Buffer and I/O Resource Pre-allocation for Implementing Batching and Buffering Techniques for Video-on-Demand Systems

Mary Y.Y. Leung John C.S. Lui *
Department of Computer Science & Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{yyleung, cslui}@cs.cuhk.edu.hk

Leana Golubchik †
Department of Computer Science
Columbia University
New York, NY 10027
leana@cs.columbia.edu

Abstract

To design a cost effective VOD server, it is important to carefully manage the system resources so that the number of concurrent viewers can be maximized. Previous research results use data sharing techniques, such as batching [5], buffering [12], and piggybacking [7], to reduce the demand for I/O resources in a VOD system. However, these techniques still suffer from the problem that additional I/O resources are needed in the system for providing VCR functionality — without careful resource management, the benefits of these data sharing techniques can be lost. In this paper, we first introduce a model for determining the amount of resources required for supporting both normal playback and VCR functionality to satisfy predefined performance characteristics. Consequently, this model allows us to maximize the benefits of data sharing techniques. Furthermore, one important application of this model is its use in making system sizing decisions. Proper system sizing will result in a more cost-effective VOD system.

1. Introduction

Recent advances in information and communication technologies have made important multimedia services such as interactive TV and video-on-demand (VOD) feasible [6]. In a VOD system, video objects are stored in a storage server and played out to the user upon request. A significant amount of research effort has been dedicated to designing VOD servers. Yet, to make VOD service popular, the VOD system has to be cost-effective in the sense that it should store thousands of videos and concurrently support many viewers at a reasonable cost.

Several techniques have been proposed for increasing the number of concurrent viewers of *popular* movies through better usage of resources such as I/O bandwidth and buffer space. One technique is to use the batching [5] method in which one I/O stream is dedicated to servicing several viewers who have arrived within the same timing window. Another technique for reducing the amount of I/O resource needed is the buffering method [12], where viewers can fetch the data blocks from the VOD system buffers instead of from the disk subsystem. Finally, the piggybacking method [7] allows viewers to have different display speed so that eventually, they can catch up with each other and thereby reduce the I/O resource requirements. All these approaches are orthogonal, and they can be implemented in a VOD system to reduce the total I/O resource requirement.

However, in addition to normal playback, a VOD system is expected to provide VCR functions such as fast forward with viewing (FF), rewind with viewing (RW), and pause (PAU). Although batching and buffering techniques can service requests for popular movies in large bulks, they do not work very well in the presence of interactive VCR functions — viewers may "fall out" of a batch during the use of VCR functionality, which can significantly increase the demand for system resources [6]. Therefore, efficient resource management and proper system sizing becomes a very important issue. In this paper, we address the following question: *If batching and buffering are both used, what is the minimum amount of memory buffers and I/O resources that need to be pre-allocated for normal playback such that after a VCR function, the system can deallocate an I/O stream (which was dedicated to a VCR function) with probability greater than or equal to P^*?* If we can increase the probability of releasing these resources when resuming normal playback, then we can use these resources to admit more users, thereby making the VOD system more cost-effective.

The contributions of this paper are as follows. We introduce a model for determining the amount of resources required for supporting both normal playback and VCR functionality, while satisfying predefined performance characteristics. Our proposed model allows us to maximize the benefits of data sharing techniques, such as batching and buffering. In other words, given a certain amount of system buffer and I/O bandwidth resources, we use our mathematical model to determine the optimum distribution of these resources among the popular movies such that normal playback and VCR functions can be provided in a satisfactory manner. The mathematical model is general in the sense that it accommodates a wide variety of probability distributions in modeling FF, RW and PAU. Another contribution of this paper is the application of the model in making system sizing decisions; proper system sizing will result in a more cost-effective VOD system.

The remainder of the paper is organized as follows. In Section 2, we present our system model for reducing I/O resource requirements. In Section 3, we present our mathematical model, which is used for predicting the probability

*This research is supported in part by the UGC Earmarked Grant UGC REF. CUHK 481/95E.

†This research is supported in part by the NSF CAREER grant CCR-96-25013.

of deallocating an I/O resource after a fast-forward operation. A comparison of our mathematical model with simulation results is presented in Section 4. In Section 5, we describe how to apply the model to performing pre-allocation of buffer and I/O resources in a VOD system. Lastly, our conclusions are given in Section 6.

2. System Model

Our system model is based on both batching [5] and partitioned buffer management strategies proposed in [12]. We adopt these methods to improve the scalability of VOD systems by servicing multiple user requests using a single I/O stream. Note that these data sharing techniques should only be applied to requests for *popular* movies. The use of batching for non-popular movies will incur unnecessary latencies while the use of buffering for non-popular movies will be a waste of buffering resources. In this section, we first briefly describe the use of static partitioned buffering in scheduling video requests, and then we present our model.

The static partitioned buffer management strategy is developed based on a buffer refreshing process. This scheduling method is similar to batching, in which the movie is restarted periodically at predefined intervals that and requests arriving during a batching interval can be served using a single I/O stream [5]. In addition, some amount of buffer space, termed a partition, is associated with each I/O stream. This allocated buffer space allows the retaining of movie frames in memory for a certain period of time termed the *viewer enrollment window*. Viewers who come before the closing of the viewer enrollment window (at this point, the frames in the partition are refreshed) read the frames from the buffer partition (we call these the type 2 viewers). Viewers who come after the window is closed (we call these the type 1 viewers) are queued up to wait for the next restart of the movie. The longest time a viewer has to wait occurs when he arrives just after the viewer enrollment window is closed. In this case, the maximum waiting time equals the batching interval (the interval between initiations of two I/O streams) minus the length of the viewer enrollment window. Figure 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 buffers.

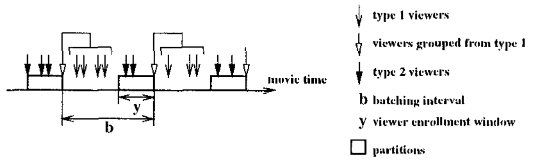


Figure 1. Different types of viewers in static partitioning.

The above scheduling method only deals with normal playback. However, VCR functions like fast-forward (FF), rewind (RW), and pause (PAU) are important features 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 when the viewer resumes to normal playback and is no longer part of any group of requests sharing the I/O and buffer resources. In this paper, we present a model which facilitates the reduction of an additional load resulting from the users resuming to normal playback after a VCR operation. In addition, this model aids in making system sizing decisions, such that the cost-effectiveness can be improved.

VCR functions can be viewed as follows. The FF/RW operations can be interpreted as having to display the movie at y times the normal playback rate. Hence, the reservation of resources, such as I/O and buffer resources, in servicing these VCR requests is unavoidable. In fact, the servicing of these VCR requests can be divided into two phases.

- *phase 1: Displaying the VCR-version of the movie*

  When a viewer issues a VCR requests other than the pause function, additional resources have to be allocated so that he can view the VCR-version of the movie. In [8], two queuing networks have been proposed to model the amount of reserved resources for required VCR requests.

- *phase 2: Resuming to normal playback*

  When resuming from the VCR operation to normal playback, the probability of releasing the resources allocated during phase 1 depends on the position in the movie at which the viewer resumes. If the frames the viewer needs are already in the buffer when he resumes, i.e, in one of the existing partitions, then the viewer can read the data directly from the partition in which he resumes. In this case, the viewer is said to *join* the partition, and the resources allocated in phase 1 can be released. Otherwise, the viewer has to continue utilizing the resources allocated in phase 1 until he can join a partition, for instance, using the piggybacking technique [1, 7, 9].

Note that the allocation of resources in phase 1 is usually inevitable. On the other hand, whether or not these allocated resources can be released depends on the resuming position. If there is a high probability that the viewers can resume at existing partitions, the additional load resulting from a resume can be reduced. Therefore, our goal is to maximize the probability of releasing the allocated resources at a resume point so as to minimize the consumption of reserved resources. Thus, more resources are available for servicing future requests, such as forthcoming VCR requests, or they can even be dedicated to serving requests for non-popular movies. Clearly, the size and the number of partitions,[1] used for servicing normal playback, together with the duration of VCR requests, will greatly affect the probability of resuming at an existing partition. In this paper, we combine the virtues of batching and buffering (for popular movies) while providing support for VCR-like interactive requests by applying the static partitioning model. Based on the distribution of the duration of VCR requests, we perform resource allocation to determine the resources needed for normal playback such that the probability of holding the resources upon resume can be minimized.

2.1. Assumptions and Principles

A viewer is said to have a *hit* if he can resume at some partitions when resuming to normal playback after a VCR request, such that additional resources allocated to him in phase 1 can be released in phase 2. Otherwise, the viewer is said to have a *miss* if he resumes at the *gaps* between partitions. The hit and miss events are illustrated in Figure 2.

[1]Note that the number of partitions is equal to the number of I/O streams associated with a movie.

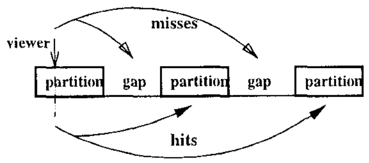


Figure 2. Hits and misses for viewers resuming from VCR requests.

Since our goal is to maximize the hit probability, it is natural to consider the behavior of VCR requests in determining the *configuration* of the system, i.e, in determining how much buffer and I/O resources 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 time spent in a VCR request. The pdf of VCR requests can be obtained by statistics while the movie is displayed, and is defined in the interval $[0, l]$, where l is the movie length (in time unit). 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 movie is restarted periodically.

We assume that the arrivals of requests for a popular movie are distributed according to a Poisson process. This is a reasonable model of the arrival process since we expect the VOD system to have a large user population. Given the above assumptions, we determine the expected hit probability upon resume to normal playback under various system configurations. The basic idea is as follows. If we could maximize the hit probability, then we could reduce the additional load resulting from VCR requests for resuming to normal playback. In other words, by increasing the hit probability, the probability of releasing reserved resources, consumed in phase 1, is increased. Consequently, the system will be able to service more requests simultaneously.

It is important to point out that, although relatively little time is spent in VCR modes as compared to the normal playback [8], inefficient schemes for providing VCR functionality can easily result in consumption of large amounts of system resources. As mentioned above, the approach, taken in this paper, to remedy the situation is to increase the hit probability. However, this is not a simple task. Intuitively, increasing the partition size will increase the fraction of a movie that resides in main memory. An extreme case would be to buffer the entire movie and thus increase the hit probability to 1. However, this will not necessarily result in the reduction of the amount of overall system resources required. Therefore, without an accurate model and a good system sizing method, we will not be able to calculate the right amount of I/O bandwidth and buffer space resources to reserve in order to design a cost-effective VOD system. In the following section, we present such a model for facilitating resource pre-allocation and system sizing.

3. Mathematical Model

In this section, we present the derivation of our mathematical model. The basic idea pursued here is to use a mathematical model to determine an 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.

Before presenting the mathematical model, let us present the criteria for *catching up* to a stream. An example situation is illustrated in Figure 3(a). We denote R_{FF} as the rate of fast forward and R_{PB} as the rate of normal playback. Suppose viewers X and Y are "traveling" at a rate of R_{FF} and R_{PB}, respectively, and X lags behind Y by Δ minutes. Since R_{FF} is greater than R_{PB}, X will eventually *catch up* with Y at the catch-up point. Similarly, Figure 3(b) 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.

$$t = \begin{cases} \alpha \cdot \Delta, & \text{where } \alpha = \frac{R_{FF}}{R_{FF}-R_{PB}} \text{ for FF,} \\ \gamma \cdot \Delta, & \text{where } \gamma = \frac{R_{RW}}{R_{PB}+R_{RW}} \text{ for RW.} \end{cases} \tag{1}$$

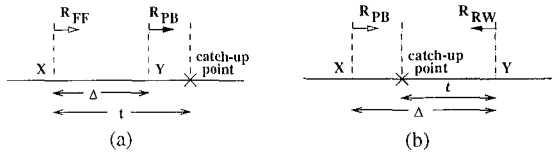


Figure 3. scenarios for catching up to viewers a) in front b) behind

In the following section, we present the buffer and I/O requirements for normal playback under static partitioning. After introducing the problem formulation, we present a complete derivation of our mathematical model for predicting the probability of deallocating an I/O resource upon a resume from a VCR operation.

3.1. Resources Requirements for Normal Playback

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 $\frac{l}{n}$ minutes. Assume that the total amount of buffers allocated for normal playback for a popular movie can store B' minutes of the movie. Therefore, the size of each partition is B'/n. Then, in the worst case, if a viewer arrives at the time when the viewer enrollment window has just been closed, the viewer has to wait for the next restart; this is the maximum amount of time a viewer has to wait. The length of the viewer enrollment window is equal to $B'/n - \delta$, where δ is the reserved amount of buffer space. This 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 [12]. 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:

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

It is important to note that $n = \frac{l}{w}$ would imply that $B = 0$, which corresponds to the pure batching case, where each batch is served by a single I/O stream. However, in this case, the hit probability will always equal to zero, unless n is infinitely large. Rearranging the above equation gives the number of I/O streams needed, n as $\frac{l}{w} - \frac{B}{w}$, 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. 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. Of course, here we are using w minutes of buffer space as a tradeoff with one I/O stream. In the case of popular movies which are restarted frequently (i.e, w is small), we can save one I/O stream using relatively little buffer space.

When referring to a particular viewer, we adopt the notation, V_c, to denote the position of a viewer currently viewing the V_c^{th} minute of the movie. Furthermore, the positions of 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 may not be actual viewers viewing the V_f^{th} and V_l^{th} minute 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 $\frac{B}{n}$ and is illustrated in Figure 4.

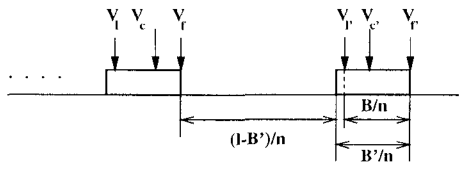
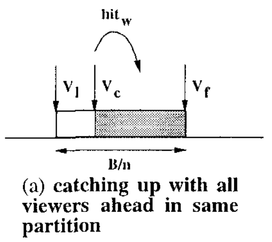


Figure 4. relative positions of viewers at V_c,V_f, V_l in a partition

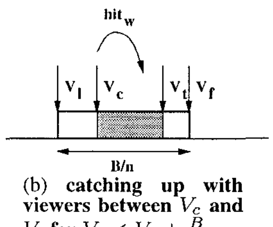
The idea behind the model formulation is to use it to determine the probability of a hit when a viewer resumes from a VCR request. We denote $P(hit|V_c, V_f)$ to be the conditional probability of a hit, given that a viewer is at position V_c and that the first possible viewer in the same partition is 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 of the duration of a FF request. The main difficulty in handling VCR requests lies in their inherently non-deterministic nature [8], 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, 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.

3.1.1. Hits occurring within the same partition (hit_w)



(a) **catching up with all viewers ahead in same partition**



(b) **catching up with viewers between V_c and V_t for $V_t < V_c + \frac{B}{n}$**

Figure 5. Two cases of hit_w for viewer V_c

An event hit_w occurs if a viewer can resume in the same partition in which the VCR request is issued. In this case, the longest fast forward duration which can still result in a hit_w is the distance between the viewer in question and the first possible viewer in his partition. This situation is illustrated in Figure 5(a) in which the viewer must resume in the shaded region of the partition. The probability of a hit_w, given that the viewer is at position V_c and that the first viewer is at position V_f, for a FF request is as follows:

$$P(hit_w|FF, V_c, V_f) = \int_0^{\alpha\Delta_w} f(x)dx \quad \text{where} \quad \Delta_w = V_f - V_c \qquad (3)$$

Our aim is to calculate the probability of a hit_w when a viewer resumes from any VCR request, i.e, we need to uncondition the above probability to obtain $P(hit_w)$. 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.

- 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 $V_f = V_c + \frac{B}{n}$).

  It is important to 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 Eq. (1), the time t, elapsed before a catch-up is accomplished is equal to the initial distance, Δ, times a constant greater than 1. Therefore, if $V_c + t > l$, 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: *can accomplish 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 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:

$$P_a(hit_w|FF, V_c) = \int_{V_f=V_c}^{V_c+\frac{B}{n}} P(hit_w|FF, V_c, V_f)P(V_f)dV_f \quad (4)$$

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 that results obtained using our mathematical model matched those obtained through simulation in Section 4.

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

In this case, we handle the boundary condition which ensures that $V_c + t \leq l$. That is, the catch-up to some viewers within the same partition is still possible before the end of the movie. In other words, consider the scenario that 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 l^{th} minute of the movie. Using Eq. (1), 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:

$$V_t \quad \leq \quad \frac{l + (\alpha - 1)V_c}{\alpha} \quad (5)$$

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 Eq. (5). Figure 5(b) 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 duration between 0 and $\alpha(V_t - V_c)$.

$$P_b(hit_w|FF, V_c) = \int_{V_f=V_c}^{V_t} P(hit_w|FF, V_c, V_f)P(V_f)dV_f + \int_{V_f=V_t}^{V_c+\frac{B}{n}} \int_0^{\alpha(V_t-V_c)} f(x)\, dx P(V_f)dV_f \quad (6)$$

Using the assumption made in case a, $P(V_f)$ is equal to $\frac{1}{B/n}$.

- 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. Here we also uncondition on V_c based on the two cases used for unconditioning on V_f. Again, we have two cases to consider:

  case a: $V_c + \frac{B}{n} \leq V_t$ (or $V_c < l - \frac{B\alpha}{n}$, using Eq. (5).)

$$P_a(hit_w|FF) = \int_{V_c=0}^{l-\frac{B\alpha}{n}} P_a(hit_w|FF, V_c)\, P(V_c)\, dV_c \quad (7)$$

The only unknown here is $P(V_c)$, which is the probability that a viewer is at the V_c^{th} 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 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 to be 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.

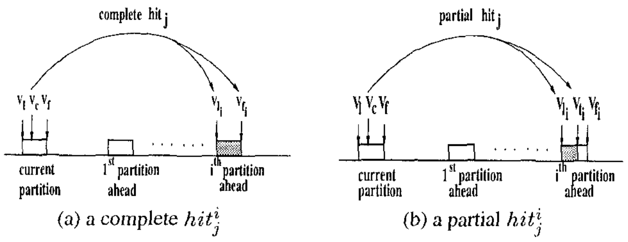
$$P_b(hit_w|FF) = \int_{V_c=l-\frac{B\alpha}{n}}^{l} P_b(hit_w|FF, V_c)\, P(V_c)\, dV_c \quad (8)$$

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 Eqs. (7) and (8).

3.1.2. Jump to other partitions (hit_j)

In addition to resuming 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 VCR request can be released. We refer to this event as a hit_j. As illustrated in Figure 6, if a viewer is to resume at the i^{th} partition ahead of its current position, then he must fast forward long enough to at least catch up with the last possible viewer in the i^{th} partition ahead, namely, at position V_{l_i}. Let us denote by V_{f_i} the position of the first possible viewer in the i^{th} partition, then the event hit_j can be divided into two cases, namely, the *complete* hit_j^i and the *partial* hit_j^i. As illustrated in Figure 6(a), a complete hit_j^i 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}. Figure 6(b) illustrates the event of a partial hit_j^i, which illustrates a viewer which is not able to catch up with the viewer at position V_{f_i} before the movie ends.



(a) a complete hit_j^i (b) a partial hit_j^i

Figure 6. A hit in another partitions (hit_j).

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_j by Δ_{jump_l} and Δ_{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.

$$\begin{aligned}\Delta_{jump_l} &= V_{l_i} - V_c \\ &= \frac{il}{n} + V_f - V_c - \frac{B}{n} \\ \Delta_{jump_f} &= V_{f_i} - V_c \\ &= \frac{il}{n} + V_f - V_c\end{aligned}$$

The probability of a hit_j at the i^{th} partition ahead, denoted by $P(hit_j^i)$, is shown below :

case 1: able to catch up with both viewers at position V_{l_i} and V_{f_i} (a complete hit_j)

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 i^{th} partition ahead:

$$P_c(hit_j^i|FF, V_c, V_f) = \int_{\alpha\Delta_{jump_l}}^{\alpha\Delta_{jump_f}} f(x)dx \tag{9}$$

case 2: able to catch up with a viewer at position V_l (a partial hit_j)

In boundary cases, a viewer at position V_c may not be able to catch up with both viewers at position V_{l_i} and V_{f_i}. However, even if a viewer at position V_c cannot catch up with all viewers in the i^{th} partition, as long as he is able to catch up with a viewer at position V_{l_i}, we will classify this event as a partial hit_j. As before, we define V_{t_i} to be the position of the last viewer in the i^{th} partition with which a viewer at position V_c can catch up. Thus, we have that $V_t = V_{t_i} - \frac{il}{n}$, resulting in $V_t = \frac{l+(\alpha-1)V_c - \frac{il\alpha}{n}}{\alpha}$:

$$\begin{aligned}P_p(hit_j^i|FF, V_c, V_f) &= \int_{\alpha\Delta_{jump_l}}^{\alpha(\frac{il}{n}+V_t-V_c)} f(x)dx \\ &= \int_{\alpha\Delta_{jump_l}}^{l-V_c} f(x)dx \qquad (10)\end{aligned}$$

- Unconditioning on V_f

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

$$P_1(hit_j^i|FF, V_c) = \int_{V_f=V_c}^{V_c+\frac{B}{n}} P_c(hit_j^i|FF, V_c, V_f)P(V_f)dV_f \tag{11}$$

$$P_2(hit_j^i|FF, V_c) = \int_{V_f=V_c}^{V_t} P_c(hit_j^i|FF, V_c, V_f)P(V_f)dV_f \tag{12}$$

Observe that a viewer at position V_c can have a complete hit_j as well a partial hit_j, depending on the value of V_f. Given a viewer at position V_c, Eq. (12) gives the probability of a complete hit_j 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_j; this is shown in Eq. (13) below. Eq. (14) corresponds to those viewers who can only have a partial hit_j for all values of V_f, where the largest V_f with respect to V_c is defined as $V_{t'} = \frac{l+(\alpha-1)V_c - \alpha(\frac{il-B}{n})}{\alpha}$.

$$P_3(hit_j^i|FF, V_c) = \int_{V_f=V_t}^{V_c+\frac{B}{n}} P_p(hit_j^i|FF, V_c, V_f)P(V_f)dV_f \tag{13}$$

$$P_4(hit_j^i|FF, V_c) = \int_{V_f=V_c}^{V_{t'}} P_p(hit_j^i|FF, V_c, V_f)P(V_f)dV_f \tag{14}$$

- 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.

$$P_1(hit_j^i|FF) = \int_0^{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}} P_1(hit_j^i|FF, V_c)\ P(V_c)\ dV_c \tag{15}$$

$$P_2(hit_j^i|FF) = \int_{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}}^{l-\frac{il\alpha}{n}} P_2(hit_j^i|FF, V_c)\ P(V_c)\ dV_c \tag{16}$$

$$P_3(hit_j^i|FF) = \int_{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}}^{l-\frac{il\alpha}{n}} P_3(hit_j^i|FF, V_c)\ P(V_c)\ dV_c \quad (17)$$

$$P_4(hit_j^i|FF) = \int_{l-\frac{il\alpha}{n}}^{l-\frac{(il-B)\alpha}{n}} P_4(hit_j^i|FF, V_c)\ P(V_c)\ dV_c \quad (18)$$

Hence, the probability of resuming at the i^{th} partition ahead for a FF request, $P(hit_j^i|FF)$, is calculated by summing Eqs. (15) through (18).

The number of partitions to which a viewer at position V_c would be able to jump ahead, resulting in hit_j, depends on in which partition this viewer is. For example, it would be impossible for a viewer in the last partition of the movie to jump ahead to another partition. Furthermore, the existence of V_t also limits the number of partitions to which a viewer can jump ahead. To find the range of i, we refer back to Eq. (15), which indicates 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 gives the range of i:

$$i \leq \lfloor \frac{n(l + w\alpha) - l\alpha}{l\alpha} \rfloor \quad (19)$$

3.1.3. Fast-forwarding to the end of a movie

Recall that our goal is to maximize the probability of the resources, allocated for a VCR request, to 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 non-zero 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 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:

$$\begin{aligned} P(end) &= \int_0^l \int_{\alpha(V_t - V_c)}^l f(x)\ dx\ P(V_c)\ dV_c \\ &= \int_0^l \int_{l-V_c}^l f(x)\ dx\ \frac{1}{l}\ dV_c \end{aligned} \quad (20)$$

Finally, $P(hit|FF)$, the probability of a hit given that the VCR request is a FF operation, is calculated as follows.

$$\begin{aligned} P(hit|FF) &= P(hit_w|FF) + \sum_{i=1}^{\lfloor \frac{n(l+w\alpha)-l\alpha}{l\alpha} \rfloor} P(hit_j^i|FF) \\ &\quad + P(end) \end{aligned} \quad (21)$$

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 i^{th} partition ahead, where $i \geq 1$. 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 [10].

3.1.4. 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}, 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, $P(hit)$, which can be expressed as follows.

$$\begin{aligned} P(hit) &= P(hit|FF)P_{FF} + P(hit|RW)P_{RW} \\ &\quad + P(hit|PAU)P_{PAU} \end{aligned} \quad (22)$$

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:

$$w = \frac{l - B}{n} \quad (C1)$$

$$P^* \leq P(hit) \quad (C2)$$

The first constraint, $C1$, corresponds to the maximum waiting time that a viewer might experience before the movie restarts. The second constraint, $C2$, ensures the average probability of a hit conform 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 pre-allocated for playback of a popular movie in order to satisfy the performance requirement (or quality of service) 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 5.

4. Model Verification

In this section, we validate 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[2] with a rate λ. As VCR requests are usually of short duration, as simple cases for illustration, we use an exponential distribution and a skewed gamma distribution to represent the duration of the VCR functions.

[2]The rationale for using a Poisson process was given in Section 2.1.

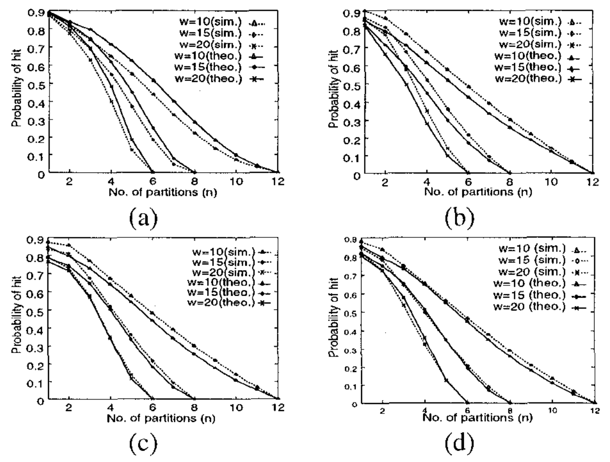


Figure 7. Simulation and theoretical results for normal playback and (a) only fast-forward VCR requests (b) only rewind VCR requests (c) only pause VCR requests (d) all kinds of VCR requests with $P_{FF} = 0.2$,$P_{RW} = 0.2$ and $P_{PAU} = 0.6$. Interarrival times are exponential and $1/\lambda = 2$ minutes; duration of VCR requests is drawn from a skewed gamma distribution with a mean = 8 minutes ($\alpha = 2, \gamma = 4$).

The rates of fast forward and rewind, R_{FF} and R_{RW}, are three times that of normal playback, R_{PB}. The movie length l is equal to 120 minutes. In the first three experiments, we simulate the 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 two 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 f is equal to P_f, and f 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. Furthermore, 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 Figures 7(a) to 7(c). 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. A mix of all three VCR requests plus requests for normal display is shown in Figure 7d.

These figures illustrate that the theoretical results match closely with the simulation results, which validates our mathematical model. The minor disagreement between the simulation and the theoretical results can be explained as follows. Firstly, we have assumed a uniformness of viewers within a single partition in deriving our mathematical model. However, in a real system, viewers who arrive after the closing of the viewer enrollment window (refer to Figure 1) will all become the first viewer (i.e., a type 1 viewer) when the movie restarts. Secondly, the hit probability in our mathematical model is calculated by considering the viewer at all positions in the movie, and a hit at a particular partition is treated as catching up with either a viewer at position V_f or at position V_l. A discrepancy occurs when a viewer is at a boundary position, namely, the 0^{th} minute of the movie. In a real system, if that viewer issues a PAU or RW request at that position, then the probability of a hit upon resume will depend on whether the enrollment window is still open. But in our model we assume that a miss occurs in this case. This explains why our model underestimates the probability of a hit for the RW and PAU cases (refer to Figures 7(b) & (c)). The final discrepancy occurs when viewers arrive after the closing of the enrollment window but before the restart of the movie. The position of all these viewers corresponds to the leading position of a partition. When one of these viewers initiates a FF request, our model will over-estimate the hit probability due to the uniformness assumption. Similarly, if one of these viewers issues a RW or PAU, our model will under-estimate the hit probability.

5. Application to system sizing

In this section, we illustrate how our mathematical model can be applied to making system sizing decisions. In managing system resources, our goal is to optimize the performance of a VOD system by optimizing the usage of system resources while providing an acceptable level of quality of service to the viewers. Note that the provision of VCR functions and the minimization of average waiting time, w, will inevitably increase system resource requirement. However, 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. Given the performance requirements of maximum waiting time, w,and minimum hit probability, P^*, the mathematical model will determine the minimum amount of buffer and I/O resources that would be needed to support these requirements. In other words, we use our model to answer questions of the following form. *Given B_s and n_s, the amount of system buffer space and the amount of I/O bandwidth available for playback respectively, how should we allocate buffer space, B_i, and I/O bandwidth, n_i, to movie i, $\forall i \in [1 \ldots m]$ such that (1) $\sum_{i=1}^{m} B_i \leq B_s$, (2) $\sum_{i=1}^{m} n_i \leq n_s$, and (3) hit probability, $P_i \geq P_i^*$, in order to support m popular movies simultaneously (where $m \leq n_s$).*

For each movie i of length l_i, given the maximum waiting time, w_i, we calculate the respective buffer size, B_i, for each n_i using Eq. (2). The hit probability corresponding to each pair, i.e, $P(B_i, n_i)_{w_i}$, can be obtained using our mathematical model. The optimal choice of $(B_i, n_i)_{w_i}$ is computed through the following steps:

Step 1: Pick a value of P_i^* for each movie i to be the minimum hit probability that the system needs to support.

Step 2: Choose all the $(B_i, n_i)_{w_i}$ pairs such that $P(B_i, n_i)_{w_i} \geq P_i^*$. These $(B_i, n_i)_{w_i}$ pairs will constitute the feasible set S_i.

Step 3: From each feasible set S_i, $\forall i \in [1 \ldots m]$, we determine an optimal $(B_i^*, n_i^*)_{w_i}$ pair such that the overall buffer space and I/O streams needed would be minimized. The choices of $(B_i^*, n_i^*)_{w_i}$ for each movie i determine the overall distribution of resources among the popular movies.

What remains to be shown is how to determine the optimal pair, $(B_i^*, n_i^*)_{w_i}$ in step 3 above. Given Eq. (2), there is always a tradeoff between the buffer space and the I/O bandwidth requirements, since they are inversely proportional in a system using batching and buffering. Since I/O bandwidth is one of the critical and crucial resources needed to start a batch, we reduce I/O bandwidth requirement at the expense of memory buffer requirement. We anticipate that this tradeoff is worthwhile for popular movies, since we are only using w minutes of buffer space as a tradeoff for one I/O stream. (Recall that for popular movies the value of w will be small). Another reason for using buffer space to reduce the amount of I/O bandwidth needed is to increase the probability of a hit. When the probability of a hit is very small, it is almost impossible for the viewer to release the I/O stream upon resume until he finishes viewing the whole movie. On the other hand, a high probability of a hit would ensure the release of I/O resources upon resume. These resources can be used to service future VCR requests or requests for normal playback of non-popular movies. Therefore, not only can we use a small amount of buffer space to save one I/O stream, but we can also use it to increase the probability of a hit, which will avoid the exhaustion of reserved resources[3] for servicing VCR functions. This implies that less resources need to be reserved. Since the cost of the memory buffers is far from negligible, in determining the optimal choices of $(B_i^*, n_i^*)_{w_i}$ pairs, we choose to minimize $\sum_{i=1}^{m} B_i^*$. Using Eq. (2), we calculate B_i^* to be $l_i - n_i^* w_i$ and formulate the following constrained optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} l_i - n_i^* w_i \\ \text{subject to} & \sum_{i=1}^{m} n_i^* \le n_s \\ & \sum_{i=1}^{m} l_i - n_i^* w_i \le B_s \\ \text{where} & (B_i^*, n_i^*)_{w_i^*} \in \mathcal{S}_i \end{array}$$

Example 1 Consider three popular movies, $1, 2, 3$ with lengths 75, 60, and 90 minutes respectively. Also, let w_i^* be 0.1, 0.5 and 0.25 minute respectively. The duration of VCR requests of movie 1 are drawn from a gamma distribution with a mean equal to 8 minutes (i.e., $\alpha = 2, \gamma = 4$) while the duration of VCR requests of movie 2 and 3 are drawn from an exponential distribution with a mean equal to 5 minutes and 2 minutes, respectively. Let the performance requirement of P_i^* for each movie i be equal to 0.5. Applying our mathematical model to each of the three movies, the feasible values for each movie are plotted in Figure 8 for a step size of a 5-minute-buffer.

In a pure batching case, the number of I/O streams needed would be equal to $\frac{75}{0.1} + \frac{60}{0.5} + \frac{90}{0.25}$, which amounts to 1230 I/O streams, where the probability of a hit is 0. On the other hand, suppose that $n_s = 1230$, then solving the optimization problem gives $[(B_1^*, n_1^*), (B_2^*, n_2^*), (B_3^*, n_3^*)] = [(39, 360), (30, 60), (44.5, 182)]$. Thus, the minimum amount of buffer space required to ensure the mean hit probability and the mean waiting times as specified above is 113.5 minutes worth and the number of I/O streams needed is only 602. Therefore, we have saved 628 I/O streams at the expense of 113.5 minutes worth of buffer space.

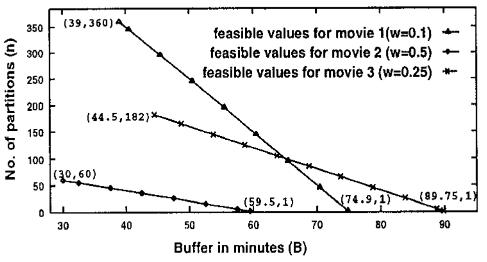


Figure 8. different (B,n) pairs for movies 1, 2, 3 for each 5 minutes of buffer space

Note that there are multiple pairs of (B^*, n^*) that can satisfy the given performance requirements but at a different cost. Hence, our next goal is to reduce the overall cost of resources, including resources needed for normal playback as well as for servicing VCR requests. Given P_i^* and w_i $\forall i \in [1 \ldots m]$, we can use our model to determine the system cost C to be equal to $C = C_b \sum_{i=1}^{m} B_i^* + C_n \sum_{i=1}^{m} n_i^*$ where C_b and C_n are the costs for memory buffers per one minute of a movie and one I/O stream, respectively. Rearranging gives the following equation.

$$C = C_n(\varphi \sum_{i=1}^{m} B_i^* + \sum_{i=1}^{m} n_i^*) \quad \text{where} \quad \varphi = \frac{C_b}{C_n} \tag{23}$$

In other words, we measure the system cost in terms of the amount of buffer space and I/O bandwidth needed to ensure a level of performance, as expressed through a minimum hit probability and a maximum waiting time. Next, we illustrate how our model can be utilized in making system sizing decisions through the following example.

Example 2 In this simple example, we assume that a 2G-SCSI disk costs around \$700 and that its transfer rate is 5 MB/sec. The data rate of an MPEG-2 movie is assumed to be 4 Mbps, and the cost of 1 MB of main memory is assumed to be \$25. We calculate C_b and C_n as follows.

$$C_b = \frac{60s * 4\text{Mbps}}{8} * \$25 = \$750$$

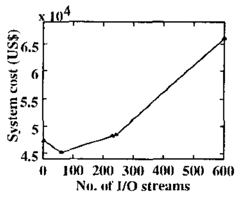
$$C_n = \frac{\$700}{5\text{M Bytes/sec} * 8/4\text{Mbps}} = \$70$$

Given the above equations, we can see that the amount of buffer space required to store 1 minute of an MPEG-2 movie is approximately 11 times as expensive as one I/O stream, i.e., $\varphi \approx 11$.

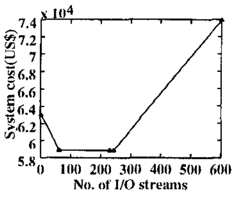
Consider the system of Example 1. The system cost for normal playback is plotted in Figure 9(e) as a function of the number of I/O streams. The minimum point on a cost curve in this figure is the optimal system sizing choice, since it not only ensures the performance requirement, i.e, P_i^* and w_i, but also has the lowest cost.

[3]In other words, if there is no chance of releasing I/O resources back to the system pool, then each VCR request will consume one I/O resource until the viewer finishes the movie. Then, more VCR requests implies more resources will be held by the viewers until they finish watching the movie.

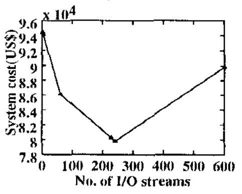
In the previous example, the minimum cost occurs when the number of I/O streams reaches its maximum feasible value because the cost of memory buffers dominate the overall system cost. However, in future technology, where cost of memory may decrease faster than the cost of I/O bandwidth, the minimum cost will occur at some other positions on the cost curve, as illustrated in Figure 9. In this experiment, we vary φ as $3, 4, 6, 10, 11$ and 16. Figures 9(a) to 9(d) illustrate the situation where the cost of the memory decreases faster than the cost of I/O bandwidth while Figures 9(e) and 9(f) illustrate the situation where the cost of I/O bandwidth decreases faster than the cost of memory. Again, this model helps system designers to make the right system sizing decisions and at the same time, meet the performance requirements.



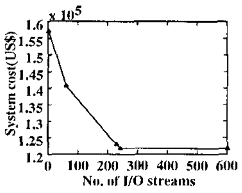
(a) $\varphi = 3$



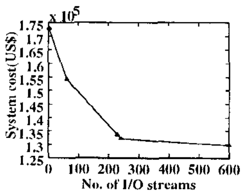
(b) $\varphi = 4$



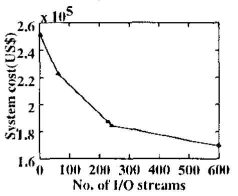
(c) $\varphi = 6$



(d) $\varphi = 10$



(e) $\varphi = 11$



(f) $\varphi = 16$

Figure 9. System cost vs number of I/O streams for different values of φ

6. Conclusions

In a VOD environment, the provision of VCR functions will inevitably require additional system resources, in the sense that resources have to be reserved in advance for serving these VCR requests. When using batching and buffering techniques, the scalability of VOD systems can be improved through data sharing. For example, the movie can be restarted periodically, after a batching interval, and certain portions of the movie can exist in the buffers. We observe that, if the viewer is able to read the data from the buffers upon resume to normal playback (we call this a hit), then no additional resources will be consumed once the VCR operation is finished. It is very important to determine the right amount of I/O streams and memory buffers needed to be allocated so as to optimize the performance of VOD systems as well as to maintain the benefits of batching and buffering techniques. In summary, we first introduced a very general mathematical model for determining the amount of resources required for supporting both normal playback and VCR functionality to satisfy predefined performance characteristics. Consequently, this model allows us to maximize the benefits of data sharing techniques. We illustrated how to use our model in performing pre-allocation of buffer space and I/O resources for popular movies such that the number of concurrent users of a VOD system is increased in a cost-effective manner. Furthermore, we illustrated an important application of this model, namely, its use in making system sizing decisions, where proper system sizing results in a more cost-effective VOD system.

References

[1] C. Aggarwal, J. Wolf, and P. Yu. Optimal Piggyback Policies for Video-on-Demand Systems. In *ACM SIGMETRICS Conference*, pages 200–209, May 1996.

[2] H. Chen, A. Krishnamurthy, T. Little, and D. Venkatesh. A Scalable Video-on-Demand Service for the Provision of VCR-Like Functions. In *Proc. 2nd Intl. Conf. on Multimedia Computing and Systems*, Washington DC, May 1995.

[3] M.-S. Chen, D. Kandlur, and P. Yu. Storage and retrieval methods to support fully interactive playout in a disk-array-based video server. *Multimedia Systems*, pages 126–135, 1995.

[4] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and Caching in Large-Scale Video Servers. *IEEE CompCon*, 1995.

[5] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In *Proc. ACM Multimedia '94*, pages 391–398, San Francisco, 1994.

[6] D. Deloddere, W. Verbiest, and H. Verhille. Interactive video on demand. *IEEE Communication Magazine*, 32(5):82–88, 1994.

[7] L. Golubchik, J. Lui, and R. Muntz. Adaptive Piggyback: A novel Technique for Data sharing in Video-On-Demand Storage Servers. *ACM Journal of Multimedia Systems*, 4(3):140–155, June 1996.

[8] J. K.D., J. Salehi, J. Kurose, and D. Towsley. Providing VCR Capabilities in Large-Scale Video Servers. In *Proc. ACM Intl. Conf. on Multimedia*, pages 25–32, San Francisco, Oct. 1994.

[9] L. Lau, J. Lui, and L. Golubchik. Merging Video Streams in a Multimedia Storage Server: Complexity and Heuristics. *Accepted for publication in ACM Journal of Multimedia Systems*, 1996.

[10] M. Leung, J. Lui, and L. Golubchik. Buffer and I/O Resource Pre-allocation for Implementing Batching and Buffering Techniques for Video-on-Demand Systems. Technical report, The Chinese University of Hong Kong, Apr. 1996. Technical Report CS-TR-96-03.

[11] R. Ng and J. Yang. Maximizing Buffer and Disk Utilizations for News on-Demand. In *Proceedings of the 20th VLDB Conference*, pages 451–462, Santiago, Chile, 1994.

[12] D. Rotem and J. Zhao. Buffer Management for Video Database Systems. In *Proc. Intl. Conference on Data Engineering*, pages 439–448, Taipei, Taiwan, Mar. 1995.

[13] K. Wu and P. Yu. Consumption-Based Buffer Management for Maximizing System Through puts of News-on-Demand Multimedia Systems. *IBM Research Report RC 20143*, Jan. 1995.

[14] P. Yu, J. Wolf, and H. Shachnai. Design and Analysis of A Look-ahead Scheduling Scheme to Support Pause-Resume for Video-on-Demand Applications. *ACM Journal of Multimedia Systems*, 3(4):137–150, 1995.