{s} \setminus \{s'_1, \ldots, s'_{k-1}\})) - \Theta(\mathcal{M}(\mathbf{s} \setminus \{s'_1, \ldots, s'_k\})) = \frac{r'_k B_{max}}{(B_{max} - (R - \sum_{i=1}^{k-1} r'_i))(B_{max} - (R - \sum_{i=1}^{k} r'_i))}. \quad (29)$$

The total decrease in the buffer requirement from changing the service mode of $\mathbf{v}$ is the sum of Eq. 28 and Eq. 29, which can be re-written as follows:

$$\Theta(\mathcal{M}(\mathbf{s})) - \Theta(\mathcal{M}(\mathbf{s} \setminus \mathbf{v})) = \frac{\sum_{i=1}^{k} r'_i B_{max}}{(B_{max} - R)(B_{max} - (R - \sum_{i=1}^{k} r'_i))}. \quad (30)$$

Hence, the claim.

Lemma 3. *Let* r_x *and* r_y *be the rates for two playbacks* s_x *and* s_y*, respectively, and* $r_x > r_y$*. Let* $\mathbf{s}$ *be the set of ongoing playbacks. Let* ΔM_x *and* ΔM_y *be the additional amounts of memory to accommodate* s_x *and* s_y *into* $\mathbf{s}$ *by disk mode, respectively. Then,* $\frac{\Delta M_x}{\Delta M_y} > \frac{r_x}{r_y}$.

Proof. Let $\mathbf{s} = \{s_1, \ldots, s_n\}$ and all playbacks are in disk mode. The buffer size for each playback is proportional to its playback rate, i.e., playback s_i is allocated $\frac{r_i}{\sum_{i=1}^{n} r_i}\mathcal{M}(\mathbf{s})$ amount of memory, and thus we can re-write $\mathcal{M}(\mathbf{s})$ as $\mathcal{M}(\mathbf{s}) = \mathcal{K}_\mathbf{s}\sum_{i=1}^{n} r_i$. To simplify the expression, we introduce another variable $\mathcal{K}_\mathbf{s}$ which is a coefficient determined by the disk operational characteristics, i.e., disk-scheduling policy, seek time, etc., and $\mathbf{s}$. Since operational characteristics of the disk are not affected by the system or above layer entity, $\mathcal{K}_\mathbf{s}$ can be interpreted as a function of only $\mathbf{s}$. Given this, the claim is written as in Eq. 31. Let $R = \sum_{i=1}^{n} r_i$.

$$\text{If } r_x > r_y, \text{ then } \frac{\mathcal{K}_{\mathbf{s}\cup\{r_x\}}(R + r_x) - \mathcal{K}_\mathbf{s}R}{\mathcal{K}_{\mathbf{s}\cup\{r_y\}}(R + r_y) - \mathcal{K}_\mathbf{s}R} > \frac{r_x}{r_y}. \quad (31)$$

Then,

$$R(\mathcal{K}_{\mathbf{s}\cup\{r_x\}} - \mathcal{K}_\mathbf{s})r_y + \mathcal{K}_{\mathbf{s}\cup\{r_x\}}r_x r_y > R(\mathcal{K}_{\mathbf{s}\cup\{r_y\}} - \mathcal{K}_\mathbf{s})r_x + \mathcal{K}_{\mathbf{s}\cup\{r_y\}}r_x r_y. \quad (32)$$

Thus,

$$\underbrace{r_x r_y(\mathcal{K}_{\mathbf{s}\cup\{r_x\}} - \mathcal{K}_{\mathbf{s}\cup\{r_y\}})}_{P1} + \underbrace{R((\mathcal{K}_{\mathbf{s}\cup\{r_x\}} - \mathcal{K}_\mathbf{s})r_y - (\mathcal{K}_{\mathbf{s}\cup\{r_y\}} - \mathcal{K}_\mathbf{s})r_x)}_{P2} > 0.$$

$\mathcal{K}_{\mathbf{s}\cup\{r_x\}}$ is greater than $\mathcal{K}_{\mathbf{s}\cup\{r_y\}}$, and hence $P1$ is greater than 0. $P2$ is greater than or equal to 0 since $\frac{\mathcal{K}_{\mathbf{s}\cup\{r_x\}} - \mathcal{K}_\mathbf{s}}{\mathcal{K}_{\mathbf{s}\cup\{r_y\}} - \mathcal{K}_\mathbf{s}} \geq \frac{r_x}{r_y}$. Hence, $\frac{\Delta M_x}{\Delta M_y} > \frac{R_x}{R_y}$ holds. Hence, the claim.

Lemma 4. *Given a set of disk-mode playbacks* $\mathbf{s}$ *and* $\mathbf{t}$ *such that* $\mathbf{t} \subset \mathbf{s}$*, if* $\mathcal{P}_\mathbf{s}(s' \mapsto \check{s}') < \mathcal{P}_\mathbf{s}(s'' \mapsto \check{s}'')$*, then* $\mathcal{P}_\mathbf{t}(s' \mapsto \check{s}') < \mathcal{P}_\mathbf{t}(s'' \mapsto \check{s}'')$.

Proof. Let $\sum_{s_i^k\in\mathbf{s}} r_i$ and $\sum_{s_i^k\in\mathbf{t}} r_i$ be R_s and R_t, respectively.

$$\mathcal{P}_\mathbf{s}(s' \mapsto \check{s}') = \frac{B_{max}r'}{(B_{max} - R_s)(B_{max} - R_s - r')} - \Delta' r'.$$

From the assumption $\mathcal{P}_\mathbf{s}(s' \mapsto \check{s}') < \mathcal{P}_\mathbf{s}(s'' \mapsto \check{s}'')$,

$$\frac{B_{max}r'}{(B_{max} - R_s)(B_{max} - R_s - r')} - \Delta' r' < \frac{B_{max}r''}{(B_{max} - R_s)(B_{max} - R_s - r'')} - \Delta'' r''$$

$$\Leftrightarrow$$

$$\frac{B_{max}r'}{(B_{max} - R_s)(B_{max} - R_s - r')} - \frac{B_{max}r''}{(B_{max} - R_s)(B_{max} - R_s - r'')} < \Delta' r' - \Delta'' r''$$

$\frac{B_{max}r'}{(B_{max}-R_s)(B_{max}-R_s-r')}$ can be written as $B_{max}\left(\frac{1}{B_{max}-R_s-r'}-\frac{1}{B_{max}-R_s}\right)$. Thus, the above equation can be written as follows:

$$B_{max}\left(\frac{1}{B_{max}-R_s-r'}-\frac{1}{B_{max}-R_s-r''}\right) < \Delta' r' - \Delta'' r'' .$$

Since $R_s > R_t$,

$$B_{max}\left(\frac{1}{B_{max}-R_t-r'}-\frac{1}{B_{max}-R_t-r''}\right) < B_{max}\left(\frac{1}{B_{max}-R_s-r'}-\frac{1}{B_{max}-R_s-r''}\right) .$$

Thus,

$$B_{max}\left(\frac{1}{B_{max}-R_t-r'}-\frac{1}{B_{max}-R_t-r''}\right) < \Delta' r' - \Delta'' r''$$
$$\Leftrightarrow \frac{B_{max}}{B_{max}-R_t-r'} - \Delta' r' < \frac{B_{max}}{B_{max}-R_t-r''} - \Delta'' r''$$
$$\Leftrightarrow \mathcal{P}_{\mathbf{t}}(s' \mapsto \check{s}') < \mathcal{P}_{\mathbf{t}}(s'' \mapsto \check{s}'') .$$

Hence, the claim.

YOUJIP WON received the B.S. and M.S. degree in Computer Science from the Department of Computer Science, Seoul National University, Seoul, Korea in 1990 and 1992, rsepectively and the Ph.D. in Computer Science from the University of Minnesota, Minneapolis in 1997. After finishing his Ph.D, He worked as Server Performance Analyst at Server Architecture Lab., Intel Corp. after the completion of his Ph.D. Since 1999, he has been on the board of faculty members in Division of Electrical and Computer Engineering, Hanyang University, Seoul, Korea. His current research interests include Multimedia Systems, Internet Technology, and Performance Modeling and Analysis. He is a member of ACM and IEEE.