Use of Analytical Performance Models for System Sizing and Resource Allocation in Interactive Video-on-Demand Systems Employing Data Sharing Techniques

Mary Y.Y. Leung, John C.S. Lui, *Member*, *IEEE*, and Leana Golubchik, *Member*, *IEEE*

Abstract—In designing cost-effective video-on-demand (VOD) servers, efficient resource management and proper system sizing are of great importance. In addition to large storage and I/O bandwidth requirements, support of interactive VCR functionality imposes additional resource requirements on the VOD system in terms of storage space, as well as disk and network bandwidth. Previous works have used "data sharing" techniques (such as batching, buffering, and adaptive piggybacking) to reduce the I/O demand on the storage server. However, such data sharing techniques complicate the provision of VCR functions and diminish the amount of benefit that can be obtained from data sharing techniques. The *main* contribution of this paper is a simple, yet powerful, *analytical modeling approach* which allows for analysis, system sizing, resource allocation, and parameter setting for a fairly general class of "data sharing" techniques which are used in conjunction with the providing of VCR-type functionality. Using this mathematical model, we can determine the proper amount of resources to be allocated for normal playback as well as for service of VCR functionality requests while satisfying predefined system performance requirements. To illustrate the usefulness of our model, we focus on a specific data sharing scheme which combines the use of batching, buffering, and adaptive piggybacking, as well as allows for the use of VCR functions. We show how to utilize this mathematical model for system sizing and resource allocation purposes—that is, how to distribute the available resources between the service of normal playback and VCR functionality requests under various workloads and resource price ratios, so as to obtain the lowest system cost.

Index Terms—Video-on-demand, data sharing techniques, resource allocation, system sizing.

1 Introduction

Advances in information and communication technologies have made multimedia services feasible. Among these services, interactive TV and video-on-demand (VOD) have been identified as two important applications [8]. In a VOD system, video objects are stored on a storage server and delivered to users upon request. Unlike textual data, video objects occupy large amounts of storage space and require large amounts of I/O bandwidth for real-time delivery. Thus, upon admission of a user, sufficient amounts of resources have to be reserved in order to provide an acceptable level of service to viewers. A significant amount of research effort has been dedicated to designing a fault-tolerant VOD server, as well as investigating techniques for the efficient use of resources in VOD servers [7], [11], [14], [19], so that the amount of resources needed to satisfy predefined performance measures can be reduced. For a survey of such approaches, please refer to [9].

Several techniques have been proposed for increasing the number of concurrent viewers of *popular* objects through the *sharing of data* and, consequently, the sharing of system resources, such as I/O bandwidth and buffer space. One such technique is batching (e.g., as in [6]), in which one I/O stream is dedicated to servicing several viewers, for instance, those that arrived within a predefined and relatively short time interval. As long as viewers are willing to wait for a small amount of time, where the maximum waiting time is the length of this time interval, all requests for the same object arriving within the time interval can be serviced using one I/O stream, thereby reducing the overall amount of I/O bandwidth necessary. Another technique for reducing the amount of needed I/O bandwidth is buffering (e.g., as in [22]), where careful buffer replacement algorithms insure that viewers can fetch the needed data blocks from the VOD system buffers instead of from the disk subsystem. Last, the adaptive piggybacking method [10] takes advantage of the fact that video display rates can be modified (without the user noticing the change in video quality) in order to *dynamically* merge ongoing displays of a popular object into a single group which can be served by one I/O stream, thereby reducing the overall I/O bandwidth requirements. In general, the basic idea behind these

- *M.Y.Y. Leung is with Accenture, 23/F Wing-On Center, 111 Connaught Rd., Central, Hong Kong. E-mail: mary.y.leung@accenture.com.*
- *J.C.S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong. E-mail: cslui@cse.cuhk.edu.hk.*
- *L. Golubchik is with the Department of Computer Science and UMIACS, University of Maryland, College Park, MD 20742. E-mail: leana@cs.umd.edu.*

Manuscript received 24 Mar. 1998; revised 6 Dec. 1999; accepted 31 Aug. 2000.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 106442.

approaches is to group users requesting the same object, so as to increase the degree of *data sharing* and, thereby, reduce the overall system I/O bandwidth demand.

Apart from normal playback, it is also desirable to provide interactive VCR functions, such as fast forward with viewing (FF), rewind with viewing (RW), and pause/resume (PAU). Part of the difficulty here is that the provision of VCR functions often requires additional system resources and it is usually not desirable to reserve such resources for each request in the system since VCR resource reservation on a per-request basis may be utilized infrequently and, hence, these resources will not be utilized efficiently. There are various schemes in the literature for supporting interactive VCR functions in VOD systems, for instance, [3], [13], [15]. However, among all these approaches, few have addressed the problem of incorporating VCR functionality in a system which uses data sharing techniques for the service of normal playback requests.

The basic problem with combining VCR type functionality with data sharing techniques is as follows: All data sharing techniques (such as batching, buffering, and adaptive piggybacking) improve the cost-effectiveness of a VOD system by serving groups of clients requesting the same video with a single set of resources. However, they all suffer from the problem of *where* to obtain resources when one of the viewers in a group requests a VCR-type functionality and, thus, is no longer part of the data sharing group since, in this case, resources are not reserved on a per-client basis but only on a per-group basis. Even if some resources are reserved for performing VCR-type functions, another problem (and perhaps a more difficult one) is where does the system obtain resources for a viewer that is ready to go back to the normal playback but is *no longer part of a group*. We refer to such viewer as a *fallen out* viewer.

The basic approaches to dealing with this problem include: 1) combining data sharing techniques (such as batching and buffering) in order to increase the probability that a client that left a group to perform VCR-type functionality will rejoin another group once they are ready to return to normal playback (e.g., as in [18], [17]) and 2) setting aside a relatively small pool of resources to provide for clients who need to perform VCR-type functionality (e.g., as in [13], [2]) and/or for those that are not able to rejoin another group. Once these basic approaches are employed, the *main difficulty* is in the setting of parameters, i.e., determining: a) the size of the pool of resources (for VCR-type functionality), b) the batching interval (or the number of I/O streams) to use per video object, c) the amount of buffer space to use per video object (or per group), etc., such that the probability of allowing VCR-type functionality and the cost-effectiveness of the whole system (given some QoS requirements) are optimized.

Related work on allowing users in a data sharing group to perform VCR functionality is as follows: In [25], the authors propose a method for allowing users in a data sharing group to pause and eventually resume back to normal playback. This method uses additional buffer space resources to gain improvements in the amount of I/O bandwidth needed to support pause/resume of batched requests. In [2], the authors propose a scheme for providing interactive functionality to different multicast groups (each multicast group can be viewed as a batch group). The main idea there is to utilize the client-side resources, i.e., provide enough buffer space in the viewer's set-top box in conjunction with reserving some emergency channels in the network during high loads such that the system is able to service an interactive request by a viewer with a reasonably high probability. In [18], the authors propose a protocol which splits an interactive user from a batch and serves the user using a dedicated I/O stream. This interactive viewer is then "attached" back to a batch of requests by using additional buffer space needed to bridge the gap between a batched group and this interactive stream. However, none of the approaches mentioned above [2], [18], [25] solve the problem of system sizing, i.e., *how* one should set design parameters and, hence, allocate different resources to different types of display modes (either normal or interactive) so as to provide a certain level of QoS and, at the same time, reduce the overall system cost.

The contributions of this paper are as follows: The *main* contribution is our analytical model which is used for system sizing and for determining the amount of resources needed for supporting normal playback and VCR functionality, while satisfying predefined system performance requirements. This model allows us to maximize the benefits of data sharing techniques in the presence of interactive VCR functions. Consequently, our paper uses the analytical model for *system sizing* purposes in interactive VOD systems using data sharing techniques. Specifically, we divide the overall resources into those required for normal playback and those required for the service of a VCR request. The goal is to reduce the probability of viewers "falling out" of a group (after resuming to normal playback), thereby increasing the benefits of data sharing, but without significantly increasing the system costs. We use our model to calculate the amount of resources required for normal playback such that, upon resumption to normal playback, additional resources (which were dedicated to the VCR function) can be released with a prespecified probability, which reflects the system's QoS requirements. To further increase this probability, we use the adaptive piggybacking technique to facilitate the joining of an existing group by the *fallen out* viewers. Hence, although a fallen out viewer may have to hold on to additional resources to resume normal playback, we attempt to reduce this resource holding time. All this implies that the total amount of resources reserved for service of VCR requests can be reduced. Thus, another contribution of our work is a novel data sharing technique that combines batching, buffering, and adaptive piggybacking in order to improve the cost-effectiveness of an *interactive* VOD system.

The organization of the paper is as follows: In Section 2, we present the background information and a brief survey of batching, buffering, and adaptive piggybacking techniques. In Section 3, we describe our system and introduce the model for VCR display. In Sections 4 and 5, we continue with the exposition of our analytical model and describe details of resource allocation for normal playback, as well as VCR modes. In Section 6, we describe how to apply our model to solving system sizing problems in interactive VOD servers which employ data sharing techniques. Last, our conclusions are given in Section 7.

2 Background on Data Sharing and VCR Support

In this section, we briefly describe several basic approaches to reducing I/O demand on a VOD server through data sharing. Note that these techniques are orthogonal, i.e., they can be combined in order to better utilize the I/O resources in a VOD system. Last, we also briefly survey several approaches to providing VCR functionality in conjunction with data sharing schemes.

1. **Data sharing via batching** (e.g., as in [6]). One way to satisfy independent playback requests is to dedicate an I/O stream to each request. However, this (potentially) requires a large number of I/O streams. Given that there will be a relatively large number of requests for popular videos, we can batch together requests separated by relatively small time intervals and serve the entire batched group using a single I/O stream. As long as the viewers are willing to tolerate a certain latency, batching can help to achieve a substantial reduction in the required server I/O bandwidth capacity. However, this reduction in I/O streams will increase the waiting time of the viewers, thus, the trade-off is between the I/O bandwidth utilization and the waiting time of viewers. There are many batching policies that one can consider. One simple approach is to start the video periodically at predetermined time intervals. In this case, we can easily evaluate the saving in I/O bandwidth. For instance, if λ is the average arrival rate of requests for a particular video and b is the batching time interval for that video, then the average fraction of I/O savings is $\frac{(\lambda b-1)}{\lambda b}$, provided $\lambda b > 1$.
2. **Data sharing via buffering** (e.g., as in [5], [12], [20], [23]). Buffering is a technique which attempts to bridge the temporal gap between "sufficiently close" successive requests for the same video using buffer space. This is achieved by retaining the video frames fetched by the previous stream in the buffer space and allowing streams corresponding to subsequent requests for the same video to be served from the buffer space rather than by using an I/O stream. The basic idea is to use buffer space to promote data sharing and, hence, reduce the demand for I/O bandwidth. Since even retaining a few minutes of the video in the buffer is costly, it is important to determine proper buffer replacement algorithms. There are several existing research results for the use of buffering techniques. In [5], [12], [20], [23], the authors explore the benefits of caching continuous media data to reduce the utilization of I/O bandwidth. By making efficient use of the data already fetched into the buffer space, the number of concurrent viewers can be increased. In addition, in [4], the cost-performance trade-offs of various buffering and caching strategies is studied by varying the buffer size, disk utilization, and disk characteristics.
3. **Data sharing via adaptive piggybacking** (e.g., as in [1], [10]). Adaptive piggybacking exploits the fact that video display rates can be altered without a user-perceived loss in the quality of that video, to "merge" requests for the same video and, thus, allow the corresponding viewers to share a single I/O stream, thereby *dynamically* reducing the I/O demand on the VOD system. Although the reduction in I/O demand is not as high as that of "traditional" batching, the advantage is that viewers do not experience the additional latency associated with batching nor is there a need for additional buffer space required by buffering techniques. This approach assumes that the storage server is capable of altering the display rate of a video. For instance, since the display proceeds at a fixed rate, the slow down in the *effective* display rate can be done by inserting additional frames; similarly, the display rate can be *effectively* increased by removing frames. It has been pointed out that differences as large as ± 5 percent in the display rates will not be perceivable by the viewers and, thus, will not result in a degradation in video quality [10]. Thus, the basic idea behind this approach is to *dynamically* merge two (or more) I/O streams into one. Justifications of the feasibility and evidence of the benefits of this approach are given in [10].

   As already mentioned, these basic data sharing techniques are orthogonal and can be combined into a variety of data sharing policies. As an example of one combination of batching and buffering, we now briefly describe the static partitioning scheme proposed in [22]. Here, we choose this particular scheme as this is our point of departure for the remainder of the paper.
4. **Data sharing via static partitioning** (e.g., as in [22], [17]). Static partitioning is one of the partitioned buffer management strategies proposed in [22], which combines the use of batching and buffering and is developed based on a buffer refreshing process. This method is similar to batching schemes in which an I/O stream for a particular video is restarted periodically at predefined time intervals. In addition, some amount of buffer space, termed a partition, is associated with each of these I/O streams. This allocated buffer space allows for the retaining of the video frames in memory for a certain period of time, which is termed the *viewer enrollment window*. Thus, this scheme allows for the sharing of data between requests which arrive during the viewer enrollment window; these requests are served using a single set of resources, i.e., a single I/O stream [6] plus the buffer space associated with it. This facilitates a reduction in the additional latency associated with "pure" batching at the cost of additional buffer space. Fig. 1 illustrates a scenario for the static partitioning scheme, where each batch of viewers is served using an I/O stream and a partition of buffer space. Viewers that arrive before the closing of the viewer enrollment window read the frames from the buffer partition (we call these type 2 viewers). Viewers that arrive after the window is closed (we call these type 1 viewers) are

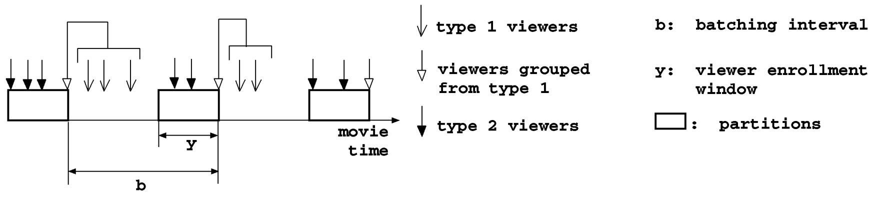


Fig. 1. Different types of viewers in static partitioning.

queued up to wait for the next restart of the video. The longest time a viewer has to wait occurs when he/she arrives just after the viewer enrollment window is closed. In this case, the maximum waiting time equals the batching interval b (the interval between initiations of two consecutive I/O streams) minus the length of the viewer enrollment window y. An interesting approach which provides an optimal value of the viewer enrollment window when VCR functionality is not used is given in [23].

Let us briefly survey schemes for combining data sharing techniques with VCR functionalities.

5. **Data-sharing with VCR support** (e.g., as in [25], [2], [18]). In [25], the authors propose a method for allowing users in a data sharing group to perform pause/resume type functionality only. This method uses additional buffer space resources to gain improvements in the amount of I/O bandwidth needed to support pause/resume functionality for batched requests. In [2], the authors propose a scheme for providing interactive functionalities to different multicast groups, where the basic technique is to utilize client-side resources, i.e., to provide buffer space in the viewer's set-top box in conjunction with reserving some emergency channels in the network during high loads such that the system is able to service an interactive request by a viewer with a reasonably high probability. In [18], the authors propose a protocol which "splits" an interactive user from a batch and serves him/her by using a dedicated I/O stream. This interactive viewer is then "attached" back to a batch of requests by using additional buffer space needed to bridge the temporal gap between a batched group and this interactive stream. We should note that part of the motivation for us considering the addition of the adaptive piggybacking scheme to the set of data sharing techniques used (refer to Section 5) is to be able to deallocate such additional resources (i.e., such as those used to aid a viewer in joining an existing batch) before the user exits the system and, thereby, improves the cost-effectiveness of the system. Most of these approaches mentioned are evaluated through extensive *simulations* to cover a wide range of possible design parameters. Thus, partly what is desirable here is a better methodology for making design decisions and evaluating resulting schemes.

3 System Model

For ease of exposition as well as ease of illustration of the mathematical model, we focus on the partitioned buffer management strategy termed static partitioning [22] as a specific combination of data sharing techniques. The static partitioning scheme, proposed in [22], addresses normal playback functionality only. However, VCR functions such as fast-forward (FF), rewind (RW), and pause (PAU) are important and necessary features, which need to be provided to the viewers. A fundamental problem in providing such interactive features in conjunction with data sharing techniques is the need for additional resources not only for servicing VCR requests, but also for allowing viewers to resume from a VCR mode to normal playback. The difficulty here is that, at this resume point, viewers are no longer part of a group of requests sharing the I/O and buffer space resources, i.e., the resources are reserved on a per-group rather than per-viewer basis. In this paper, we present a model which facilitates the calculation of distribution of resources between normal playback and VCR functionality, so as to reduce the additional load resulting from the users resuming from a VCR operation to normal playback. In addition, this model aids in making system sizing decisions, such that the overall cost-effectiveness of the system can be improved.

3.1 VCR Functionality

The service of VCR requests can be divided into the following two phases.

Phase 1. Displaying the VCR-version of the video. When a viewer issues a VCR requests, other than a pause, additional resources have to be allocated so that the viewer can watch the VCR-version of the video. Let μ_{VCR} denote the mean time spent in phase 1.

Phase 2. Resuming to normal playback. When resuming from a VCR operation to normal playback, the possibility of releasing the resources allocated during Phase 1 depends on the time point in the video at which the viewer resumes. If the frames the viewer needs are already in the VOD system buffer when the viewer resumes, i.e., in one of the existing partitions, then the viewer can read the data directly from the partition in which he/she resumes. In this case, the viewer is said to *join* that partition, and the resources allocated in Phase 1 can be released. We denote the probability of a viewer resuming from a VCR operation at an *existing* partitions by P^*. On the other hand, if the viewer cannot resume at any of the existing partitions, then to continue normal playback without any delay, the viewer has to make use of the resources allocated in Phase 1 until a viewer can join an existing partition (if that is possible). Thus, a viewer is said to have a *hit* if he/she can join an existing partition when resuming to normal playback after a VCR request, such that additional resources allocated to that viewer in Phase 1 can be released in Phase 2. Otherwise, the viewer is said to have a *miss* if he/she

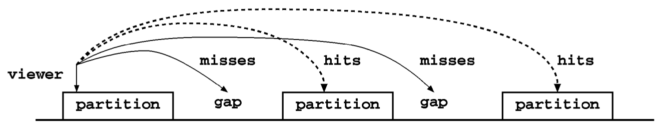


Fig. 2. Hits and misses for viewers resuming from VCR requests.

resumes at a *gap* between partitions. The hit and miss events and their corresponding probabilities are illustrated in Fig. 2. Let μ_{hold} denote the mean time before the resources allocated in Phase 1 can be released. In this paper, the allocation of resources for normal playback ensures that there is a minimum probability of P^* for the viewers to resume at an existing partition after a VCR request. Therefore, with the probability of $(1 - P^*)$, the viewer must hold on to the resources intended for VCR requests when he/she resumes. Thus, the expected time spent in Phase 2 is given by $(1 - P^*)\mu_{hold}$.

3.2 Normal Playback

As already stated, the allocation of resources in Phase 1 of performing a VCR operation is usually inevitable. On the other hand, whether or not these allocated resources can be released depends on the resuming position of the viewer. Therefore, our goal is to maximize the probability of releasing the allocated resources at a resume point so as to reduce the consumption of reserved resources. Consequently, more resources will be available for serving future requests, such as forthcoming VCR requests or even requests for nonpopular videos. Clearly, the size and the number of partitions used for servicing normal playback, together with the duration of VCR requests will greatly affect the probability of resuming at an existing partition. The normal playback scheme, derived based on the static partitioning technique, is used to calculate the amount of resources allocated to each video so as to ensure the following: *When a viewer resumes from a VCR request, there is a probability of at least P^* that the additional resources allocated in Phase 1 can be deallocated in Phase 2.*

Since our goal is to maximize the probability of a hit, it is natural to consider the behavior of VCR requests in determining the *configuration* of the system, i.e, in determining how much buffer resources and I/O streams should be reserved for normal playback. We study the behavior of each VCR operation as a function of its respective probability density function (pdf), $f(x)$, where x is the amount of video time spent in a VCR request. The pdf of VCR requests can be obtained by collecting statistics and is defined in the interval $[0, l]$, where l is the length of the video (in units of time). Note that a pause of x time units where $x > l$ is equivalent to a pause of $x \bmod l$ time units, e.g., the situation where $l = 120$ minutes and $x = 130$ minutes is equivalent to pausing for 10 minutes since a video is restarted periodically.

Based on the distribution of the duration of VCR requests, we perform resource allocation for normal playbacks such that the probability of holding on to the resources upon resume can be minimized. This resource allocation is calculated based on the mathematical model we propose, which is presented in Section 4. In this model, we assume that the arrivals of requests for a popular video are distributed according to a Poisson process with rate λ. This is a reasonable model of the arrival process since we expect the VOD system to have a large user population. Based on the above principles, we determine the expected hit probability upon resume to normal playback under various system configurations. Intuitively, increasing the size of the partitions will increase the hit probability because the fraction of a video residing in buffers can be increased. Consider the extreme case where the entire video is buffered, thus increasing the hit probability to 1. However, this may not yield a system with the lowest operating cost if relatively little time is spent in a VCR request, in which case, buffering the entire video would be too costly. Therefore, we use our model to determine a "configuration" which yields the lowest cost for supporting a given number of concurrent viewers for both normal playback and VCR functions.

3.3 Model for VCR Display

In this paper, we model the additional I/O resources needed for VCR functionality by an $M/M/m$ queue, so as to provide VCR service with a predefined level of QoS guarantee. For instance, we could ensure that the average time that a viewer has to wait to switch to VCR mode is less than three seconds. The details of QoS guarantees are presented at the end of this section. The notation related to normal and VCR control is summarized in Table 1. For simplicity of presentation, let us concentrate on the derivation of resource distribution based on a single video. We assume the arrival rate of VCR requests to be Poisson with a rate of λ_{VCR} per minute for each viewer, i.e., if, on the average, a viewer issues z VCR requests per video of length l minutes, we have:

$$\lambda_{VCR} = z/l. \tag{1}$$

TABLE 1
Major Notations Involved in VCR Model and Normal Playback Model

Notation	Definition
m	no. of I/O streams for servicing VCR requests
λ_v	total arrival rate of VCR requests
μ_{VCR}	mean time spent in phase 1 of VCR functions
μ_{hold}	mean time spent in phase 2 of VCR functions
μ_v	mean time spent in utilizing VCR resources
P^*	Probability of hit upon resume
B	Buffer reserved for playback (mins)
n	no. of I/O streams for normal playback
l	length of a movie (mins)
w	maximum waiting time for starting a movie
λ	arrival rate of normal playback requests
R_{FF}	Rate of fast forward
R_{RW}	Rate of rewind
R_{PB}	Rate of normal playback

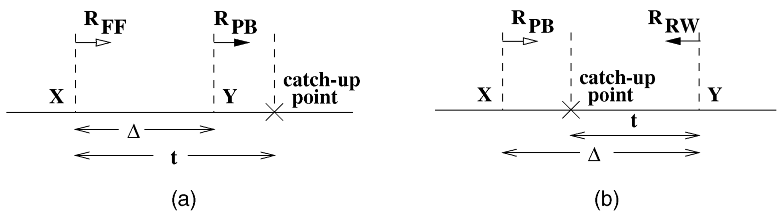


Fig. 3. Scenarios for catching up to viewers (a) in front (b) behind.

Since we assumed that the arrival rate of the normal playback follows the Poisson distribution of rate λ, the total arrival rate of VCR requests to the VOD system, λ_v, is given by $\lambda l \lambda_{VCR}$. Using (1), we have $\lambda_v = \lambda z$. The time spent in utilizing the servers in the $M/M/m$ queue corresponds to the time spent by the viewers utilizing additional resources for servicing VCR requests and for continuing normal playback (in case of a miss) before these viewers can join an existing partition. It is assumed to be exponentially distributed with mean μ_v. As mentioned in Section 3.1, a VCR function is a two-phase process. Therefore, μ_v is calculated by summing up the mean time spent in each phase:

$$\mu_v = \mu_{VCR} + (1 - P^*)\mu_{hold}. \tag{2}$$

To simplify the calculation of the average time for releasing the hold on resources, μ_{hold}, we also assume that each position of the video has the same probability to be the point of resume. We obtain the probability of queuing (Prob[queueing]), the expected queue length (N_q) and the expected waiting time for service (T_q) as follows:

$$\text{Prob[queueing]} = \frac{\left(\frac{(m\rho)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)}{\left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \left(\frac{(m\rho)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)\right]}, \tag{3}$$

$$N_q = \frac{\rho}{1-\rho}\left(\frac{\left(\frac{(m\rho)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)}{\left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \left(\frac{(m\rho)^m}{m!}\right)\left(\frac{1}{1-\rho}\right)\right]}\right), \tag{4}$$

$$T_q = N_q/\lambda_v, \tag{5}$$

where $\rho = \lambda_v \mu_v / m$. In applying the above results of the $M/M/m$ queue in calculating the additional I/O resources needed, we have to ensure that the service performance of handling VCR requests reaches a certain satisfactory level. In other words, when the viewer issues a FF/RW request or when he/she resumes from any VCR request, it is not desirable if the viewer has to wait for a long time for the display of the VCR-version of the video or for normal playback to resume. Therefore, one of the following performance levels has to be satisfied. They are:

1. When the viewer issues a VCR request, the probability that a viewer has to wait for the VCR request to be serviced must not be greater than ϵ.
2. If the viewer has to wait, we ensure that the expected waiting time is within the tolerance level of the viewers, t^*.
3. The average number of viewers waiting for their VCR requests to be serviced is no more than l^*.

Given (3), (4), and (5), the above three performance requirements can be formulated using the following equations:

$$\text{Prob[queueing]} \le \epsilon, \tag{6}$$

$$T_q \le t^*, \tag{7}$$

$$N_q \le l^*. \tag{8}$$

These three inequalities will ensure that the system provides service at a predefined performance level. Note that the system utilization is ρ must be less than 1, therefore the smallest possible value of m is $\lfloor \lambda_v \mu_v \rfloor + 1$. To determine the number of servers needed to ensure a performance level, the value of m is incremented until one of the above three inequalities is satisfied.

4 Resource Allocation for Normal Playback

In this section, we present the derivation of our model for determining the configuration of the system for normal playback. Specifically, we want to determine the allocation of resources which, under various VCR operations (e.g., FF, RW, PAU), will insure the probability of a viewer resuming from a VCR request at any existing partitions to be greater than or equal to P^*, where P^* is a given system design parameter.

To maintain the I/O reduction benefit of the data sharing technique, we use the adaptive piggybacking with buffering techniques to join a fallen out viewer with an existing partition. Let us first describe the criteria for one stream to *catch up* with another (for simplicity of illustration, we will use streams rather than partitions). An example situation is illustrated in Fig. 3a. We denote R_{FF} as the rate of fast forwarding and R_{PB} as the rate of normal playback. Suppose viewers X and Y are viewing the movie at a rate of R_{FF} and R_{PB}, respectively, and X lags behind Y by Δ minutes (in movie time). Since R_{FF} is greater than R_{PB}, X will eventually *catch up* with Y at the catch-up point, denoted by a cross in the figure. Similarly, Fig. 3b illustrates a scenario where viewer Y rewinds to catch up with viewer X. Thus, the amount of movie time, t, through which viewer X must fast-forward (or viewer Y must rewind) before a catch-up is accomplished is as follows:

$$t = \begin{cases} \alpha \cdot \Delta, & \text{where } \alpha = \frac{R_{FF}}{R_{FF} - R_{PB}} \text{ for fast forward} \\ \gamma \cdot \Delta, & \text{where } \gamma = \frac{R_{RW}}{R_{PB} + R_{RW}} \text{ for rewind.} \end{cases} \tag{9}$$

In what follows, we derive the buffer and I/O resource requirements for normal playback under static partitioning. After formulating the problem, we present the derivation of our mathematical model for predicting the probability of deallocating an I/O resource upon a resume from a VCR operation (which will be used later in the paper for system sizing). This probability is derived by first considering the conditional probability of a hit for a viewer at a particular point in the movie, which is unconditioned to obtain the general probability of a hit for any type of a VCR request. The notations used are summarized in Table 1.

4.1 Mathematical Model

The static partitioning model adopts the idea of batching requests for a popular movie such that I/O streams are initiated periodically. If n is the number of I/O streams that are available, then for a movie with a length of l minutes, one way to guarantee that the viewers have the same maximum waiting time is to start the movie every l/n minutes. Assume that the total amount of buffers allocated for normal playback for a popular movie can store B' minutes of the movie. Then, the size of each partition is B'/n, and, in the worst case, a viewer arriving at the closing of the viewer enrollment window has to wait for the next restart of the movie. The length of the viewer enrollment window is equal to $B'/n - \delta$, where δ is the reserved amount of buffer space, which insures that when the first viewer in a partition replaces the frames in the buffer, the system will not overwrite the frames not yet viewed by the last viewer in the same partition [22]. Thus, the maximum waiting time, w, is equivalent to the gap between partitions, which is equal to $\frac{l-B'}{n} + \delta$. Let $B = B' - n\delta$, then we have:

$$w = \frac{l - B}{n}, \text{ where } n = 1, 2, \ldots, \frac{l}{w}. \tag{10}$$

Note that $n = \frac{l}{w}$ would imply that $B = 0$, which corresponds to the pure I/O or pure batching case, where each batch is served by a single I/O stream. However, in this case, the hit probability is always equal to zero (unless n is infinitely large). Rearranging the above equation gives n, the number of I/O streams needed is $\frac{l}{w} - \frac{B}{w}$, where $B \leq l$.

The difference between using buffering in conjunction with batching and pure batching lies in the amount of I/O and buffer resources required.[1] When we dedicate B minutes worth of buffer space for normal playback, then we can save $\frac{B}{w}$ I/O streams which can be used to service VCR requests or requests for other movies, where we consider using w minutes of buffer space as a tradeoff for one I/O stream.

1. We will consider the pure buffering in the later section of the paper.

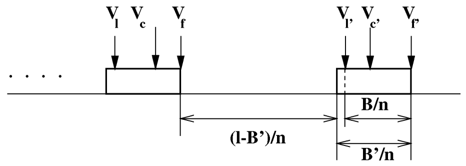


Fig. 4. Relative positions of viewers at V_c, V_f, and V_l in a partition.

Let V_c denote the position of our tagged viewer currently viewing the V_c^{th} frame[2] of the movie. Furthermore, the positions of the first and the last viewer that can exist in a partition are denoted by V_f and V_l, respectively. These two viewers may be *virtual*, in the sense that there not be any actual viewers viewing the V_f^{th} and V_l^{th} frame of the movie, yet they are the two extreme positions in the partition in which a viewer can possibly exist. The maximum difference (in time units) between these two extreme viewers is B/n, which is illustrated in Fig. 4. The idea behind this model formulation is to determine the probability of a hit when a viewer resumes from a VCR request. We define $P(hit|V_c, V_f)$ be the conditional probability of a hit given that a viewer is at position V_c and that the first possible viewer in the same partition is at position V_f. Among the three types of VCR requests (FF, RW, and PAU), let us first consider the FF operation.

In order to determine $P(hit|V_c, V_f)$ given that the VCR function is FF, we need to know the probability density function (pdf) of the duration of a FF request. The main difficulty in handling VCR requests lies in their inherently nondeterministic nature [13], where the retrieval pattern is not known. Rather than assuming a particular distribution, we allow a general distribution in our model and we let the pdf of the duration of FF requests be denoted by $f(x)$, where $x \in [0, l]$. Note that, our goal is not to obtain the exact distribution or model for VCR behavior, but rather we assume that the VCR behavior has a general distribution and construct a model which is able to handle a general probability distribution and, thus, not be limited by any particular distribution. Given this general probability distribution, $f(x)$, we derive the probability of a hit. We subdivide a hit into the following two mutually exclusive cases: 1) a hit within the same partition i.e, within the partition in which the VCR operation is initiated and 2) a hit in another partitions.

4.1.1 *Hits Occurring within the "Same" Partition (hit_w)*

The event hit_w occurs if a viewer can resume in the same partition in which the VCR request is issued. Our aim is to calculate the probability of a hit_w when a viewer resumes from any VCR request, denoted by $P(hit_w)$. We begin by considering the FF situation. The longest fast forward duration which can still result in a hit_w is the distance between the viewer, V_c, and the first possible viewer in his/her partition. This is illustrated in Fig. 5a in which the viewer must resume in the shaded region of the partition. The probability of a hit_w, given that: 1) the viewer is at

2. Note that we model the movie as a continuous segment. We use the term frame for ease of exposition.

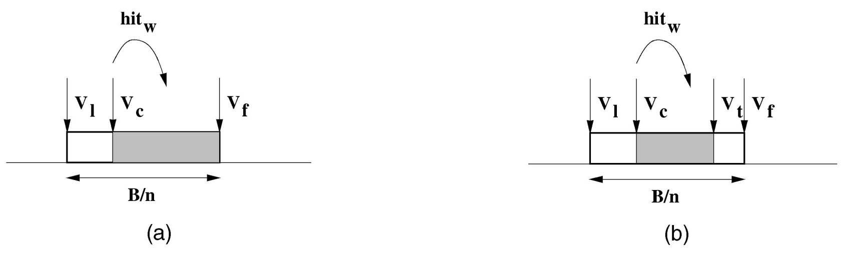


Fig. 5. Two cases of hit_w for viewer V_c: (a) catching up with all viwers ahead in the same partition and (b) catching up with viewers between V_c and V_t for $V_t < V_c + \frac{B}{n}$.

position V_c, 2) the first viewer is at position V_f, and 3) the requested VCR function is FF is as follows:

$$P(hit_w|FF, V_c, V_f) = \int_0^{\alpha\Delta_w} f(x)dx \text{ where } \Delta_w = V_f - V_c$$

and the upper limit is the time to enable a catch-up, as defined in (9). The remainder of this section is dedicated to determining how to uncondition on V_f and V_c so as to obtain $P(hit_w|FF)$, which is the expected probability of a hit within the same partition after a viewer resumes from a FF request.

1. **Unconditioning on** V_f. Recall that V_f is the position of the first possible viewer that can exist in the same partition as the viewer at position V_c. Thus, we can uncondition on V_f with respect to V_c. Since V_c can be any position within the partition, it can range from being the position of the first viewer in the partition (in which case $V_f = V_c$) to being the position of the last viewer in the partition (in which case $V_f = V_c + \frac{B}{n}$).

   Note that there are boundary cases where the viewer at position V_c may not be able to catch up with all the possible viewers ahead of him in the same partition. As shown in (9), the time t, elapsed before a catch-up is accomplished, is equal to the initial distance, Δ, times a constant greater than 1. If the catch up point is after the end of the movie, or $V_c + t > l$, then the viewer at position V_c cannot catch up with the target viewer *before the movie ends*. Thus, we divide the unconditioning of V_f into two cases.

   Case A. *Viewer V_c can accomplish a catch-up with all possible positions of the first viewer in the same partition, i.e., the largest value of V_f is equal to $V_c + \frac{B}{n}$.*

   In this case, even if V_c corresponds to the last possible viewer, he/she will be able to catch up with the first possible viewer, who is $\frac{B}{n}$ units of time ahead, before the movie ends. Thus, V_f ranges from V_c to $V_c + \frac{B}{n}$, and clearly, we have:

$$P_a(hit_w|FF, V_c) = \int_{V_f=V_c}^{V_c+\frac{B}{n}} P(hit_w|FF, V_c, V_f)P(V_f)dV_f, \tag{12}$$

   where $P(V_f)$ is the probability that the first viewer, who is in the same partition as V_c, is at the V_f^{th} minute of the movie. Here, we approximated $P(V_f)$ to be equal to $\frac{1}{B/n}$. Note that, this quantity is only an approximation since those viewers who arrive before the next restart of the movie all become "part of" the first viewer of the partition. Another reason why this is an approximation is that after viewers have resumed from VCR requests, the position of viewers may not be uniformly distributed within a partition. Nevertheless, we will show in a later section that results obtained using our mathematical model match well with those obtained through simulation.

   Case B. *Viewer V_c cannot catch up with the farthest possible position of the first viewer in the same partition, i.e., the largest value of V_f that the viewer can catch up to is equal to $\frac{l+(\alpha-1)V_c}{\alpha}$.*

   In this case, we need to handle the boundary condition which ensures that $V_c + t \leq l$—that is, a catch-up to some viewers within the same partition is still possible before the end of the movie. In other words, consider the scenario where, given a viewer at position V_c, there exists a viewer at position V_t, (where $V_t \geq V_c$) such that when the former viewer catches up to the latter, both viewers are at the lth minute of the movie. Using (9), we can describe a relationship between V_t and V_c by the equation $V_c + \alpha(V_t - V_c) \leq l$, which gives the upper bound for V_t:

$$V_t \leq \frac{l + (\alpha - 1)V_c}{\alpha}. \tag{13}$$

   Note that the viewer at position V_c will not be able to catch up with a viewer at position V_f, where $V_f > V_t$ for $V_f \in [V_c, V_c + \frac{B}{n}]$. The furthest position V_f with which the viewer at position V_c is able to catch up is V_t, as given by (13). Fig. 5b illustrates the region where a hit_w is ensured for $V_f \leq V_t$. Therefore, we have a second equation for unconditioning on V_f. The first term below corresponds to one case where $V_t \geq V_f$. The second term corresponds to the case where $V_t < V_f \leq V_c + \frac{B}{n}$, and the event hit_w is ensured for the duration 0 to $\alpha(V_t - V_c)$.

$$\begin{aligned} &P_b(hit_w|FF, V_c) = \\ &\int_{V_f=V_c}^{V_t} P(hit_w|FF, V_c, V_f)P(V_f)dV_f \\ &+ \int_{V_f=V_t}^{V_c+\frac{B}{n}} \int_0^{\alpha(V_t-V_c)} f(x)\ dx P(V_f)dV_f. \end{aligned} \tag{14}$$

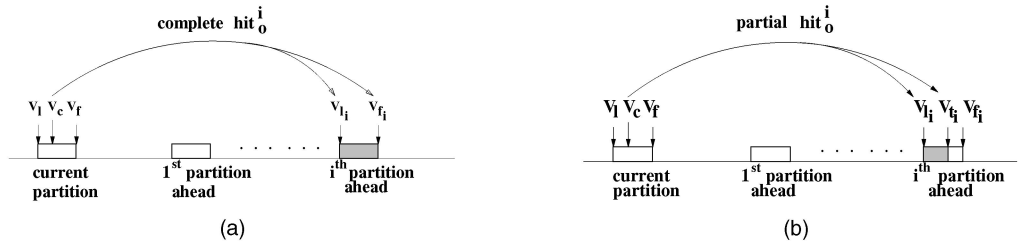


Fig. 6. A hit in another partitions (hit_o). (a) A complete hit^i_o and (b) a partial hit^i_o.

Again, we use the assumption made in Case A, that is, $P(V_f)$ is equal to $\frac{1}{B/n}$.

2. **Unconditioning on** V_c. As long as $V_c + \frac{B}{n}$ is smaller than V_t, a viewer at position V_c would be able to catch up with a viewer at all possible positions of V_f in the same partition. Otherwise, the farthest possible position of V_f will be at V_t (refer to Fig. 5b). Here, we also uncondition on V_c based on the two cases used for unconditioning on V_f.

Case A. $V_c + \frac{B}{n} \leq V_t$ (or $V_c < l - \frac{B\alpha}{n}$, using (13)), we have:

$$P_a(hit_w|FF) = \int_{V_c=0}^{l-\frac{B\alpha}{n}} P_a(hit_w|FF, V_c)\ P(V_c)\ dV_c. \tag{15}$$

Here, the only unknown is $P(V_c)$, which is the probability that a viewer is at the V_cth minute of the movie. Since a viewer can be at any position of the movie, we assume that $P(V_c) = \frac{1}{l}$. In other words, we assume that all positions of the movie have an equal probability of being viewed.

As V_c is always smaller than or equal to V_f, in the second case, we have the farthest V_f to which a viewer at position V_c can catch up equal to V_t. In this case, we uncondition as follows:

Case B ($V_c \leq V_f$ and $V_f = V_t$ (or $V_c \leq V_t$ which gives $V_c \leq l$)). Therefore, unconditioning on V_c, where $l - \frac{B\alpha}{n} < V_c \leq l$ gives the following equation for $P_b(hit_w|FF)$, where $P(V_c)$ is equal to $\frac{1}{l}$, as in Case A.

$$P_b(hit_w|FF) = \int_{V_c=l-\frac{B\alpha}{n}}^{l} P_b(hit_w|FF, V_c)\ P(V_c)\ dV_c. \tag{16}$$

Hence, given that the VCR request is a FF, the probability of a hit within the same partition, $P(hit_w|FF)$, is obtained by summing up the two expressions in (15) and (16).

4.1.2 *Jump to Other Partitions* (hit_o)

Beside the possibility of being able to resume in his own partition, a viewer can also resume in another partition such that the I/O resources allocated in Phase 1 for serving his/her VCR request can be released. We refer to this event as a hit_o. As illustrated in Fig. 6, if a viewer is to resume at the ith partition ahead of its current position, then he/she must fast forward long enough to at least catch up with the last possible viewer in the ith partition ahead, namely, at position V_{l_i}. Let us denote by V_{f_i} the position of the first possible viewer in the ith partition, then the event hit_o can be divided into two cases, namely, the *complete* hit^i_o and the *partial* hit^i_o. As illustrated in Fig. 6a, a complete hit^i_o is an event corresponding to a viewer being able to catch up not only to the viewer at position V_{l_i} but also to the first possible viewer, i.e., viewer at position V_{f_i}. Fig. 6b depicts the event of a partial hit^i_o, which illustrates a viewer who is not able to catch up with the viewer at position V_{f_i} before the movie ends. Since the movie is started every $\frac{l}{n}$ minutes, viewers at the same relative position in different partitions have a phase difference which is a multiple of $\frac{l}{n}$, e.g., viewers at position V_l and V_{l_i} have a phase difference of $i \cdot \frac{l}{n}$ minutes. Let us denote the shortest and longest duration a viewer must fast forward to have a hit^i_o by $\Delta^i_{jump_l}$ and $\Delta^i_{jump_f}$, respectively. To make our derivation consistent with that of $P(hit_w)$, we express these two quantities in terms of V_f and V_c as follows:

$$\Delta^i_{jump_l} = V_{l_i} - V_c = \frac{il}{n} + V_f - V_c - \frac{B}{n}, \tag{17}$$

$$\Delta^i_{jump_f} = V_{f_i} - V_c = \frac{il}{n} + V_f - V_c. \tag{18}$$

The probability of a hit_o at the ith partition ahead, denoted by $P(hit^i_o)$, is shown below.

Case 1. The ability to catch up with both viewers at position V_{l_i} and V_{f_i} (a complete hit^i_o). In this case, a viewer at position V_c is able to catch up with viewers at both, position V_{l_i} and V_{f_i} in the ith partition ahead:

$$P_c(hit^i_o|FF, V_c, V_f) = \int_{\alpha\Delta^i_{jump_l}}^{\alpha\Delta^i_{jump_f}} f(x)dx, \tag{19}$$

where the upper and lower limit of the integral are derived based on (9), (17), and (18).

Case 2. The ability to catch up with a viewer at position V_{l_i} but not all at the ith partition (a partial hit^i_o). In boundary cases, a viewer at position V_c may not be able to catch up with all viewers between position V_{l_i} and V_{f_i}. However, as long as the viewer is able to catch up with a viewer at position V_{l_i}, we will classify this event as a partial hit^i_o. As before, we define V_{t_i} to be the position of the last viewer in the ith partition with which a viewer at position V_c can catch up. Constraining $V_c + \alpha(V_{t_i} - V_c) \leq l$ and

$V_t = V_{t_i} - \frac{il}{n}$, results in $V_t = \frac{l+(\alpha-1)V_c - \frac{il\alpha}{n}}{\alpha}$. Then, the probability of a partial hit in this case is:

$$P_p(hit_o^i|FF, V_c, V_f) = \int_{\alpha\Delta^i_{jump_l}}^{\alpha(V_{t_i}-V_c)} f(x)dx = \int_{\alpha\Delta^i_{jump_l}}^{\alpha(\frac{il}{n}+V_t-V_c)} f(x)dx = \int_{\alpha\Delta^i_{jump_l}}^{l-V_c} f(x)dx. \quad (20)$$

Unconditioning on V_f. When unconditioning on V_f, we consider a complete hit_o^i and a partial hit_o^i separately. For a complete hit_o^i, we uncondition on V_f using cases a and b as in the derivation for event hit_w. For event hit_o^i, $V_t = \frac{l+(\alpha-1)V_c-\frac{il\alpha}{n}}{\alpha}$:

$$P_1(hit_o^i|FF, V_c) = \int_{V_f=V_c}^{V_c+\frac{B}{n}} P_c(hit_o^i|FF, V_c, V_f)P(V_f)dV_f, \quad (21)$$

$$P_2(hit_o^i|FF, V_c) = \int_{V_f=V_c}^{V_t} P_c(hit_o^i|FF, V_c,\ V_f)P(V_f)dV_f. \quad (22)$$

Observe that a viewer at position V_c can have a complete hit_o, as well as a partial hit_o, depending on the value of V_f. Given a viewer at position V_c, (22) gives the probability of a complete hit_o for $V_c \leq V_f \leq V_t$. But, $V_t < V_f < V_c + \frac{B}{n}$ can also result in a partial hit_o; this is shown in (23) below. Equation (24) corresponds to those viewers who can only have a partial hit_o for all values of V_f, where the largest V_f with respect to V_c is defined as $V_{t'} = \frac{l+(\alpha-1)V_c-\alpha(\frac{il-B}{n})}{\alpha}$.

$$P_3(hit_o^i|FF, V_c) = \int_{V_f=V_t}^{V_c+\frac{B}{n}} P_p(hit_o^i|FF, V_c, V_f)P(V_f)dV_f, \quad (23)$$

$$P_4(hit_o^i|FF, V_c) = \int_{V_f=V_c}^{V_{t'}} P_p(hit_o^i|FF, V_c, V_f)P(V_f)dV_f. \quad (24)$$

Unconditioning on V_c. The range of V_c is calculated using the same technique as in the hit_w case to yield the following four equations:

$$P_1(hit_o^i|FF) = \int_0^{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}} P_1(hit_o^i|FF, V_c)\ P(V_c)\ dV_c, \quad (25)$$

$$P_2(hit_o^i|FF) = \int_{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}}^{l-\frac{il\alpha}{n}} P_2(hit_o^i|FF, V_c)\ P(V_c)\ dV_c, \quad (26)$$

$$P_3(hit_o^i|FF) = \int_{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}}^{l-\frac{il\alpha}{n}} P_3(hit_o^i|FF, V_c)\ P(V_c)\ dV_c, \quad (27)$$

$$P_4(hit_o^i|FF) = \int_{l-\frac{il\alpha}{n}}^{l-\frac{(il-B)\alpha}{n}} P_4(hit_o^i|FF, V_c)\ P(V_c)\ dV_c. \quad (28)$$

Hence, the probability of resuming at the ith partition ahead for a FF request, $P(hit_o^i|FF)$, is calculated by summing (25), (26), (27), and (28):

$$P(hit_o^i|FF) = \sum_{a=1}^{4} P_a(hit_o^i|FF). \quad (29)$$

The number of partitions to which a viewer at position V_c would be able to jump ahead, resulting in hit_o, depends on which partition this viewer is before the initiation of the FF, e.g., it would be impossible for a viewer in the last partition of the movie to jump ahead to another partition. Also, the existence of V_t limits the number of partitions to which a viewer can jump ahead. To find the range of i, we refer back to (25), which indicates that the upper limit on V_c, namely, $l - \frac{B\alpha}{n} - \frac{il\alpha}{n}$, must be greater than or equal to 0. Therefore, we have to satisfy the condition $l - \frac{B\alpha}{n} - \frac{il\alpha}{n} \geq 0$. Rearranging the equation, we have:

$$i \leq \left\lfloor \frac{n(l+w\alpha)-l\alpha}{l\alpha} \right\rfloor. \quad (30)$$

4.1.3 Fast-Forwarding to the End of a Movie

Recall that our goal is to maximize the probability that the resources allocated for a VCR request will be released upon resume to normal playback. Consider the case where a viewer is at position V_c. Then, the longest FF duration is given by $\alpha(V_t - V_c)$. As the pdf of a FF duration is defined in the interval $[0, l]$, there is a nonzero probability that a viewer issuing a FF request will fast-forward to the end of the movie. In this case, the resources allocated to him/her in Phase 1 can also be released. We define the probability of fast forwarding beyond the end of a movie to be $P(end)$; it is given by the following equation:

$$\begin{aligned} P(end) &= \int_0^l \int_0^l \int_{l-V_c}^{l} f(x)\ dx\ \frac{1}{l}\ dV_c \\ &= \int_0^l \int_{l-V_c}^{l} f(x)\ dx \frac{1}{l} dV_c. \end{aligned} \quad (31)$$

Finally, $P(hit|FF)$, the probability of a hit given that the VCR request is a FF operation is:

$$P(hit|FF) = P(hit_w|FF) + \sum_{i=1}^{\lfloor\frac{n(l+w\alpha)-l\alpha}{l\alpha}\rfloor} P(hit_o^i|FF) + P(end). \quad (32)$$

The first term of the RHS corresponds to the probability of a hit within the same partition, i.e., the partition in which the FF request was initiated, and the second term corresponds to the total probability of hit in the ith partition ahead, where $i \geq 1$. We assume that the probability of a hit in partition i, where i is less than the current partition is defined to be zero. The last term corresponds to the probability of a fast-forward to the end of a movie. All three terms sum up to the probability of releasing the I/O resource upon resume from a FF request. For VCR requests like rewind and pause, we derive $P(hit|RW)$ and $P(hit|PAU)$ in a manner similar to the derivation of $P(hit|FF)$. Due to limitation of space, we do not repeat the derivations here. The detailed derivations can be found in [16].

4.1.4 The Expected Hit Probability $P(hit)$

Whenever a viewer issues a VCR request, there is some probability that the request is of FF, RW, or PAU type; we denote these probabilities by P_{FF}, P_{RW}, and P_{PAU}, respectively. Note that the values of these probabilities can be determined by measuring user behavior using statistical techniques. Given these probabilities, we can obtain the probability of a hit of a resume from a VCR request to normal operation, $P(hit)$, which can be expressed as follows:

$$P(hit) = P(hit|FF)P_{FF} + P(hit|RW)P_{RW} + P(hit|PAU)P_{PAU}. \tag{33}$$

The quantity, $P(hit)$, is a function of seven system parameters. Specifically,

$$P(hit) = \xi(l, B, n, w, R_{FF}, R_{PB}, R_{RW}).$$

Our goal is to determine the proper values of B (the total size of buffer space needed for normal playback) and n (the number of I/O streams needed for normal playback) such that $P(hit)$ is greater than or equal to P^*, which is a given design parameter. To determine the values of B and n, we use the following two constraints: **(C1)** $w = \frac{l-B}{n}$, **(C2)** $P^* \leq P(hit)$. The first constraint, $(C1)$, corresponds to the maximum waiting time that a viewer might experience before the movie restarts. The second constraint, $(C2)$, ensures that the average probability of a hit conforms to a specified probability P^*. Substituting the system parameters into the two constraints will yield two equations in two unknowns, namely, B and n. By solving these two equations numerically, we can determine the size of the system buffer, B, and the number of I/O streams, n, which need to be preallocated for playback of a popular movie in order to satisfy the performance (or quality of service) requirements of P^* and w. Although the above formulation only deals with a single movie, we will show how to apply our model to handle multiple movies in solving the system sizing problem in Section 6.

4.2 Model Verification

In this section, we verify our mathematical model using simulation. We first describe the system parameters used in this section. The arrivals of viewer requests for a popular movie are modeled as a Poisson process with a rate λ. As VCR requests are usually of relatively short duration as simple cases for illustration purposes, we use an exponential distribution and a skewed gamma distribution to represent the duration of the VCR functions.

In the following examples, the rates of fast forward and rewind, R_{FF} and R_{RW}, are three times that of normal playback, R_{PB}, and the movie length l is equal to 120 minutes. In the first three experiments, we simulate a system with two types of requests, normal playback and *one* of the VCR functions, namely, either fast forward with viewing (FF), or rewind with viewing (RW), or pause (PAU). In the last experiments, the requests can be of type normal playback or *any* of the three VCR functions, where the probability of a request for a VCR function r is equal to P_r, and r can be FF, RW, or PAU. In these experiments, we vary both the arrival rate of requests and the mix of FF, RW, and PAU. Also, in each experiment, we vary the maximum waiting time, w. The probability of a hit is plotted as a function of the number of partitions, n, where each curve corresponds to a specific value of the maximum waiting time, w. The results obtained from our mathematical model and from simulations are plotted in Figs. 7a, 7b, and 7c. These figures illustrate both the theoretical and the simulation results where the only type of a VCR request issued is either FF, RW, and PAU, respectively. Results for a mix of all three VCR requests plus requests for normal display are depicted in Fig. 7d. These figures illustrate that the theoretical results, for the most part, closely match the simulation results, which verifies the accuracy of our mathematical model.

5 Resource Allocation for VCR Mode

In this section, we present the computation of resources needed to handle VCR requests so as to satisfy the performance level defined by (6) and (7). Notice that ρ is defined by $\lambda_v \mu_v / m$, and it denotes the fraction of time any one of the reserved I/O resources is busy. Therefore, we aim to decrease the value of ρ. Since 1) the total VCR arrival rate to the system, λ_v, is beyond our control (unless we want to lower the quality of service) and 2) maximizing the number of I/O resources for VCR functionality, m, is not favorable because it implies a more expensive system; we are going to reduce ρ by minimizing μ_v. If we examine μ_v in (2), we see that μ_v is comprised of two components. The first component cannot be reduced because it denotes the mean time a viewer will spend in the fast forward, rewind, or pause operation whenever the viewer issues a VCR request, which depends solely on the users' behavior. For the second component, it denotes the expected time a viewer spends holding onto additional resources, allocated to him when he resumes to normal playback before he can rejoin an existing partitions, if normal playback is expected to resume immediately. Notice that this amount of time greatly depends on how the system handles the fallen out viewers. So, in devising methods for how to handle VCR requests, we aim to implement a method which minimizes the value of μ_{hold}. In the remaining section, we will present two schemes for handling VCR requests, both of which ensure that the resumption to normal playback from a VCR operation is immediate and show the derivations of the corresponding values of μ_{hold}.

Scheme 1 (No merging). If a viewer cannot resume at any of the existing partitions after a VCR request, i.e., resume with a miss, the system will not do anything to facilitate the release of resources. In other words, for these viewers to resume to normal playback without any delay, they will have to continue utilizing I/O resources (allocated during the VCR phase) after the resume, until they finish viewing the movie. Assume that each point of the movie has an equal probability of being the point of resume, we obtain the value of μ_{hold} by calculating the expected time to hold on to the VCR resource when the viewer resumes with a miss. Thus, we have $\mu_{hold} = l/2$ and this value also serves as an upper bound on μ_{hold}.

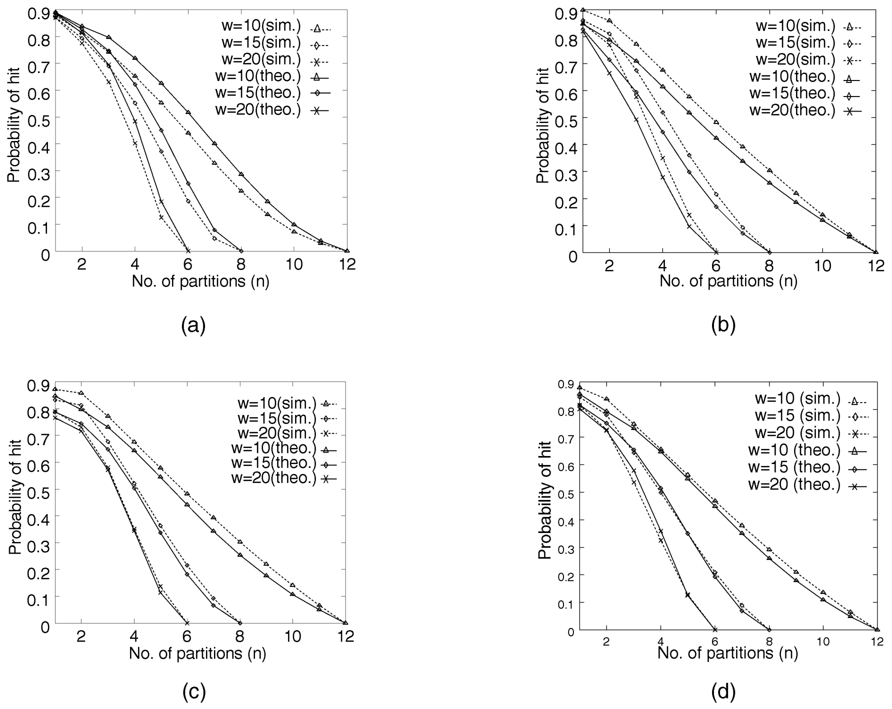


Fig. 7. Comparing simulation and theoretical results for normal playback and various VCR requests wherein the interarrival times are exponential and $1/\lambda = 2$ minutes; duration of VCR requests is drawn from a skewed gamma distribution with a mean = 8 minutes ($\alpha = 2, \gamma = 4$). (a) Only fast-forward VCR request, (b) only rewind VCR requests, (c) only pause VCR requests, and (d) all types of VCR requests with $P_{FF} = 0.2, P_{RW}$ and

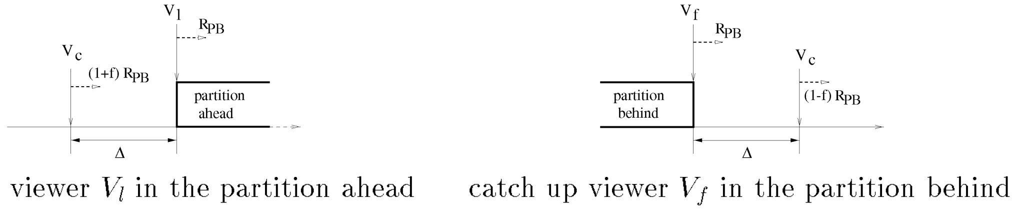


Fig. 8. Catch-up through adaptive piggybacking when the viewer resumes with a miss.

Scheme 2 (Merging through piggybacking and buffering). This scheme is to reduce the value of μ_{hold} by applying the techniques of adaptive piggybacking and buffering. The *adaptive piggybacking* approach [1], [10] alters display rates of requests *in progress*, for the purpose of *merging* their respective I/O streams into a single stream, which can serve the entire group of merged requests. After two viewers are "merged," one of them can release his/her resources. Thus, the idea is similar to that of *batching* except the grouping is done *dynamically* while the displays are *in progress* so no latency is experienced by the user.

Using adaptive piggybacking, we will alter the display rate of the fallen out viewer such that this viewer can eventually catch up with either the partition ahead or the partition behind. In particular, the viewer will either slow down to enable a catch-up by the partition behind or speed up to catch up with the partition ahead, depending on which partition he is closer to. In this case, the probability of releasing the allocated VCR resources before the viewer finishes the movie can be increased, thereby decreasing the value of μ_{hold}.

Fig. 8 illustrates two scenarios where the viewer resuming with a miss at position V_c of the movie and consumes the content of the movie at altered rates as compared to the viewers in the partitions, who are consuming movie content at the normal rate. Let us denote the normal rate of consumption by R_{PB}. Fig. 8a illustrates that the fallen out viewer will speed up the consumption at a rate of $(1 + f)R_{PB}$ in order to catch up with the last possible viewer, V_l, in the partition ahead. Fig. 8b illustrates that the viewer will be consuming the movie content at a slower rate of $(1 - f)R_{PB}$ in order to enable a catch-up by the first possible viewer, V_f, in the partition behind. In each case, the elapsed movie time, C_t, until a join of the fallen out viewer with either of the partitions can be accomplished through adaptive piggybacking is $\frac{1}{f}\Delta$, where Δ is the initial distance between V_c and the target viewer.

Since the calculation of μ_{hold} depends on different values of C_t for viewers resuming at different positions of the movie, it would be desirable to further reduce this quantity. Thus, in addition to adaptive piggybacking, we can also make use of the buffering technique to reduce the catch-up time. The motivation here is that a viewer resuming with a miss "near" to an end of an existing partition, can reduce his catch-up time by "bridging" the gap between him and the partition through the use of additional buffer space, i.e., to *merge* with that partition earlier. After the merge, the allocated I/O stream can be released since the partition and the additional buffer space will now be treated as one *extended partition*. This additional buffer space can be gradually released by continuing to apply adaptive piggybacking until the viewer can join in the original (i.e., rather than the extended) partition. Furthermore, viewers resuming with a miss "far" from the ends of the partitions can perform adaptive piggybacking until they are "close enough" to the partition to utilize additional buffer space in order to perform an early merge (as suggested above). In this case, the catch-up time, or C_t, can be further reduced and by using buffering, the allocated I/O streams can be released as soon as possible upon resume, thereby enhancing the availability of I/O streams for other VCR requests. The combined use of buffering and adaptive piggybacking techniques is accomplished in the following manner.

We define a threshold of k minutes (ahead of behind a partition) within which the use of buffering will enable a merge with the partition. Also, adaptive piggybacking will be used simultaneously to further facilitate the joining with the partition. Thus, we divide the handling of resume with a miss into two cases, namely, resume within the threshold k and resume beyond that threshold.

Case A: Resuming within the threshold ($\Delta \leq k$). When the viewer resumes with a miss at a position which is within the k minute threshold ahead or behind a partition, we will use Δ minutes of buffer space to join the viewer with the existing partition. Since that viewer is now consuming movie frames that have either not been fetched yet into the buffer (in the case of a resume within k minutes *ahead* of a partition) or have been already replaced (in the case of a resume within k minutes *behind* a partition), we will have to continue utilizing the I/O stream until the partition and the additional buffer space can be merged together as a single *extended partition*. At this point, the merge is said to be completed and the I/O stream allocated to the viewer can be released. Meanwhile, we will 1) continue performing adaptive piggybacking so the viewer can continue to catch up with the original partition and 2) release the additional buffer space gradually.

If the viewer resumes at a point R that is Δ minutes in front of a partition, he will continue to use the allocated I/O stream to fetch the movie frames, which will be retained in the additional buffer space. The maximum size of the additional buffer space is determined by Δ since the first viewer in the partition behind takes Δ minutes to reach point R. Similarly, the maximum size of additional buffer space and the time needed to enable a merge for a viewer resuming behind a partition is given by $\Delta/(1+f)$ minutes as the viewer is adopting a slightly faster rate with respect to the normal playback rate.

Given a viewer at position V_c of the movie, the average time to release the I/O stream for a resume ahead of the partition and a resume behind the partition is given by $\int_{\Delta=0}^{k} \Delta \frac{d\Delta}{\frac{l-B}{n}}$ and $\int_{\Delta=0}^{k} \frac{\Delta}{1+f} \frac{d\Delta}{\frac{l-B}{n}}$, respectively, where $\frac{l-B}{n}$ is the length of the gap between partitions. Therefore, to calculate the average time to release the I/O stream, we further uncondition by taking the probability of resuming at any position of the movie to be equal to $1/l$. Here, we separate the calculation into two cases. The first case corresponds to a viewer resuming at a position which is Δ units behind a partition, that is, this viewer will catch up with V_l in the partition ahead. Therefore, the average catch-up time is given by:

$$T_{1k} = \left(\frac{n}{l-B}\right)\left(\frac{1}{l}\right)\left(\frac{1}{1+f}\right) \left[\int_{V_c=0}^{l-\left(\frac{k}{1+f}+k\right)} \int_{\Delta=0}^{k} \Delta d\Delta dV_c + \int_{V_c=l-\left(\frac{k}{1+f}+k\right)}^{l} \int_{\Delta=0}^{\frac{l-V_c}{2}} \Delta d\Delta dV_c\right]. \quad (34)$$

The first term corresponds to the average time to catch up with the partition ahead when resuming at any position of the movie from the 0th position to the $\left(l - \left(\frac{k}{1+f} + k\right)\right)$th position. The second term is needed for handling boundary case of resuming at any position within the range of the $\left(l - \left(\frac{k}{1+f} + k\right)\right)$th position to the end of the movie. Therefore, the distance of the resuming viewer from the partition ahead only ranges from 0 to $\left(\frac{l-V_c}{2}\right)$ minutes.

The second case corresponds to a viewer resuming at a position of Δ units ahead of a partition. This viewer will slow down the consumption to enable a catch-up by V_f in the partition behind. The average time to release the I/O stream is calculated in a similar manner and is given by:

$$T_{2k} = \left(\frac{n}{l-B}\right)\left(\frac{1}{l}\right) \left[\int_{V_c=0}^{l-k} \int_{\Delta=0}^{k} \Delta d\Delta dV_c + \int_{V_c=l-k}^{l} \int_{\Delta=0}^{l-V_c} \Delta d\Delta dV_c\right]. \quad (35)$$

Case B: Resuming beyond the threshold ($\Delta > k$). When the viewer resumes at a position which is more than k minutes away from a partition, the viewer will perform adaptive piggybacking [1], [10] until the separation distance reduces to k minutes. At this point, we can handle the situation as in previous cases. For resuming at a position beyond the threshold, we consider the average time for releasing the I/O stream through the following three cases:

Case 1: Catch up with V_l in the partition ahead. The viewer resuming at position V_c of the movie will speed up the consumption of movie frames at a rate of $(1+f)R_{PB}$ to catch up with the last viewer in the partition ahead. The time elapsed before a merge can be accomplished is $(\Delta - k)/f$, which is the time needed to decrease the distance between the viewer and the partition to k minutes. At this point, $k/(1+f)$ more minutes are needed for the merge to complete. This scenario is illustrated in Fig. 9. Therefore, the total time, C_t, needed to release the I/O stream is given by $(\Delta - k)/f + k/(1+f)$. However, C_t must not be greater than the time for V_l to finish the movie; otherwise, it simply means that the viewer cannot release the allocated I/O resource before finishing the movie. In other words, C_t must not be greater than $(l - V_l)$. In this case, the following inequality must be satisfied:

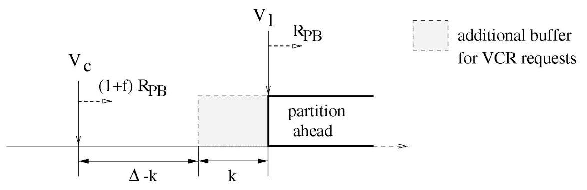


Fig. 9. Speed up to catch up V_l until the distance between the two is reduced to k.

$$\Delta \leq \frac{l - V_c - k\Big(1 - \frac{1}{f(1+f)}\Big)}{\frac{1}{f} + 1}. \tag{36}$$

Case 2: Catch up with V_f in the partition behind. The viewer resuming at position V_c of the movie will slow down the consumption of movie content at a rate of $(1-f)R_{PB}$ to facilitate the catch-up by the first viewer, V_f, in the partition behind, as illustrated in Fig. 10. Using the same calculations as in the above case, we can see that the time elapsed before a merge can be accomplished and must not be greater than the time for V_f to finish the movie, i.e.,

$$\frac{1}{f}(\Delta - k) + k \leq l - (V_c - \Delta).$$

However, if the time taken to merge with the partition behind is greater than that taken to finish the movie by V_c, we cannot benefit from performing adaptive piggybacking. Therefore, the following inequality has to be satisfied instead:

$$\Delta \leq \left[l - V_c - k\Big(1 - \frac{1}{f}\Big)\right] f. \tag{37}$$

Case 3: Cannot catch up with either of partitions. In this case, viewer V_c cannot perform a merge before he finishes the movie. Then, the time elapsed before the I/O resources can be released and is given by $l - V_c$.

The average time to release a resources, given that the viewer is at position V_c of the movie, is computed using (38). The first term below corresponds to the average time a viewer takes to catch up with the partition ahead while the second term below corresponds to the average time a viewer takes to enable a catch-up by the first possible viewer in the partition behind. In this equation, k_1 denotes the largest distance from V_l in the partition ahead where a catch-up to the partition ahead can be accomplished before the end of the movie and k_2 denotes the largest distance from V_f in the partition behind where a catch-up from behind is still possible. The last term of the equation corresponds to the average time spent to release an I/O resource when a viewer cannot perform adaptive piggybacking in order to enable a catch-up with either of the partitions:

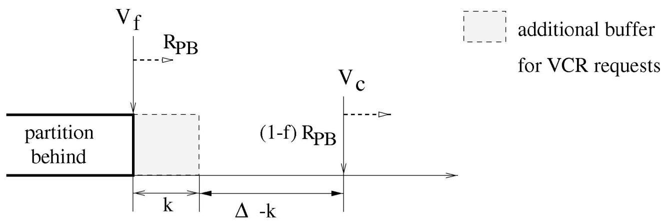


Fig. 10. Slow down to enable a catch-up by V_f until the distance between the two is reduced to k.

$$\begin{aligned}\int_{\Delta=k}^{k_1} \left[\frac{1}{f}(\Delta - k) + \frac{k}{1+f}\right]\left(\frac{d\Delta}{\frac{l-B}{n}}\right) \\ + \int_{\Delta=k}^{k_2} \left[\frac{1}{f}(\Delta - k) + k\right]\left(\frac{d\Delta}{\frac{l-B}{n}}\right) \\ + (l - V_c)\left(\frac{\frac{l-B}{n} - k_1 - k_2}{\frac{l-B}{n}}\right).\end{aligned} \tag{38}$$

To find the average time to release resources at any position of the movie, we uncondition on V_c in the three cases as stated above, which corresponds to the three terms given in (38). The range of V_c is determined by the range of Δ. When the viewer resumes at a distance of $\frac{l-B}{2n}$ from either of the partitions, i.e., half way between two partitions, it takes the same amount of time for him to join either the partition ahead or the one behind. In each of the three cases, we determine the range of V_c by setting the inequality: $\frac{l-B}{2n} \leq \Delta$ to equality for Cases 1 and 2 using (36) and (37), respectively.

For Case 1:

$$V_c \leq l - k\left(\frac{1}{f(1+f)}\right) - \left(\frac{1}{f} + 1\right)\left(\frac{l-B}{2n}\right) = \eta_1. \tag{39}$$

For Case 2:

$$V_c \leq l - k\left(1 - \frac{1}{f}\right) - \left(\frac{1}{f}\right)\left(\frac{l-B}{2n}\right) = \eta_2. \tag{40}$$

Let us consider the unconditioning of V_c in (38) term by term.

Case 1. For $V_c \leq \eta_1$, this corresponds to the viewer resuming at the maximum distance of $\frac{l-B}{2n}$ from the end of the partition where a merge with the partition ahead can still be accomplished before the end of the movie. However, for $V_c > \eta_1$, the maximum distance from the end of the partition is given by (36). Thus, we have the following:

$$\begin{aligned}T_1 =& \left(\frac{n}{l-B}\right)\left(\frac{1}{l}\right)\Bigg[\int_{V_c=0}^{\eta_1}\int_{\Delta=k}^{\frac{l-B}{2n}}\left[\frac{1}{f}(\Delta - k) + \frac{k}{1+f}\right] d\Delta dV_c \\ &+ \int_{\eta_1}^{l-\left(\frac{k}{1+f}+k\right)}\int_{\Delta=k}^{\frac{1-V_c-k\left(1-\frac{1}{f}\right)}{\frac{1}{f}+1}}\left[\frac{1}{f}(\Delta - k) + \frac{k}{1+f}\right] d\Delta dV_c\Bigg].\end{aligned} \tag{41}$$

Case 2. The equation given below is derived in the same manner as in Case 1 with reference to (37) and the range of V_c defined in (40):

TABLE 2
Parameters of the Two Popular Movies for Simulation

	l (mins)	w (mins)	P^*	B (mins)	n	μ_{VCR} (mins)
Movie 1	60	0.5	0.5	30	60	5
Movie 2	90	0.25	0.5	44.5	182	2

$$T_2 = \Big(\frac{n}{l-B}\Big)\Big(\frac{1}{l}\Big)\left[\int_{V_c=0}^{\eta_2}\int_{\Delta=k}^{\frac{l-B}{2n}}\left[\frac{1}{f}(\Delta-k)+k\right]d\Delta dV_c \right. \\ \left. + \int_{V_c=\eta_2}^{l-k}\int_{\Delta=k}^{\left[l-V_c-k\left(1-\frac{1}{f}\right)\right]f}\left[\frac{1}{f}(\Delta-k)+k\right]d\Delta dV_c\right]. \tag{42}$$

Case 3. The following equation is calculated with respect to (34), (35), (41), and (42). Making use of the upper bound of Δ in each equation and uncondition on the corresponding values of V_c, the average time to release the I/O stream when adaptive piggybacking cannot be performed to facilitate the joining of partitions is given by the following equation:

$$T_3 = \Big(\frac{n}{l-B}\Big)\Big(\frac{1}{l}\Big) \\ \left[\int_{V_c=\eta_1}^{\eta_2}(l-V_c)\left(\frac{l-B}{n} - \frac{l-B}{2n} - \frac{1-V_c-k\left(1-\frac{1}{f}\right)}{\frac{1}{f}+1}\right)dV_c \right. \\ + \int_{V_c=\eta_2}^{l-\left(\frac{k}{1+f}+k\right)}(l-V_c)\left(\frac{l-B}{n} - \frac{1-V_c-k\left(1-\frac{1}{f(1+f)}\right)}{\frac{1}{f}+1} \right. \\ \left. - \left[l-V_c-k\left(1-\frac{1}{f}\right)\right]f\right)dV_c \\ + \int_{V_c=l-\left(\frac{k}{1+f}+k\right)}^{l-k}(l-V_c)\left(\frac{l-B}{n} - \frac{l-V_c}{2} \right. \\ \left. - \left[l-V_c-k\left(1-\frac{1}{f}\right)\right]f\right)dV_c \\ \left. + \int_{V_c=l-k}^{l}(l-V_c)\left(\frac{l-B}{n}-(l-V_c)-\frac{l-V_c}{2}\right)dV_c\right]. \tag{43}$$

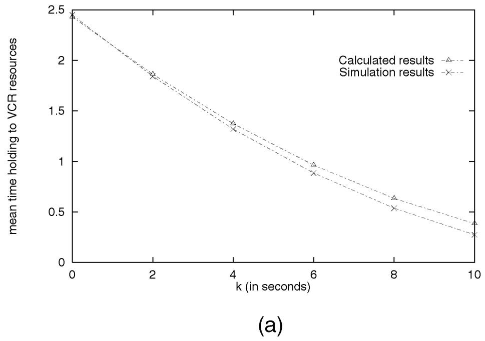
Thus, the average time to release the I/O stream for Case 3 is given by (43). The first two terms are derived with reference to (41) and (42); the last two terms are derived with reference to the second term in (34) and (35). Therefore, the average time to release the I/O stream, μ_{hold}, in this scheme, is given by the following equation:

$$\mu_{hold} = T_{1k} + T_{2k} + T_1 + T_2 + T_3. \tag{44}$$

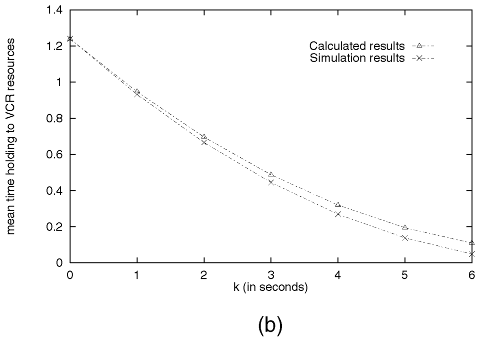
5.1 Verification

In the previous section, we have derived the equations for calculating the value of μ_{hold} for Scheme 2. These equations help to calculate the overall time the viewers will hold on to the I/O resources before they can join an existing partition. In this section, we will verify our derivation using simulation. We carry out our simulation using two popular movies, 1 and 2, with movie length l being 60 and 90 minutes, respectively. We also set the maximum time that the viewers can wait for the start of the movies, w to be 0.5 and 0.25 minute, respectively. The duration of VCR requests of movie 1 and 2 are drawn from an exponential distribution with a mean equal to five minutes and two minutes, respectively. Let the performance requirement of P^* of each movie be equal to 0.5. We apply our mathematical model to calculate the amount of buffer and I/O resources for normal playback; these are summarized in Table 2.

The arrival rate of viewers into system, λ, is $15/min$ and the simulation is run for 50,000 viewers. The average time for a fallen out viewer to hold on to the allocated resources, (μ_{hold}), is calculated for different values of the threshold k. The respective values of μ_{hold} obtained by simulation and through our derivation of both movies are plotted in Fig. 11. From Fig. 11, we can see that the calculated values of μ_{hold} match closely the simulation results, which verifies our derivation. Yet, the values of μ_{hold} obtained from simulation are smaller than the calculated counterparts, which can be explained as follows: In our derivation, we try to calculate the time taken for the viewers to join an existing partition. However, in the simulation, there also exist other fallen out

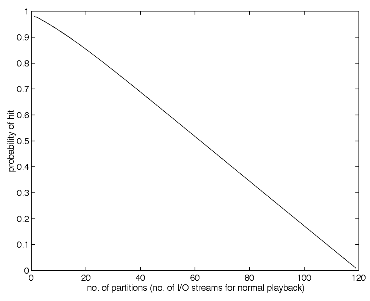


(a)

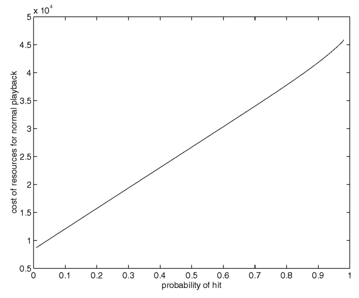


(b)

Fig. 11. Simulated and calculated results of μ_{hold} for different values of the threshold k. (a) Movie 1 and (b) movie 2.



(a)



(b)

Fig. 12. Calculating the resources for normal playback. (a) The relation between the probability of hit and number of I/O streams required for normal playback. (b) The relation between the cost of resources for normal playback and the probaility of a hit.

viewers trying to piggyback on an existing partition and they may have already formed an extended partition. These extended partitions reduce the time for other fallen out viewers to join an existing partition, which explains why our derivation slightly overestimates the values of μ_{hold} as compared with the simulated values.

6 Applications to System Sizing

Let us illustrate how we apply the mathematical model described in Sections 4 and 5 to making system sizing decisions. In determining the amount of resources required for a VOD system, we divide the system resources into two categories. The first category of resources is for handling normal playback and is calculated by applying the mathematical model described in Section 4, while the second category of resources is for servicing VCR requests and is calculated by the model described in Section 5. Our goal is to optimize the performance of a VOD system by reducing the demand of system resources while providing an acceptable level of quality of service to the viewers. The total system cost is determined by the amount of buffer space and I/O streams (both for servicing normal and VCR playbacks) needed to ensure a required level of performance. Thus, the overall cost of resources will be a function of the costs for 1) memory buffers per minute of a movie denoted by C_b and 2) an I/O stream denoted by C_n.

Although the two categories of resources serve different purposes, the resources allocated for normal playback will directly affect the amount of resources required for handling VCR requests while the amount of resources for servicing VCR requests will affect the QoS. Thus, in calculating the system resources required for normal playback, we also consider the behavior of viewers in issuing VCR requests. This is achieved by ensuring a minimum probability of a hit, P^*: Whenever a viewer resumes from a VCR request, we can ensure that the VCR resources will not be consumed for normal playback with a probability greater than or equal to $(1 - P^*)$. Therefore, if we can design our system such that the probability of a hit upon resume from a VCR request is high, then the amount of resources which need to be reserved for serving VCR functions can be reduced. If the probability of hit is very small, it is unlikely that the viewer can release the I/O stream upon resume until he finishes viewing the whole movie. However, to ensure a high probability of a hit requires more resources to be reserved for normal playback, which also implies a higher system cost. Therefore, it is very important to determine the optimal probability of a hit which will yield a system with the lowest overall cost, but one that still satisfies the required performance and quality of service levels. In the remaining section, we will show how to determine the cost of resources for normal playback as well as those for servicing VCR requests.

6.1 Cost of Resources for Normal Playback

For a movie of length l and a given maximum waiting time, w, we can calculate the amount of resources needed for normal playback as follows: Using our mathematical model, we calculate the corresponding hit probability for each possible number of I/O streams needed for normal playback, n (where $n = 1, 2, \ldots \frac{l}{w}$). The respective buffer size, B, for each n can be obtained by using (10). We can then determine the cost of resources for normal playback, C_{PB}, for each (B, n) pair, by solving $C_b\, B + nC_n$, which on rearranging gives $C_{PB} = C_n(\varphi B + n)$, where $\varphi = \frac{C_b}{C_n}$ and can be viewed as a price ratio of one minute of memory buffer to one I/O stream. For each possible (B, n) pair, there is a corresponding probability of a hit, P^*, and cost, C_{PB}. We express each possible configuration of resources for normal playback as a *PB-tuple*, which is in the form of (B, n, P^*, C_{PB}).

6.1.1 Experiment 1

A movie is of length, 60 minutes, and the maximum time a viewer is willing to wait before a movie starts is 30 seconds. Assume that the time a viewer spends in a VCR function follows an exponential distribution with the mean equal to 1 minute of movie time. By applying our mathematical model, the respective probability of a hit is plotted in Fig. 12a. We also assume that a 2G-SCSI disk is used in this experiment and that the cost of each disk is around \$700 and that its transfer rate is 5 MB/sec. The data rate of an MPEG-2 stream is assumed to be 4 Mbps, and the cost of 1 MB of main memory is \$24. We calculate C_b and C_n as follows: $C_b = \frac{60s*4\text{Mbps}}{8} * \$24 = \$720$ and $C_n = \frac{US\$700}{5\text{MBytes/sec}*8/4\text{Mbps}} = \$70$. Given the above equations, we can see that the amount of buffer space required to store one minute of an MPEG-2 movie is approximately 10 times as expensive as one I/O stream, i.e., $\varphi \approx 10$. The cost for normal playback is plotted in Fig. 12b. From Fig. 12a, we can see that decreasing the

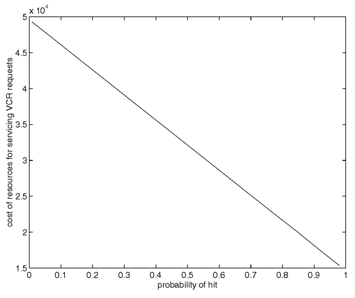


Fig. 13. Cost of resources for servicing VCR requests.

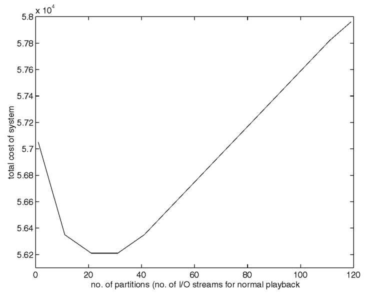


Fig. 14. Total cost of the system.

number of partitions will increase the probability of a hit. More buffer space is utilized to compensate for the saved I/O streams, which means that a larger portion of the movie will reside in the buffer space thereby increasing the probability of a hit. However, we can see from Fig. 12b that the cost of resources for normal playback also increases with the probability of a hit. So, it may seem that adopting a lower probability of hit will yield a lower system cost. We will show, in the next experiment, that this is not true because we are trying to reduce the overall system cost. In determining how much resources need to be allocated for normal playback, we have to consider the overall system cost to obtain the required P^*.

6.2 Cost of Resources for VCR Functions

As mentioned in Section 3.3, the provision of VCR functionality should ensure the satisfaction of a certain performance level, given by (6), (7), and (8). As in the normal playback case, we express each possible configuration of resources for servicing VCR requests as a *VCR-tuple*, which is in the form of $(B_v, m, \lambda_v, \mu_{VCR}, 1 - P^*, C_v)$, where C_v is the cost of resources needed for servicing VCR requests.

6.2.1 Experiment 2

Consider the system in Experiment 1. The system will make use of adaptive piggybacking only to facilitate the joining of fallen out viewers with existing partitions. In other words, we will adopt Scheme 2 but set the threshold value $k = 0$. We also assume that the average waiting time for servicing any VCR request is less than three seconds. This performance level is chosen because the waiting time is a measure that is noticeable to the viewers as compared to the queue length and probability of waiting. The arrival rate of VCR requests, λ_v is assumed to be 200/min. For a movie of 60 minutes, the expected number of viewers in the system is 900. Therefore, each viewer will, on average, issue a VCR request once every $4.5\,min$. We will show later in this section the effect of altering VCR arrival rates. Fig. 13 shows the relationship between the cost of resources for servicing VCR requests and the probability of a hit. From Fig. 13, we can see that the cost of resources decreases with the increase of probability of a hit, which is opposite to the effect observed in Fig. 12b. A lower probability of a hit will increase the average time a viewer spends in holding on to the additional resources, which implies that more resources are needed to be reserved for servicing VCR requests. Therefore, in making system sizing decisions, we have to consider both the cost of resources for normal playback as well as the cost of resources for servicing VCR requests (the overall system cost) before we can determine how much resources are to be allocated for normal playback and for servicing VCR requests.

6.3 Overall System Cost

The overall system cost is calculated by summing up the cost of 1) resources for normal playback and 2) resources for servicing VCR requests. Let us describe how to determine the optimal system configuration by considering the overall system cost. The overall system costs for a system using Scheme 1 and Scheme 2 are compared. In addition, we also show that our proposed mathematical model, together with Scheme 2 adopted for servicing VCR requests, will yield a lower cost as compared with systems using pure I/O and pure buffer space resources. Last, we describe how to choose the right value of a hit probability, P^* and of the threshold k.

Assume that we want to determine the amount of system resources which need to be allocated for a single movie, we have to calculate the overall system cost in the following manner:

Step 1. Generate all the PB-tuples by determining all the possible configurations for normal playback by applying the model presented in Section 4 and calculate the respective hit probabilities and cost.

Step 2. Obtain μ_{hold} for each of the configurations generated in Step 1 using (44). With reference to the hit probability in each PB-tuple obtained in Step 1, we can μ_v by (2). Then, we can determine a corresponding VCR-tuple by applying the $M/M/m$ queue model.

Step 3. For each pair of PB-tuple and VCR-tuple, obtain the overall system cost, C, by summing up C_{PB} in the PB-tuple and C_v in the VCR-tuple.

Step 4. The optimal choice of system resources allocation is the tuple pair with the lowest cost.

6.3.1 Experiment 3

Consider the system in Experiment 1 and 2. The total cost of the system is plotted against the number of I/O streams needed for normal playback and is shown in Fig. 14. In this figure, we can see that the overall cost of the system decreases initially for the smaller number of I/O streams and increases toward the larger number of I/O streams. The system yields the lowest cost when the number of I/O streams for normal playback lies within the range of $20 - 30$. With reference to Fig. 12a, the probability of a hit lies approximately within the range of $0.75 - 0.85$. This illustrates that in determining the proper amount of resources to be allocated for normal playback, we should also consider the cost of resources for VCR requests. The allocation of corresponding resources should yield a system with the lowest cost while satisfying a predefined performance level.

6.4 Comparison of Different Schemes

Let us now demonstrate the effectiveness of our approach by running several experiments to determine the overall system cost. First, we incorporate Scheme 1 and Scheme 2 into our normal playback model, where their total system cost is plotted against different arrival rates of VCR requests while other parameters remain the same as those described in the previous experiments. Second, we examine two other systems, namely, a system using only I/O streams for normal playback (referred to as *pure I/O* hereafter) and a system using only buffer space for normal playback (referred to as *pure buffer* hereafter). These are the two extreme cases where only I/O streams or only memory buffers are used for normal playback. We aim to show that a proper balance between I/O and memory resource will yield a system with a lower cost.

6.4.1 *Experiment 4*

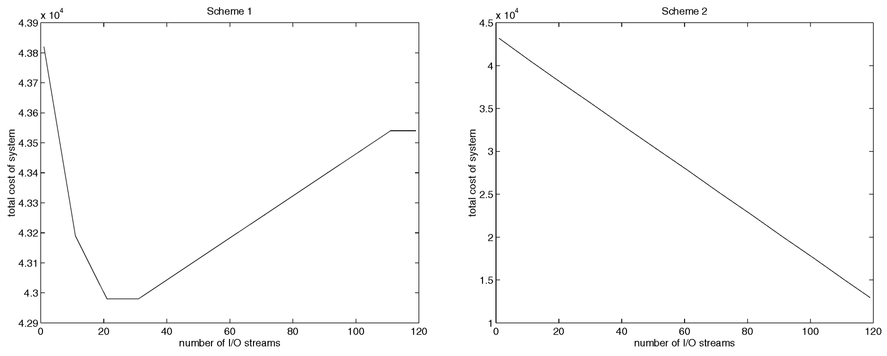
In this experiment, we determine the overall system cost of each scheme for different arrival rates of VCR requests, ranging from very low arrival rates to very high arrival rates. We assume that the arrival rate of a normal playback request is $15/\mathrm{min}$, so that the expected number of viewers in the system is given by $15 \times 60 = 900$. We plot the overall system cost of each scheme against the number of I/O streams needed for normal playback with the average VCR arrival rates λ_v of $15/\mathrm{min}$, $100/\mathrm{min}$, $200/\mathrm{min}$, and $400/\mathrm{min}$ in Figs. 15a, 15b, 15c, and 15d, respectively. From these figures, we can see that different arrival rates will yield a system with the lowest cost occurring at different positions on the cost curve. For Scheme 1, except for very low VCR arrival rates, the minimum cost is always located at the smallest value of I/O streams, that is, with the largest amount of buffer space for normal playback. This can be explained as follows: Recall that μ_{hold} for Scheme 1 is $\frac{l}{2}$, which is 30 minutes in this case. That is to say, if the mean time for a fallen out viewer to hold on to the additionally allocated resource is large, more resources need to be reserved for servicing VCR requests and handling the fallen out viewers. This amount of resources becomes so large that it is more beneficial to allocate more buffer space for normal playback, such that the probability of a hit can be increased. By increasing the probability of a hit, the mean time a viewer spends in utilizing the additional resources can be reduced, and we can make sure that the reserved resources are used for servicing VCR requests rather than servicing the normal playback of the fallen out viewers. The same rational applies to Scheme 2 when the VCR arrival rate is very high. On the contrary, for low VCR arrival rates in Scheme 2, the lowest cost is located at the largest value of I/O streams, that is, when the least amount of buffer space is used for normal playback. A low arrival rate of VCR requests means that not many viewers will be issuing VCR requests. Even if the viewers fall out of a partition upon resume, Scheme 2 can facilitate the joining of these viewers with existing partitions and, in turn, the release of allocated resources in a very short time. Therefore, only a small amount of resources is needed for servicing VCR requests and handling the fallen out viewers. In this case, using a large amount of buffer space for normal playback to ensure a larger probability of a hit is not beneficial.

When we compare Scheme 1 and Scheme 2, we can see that Scheme 2 always outperforms Scheme 1 in the sense that the minimum cost for a system using Scheme 2 is more smaller than that of a system using Scheme 1. To further demonstrate the effectiveness of Scheme 2, we plot the percentage of total system cost saved by using Scheme 2 as compared to using Scheme 1. The percentage saved through Scheme 2 for different arrival rates, as shown in Fig. 16 is calculated by comparing the minimum overall cost in each of the two schemes, with reference to Scheme 1. The percentage saved by Scheme 2 as compared to Scheme 1 ranges approximately from 13 percent to as high as 70 percent.

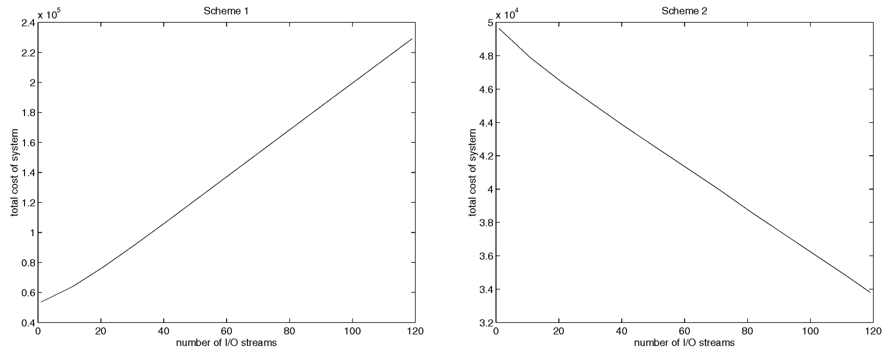
6.4.2 *Experiment 5*

In this experiment, we compare the overall system cost of our proposed system with that of pure I/O and pure buffer. We assume that the arrival rate of normal playback requests is $15/min$, thus, a total of 900 I/O streams are needed for normal playback in the pure I/O case. The rate of fast-forward or rewind is assumed to be three times faster than that of normal playback. The percentage of system cost saved by Scheme 2 for the case where the total cost is dependent on the VCR requests arrival rate is plotted against the different VCR requests arrival rates and is shown in Fig. 17a, while the independent case is plotted in Fig. 17b. The percentage saved is determined by the difference between the minimum cost of Scheme 2 and that of the pure I/O and the pure buffer schemes.

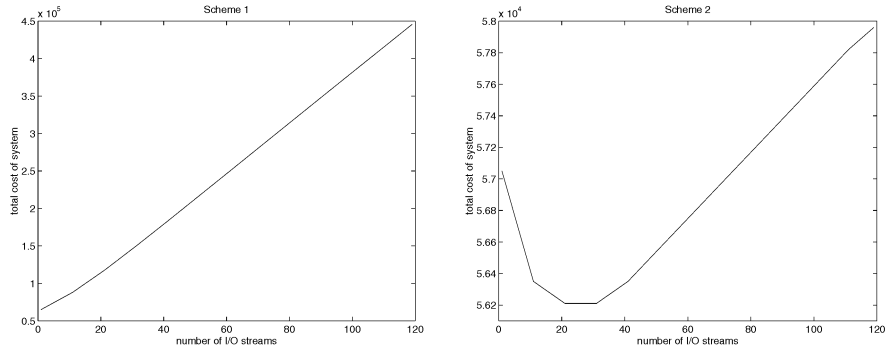
As shown in Fig. 17a, the cost curve decreases rapidly as the VCR arrival rate increases and nearly levels off for very high VCR arrival rates. Scheme 2 outperforms the pure I/O method by at least approximately 21 percent and can outperform it by as high as 83 percent. Scheme 2 can also outperform the pure buffer scheme by up to 75 percent for low to moderate VCR arrival rates, although beyond a VCR arrival rate of around $220/min$, pure buffer has a lower cost than Scheme 2, which is demonstrated by a negative percentage in the graph. As for the independent case illustrated in Fig. 17b, the cost curves also decrease rapidly at first from around 80 percent until the VCR arrival rate reaches around $200/min$, after which the decrease is slowed down. In contrast with the dependent case, both the pure I/O and the pure buffer method perform better than Scheme 2 when the VCR arrival rate increases beyond a certain point. The percentage saved becomes negative when the VCR arrival rate increases to around $200/min$ and $290/min$ for pure buffer and pure I/O, respectively. Furthermore, instead of leveling off as in the dependent case, the negative percentage continues to drop for very high VCR arrival rates because when the VCR arrival rate increases, more and more resources are needed in Scheme 2, which means the cost also



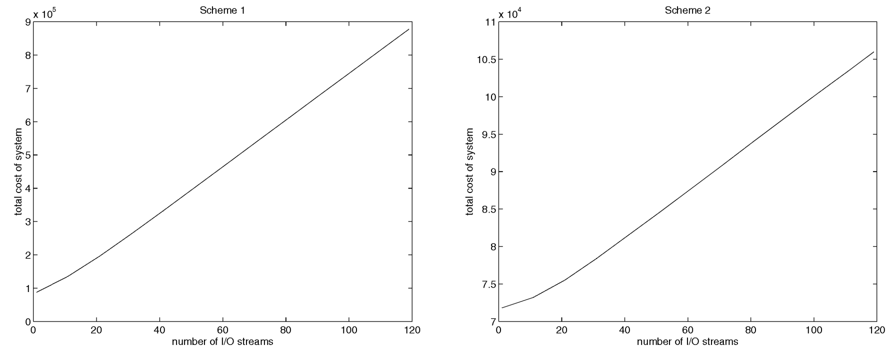
(a)



(b)



(c)



(d)

Fig. 15. Total cost of system for Scheme 1 and Scheme 2 for different VCR arrival rates. (a) Average arrival rate of VCR requests $= 15/min$. (b) Average arrival rate of VCR requests $= 100/min$. (c) Average arrival rate of VCR requests $= 200/min$. (d) Average arrival rate of VCR requests $= 400/min$.

increases. On the other hand, since the cost of pure I/O and pure buffer is independent of the VCR arrival rate, this means that their cost is fixed, thereby resulting in the continuous decrease in the percentage saved. We also plot the same graphs for two more system settings, where the movie is of length $90\ min$ and $120\ min$; these are shown in Figs. 18 and 19, respectively. From these figures, we can see

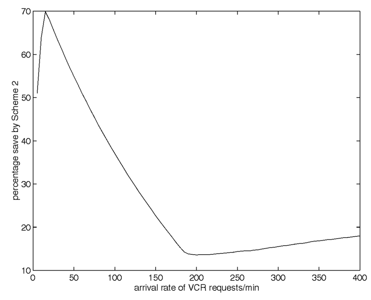


Fig. 16. Percentage of system cost saved by Scheme 2 as compared to Scheme 1.

that the percentage saved by Scheme 2 becomes more significant as the movie length increases.

The percentage saved by Scheme 2 is more significant at lower VCR arrival rates because for systems using pure I/O or pure buffer method, the cost is dominated by the resources allocated for normal playback. However, as the VCR arrival rate increases, the requirement of VCR I/O resources becomes larger and larger, until it becomes the dominant factor in the case of Scheme 2. In spite of that, we can claim that Scheme 2 outperforms the pure I/O and pure buffer methods in a VOD system. This can be explained as follows: Consider the movie of length $60\,min$ and the range of VCR arrival rates where the percentage saved by Scheme 2 is positive, i.e., at most $200/min$, where the expected number of viewers in the system is 900 in this example. That means that approximately $\frac{1}{4}$ of the viewers are doing VCR functions per minute or the viewers are issuing VCR requests once every 4.5 minutes on the average, and this is a very high VCR rate for viewing movies. In a VOD system, relatively less time will be spent in doing VCR functions as compared with normal playback. For a movie of length 60 minutes, we expect

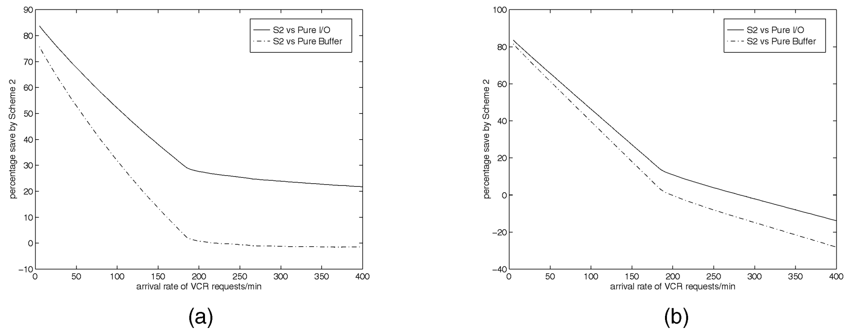


Fig. 17. Percentage of system cost saved by Scheme 2 over pure I/O and pure buffer. (a) VCR arrival rate dependant and (b) VCR arrival rate independent.

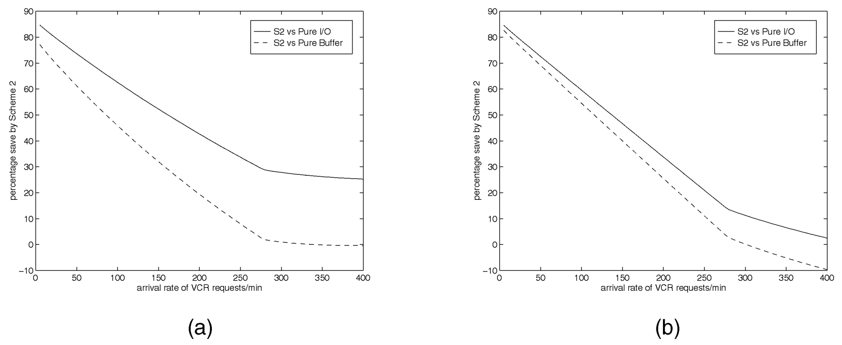


Fig. 18. Percentage of system cost saved by Scheme 2 as compared to pure I/O and pure buffer for a movie of length of $90\,min$. (a) VCR arrival rate dependent and (b) VCR arrival rate independent.

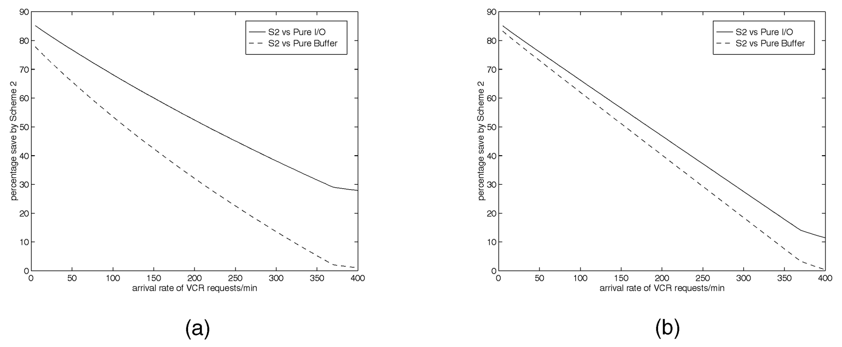


Fig. 19. Percentage of system cost saved by Scheme 2 over pure I/O and pure buffer for a movie of length of $120\,min$. (a) VCR arrival rate dependent and (b) VCR arrival rate independent.

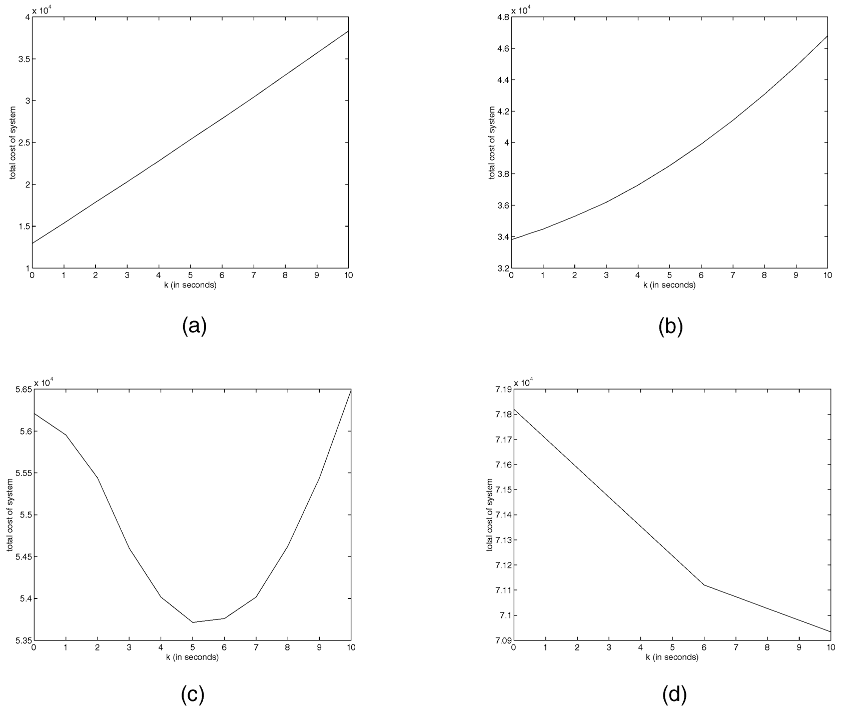


Fig. 20. Total system cost for different values of k. (a) $\text{VCR arrival rate} = 15/min$. (b) $\text{VCR arrival rate} = 100/min$. (c) $\text{VCR arrival rate} = 200/min$. (d) $\text{VCR arrival rate} = 400/min$.

the range of VCR arrival rates to be far less that $200/min$. In this region, we can have a substantial reduction in the cost of resources by using Scheme 2. In addition, the percentage saved as compared to the pure I/O method is greater than the percentage saved as compared to the pure buffer method, which means that the cost of using pure buffer is lower than that of using pure I/O. This is mainly due to the fact that the cost of the pure I/O method will depend on the arrival rate of normal playback requests, whereas that of pure buffer is fixed for any arrival rate. As both the pure I/O and the pure buffer method require the same amount of VCR resources (since they have the same μ_v in the dependent case), the main difference lies only in the cost of normal playback resources. Therefore, the pure I/O method will have a lower cost than the pure buffer method for smaller values of arrival rates.

6.4.3 Experiment 6 (Different Values of Threshold Buffer k)

As mentioned before, the use of additional buffers can further facilitate the joining of the fallen out viewer with existing partitions. In this experiment, we plot the overall system cost against different values of k for different arrival rates, as depicted in Fig. 20. For different values of k, we determine the corresponding values of μ_{hold} and calculate the lowest cost of supporting such a system. We plot the total cost of the system with different values of k by varying the VCR arrival rate. In this figure, we can see that the minimum point of the cost curve occurs at different values of k for different values of VCR arrival rates. For smaller VCR arrival rates, the minimum cost occurs at the smallest k. When VCR arrival rate increases, the minimum cost shifts to larger values of k. For large VCR arrival rates, the minimum cost occurs at the largest k. This means that the use of more buffer space is beneficial when the VCR arrival rate is high. The use of buffer space has reduced the time for the viewers to join an existing partition, which implies that fewer I/O streams are needed for servicing VCR requests. For a system with very high VCR arrival rates, we can anticipate that a large amount of I/O streams are needed for handling VCR requests. Although the use of additional buffer space will impose more cost on the VCR resources, this cost can be compensated by the reduction in I/O streams. This reduction becomes very significant when the VCR arrival rate is high. On the other hand, if the VCR arrival rate is small, only a small amount of additional I/O streams are needed. Thus, the use of additional buffer space cannot be compensated for by the cost of the I/O streams saved. Therefore, for systems with small VCR arrival rates, using only I/O streams for servicing VCR requests can yield a lower cost.

6.4.4 Experiment 7: Different Values of Memory-I/O Cost Ratio, φ.

In the previous experiments, we have assumed that the cost of memory buffers per minute of a movie is approximately 10 times that of one I/O stream needed to retrieve that movie from the disk subsystem. However, as technological changes occur, the cost of memory may decrease faster (or slower) than the cost of I/O bandwidth; thus, the minimum cost might occur at other positions on the cost curve, as illustrated in Fig. 21. In this experiment, we vary φ as follows: $3, 8, 10$, and 11. Figs. 21a and 21b illustrate the situation where the cost of the memory decreases faster than the cost of I/O bandwidth while Fig. 21d illustrates the

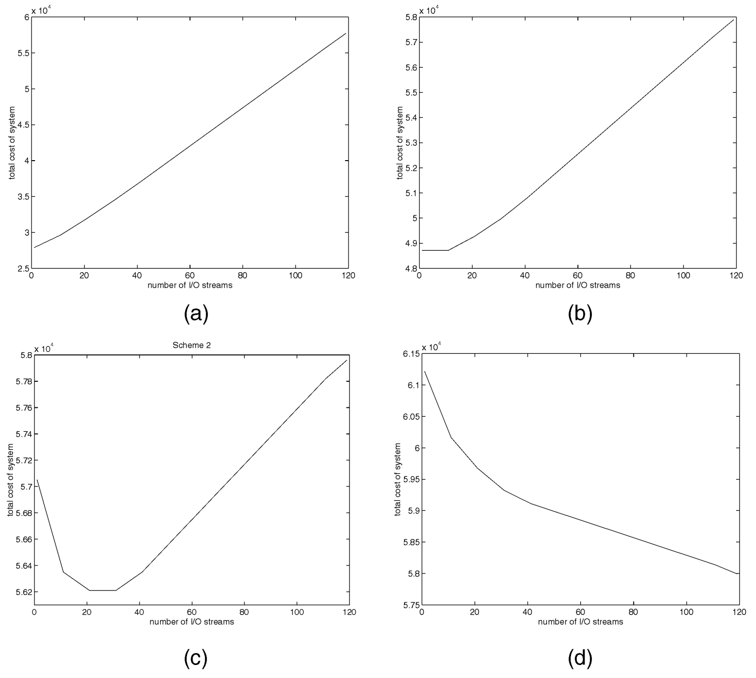


Fig. 21. Overall system cost versus the number of I/O streams for different values of φ. (a) $\varphi = 3$, (b) $\varphi = 8$, (c) $\varphi = 10$, and (d) $\varphi = 11$.

situation where the cost of I/O bandwidth decreases faster than the cost of memory. As we can see from the graph, the minimum cost tends to shift to the right as φ increases, i.e., more I/O streams should be used instead of memory buffers for large values of φ, but more memory buffers should be used for smaller values of φ. Smaller values of φ indicate that the use of more buffer space can increase the probability of a hit and, consequently, causes a reduction in the amount of VCR resources to be reserved. Recall that, in normal playback, we are using w minutes of buffer space as a trade-off for one I/O stream, therefore, the decrease of φ simply means that we can trade-off one I/O stream with an even lower cost. Therefore, in choosing an optimal system configuration, our model aids system designers in making proper system sizing decisions and, at the same time, allows them to meet the performance requirements.

7 Conclusions

Determining the proper amount of resources to be allocated is crucial for optimizing the performance of VOD systems so as to maintain the benefits of data sharing techniques. In this paper, we choose specific combinations of data sharing techniques (e.g, batching with static partitioning) which result in a reasonably representative of data sharing scheme. We then present our mathematical model so as to illustrate how we can perform system sizing. The mathematical model we present can help designers to perform efficient resource management and system sizing for a large scale VOD system. Although the amount of resources needed will also depend on the arrival rate of normal playback requests and the arrival rate of VCR requests, our mathematical model is general enough so that it can be applied in solving the system sizing problem. The amount of resources calculated ensures that the service to be provided satisfies a given performance level, such as ensuring that the maximum waiting time for starting or resuming the movie is within the viewers' tolerance level. Furthermore, we use our model in determining the *optimal distribution* of resources for normal playback and servicing of VCR requests, so as to reduce the overall system cost. We show, through experiments, that different arrival rates of VCR requests will affect the overall costs and the distribution of resources for servicing normal playback requests and VCR requests. Using the mathematical model we proposed, system designers can make proper system sizing and resource distribution decisions. Our future research work includes extending the proposed mathematical model to encompass other data sharing and VCR functionality schemes. For instance, one promising direction seems to be the possibility of including, *in the model*, schemes which utilize client-side resources to improve overall system performance, e.g., such as buffering of video data at the client's set-top box for provision of VCR functions (as proposed in [2]).

Acknowledgments

The research work of John C.S. Lui was supported in part by the RGC Earmarked Grant and the CUHK Direct Grant. The research work of Leana Golubchik was supported in part by the US National Science Foundation Grant CCR-98-96232 and the ANI-00-70016 grant.

References

[1] C. Aggarwal, J. Wolf, and P. Yu, "Optimal Piggyback Policies for Video-on-Demand Systems," *Proc. ACM SIGMETRICS Conf.*, pp. 200-209, May 1996.

[2] K.C. Almeroth and M.H. Ammar, "The Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service," *IEEE J. Selected Areas in Comm.*, vol. 14, no. 6, pp. 1110-1122, Aug. 1996.

[3] M-S. Chen, D.D. Kandlur, and P.S. Yu, "Storage and Retrieval Methods to Support Fully Interactive Playout in a Disk-Array-Based Video Server," *Multimedia Systems,* pp. 126-135, 1995.
[4] A. Dan, D.M. Dias, R. Mukherjee, D. Sitaram, and R. Tewari, "Buffering and Caching in Large-Scale Video Servers," *Proc. IEEE CompCon,* 1995.
[5] A. Dan, D. Sitaram, "Buffer Management Policy for an On-Demand Video Server," IBM Research Report RC 19347, Jan. 1994.
[6] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," *Proc. ACM Multimedia Conf.,* pp. 391-398, 1994.
[7] E. de Souza e Silva, H.R. Gail, L. Golubchik, and J.C.S. Lui, "Analytical Models for Mixed Workload Multimedia Storage Servers," *J. Performance Evaluation,* vols. 36-37, nos. 1-4, pp. 185-211, 1999.
[8] D. Deloddere, W. Verbiest, and H. Verhille, "Interactive Video on Demand," *IEEE Comm. Magazine,* vol. 32, no. 5, pp. 82-88, 1994.
[9] S. Ghandeharizadeh and R.R. Muntz, "Design and Implementation of Scalable Continuous Media Servers," special issue of *Parallel Computing J. Parallel Data Servers and Applications,* Jan. 1998.
[10] L. Golubchik, J.C.S. Lui, and R. Muntz, "Adaptive Piggyback: A Novel Technique for Data sharing in Video-On-Demand Storage Servers," *ACM J. Multimedia Systems,* vol. 4, no. 3, pp. 140-155, June 1996.
[11] L. Golubchik, J.C.S. Lui, and M. Papadopouli, "Designing Efficient Fault Tolerant VOD Storage Server: Techniques, Analysis and, Comparisons," *J. Parallel Computing,* vol. 24, no. 1, pp. 123-155, 1998.
[12] M. Kamath, K. Ramamritham, and D. Towsley, "Buffer Management for Continuous Media Sharing in Multimedia Database Systems," technical report, Univ. of Massachusetts, Feb. 1994.
[13] K.D. Jayanta, J.D. Salehi, J.F. Kurose, and D. Towsley, "Providing VCR Capabilities in Large-Scale Video Servers," *Proc. ACM Int'l Conf. Multimedia,* pp. 25-32, Oct. 1994.
[14] S.W. Lau and J.C.S. Lui, "Scheduling and Data Layout Policies for a Near-Line Multimedia Storage Architecture," *The ACM J. Multimedia Systems,* vol. 5, no. 5, pp. 310-323, Sept. 1997.
[15] K.W. Law, J.C.S. Lui, and L. Golubchik, "Efficient Support for Interactive Services in Multi-Resolution VOD Systems," *The Very Large Data Base J.,* vol. 8, no. 2, pp. 133-153, 1999.
[16] M.Y.Y. Leung, "System Sizing and Resource Allocation for Video-on-Demand Systems," master's thesis, The Chinese University of Hong Kong, June 1997.
[17] M.Y.Y. Leung, J.C.S. Lui, and L. Golubchik, "Buffer and I/O Resource Pre-Allocation for Implementing Batching and Buffering Techniques for Video-on-Demand Systems," *Proc. Int'l Conf. Data Eng.,* Apr. 1997.
[18] W. Liao and V.O.K. Li, "The Split and Merge Protocol for Interactive Video-on-Demand," *IEEE Multimedia,* vol. 4, no. 4, pp. 51-62, Oct. 1997.
[19] P.W.K. Lie, J.C.S. Lui, and L. Golubchik, "Threshold-Based Dynamic Replication in Large-Scale Video-on-Demand Systems," *J. Multimedia Tools and Applications,* vol. 11, no. 1, May 2000.
[20] D.J. Makaroff and R.T. Ng, "Buffer Sharing Schemes for Continuous-Media Systems," Technical Report TR-95-01, Dept. Computer Science, Univ. of British Columbia, 1995.
[21] R.T. Ng and J. Yang, "Maximizing Buffer and Disk Utilizations for News on-Demand," *Proc. 20th Very Large Data Base Conf.,* pp. 451-462, 1994.
[22] D. Rotem and J.L. Zhao, "Buffer Management for Video Database Systems," *Proc. Int'l Conf. Data Eng.,* pp. 439-448, Mar. 1995.
[23] W. Shi and S. Ghandeharizadeh, "Buffer Sharing in Video-on-Demand Servers," *ACM Sigmetrics Bull.,* 1997.
[24] K.L. Wu and P.S. Yu, "Consumption-Based Buffer Management for Maximizing System Through Puts of News-on-Demand Multimedia Systems," IBM Research Report RC 20143, Jan. 1995.
[25] P.S. Yu, J.L. Wolf, and H. Shachnai, "Design and Analysis of A Look-Ahead Scheduling Scheme to Support Pause-Resume for Video-on-Demand Applications," *ACM J. Multimedia Systems,* vol. 3, no. 4, pp. 137-150, 1995.

Mary Y.Y. Leung received the bachelor's degree (with honors) in computer science and the MPhil degree in computer science from the Chinese University of Hong Kong in 1995 and 1997, respectively. Since then, she has been working with Andersen Consulting on various projects, including the digital television and video-on-demand in London from March 1998 to July 2000. Currently, she is working on the interactive television project in Sydney.

John C.S. Lui received PhD degree in computer science from the University of California at Los Angeles. During the summer of 1990, he participated in a parallel database project in the IBM Thomas J. Watson Research Center. After his graduation, he joined a team at the IBM Almaden Research Laboratory/San Jose Laboratory and participated in the research and development of a parallel I/O architecture and file system project. He later joined the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His current research interests are in communication networks, distributed multimedia systems, OS design issues, parallel I/O and storage architectures, and performance evaluation theory. His personal interests include general reading and movies. He is a member of the IEEE.

Leana Golubchik received the BS, the MS, and the PhD degrees from the Computer Science Department at the University of California at Los Angeles in 1989, 1992, and 1995, respectively. She is currently an assistant professor in the Department of Computer Science at the University of Maryland at College Park. From the Fall of 1995 until the Summer of 1997, she was an assistant professor in the Department of Computer Science at Columbia University. Her research interests include computer systems modeling and performance evaluation, Internet-based computing, and multimedia storage systems. She is a guest coeditor for special issues of the *Parallel Computing Journal* and the *International Journal of Intelligent Systems*, a program cochair of SIGMETRICS/Performance 2001 and MIS'99, as well as a program committee member of several conferences, including SIGMETRICS, SIGMOD, ICDCS, ICDE, and PDIS. Leana has received several awards, including the US National Science Foundation (NSF) CAREER award, the IBM Doctoral Fellowship, and the US NSF Doctoral Fellowship. She is a member of Tau Beta Pi, the ACM, and the IEEE.