System Sizing and Resource Allocation for Video-on-Demand Systems

By
Mary Y.Y. Leung

Supervised By :
Prof. John C.S. Lui

Submitted to the Division of Department of Computer Science & Engineering
in partial fulfillment of the requirements for the
degree of Master of Philosophy
at the
Chinese University of Hong Kong
June 1997

香港中文大學
大學
圖書館系統
22 MAY 1998
UNIVERSITY
LIBRARY SYSTEM
THE CHINESE UNIVERSITY OF HONG KONG

System Sizing and Resource Allocation for Video-on-Demand Systems

submitted by

Mary Y.Y. Leung

for the degree of Master of Philosophy
at the Chinese University of Hong Kong

Abstract

As video data requires a large storage space and stringent real-time delivery, in order to design a cost-effective VOD server, efficient resource management and proper system sizing is very important. However, supporting interactive VCR functions such as fast forward, rewind, pause and resume features will impose additional resource requirement on the VOD system in terms of storage space, disk and network bandwidth, etc. Previous research results use batching, buffering and adaptive piggybacking in reducing the I/O demand for normal playback requests. However, these data sharing techniques complicate the provision of VCR functions because when a viewer in a batched group issues a VCR request, a new stream needs to be initiated for displaying the fast forward/rewind version of the movie to the viewer. Yet even worse is that when the viewer resumes to normal playback, it is impossible to merge the individual to the batched streams again, i.e. viewers may 'fall out' of the batches upon resume to normal playback. The only way for continuing their normal playback is to allocate new streams to these viewers, which is not desirable. Therefore, without an efficient scheme to handle VCR requests and careful resource management, the benefits of these data sharing techniques will be lost.

In our work, we solve the system sizing problem of a VOD system by determining the right amount of resources to be allocated for normal playback and servicing VCR requests while satisfying predefined performance characteristics. We propose a novel, general mathematical model to calculate the amount of resources needed for normal playback by ensuring a certain probability for the viewers to join in the existing batches upon resume from VCR requests. This approach has not been investigated in any existing literature before. We also propose a scheme to support the VCR functionality: the interactive viewers are supported by new streams whenever they issue fast forward/rewind request. Upon resume, if the viewers can join in the existing groups,

the newly allocated streams will be released, otherwise, they will hold on to the resource for their normal playback. Meanwhile, these fallen out viewers will be merged back to the batched streams by using adaptive piggybacking. We further calculate the amount of VCR resources needed by applying a queuing model. We show by experiments on how to use our proposed approach to distribute the resources for normal playback and servicing VCR requests under different arrival rates of VCR requests and different price ratios of memory buffer space to I/O bandwidth.

Acknowledgments

I am indebted primarily to my academic and research supervisor, John C.S. Lui for his guidance, support, encouragement and endless patience, which are essential to the quality and timely completion of the work presented in this thesis.

I would also like to thank Leana Golubchik from Columbia University, who read my paper and provided helpful comments and constructive criticism in my research.

A lot of my colleagues have to be especially mentioned, because things would have been different without them. I would like to thank Cedric Fong, Dilys Hung, Kuok Chan Man, Cheung Chi Chiu, Law Kwok Wai, Lam Sze Kin, Alan Tung, Wong Tai Man, Jason Wong, Terence Wong, Elton Tsang, in providing a wonderful environment for my study, with whom I interacted for their friendship, and for making life at CUCSE fruitful and enjoyable. Lastly, but certainly not the least, I wish to extend my gratitude to my parents and family for their love, and never fading spiritual support in the study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent technological advances in information and communication technologies have made fiber optic networks and high bandwidth available at lower costs. This has stimulated the integration of digital continuous media, such as audio and video, into existing computing environments. Multimedia systems play a major role in a wide range of applications such as education domain, diagnostic imaging, scientific visualization and commercial services. Among these services, interactive TV and video-on-demand (VOD) have been identified as two important services in the coming few years [DVV94]. Here we will mainly focus on video-on-demand systems, the most challenging aspect of which is to provide *on-demand* service to multiple clients simultaneously and to support interactive VCR functionalities. In both cases, the users expect that their requests can be satisfied within a short and reasonable time. However, several difficulties need to be dealt with. Firstly, unlike textual data, video objects require large storage, high disk bandwidth for retrieval and high network bandwidth for transmission. Secondly, video data also requires stringent real-time delivery, i.e. it is delay-sensitive. Unlike conventional computer applications, continuous media playback requires that data be delivered at a steady rate without significant variations, otherwise, it may result in jerkiness and jitters. Therefore, sufficient amount of resources has to be reserved ahead in order to provide an acceptable level of service to the viewers. Much research effort has been dedicated to design new techniques for the efficient use of resources, so that the resources needed to satisfy predefined performance measure can be reduced.

1.1 Video-On-Demand Environment

In a VOD system, video objects are stored in a storage server and played out to the users upon request. The storage server consists of a set of disks, a set of processors,

buffer space and a tertiary storage device. The more frequently accessed objects are cached on disks for high performance access and they provide video streams to the viewers with some form of quality-of-service guarantee to ensure a smooth playback. We assume that the caching on disks is done on-demand, i.e. a non-disk resident object is fetched from tertiary storage only when it is referenced. The server will store the whole database of digitized and compressed video data on tertiary storage. Due to the long latency encountered and the high bandwidth cost of tertiary storage, video objects cannot be transmitted to the viewers directly from the tertiary devices. If the requested video object is not disk-resident, it has to be retrieved from the tertiary device and placed on disk before initiating its display. In other words, the server should provide transparent access to objects in tertiary storage for the viewers. Besides, an I/O stream is needed to be scheduled to read the disk-resident video data from the disk to display to the viewer.

Several basic storage architectures have been proposed for constructing low-cost VOD servers and can be found in [TPBG93, OBRS94, FR94, BP94, LL95]. The authors propose different data layouts, scheduling techniques and playout control with their proposed hierarchies to design low-cost VOD servers. A storage server can be described in terms of the following parameters: 1) total available I/O bandwidth 2) total available disk storage space 3) total available buffer space, and 4) total available tertiary storage space. These system resources, as well as the data layout and scheduling schemes, determine the cost of the server and the QoS it can offer to the viewers. One useful measure of the QoS is the *latency* induced in servicing a user request. The latency for servicing a request is defined as the time between the viewer's request to the time the reading of the requested object from the disk is initiated. We assume that the additional delay until data is displayed to the viewer is relatively negligible. Latency arises because there is insufficient bandwidth for servicing the viewers' requests and insufficient buffer space for scheduling the reading of video objects from the disks. Besides, as mentioned before, to access non-disk resident video objects, they must be fetched from the tertiary store to the disk, therefore latency can also be attributed to insufficient disk storage. In general, the more resources we have, the lower will be the average latency for servicing a user request. Yet more resources simply means a larger cost. Therefore, in order to design a cost-effective VOD server, it is very important to carefully manage the system resources such that the number of viewers serviced can be increased. Among the four system resources we described above, we will focus our study on the total available I/O bandwidth and buffer space.

1.2 Problem Definition

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 [DSS94] method in which one I/O stream is dedicated to servicing several viewers who have arrived within a predefined short interval. As long as viewers are willing to wait for a small amount of time, all these requests to the same movie can be serviced using one I/O stream and thereby saving many I/O resources. Another technique for reducing the amount of I/O resources needed is the buffering method [RZ95], where viewers can fetch the data blocks from the system buffers instead of from the disk subsystem. Finally, adaptive piggybacking method [GLM96] allows viewers to have different displaying speeds so that eventually, they can catch up with each other and thereby reduce the I/O resource requirements. The basic idea behind these approaches is to group up the users so as to increase the degree of data sharing. All the approaches mentioned above are orthogonal and they can be implemented in a VOD system to reduce the total I/O resource requirement.

Apart from normal playback, it is also desirable to provide interactive VCR functions like fast forward with viewing (FF), rewind with viewing (RW) and pause/resume (PAU). There are various research results on supporting interactive VCR functions for the VOD system. In [KSKT94], the authors propose two queuing networks to model the amount of reserved resources for servicing VCR requests with an associated statistical guarantee on a quality-of-service metric: one is to delay the service while the other is to provide the service with a lower quality, in case there is insufficient bandwidth. In [CKLV95], the authors propose a *selected access scheme*, combined with data placement strategies for the server to dynamically access desired video frames from a shared storage device and this can be applied to support VCR-like functions. In [CKY95], the authors propose methods to support variable rate browsing and minimize the additional resources required. This is achieved by considering both the storage and retrieval for video data. The overall approach comprises storage methods, sampling and placement methods and playout methods. In [LL96], the authors propose the use of sub-band coding techniques to store video objects such that VCR functionality can be provided at multi-resolution levels with no additional load on system resources. However, relatively few research has addressed the problem of incorporating VCR functions in a system using data sharing techniques in servicing normal playback. The problem of allowing users in a batch to pause and eventually resume has been investigated in [YWS95] by making use of a look-ahead scheduling with look-aside buffering. Look-ahead scheduling backs up the viewers with a (look-ahead)

stream currently being used for another showing but is close to completion, consequently allowing the viewer to pause and resume at any time. Before the look-ahead stream becomes available, the pause and resume feature is supported by the original stream through (look-aside) buffering of the missed content. In [DSST94], the authors propose to reserve some resources ahead (known as contingency channels) so as to provide short response time to resume requests from VCR functions. An analytical model that predicts the reneging probability and expected resume delay is developed and it is applied to allocate channels for batching, on-demand playback and contingency. In [LL97], the authors propose a protocol which splits an interactive user from the batch, whom will be serviced by dedicated video streams known as **I** streams. These individuals are merged back to the batching streams using *synch buffer*, which is used to bridge the gap between a batching stream and the **I** stream.

That is to say, although data sharing techniques such as batching, buffering and adaptive piggybacking can service requests for popular movies in large bulks, they do not work very well in the presence of interactive VCR functions because the behavior of viewers in issuing and resuming from VCR requests is unpredictable. For example, in batching, users arriving within a certain interval are grouped together to be serviced by the same I/O stream. However, in the presence of interactive VCR functions, the benefit of data sharing is lost because when the viewers resume to normal playback, there may not be any existing batch of viewers that they can join in to share the same I/O stream to continue the normal playback of the movie. These viewers are said to *fall out* of a batch and are referred to as *fallen out* viewers. As a result, the demand for system resources required can be significantly increased [DVV94] since additional resources are required to service the normal playback of these fallen out viewers in order not to sacrifice the quality of service. Therefore an efficient scheme is needed in order to incorporate VCR functionality.

Consequently, we address the problem of system sizing in designing a VOD system, when VCR functionalities are provided in conjunction with data sharing techniques like batching and buffering. Efficient resource management and proper system sizing is crucial for designing a cost-effective VOD server. We divide the overall resources into those required in normal playback and those for servicing VCR requests. Notice that the allocation of additional resources for servicing VCR requests is inevitable since in the case of FF and RW, the VCR version of the movie has to be displayed to the viewer. However, if the chance of viewers falling out of a batch after the VCR requests can be minimized, these allocated additional resources can be released at once upon resume and consequently, the viewers will not be consuming additional resources for

their normal playback. Based on this idea, we calculate the resources required for normal playback by satisfying the following requirement: when batching and buffering techniques are both used, the amount of memory buffers and I/O resources to be preallocated for normal playback should ensure that after a VCR function, there is a certain probability, say, P^*, that the additional resources (which are dedicated to a VCR function) can be released. If we can increase the probability of releasing these resources when resuming to normal playback, then we can use these resources to admit more users, thereby making the VOD system more cost-effective. To calculate the amount of resources needed for servicing VCR requests, we propose a scheme to facilitate the fallen out viewers to join in the batches again. Therefore, even though the viewer may have to hold on to the additional resources for normal playback, we try to reduce the time for these fallen out viewers to release the resources. In this way, we can reduce the time that the VCR resources will be used for normal playback instead of for servicing VCR requests, which implies that the amount of reserved VCR resources can be reduced. This is achieved by incorporating the techniques of adaptive piggybacking and buffering, where the displaying speeds of the fallen out viewers are altered such that they can join in the existing batches in front or behind. Based on this proposed scheme, we calculate the amount of VCR resources needed.

The contributions of our work are as follows.

- In calculating the resources required for normal playback, we introduce a model for determining the amount of resources required for supporting normal playback while satisfying predefined performance characteristics. The amount of resources needed for normal playback is calculated by taking into consideration the VCR functionality to be provided to the viewers.

- Our proposed model allows us to maximize the benefits of data sharing techniques, such as batching and buffering, in the presence of interactive VCR functions. In other words, given a certain amount of system buffer, and I/O bandwidth resources, we use our mathematical model to determine the distribution of these resources among the popular movies such that normal playback and VCR functionality can be provided in a satisfactory manner.

- The mathematical model is general in the sense that when deriving the amount of resources needed for normal playback, we do not assume any particular distribution for modeling fast forward, rewind, and pause in order to guarantee the probability to be at least P^*. In calculating the amount of VCR resources, we try to follow the same technique in [KSKT94] to model these resources by the

application of an $M/M/m$ queue. Although we have to assume the VCR duration is of exponential distribution, our normal playback model does not make any assumption on the distribution.

- Besides, we propose an effective scheme to restore the benefits of data sharing that is lost resulted from the fallen out viewers. This is achieved by trying to batch these fallen out viewers with the existing batches again and our proposed scheme is efficient in significantly reducing the time to accomplish this task.

- It is worth pointing out that, by ensuring a larger P^*, (which means whenever a viewer resumes, there is a greater probability that he will be able to join back to any of the existing batches), less resources are needed to be reserved for servicing VCR requests. However, this will not necessarily result in the reduction of the overall amount of system resources required because we may need to reserve a large amount of resources for normal playback in order to ensure such a probability. Without an accurate model and a good system sizing method, we will not be able to calculate the right amount of resources needed. Therefore, another important contribution of our work is the application of the mathematical model we proposed, combined with our proposed scheme to handle fallen out viewers, in calculating the amount of resources for normal playback and servicing VCR requests respectively.

The remainder of the thesis is organized as follows. In Chapter 2, we present a review on some of the existing techniques in reducing the I/O demand of the user requests and how to support interactive VCR functionalities. In Chapter 3, we present our system model for normal playback and servicing VCR requests. The mathematical model for calculating the resources required for normal playback, while ensuring a probability to deallocate the additional resources required for servicing VCR requests, is presented in Chapter 4. In Chapter 5, we present the approach in calculating the resources needed for servicing VCR request and introduce our proposed scheme on facilitating the fallen out viewers to join in the batches. In Chapter 6, we describe how to apply our research results in solving system sizing problems in a VOD system. Lastly, our conclusions are given in Chapter 7.

Chapter 2

Related Work

There has been significant amount of research work dedicated to designing cost-effective VOD servers. In this chapter, we present some of the existing research results that are related to our work. These results are separated into two categories, namely the data sharing techniques and existing techniques on how to incorporate VCR functionality into the VOD system.

2.1 Data Sharing Techniques

There are several approaches on how to reduce the I/O demand on the storage server through data sharing, or, in effect, increasing the number of concurrent viewers. These techniques are orthogonal, in the sense that they can be implemented together to save the I/O resources in a VOD system.

2.1.1 Batching

The most simplest way to satisfy independent playback requests is to dedicate an I/O stream to each request, yet this will require a very large number of I/O streams. For popular movies, we can anticipate that there will be a large number of viewers, therefore if the playback requests are separated by small time intervals, they can be grouped together by using one single I/O stream. Batching is an I/O scheduling policy where the initiation of an I/O stream is delayed so as to group up the normal playback requests of the same movie. As long as the viewers are willing to wait for a certain period of time, batching can help to achieve a substantial reduction in the required server capacity. However, this reduction in I/O streams will increase the average

waiting time of the viewers, thus there is a trade off between the I/O bandwidth utilization and the average waiting time of viewers.

One way to batch the playback requests is to start the movie when the number of waiting viewers has reached a certain level, which is known as *batching by size.* Yet this may not be fair since there is no guarantee that the viewer who have been waiting the longest is serviced. Another way will be to start the movie periodically at predetermined intervals, known as *batching by timeout.* The use of batching is appropriate only for popular movies as it will incur unnecessary latencies for non-popular movies. Figure 2.1 shows a scenario of batching by timeout, where an I/O stream is initiated to service a batch of viewers after a fix period of time.

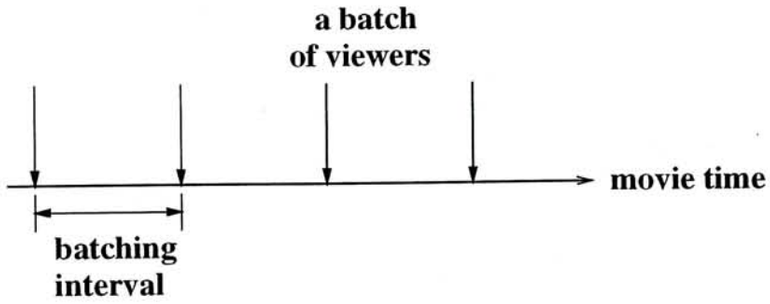


Figure 2.1: Batching by timeout

By exploiting the fact that modern communication networks such as ATM are equipped with a multicast facility[1], in [DSS94], the authors propose the batching of clients arriving within a short time duration to be serviced using a single I/O stream. Generally, increasing the batching interval reduces the resources required but at the same time increases the average waiting time of the viewers. Thus, it may result in the case that a viewer may decide to leave the system (renege) if he does not receive service within some amount of time. Various scheduling policies are also proposed for selecting which movie to multicast. The objective of these policies is to reduce the average waiting time for starting the service for a normal playback request, to reduce the reneging probability as well as to increase fairness over requests for all movie types, i.e. the reneging probability is defined to be equal for all movie requests. The two orthogonal classes of policies being studied are FCFS, which schedules the movie with the longest outstanding request, and MQL (maximum queue length), which chooses the movie with the maximum number of outstanding requests.

[1] the same message can be sent to multiple clients without causing any extra overhead to the server

2.1.2 Buffering

Buffering is a technique which tries to bridge the gap between close successive requests using buffer. This is achieved by retaining the movie frames fetched by the previous stream in the buffer space and subsequent streams will read the frames from the buffer rather than using another I/O stream. The basic idea is to use buffer space to promote sharing and hence reducing I/O bandwidth utilization. Since even retaining a few minutes of the video in the buffer is costly, it is very important to carefully decide what and when data should be retained in the buffer. Figure 2.2 shows a scenario where the frames fetched by the leading stream is retained in the buffer and is read by subsequent streams.

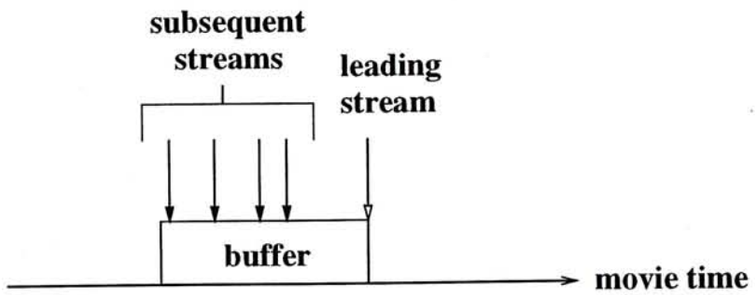


Figure 2.2: Buffering technique

There are several existing research results on using buffering techniques to reduce disk utilization and increase the capacity of the video server. In [DS94, KRT94, MN95, DS96], 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, the number of concurrent viewers can be increased. They propose buffer sharing schemes which help to determine when and where to retain the data of each stream in the buffer for use by subsequent streams. Besides, in [DDM+95], the cost performance tradeoff of various buffering and caching strategies is studied by varying the buffer size, disk utilization and the disk characteristics on the overall capacity of the system. In [ORS95], the authors propose novel demand paging algorithms that guarantee a given transfer rate as well as utilize the limited buffer space effectively and eliminate the disk bandwidth limitation. In [NY96], the authors show that buffer sharing can lead to as much as 50% reduction in total buffer requirement. Prefetching strategies are also proposed to maximize the effective use of buffer space and disk bandwidth.

2.1.3 Static Partitioning

Static Partitioning is one of the partitioned buffer management strategies proposed in [RZ95], which combines the use of batching and buffering and is developed based on a buffer refreshing process. This scheduling method is similar to batching, in which the move is restarted periodically at predefined intervals. Thus, it allows the sharing of data between requests such that requests arriving during a batching interval can be serviced using a single I/O stream [DSS94]. 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 as the *viewer enrollment window.* Figure 2.3 illustrates the scenario for the static partitioning scheme, where each batch of viewers is serviced using an I/O stream and a partition of buffer. Viewers who come before the closing of the viewer enrollment window[2] 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 consecutive I/O streams) minus the length of the viewer enrollment window.

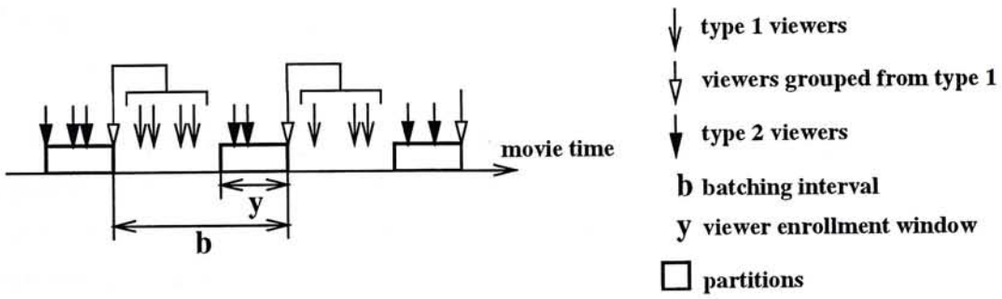


Figure 2.3: Different types of viewers in static partitioning.

2.1.4 Adaptive Piggybacking

Adaptive Piggybacking allows the viewers to have different display rates so that eventually viewers with their respective I/O streams can be merged together to share a single I/O stream. Notice that the display adjustment is done gradually and slow enough so that it is not noticeable to the viewers, therefore some time must elapse before the streams can merge. This may result in the fact that the reduction in I/O

[2]At this point, the frames in the partition are refreshed.

demand is not as high as that in batching, however, as the grouping of the viewers is done dynamically while the display is in progress, the viewers will not experience any latency.

This approach assumes that the storage server is capable of dynamically altering the display rate of a request; in other words, to *time compress* or *time expand* some portions of an object's display. As the display device displays at a fixed rate, the slow down in effective display rate can be done by adding additional frames. Similarly, the display rate can be increased by removing movie frames. It has been pointed that differences as small as $\pm 5\%$ in the display rates will not sacrifice the quality of display, thus is not perceivable by the viewers [GLM96]. The idea behind this approach is to dynamically merge two or more I/O streams into one. Before the merge, there are two I/O streams displaying the same video, and are separated by a small number of frames. In order to close the temporal gap between these two streams, the display rate of the leading stream is decreased while the display rate of the trailing stream is increased. Eventually, the faster trailing stream will be able to catch up with the slower leading stream. At this point, the streams are adaptively piggybacked, thereby merging two I/O streams into one.

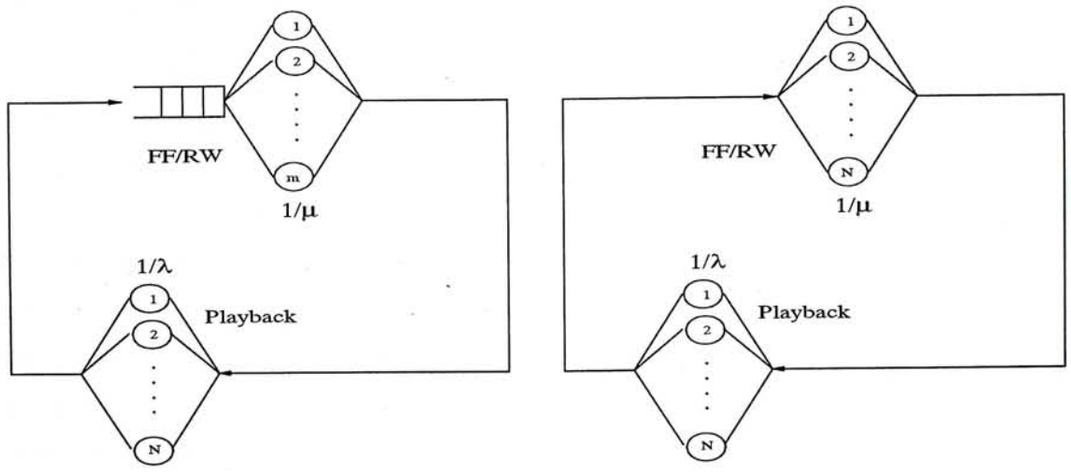
The altered stream of frames can be provided to the viewers by on-line creation of an altered version of the movie, yet it may require complicated scheduling and/or additional buffer storage and specialized hardware. Another way is to create it off-line and store on disk with the original version, yet it requires additional disk storage.

Adaptive piggybacking is first proposed in [GLM96], where three basic types of piggybacking policies, in conjunction with batching policies, are investigated and evaluated with respect to the reduction in I/O bandwidth utilization. The results indicate convincingly that small variations in the delivery rate can enable enough merging of I/O streams that significant reduction of I/O bandwidth is realized. Besides, in [AWY96], based on the simple merging policy proposed in [GLM96], a generalization and optimal variant of which is developed. Another new policy is also proposed which is seen to be optimal over a large class of any known piggybacking policies. In [LLG97a], the authors consider two possible versions of the stream merging problem, namely the *static* version and the *dynamic* version. The former one is formalized and the upper bound on the I/O demand of which is derived. The authors also propose efficient heuristic algorithms for both static and dynamic versions of the stream merging problem.

2.2 Providing VCR Functionalities

To provide on-demand VCR-like user interactions, one approach is to have a dedicated I/O stream for each user. However, this is not cost-effective as in commercial environments, there may be a lot of users, thus requiring a large amount of resources. Not much work has been done in investigating the support of interactive VCR functionalities in a video-on-demand system efficiently. We describe several of these below.

In [KSKT94], the authors propose and analyze the provision of FF/RW capabilities with an associated statistical QoS guarantee by proposing two schemes which provide immediate access to full-resolution FF/RW bandwidth with high probability. When there is not enough bandwidth for servicing a FF/RW request, the service is delayed and this scheme is called the Delay Scheme (DS). Another possible way is to provide the service immediately but with a loss in resolution, referred to as the Loss Scheme (LS). In order to compute the maximum number of users that the system can support while maintaining a certain performance level, they model the video server as a closed queuing network with 2 queues and N customers, where Figure 2.4a and Figure 2.4b represents DS and LS respectively. The lower queue represents the playback queue while the upper queue represents a user either queuing or receiving FF/RW service. The service time in the lower queue and the upper queue is assumed to be exponentially distributed with mean $1/\lambda$ and $1/\mu$ respectively. The authors have shown that their



(a) Delay Scheme (DS) (b) Loss Scheme (LS)

Figure 2.4: Video server models for the two schemes

approach to support VCR functionalities with statistical guarantees on the QoS metrics results in a significant increase in the number of supportable viewers, when compared to systems in which FF/RW bandwidth is statically reserved for each viewer. Moreover, a

playback-only server can be extended to provide this service by reserving only a small portion of its total bandwidth, which is dynamically shared among FF/RW requests.

In [DSST94], the authors pointed out that the technique of batching complicates the provision of VCR functionality, because a new stream needs to be started when the client resumes. The authors consider reserving a pool of resources known as contingency channels, for providing short response time to resume requests. The authors first develop an analytical model that predicts the reneging probability and expected resume delay, and this model is used to optimally allocate resources for batching, on-demand playback, and contingency.

In [OBRS94], the authors propose a low-cost storage architecture for a MOD server, which satisfies multiple concurrent requests for the retrieval of data. A novel storage allocation scheme is proposed that can support multiple different portions of a movie to be concurrently retrieved from the disk. This is achieved by dividing a movie into p phases, the duration of which is determined by the bandwidth requirements of the movie as well as the bandwidth capacity of the disk or disk array which stores that movie. The p phases of the movie are concurrently supported by p concurrent I/O streams. With the proposed scheme, VCR functions can be easily provided. A wide range of schemes are presented for implementing the VCR functions. The authors point out that the quality of MOD services (e.g. frame rate of fast forward as compared to normal display, delay incurred in resuming from a pause) can be improved by the allocation of additional buffers for each viewer.

In [YWS95], the authors address the problem of pause/resume functionality in a system where batching is used to conserve resources. They propose a mechanism called look-ahead scheduling with look-aside buffering. The idea is to back up each viewer with a (look-ahead) stream that is currently being used for another showing that is near completion, rather than backing up with a real stream capacity. The look-aside buffer is used to buffer the missed contents so as to support the pause and resume features before the look-ahead stream becomes available. Although this scheme can provide a substantial improvement in throughput, this is accomplished by making use of a very large buffer pool. Therefore, whether it is more cost-effective to purchase additional disks or increasing the buffer pool need to be investigated and it certainly depends on the relative cost of buffer and I/O streams.

In [CKY95], the authors point out that the support of interactive VCR functions will impose additional resources requirements on the VOD system. They propose methods to support variable rate browsing and minimize the additional resources re-

quired. This is achieved by considering both the storage and retrieval methods for video data. The overall approach adopted comprises 1) a storage method, 2) sampling and placement methods, and 3) a playout method, in which the sampling and placement methods are two alternatives for video-segment selection. The segment-sampling scheme supports browsing at any desired speed and balances the load on the disk array while the segment-placement scheme supports completely uniform segment sampling across the disk array for specific speed-up rates.

The Split and Merge (SAM) Protocol is proposed in [LL97] to address the problem of fallen out viewers when batching is used. This protocol allows multiple users to transparently share the same video stream, as if each user has his own dedicated I/O stream. The I/O streams are divided into **S** streams and **I** streams; the former is for servicing the normal playback of viewers and the latter is to service the users' interactive VCR requests. Initially, the viewers are batched together and serviced by an **S** stream. The basic idea behind is to allocate a dedicated I/O stream (**I** stream) for the viewer issuing the VCR request, thereby splitting him from the batch. After these individuals resume, they are merged back to the batching streams by using *synch buffer*. The synch buffer is filled up with movie frames by the nearest **S** stream and it is used to bridge the gap between the **S** stream and the dedicated **I** stream. However, the authors do not address the problem that what the optimal amount of synch buffer is and how the amount will be affected by different arrival rates of VCR requests. Besides, how the resources should be distributed has not been discussed.

Chapter 3

System Model

Our system model is based on a partitioned buffer management strategy called static partitioning, as described in the previous chapter, which combines the use of batching and buffering. We adopt these methods to improve the scalability of VOD systems by servicing multiple user requests using a single I/O stream. Furthermore, the addition of buffering to batching reduces the average waiting time and the number of I/O streams as compared to using batching alone. Note that these data sharing techniques should only be applied to requests for *popular* movies. The use of batching technique for non-popular movies will incur unnecessary latencies while the use of buffering technique for non-popular movies will be a waste of buffering resources. In this section, we will first describe the requirements in providing VCR functionality, and then we present our system model for normal playback and servicing VCR requests.

3.1 Operations involved in VCR Control

The static partitioning scheduling method only deals with normal playback. However, VCR functions like fast-forward (FF), rewind (RW), pause (PAU) and resume are important and necessary features to be provided to the viewers. A fundamental problem in providing such interactive features in conjunction with the data sharing techniques is the need for additional resources when the viewer resumes to normal playback from the VCR mode and is no longer part of any group of requests sharing the I/O and buffer resources. In this thesis, we present a model which facilitates the calculation of distribution of resources so as to reduce the 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.

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 request other than the pause function, additional resources have to be allocated so that he can view the VCR-version of the movie. (The pause operation, on the other hand, does not require information to be displayed. So there may be a possibility that the resources associated with the paused stream can be released to service other requests, and allocate resources again when the user resumes to normal playback). As described in section 2, two queuing networks have been proposed to model the amount of reserved resources for VCR requests in [KSKT94]. We denote the mean time spent in this phase as μ_{VCR}.

- *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 at what time and which position 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 and we denote the probability of resuming at any of the existing partitions as P^*. On the other hand, if the viewer cannot resume at any of the existing partitions, to continue his normal playback without any delay, the viewer has to make use of the resources allocated in phase 1 until he can join a partition.

  We define that a viewer is said to have a *hit* if he can resume at any existing partition 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 3.1.

  The utilization of VCR resources after resume from VCR requests is not desirable because the VCR resources are now used to service normal playback rather than VCR requests. Here we denote the mean time spent in this phase, that is, the time elapsed before the resources allocated in phase 1 can be released, as μ_{hold}. As mentioned before, the allocation of resources for normal playback ensures that there is a minimum probability of P^* for the viewers to resume at any of the existing partitions after the VCR requests. Therefore, there is only a probability

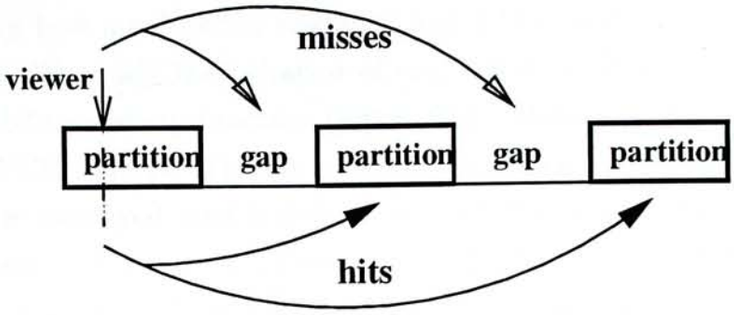


Figure 3.1: Hits and misses for viewers resuming from VCR requests.

of $(1 - P^*)$ that the viewer has to hold on to the additional resources when he resumes. Thus, the expected time spent in phase 2 is given by $(1 - P^*)\mu_{hold}$.

3.2 Normal Playback Model

As mentioned before, the allocation of resources in phase 1 for servicing a VCR request 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 guarantee a certain probability of releasing the allocated resources at a resume point so as to reduce the consumption of reserved resources. Thus, more resources are available for servicing future requests, such as forthcoming VCR requests or even be dedicated to service 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 thesis, 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. Therefore, our normal playback model is derived based on the static partitioning model, to which the resources allocated are calculated 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 guarantee the probability of a hit is at a certain level, we also consider the behavior of VCR requests in determining the *configuration* of the system,

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

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 movie 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.

Therefore, 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 controlled. This resource allocation is calculated based on the mathematical model we propose, which will be presented in Chapter 4. In this model, we assume that the arrivals of normal playback requests for a popular movie 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. The basic idea is as follows. If we could maximize the hit probability, then we could reduce the additional load resulting from 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. Intuitively, increasing the size of the partitions will increase the hit probability because the fraction of movie in the buffer can be increased. Consider the extreme case that the entire movie is buffered, thus increasing the hit probability to 1. However, this may not yield a system with the lowest cost because considerably less time will be spent in a VCR request and buffering the whole movie will be too costly. Therefore, we try to find out a balance between I/O streams and buffer space which will yield the lowest cost for supporting the same number of concurrent viewers for both normal playback and VCR functions.

3.3 VCR Model

It is important to point out that, although relatively little time is spent in VCR modes as compared with normal playback [KSKT94], inefficient schemes for providing VCR functionality can easily result in consumption of large amounts of system resources and diminish the benefits of the data sharing techniques employed. Besides, the modeling

Notation	Definition
m	no. of I/O streams for servicing VCR requests
λ_v	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

Table 3.1: Major notations involved in VCR control

of VCR resources is a problem because the behavior of viewers in doing VCR functions is completely unpredictable. That is, we cannot determine when the viewers will issue VCR requests, how long the VCR request will last, when they will resume to normal playback and so on. Nevertheless, we adopt the same approach as in [KSKT94], that is, we model the additional I/O resources by an $M/M/m$ queue, so as to provide VCR service with a predefined level of QoS guarantee. For example, we ensure that the average time that a viewer has to wait for switching to VCR mode is within 3 seconds. The details of these QoS will be presented at the end of this section. The notations involving VCR controls are summarized in Table 3.1. 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 average a viewer issues z VCR requests per movie of length l minutes, we have:

$$\lambda_{VCR} = \frac{z}{l} \tag{3.1}$$

As we have also assumed that the arrival of normal playback follows the Poisson distribution of rate λ, the total arrival rate of VCR requests in the system, λ_v, is given by $\lambda l \lambda_{VCR}$. Using Equation (3.1), we have:

$$\lambda_v = \lambda z \tag{3.2}$$

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 they can join in the partitions, and it is assumed to be exponentially distributed with mean μ_v [KSKT94]. As mentioned in Section 3.1, a VCR function is considered as a two-phase process. Therefore, μ_v is calculated by summing up the mean time spent in each phase:

$$\mu_v = \mu_{VCR} + (1 - P^*)\mu_{hold} \tag{3.3}$$

In order to simplify the calculation of the average time for releasing the hold on resources, μ_{hold}, we further assume that each position of the movie bears the same probability to be the point of resume. We obtain the probability of queuing, $P(queueing)$,

the expected waiting time for VCR service, T_q, and the mean queue length, N_q, of a $M/M/m$ queue as follows, where $\rho = \frac{\lambda_v \mu_v}{m}$.

$$P(queueing) = \frac{(\frac{(m\rho)^m}{m!})(\frac{1}{1-\rho})}{[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + (\frac{(m\rho)^m}{m!})(\frac{1}{1-\rho})]} \quad (3.4)$$

$$N_q = \frac{\rho}{1-\rho} P(queueing) \quad (3.5)$$

$$T_q = \frac{N_q}{\lambda_v} \quad (3.6)$$

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 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 movie or for the normal playback to resume respectively. In other words, when providing VCR functionality, one of the following performance level has to be satisfied.

- 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.
- Even the viewer has to wait, we ensure that the waiting time is within the tolerance level of viewers, t^*.
- The average number of people waiting for their VCR requests to be serviced is l^*.

Referring to Equations (3.4)- (3.6), the above three performance requirements can be formulated as the following equations:

$$P(queueing) \leq \epsilon \quad (3.7)$$

$$T_q \leq t^* \quad (3.8)$$

$$N_q \leq l^* \quad (3.9)$$

These three equations will ensure the system to provide the service at a predefined performance level. However, as the queue length will not be noticeable by the viewers, therefore, in our calculation, we only use the first and second criteria for our performance level.

Notice that $\rho = \frac{\lambda_v \mu_v}{m}$ and must be smaller than 1, therefore the smallest value of m possible is $\lfloor \lambda_v \mu_v \rfloor + 1$. To find out the number of servers needed to ensure the performance level, the value of m is incremented until either the designated inequality defined above is satisfied.

Chapter 4

Resource Allocation for Normal Playback

In this chapter, we present the derivation of our mathematical model for determining the configuration of the system for normal playback. 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.

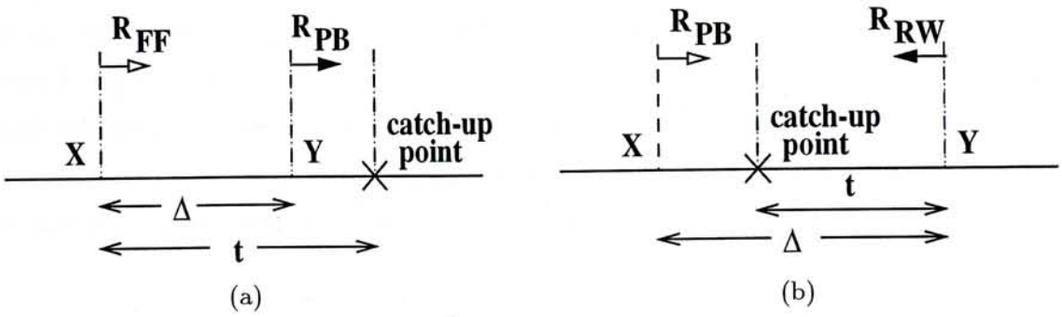


Figure 4.1: scenarios for catching up to viewers a) in front b) behind

Before presenting the mathematical model, let us present the criteria for *catching up* to a stream. An example situation is illustrated in Figure 4.1(a). We denote R_{FF} *sec/sec* as the rate of fast forward and R_{PB} *sec/sec* 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, denoted by a cross in the figure. Similarly, Figure 4.1(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

Notation	Definition
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

Table 4.1: Major notations involved in normal playback model

viewer Y must rewind) before a catch-up is accomplished is as follows.

$$t = \begin{cases} \alpha \cdot \Delta, & \text{where } \alpha = \frac{R_{FF}}{R_{FF}-R_{PB}} \text{ for FF,} \\ \gamma \cdot \Delta, & \text{where } \gamma = \frac{R_{RW}}{R_{PB}+R_{RW}} \text{ for RW.} \end{cases} \tag{4.1}$$

In the remaining chapter, 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, which will be used for system sizing. This probability is derived by first considering the conditional probability of hit for a viewer at a particular point in the movie, which will be unconditioned to obtain the general probability of hit for any type of VCR request. The major notations used for the derivation of our mathematical model for modeling normal playback resources are summarized in Table 4.1.

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 $\frac{l}{n}$ 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, 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 [RZ95]. 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:

$$w = \frac{l - B}{n}, \quad \text{where} \quad n = 1, 2, \ldots, \frac{l}{w} \tag{4.2}$$

It is important to note that $n = \frac{l}{w}$ would imply that $B = 0$, which corresponds to the pure I/O case, where each batch is serviced 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:

$$n = \frac{l}{w} - \frac{B}{w}, \quad \text{where} \quad B \leq l \tag{4.3}$$

The difference between using buffering in conjunction with batching and pure I/O lies in the amount of I/O and buffer resources required. When we dedicate B minutes worth of buffer space for normal playback, with reference to Equation (4.3), we can see that 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 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 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 4.2. 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 (pdf) of the duration of a FF request. The

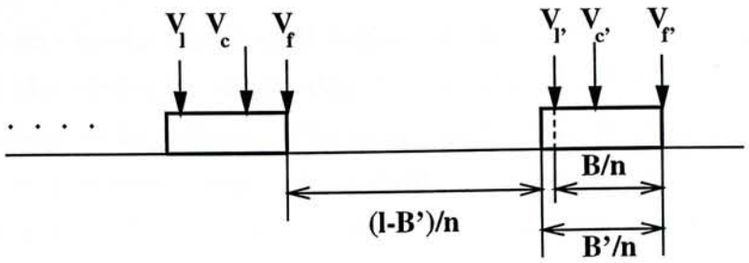
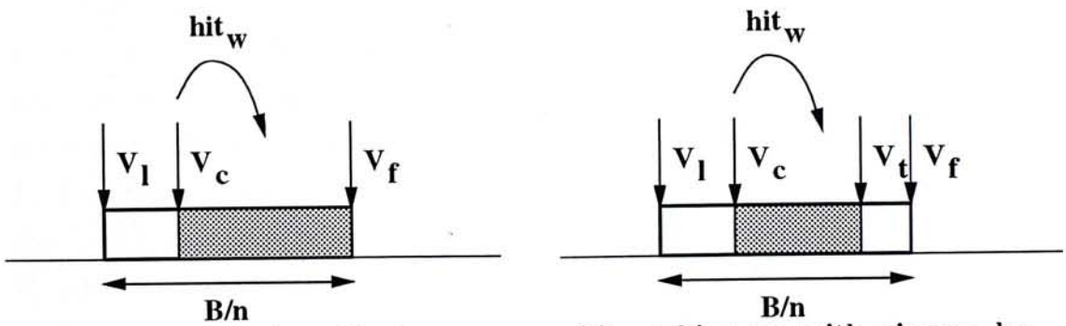


Figure 4.2: relative positions of viewers at V_c,V_f, V_l in a partition

main difficulty in handling VCR requests lies in their inherently unpredictable nature [KSKT94], 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)



(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 4.3: 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 at V_c, and the position of the first possible viewer in his partition. This situation is illustrated in Figure 4.3(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. The upper and lower limit denotes the duration of movie time that the viewer can fast forward so as to resume in the shaded region, where the upper limit is the time to enable a catch-up with the first possible viewer, V_f, as defined in Equation (4.1).

$$P(hit_w|FF, V_c, V_f) = \int_0^{\alpha\Delta_w} f(x)dx \quad \text{where} \quad \Delta_w = V_f - V_c \tag{4.4}$$

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 Equation (4.1), 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: *viewer at position V_c 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:

$$P_a(hit_w|FF, V_c) = \int_{V_f=V_c}^{V_c+\frac{B}{n}} P(hit_w|FF, V_c, V_f)P(V_f)dV_f \tag{4.5}$$

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 4.2.

case b: *viewer at position 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 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 with the latter, both viewers are just at the end of the movie. Using Equation (4.1), 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 limit for V_t:

$$V_t \leq \frac{l + (\alpha - 1)V_c}{\alpha} \tag{4.6}$$

Note that the viewer at position V_c will not be able to catch up with a viewer at position V_f if $V_f > V_t$ 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 Equation (4.6). Figure 4.3(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 fast forward for a duration between 0 and $\alpha(V_t - V_c)$.

$$P_b(hit_w|FF, V_c) = \int_{V_f=V_c}^{V_t} P(hit_w|FF, V_c, V_f)P(V_f)dV_f + \int_{V_f=V_t}^{V_c+\frac{B}{n}} \int_0^{\alpha(V_t-V_c)} f(x)\ dx P(V_f)dV_f \tag{4.7}$$

Using the same 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 Equation (4.6).)

$$P_a(hit_w|FF) = \int_{V_c=0}^{l-\frac{B\alpha}{n}} P_a(hit_w|FF, V_c)\ P(V_c)\ dV_c \tag{4.8}$$

  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.

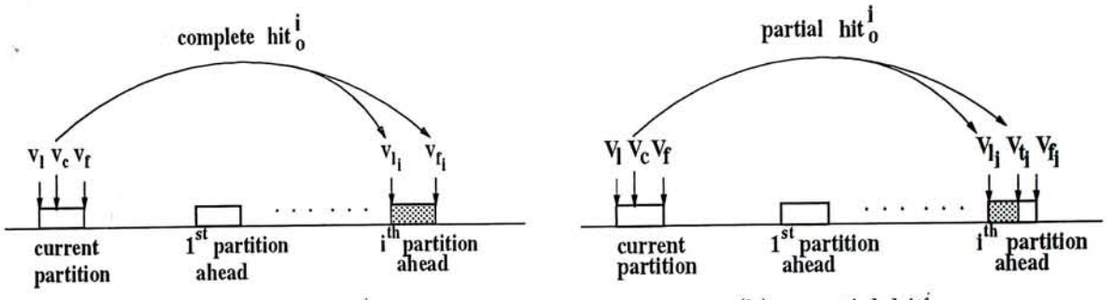
$$P_b(hit_w|FF) = \int_{V_c=l-\frac{B\alpha}{n}}^{l} P_b(hit_w|FF, V_c)\ P(V_c)\ dV_c \tag{4.9}$$

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 Equations (4.8) and (4.9).

4.1.2 Jump to other partitions (hit_o)

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_o. As illustrated in Figure 4.4, 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_o can be divided into two

cases, namely, the *complete* hit_o^i and the *partial* hit_o^i. As illustrated in Figure 4.4(a), a complete hit_o^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 4.4(b) illustrates the event of a partial hit_o^i, which illustrates a viewer not able to catch up with the viewer at position V_{f_i} before the movie ends.



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

Figure 4.4: A hit in another partitions (hit_o).

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_o 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_o at the i^{th} partition ahead, denoted by $P(hit_o^i)$, is shown below :

case 1: *able to catch up with both viewers at position* V_{l_i} *and* V_{f_i} *(a complete* hit_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 i^{th} partition ahead:

$$P_c(hit_o^i | FF, V_c, V_f) = \int_{\alpha\Delta_{jump_l}}^{\alpha\Delta_{jump_f}} f(x)dx \tag{4.10}$$

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

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 the viewers in the i^{th} partition, as long as he is able to catch up with the viewer at position V_{l_i}, we will classify this event as a partial hit_o. 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 $V_{t_i} = V_t + \frac{il}{n}$, resulting in $V_t = \frac{l+(\alpha-1)V_c-\frac{il\alpha}{n}}{\alpha}$:

$$\begin{aligned} P_p(hit_o^i|FF, V_c, V_f) &= \int_{\alpha\Delta_{jump_l}}^{\alpha(\frac{il}{n}+V_t-V_c)} f(x)dx \\ &= \int_{\alpha\Delta_{jump_l}}^{l-V_c} f(x)dx \end{aligned} \tag{4.11}$$

- Unconditioning on V_f

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

$$P_1(hit_o^i|FF, V_c) = \int_{V_f=V_c}^{V_c+\frac{B}{n}} P_c(hit_o^i|FF, V_c, V_f)P(V_f)dV_f \tag{4.12}$$

$$P_2(hit_o^i|FF, V_c) = \int_{V_f=V_c}^{V_t} P_c(hit_o^i|FF, V_c,\ V_f)P(V_f)dV_f \tag{4.13}$$

  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, Equation (4.13) 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 Equation (4.14) below. Equation (4.15) 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 $V_{t'} = \frac{l+(\alpha-1)V_c-\alpha(\frac{il-B}{n})}{\alpha}$.

$$P_3(hit_o^i|FF, V_c) = \int_{V_f=V_t}^{V_c+\frac{B}{n}} P_p(hit_o^i|FF, V_c, V_f)P(V_f)dV_f \tag{4.14}$$

$$P_4(hit_o^i|FF, V_c) = \int_{V_f=V_c}^{V_{t'}} P_p(hit_o^i|FF, V_c, V_f)P(V_f)dV_f \tag{4.15}$$

- 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.

$$P_1(hit_o^i|FF) = \int_0^{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}} P_1(hit_o^i|FF, V_c)\ P(V_c)\ dV_c \tag{4.16}$$

$$P_2(hit_o^i|FF) = \int_{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}}^{l-\frac{il\alpha}{n}} P_2(hit_o^i|FF, V_c)\ P(V_c)\ dV_c \tag{4.17}$$

$$P_3(hit_o^i|FF) = \int_{l-\frac{B\alpha}{n}-\frac{il\alpha}{n}}^{l-\frac{il\alpha}{n}} P_3(hit_o^i|FF, V_c)\ P(V_c)\ dV_c \tag{4.18}$$

$$P_4(hit_o^i|FF) = \int_{l-\frac{il\alpha}{n}}^{l-\frac{(il-B)\alpha}{n}} P_4(hit_o^i|FF, V_c)\ P(V_c)\ dV_c \tag{4.19}$$

Hence, the probability of resuming at the i^{th} partition ahead for a FF request, $P(hit_o^i|FF)$, is calculated by summing Equations (4.16) through (4.19).

The number of partitions to which a viewer at position V_c would be able to jump ahead, resulting in hit_o, 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 Equation (4.16), 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:

$$i \leq \lfloor \frac{n(l+w\alpha)-l\alpha}{l\alpha} \rfloor \tag{4.20}$$

4.1.3 Fast-forwarding to the end of a movie

Recall that our goal is to guarantee the probability of the resources, allocated for a VCR request, to be released upon resume to normal playback at a certain level. 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:

$$\begin{aligned} P(end) &= \int_0^l \int_{\alpha(V_t-V_c)}^{l} f(x)\ dx\ P(V_c)\ dV_c \\ &= \int_0^l \int_{l-V_c}^{l} f(x)\ dx\ \frac{1}{l}\ dV_c \end{aligned} \tag{4.21}$$

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

$$P(hit|FF) = P(hit_w|FF) + \sum_{i=1}^{\lfloor \frac{n(l+w\alpha)-l\alpha}{l\alpha} \rfloor} P(hit_o^i|FF) + P(end) \qquad (4.22)$$

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 is 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 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 $P(hit|FF)$. The detailed derivations can be found in Appendix A.

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}, 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 for a resume from a VCR request, $P(hit)$, which can be expressed as follows.

$$P(hit) = P(hit|FF)P_{FF} + P(hit|RW)P_{RW} + P(hit|PAU)P_{PAU} \qquad (4.23)$$

The quantity, $P(hit)$, is a function of seven system parameters. Specifically, $P(hit) = \xi(l, B, n, w, R_{FF}, R_{PB}, R_{RW})$, of which 5 of them are known. 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:

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

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

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, the handling of multiple movies is carried out in a similar manner.

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[1] 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. The use of skewed gamma distribution is to illustrate that our mathematical model does not assume any particular distribution.

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 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. 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 4.5(a) to 4.5(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 4.5d and 4.6.

These figures illustrate that the theoretical results match closely with the simulation

[1]The rationale for using a Poisson process was given in Chapter 3.

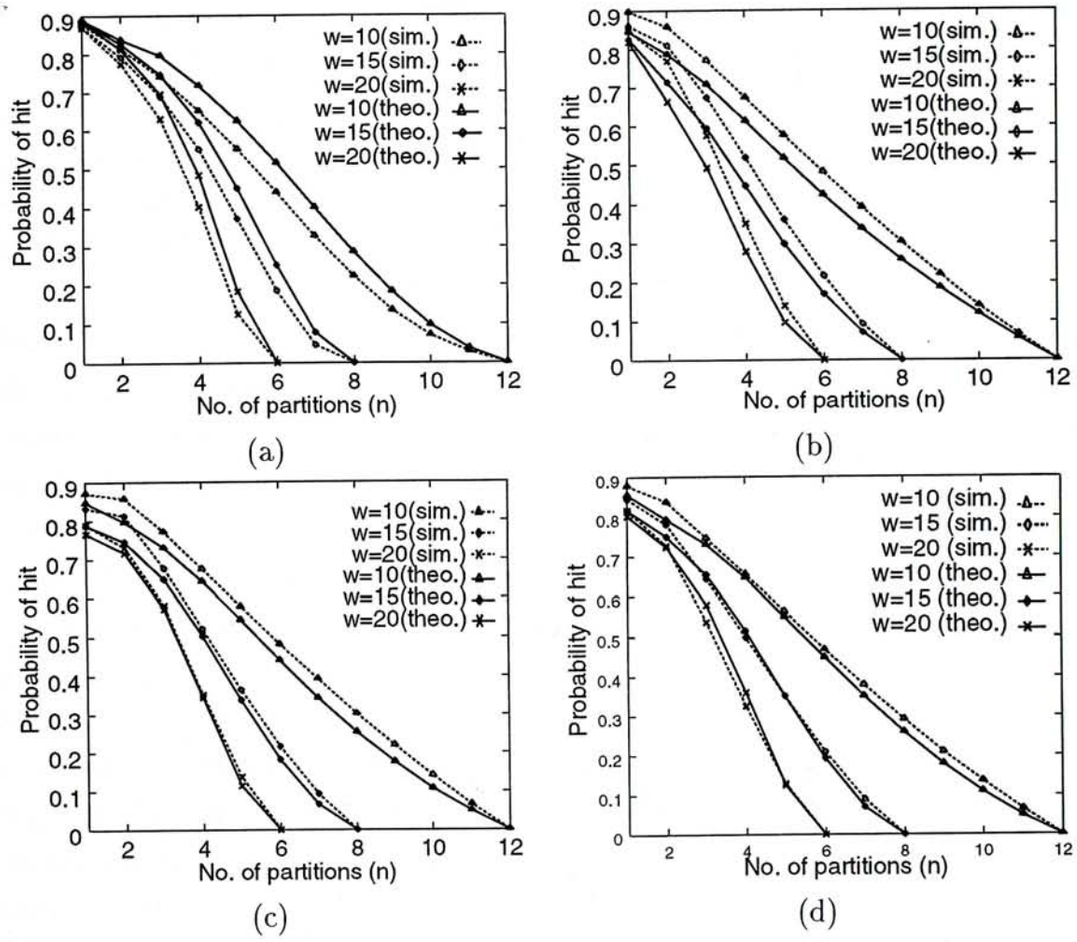


Figure 4.5: Simulation and theoretical results for normal playback and (a) only fast-forward VCR requests (b) only rewind VCR requests (c) only pause VCR requests (d) all kinds of VCR requests with $P_{FF} = 0.2, P_{RW} = 0.2$ and $P_{PAU} = 0.6$. Interarrival times are exponential and $1/\lambda = 2$ minutes; duration of VCR requests is drawn from a skewed gamma distribution with a mean = 8 minutes ($\alpha = 2, \gamma = 4$).

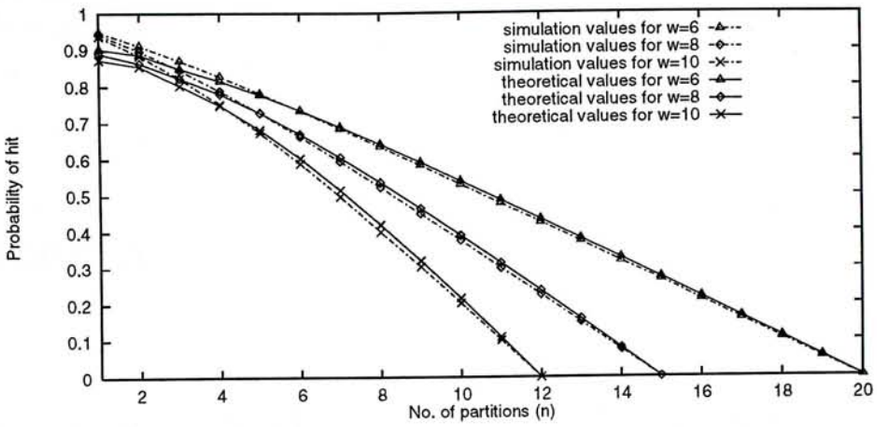


Figure 4.6: Simulation and theoretical results for requests drawn from an exponential distribution with mean = 5 minutes, and $P_{FF} = 0.4, P_{RW} = 0.3$ and $P_{PAU} = 0.3$. Poisson arrival $1/\lambda = 1$ minute

results, which verify 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 2.3) 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 4.5(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.

Chapter 5

Resource Allocation for VCR mode

In this chapter, we present the calculation of resources needed to handle VCR requests so as to satisfy the performance level defined by Equations (3.7) - (3.9). As mentioned in section 3.3, the additional I/O resources will be calculated by using an $M/M/m$ queuing model. Notice that ρ is defined by $\frac{\lambda_v \mu_v}{m}$ and it denotes the fraction of time the reserved I/O resources are busy. Therefore we aim to decrease the value of ρ. This can be achieved by the minimization of μ_v only: λ_v is out of our control because the behavior of the users in issuing VCR requests is inherently unpredictable; maximizing m is not favorable as well because it simply means more I/O resources are needed.

With reference to Equation (3.3), μ_v comprises two components. The first component cannot be minimized because it denotes the mean time a viewer will spend in the fast forward, rewind or pause function whenever he issues a VCR request, which depends solely on users' behavior. For the second component, it denotes the expected time a viewer spends in holding onto the additional resources for normal playback before he can join back to the 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 on how to handle VCR requests, we aim to implement a method which minimizes the value of μ_{hold}. In the remaining chapter, we will present two schemes in handling VCR requests, both of which will ensure that the resumption to normal playback from VCR requests without any delay, and show the derivations of the corresponding values of μ_{hold}.

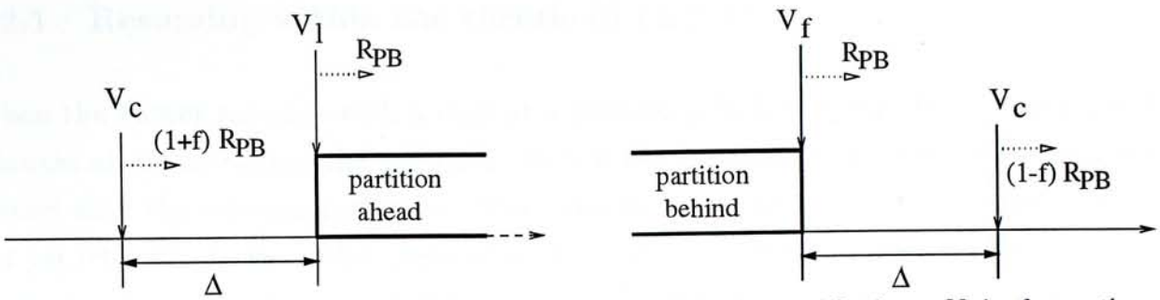
5.1 Scheme 1: No merging

In this scheme, 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 continue utilizing the VCR resources for normal playback after they resume, until they finish viewing the movie. For example, if the viewer resumes with a miss at the t^{th} minute of a movie of length l, the viewer has to hold on to the resource for $(l-t)$ minutes. Based on the assumption that each point of the movie has equal probability to be 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 $\mu_{hold} = \frac{l}{2}$ and this value also serves as an upper bound for μ_{hold}.

5.2 Scheme 2: Merging by adaptive piggybacking and buffering

In this scheme, we aim to reduce the value of μ_{hold} by applying the techniques of adaptive piggybacking and buffering. The adaptive piggybacking method [GLM96] allows different viewers to have different displaying speeds so that eventually, the viewers can catch up with each other for a merge. Therefore, by using this technique, we will alter the display rate of the fallen out viewer such that he can eventually be merged with either the partition ahead or 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 way, the chance of releasing the allocated VCR resources before the viewer finishes the movie can be increased, thereby decreasing the value of μ_{hold}.

Figure 5.1 illustrates two scenarios that the viewer resuming with a miss at position V_c of the movie will consume the movie frames at rates with altering speed as compared to the normal playback viewers in the partitions. As before, the rate of consumption of movie frames for normal playback is denoted as R_{PB}. Figure 5.1a shows that the fallen out viewer will speed up the consumption of the movie frames at a rate of $(1+f)R_{PB}$ in order to catch up with the last possible viewer at position V_l in the partition ahead. Figure 5.1b shows that the viewer will be consuming the movie frames at a slower rate of $(1-f)R_{PB}$ in order to enable a catch-up by the first possible viewer at position V_f in the partition behind. Therefore, in each case, the movie time elapsed, C_t, so as



(a) catch up with viewer V_l in the partition ahead

(b) catch up with viewer V_f in the partition behind

Figure 5.1: Scenarios illustrating the catch-up by adaptive piggybacking when the viewer resumes with a miss

to allow for a join at either of the partitions by using adaptive piggybacking is $\frac{1}{f}\Delta$, where Δ is the initial distance between V_c and the target viewer (either at position V_f in the partition behind or at position V_l in the partition ahead). Since the calculation of μ_{hold} depends on different values of C_t for viewers resuming at different positions of the movie, so it will be favorable if we can further reduce this quantity.

In addition to adaptive piggybacking, we can also make use of the buffering technique to decrease the catch-up time. The motivation behind is that for viewers resuming with a miss very near to the ends of the partitions, their catch-up time can be reduced by using bridging to *merge* them with the partitions. After the merge, the allocated I/O stream can be released because the partition and the additional buffer will now be treated as one *extended partition.* The buffer used for bridging can be gradually released by also applying adaptive piggybacking until the viewer can really join in the partition. Besides, for viewers resuming with a miss far from the ends of the partitions, they can perform adaptive piggybacking until they are close enough to the partition for applying the same technique to perform a merge before the actual joining. In this way, the catch-up time, or C_t, can be further reduced. It is anticipated that by using buffering, the allocated I/O streams can be released as soon as possible upon resume, thereby enhancing the availability of the VCR I/O streams for other VCR requests. The use of buffering and adaptive piggybacking techniques is accomplished in the following way.

We define a threshold, k minutes, within which ahead or behind a partition will cause the use of buffering to enable a merge with the partition. Besides, adaptive piggybacking will be used simultaneously to further facilitate the joining of the partition. Therefore we divide the handling of VCR requests into two cases, namely resume within the threshold and resume beyond the threshold.

5.2.1 Resuming within the threshold ($\Delta \leq k$)

When the viewer resumes with a miss at a position which is within the threshold of k minutes ahead or behind the partition, we will use Δ minutes of buffer in merging the viewer with the existing partitions. Since the viewer is now consuming movie frames not yet fetched into the buffer (resume within k minutes ahead of the partition) or have been already replaced (resume within k minutes behind a partition), an I/O stream has to be consumed temporarily, until the partition and the additional buffer 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, adaptive piggybacking will also be performed to enable the viewer to continue the catch-up with the partitions and release the buffer 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. The maximum size of the additional buffer is given by Δ minutes because the first viewer in the partition behind takes Δ minutes to reach point R. In other words, the time taken for this viewer to release the allocated I/O stream is also Δ minutes. Similarly, the maximum size of buffer 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 resuming ahead and 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 unconditioning V_c by taking the probability of resuming at any position of the movie to be equal to $\frac{1}{l}$. Here we separate the calculation into two cases. The first of which is when the viewer resumes at a position with a distance of Δ behind the partition, that is, this viewer will catch up with V_l in the partition ahead. Therefore, the average catch-up time is given by:

$$
\begin{aligned}
T_{1k} &= \left(\frac{n}{l-B}\right)\left(\frac{1}{l}\right)\left(\frac{1}{1+f}\right)\left[\int_{Vc=0}^{l-\left(\frac{k}{1+f}+k\right)} \int_{\Delta=0}^{k} \Delta d\Delta dV_c \right. \\
&+ \left. \int_{Vc=l-\left(\frac{k}{1+f}+k\right)}^{l} \int_{\Delta=0}^{\frac{l-Vc}{2}} \Delta d\Delta dV_c \right]
\end{aligned} \tag{5.1}
$$

The first term corresponds to the average time to catch up with the partition ahead when resuming at any position of the movie from 0^{th} position to $(l - (\frac{k}{1+f} + k))^{th}$

position in the movie. The second term is for handling the boundary case of resuming at any position within the range of $(l - (\frac{k}{1+f} + k))^{th}$ position to the end of the movie. Therefore the distance of the resuming viewer from the partition ahead only ranges from 0 to $(\frac{l-V_c}{2})$ minutes if he is to catch up with the partition ahead before the movie ends.

The second case is when the viewer resumes at a position of distance Δ ahead of the 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:

$$T_{2k} = (\frac{n}{l-B})(\frac{1}{l}) \left[\int_{Vc=0}^{l-k} \int_{\Delta=0}^{k} \Delta d\Delta dV_c + \int_{Vc=l-k}^{l} \int_{\Delta=0}^{l-Vc} \Delta d\Delta dV_c \right] \tag{5.2}$$

5.2.2 Resuming beyond the threshold ($\Delta > k$)

When the viewer resumes at a position with a distance greater than k minutes from the partitions, the viewer will perform adaptive piggybacking until the distance reduces to k minutes. At this point, it is handled as in previous case. For resuming at a position beyond the threshold, we consider the average time for releasing the I/O stream by the following three cases.

1. **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

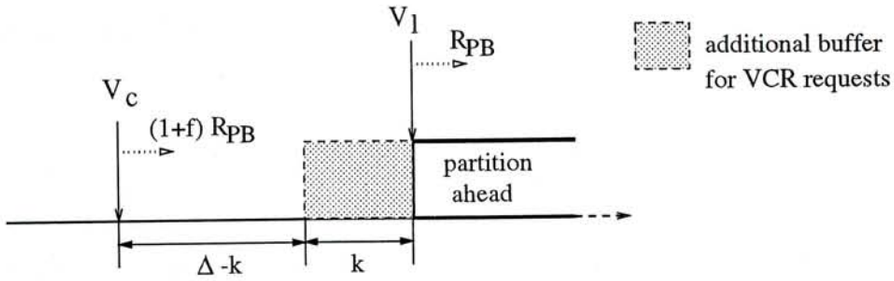


Figure 5.2: speed up to catch up with V_l in the partition ahead until distance become k

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 $\frac{1}{f}(\Delta - k)$, which is the time needed to decrease the distance of the viewer from the partition to k minutes. At this point, $\frac{k}{1+f}$ more minutes are needed for the merge to complete. This scenario is illustrated in Figure 5.2. Therefore the total time,

C_t, needed to release the VCR I/O stream is given by $\frac{1}{f}(\Delta - k) + \frac{k}{1+f}$. 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 VCR 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:

$$\begin{aligned} \frac{1}{f}(\Delta - k) + \frac{k}{1+f} &\leq l - (Vc + \Delta) \\ \Delta &\leq \frac{l - Vc - k(1 - \frac{1}{f(1+f)})}{\frac{1}{f} + 1} = L_1 \end{aligned} \tag{5.3}$$

2. **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

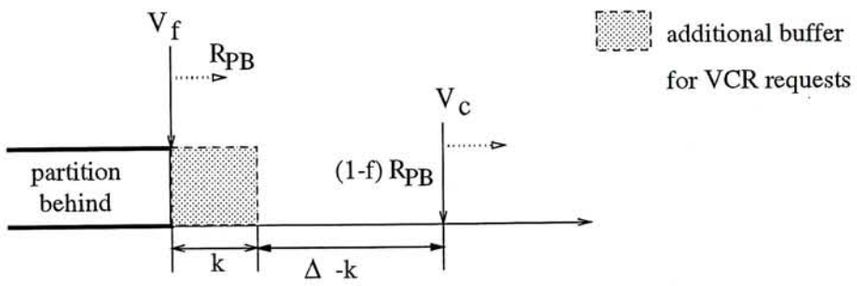


Figure 5.3: slow down to enable a catch-up by V_f in the partition behind until distance become k

of movie frames 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 Figure 5.3. Using the same calculation as the above case, we can see that the time elapsed before a merge can be accomplished 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 in performing adaptive piggybacking. Therefore, the following inequality has to be satisfied instead:

$$\begin{aligned} \frac{1}{f}(\Delta - k) + k &\leq l - Vc \\ \textit{which gives} \quad \Delta &\leq [l - Vc - k(1 - \frac{1}{f})]f = L_2 \end{aligned} \tag{5.4}$$

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

The average time to release the resources, given that the viewer is at position V_c of the movie, is calculated by Equation (5.5). The first term below calculates the average time a viewer takes to catch up with the partition ahead while the second term below calculates the average time a viewer takes to enable a catch-up by the first possible viewer in partition behind. 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. 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 corresponds to the average time spent to release the I/O when a viewer cannot perform adaptive piggybacking to enable a catch-up with either of the partitions.

$$\int_{\Delta=k}^{k_1}[\frac{1}{f}(\Delta-k)+\frac{k}{1+f}](\frac{d\Delta}{\frac{l-B}{n}})+\int_{\Delta=k}^{k_2}[\frac{1}{f}(\Delta-k)+k](\frac{d\Delta}{\frac{l-B}{n}})+ (l-V_c)(\frac{\frac{l-B}{n}-k_1-k_2}{\frac{l-B}{n}}) \tag{5.5}$$

To find the average time to release the resources at any position of the movie, we uncondition V_c in three cases as stated above, which corresponds to the three terms as defined in Equation (5.5). 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 in the partition ahead or behind. In each of the three cases, we find out the range of V_c by equating the inequality: $\frac{l-B}{2n} \leq \Delta$ for cases 1 and 2 using Equations (5.3) and (5.4) respectively.

$$\textit{For case } 1: \quad Vc \leq l-k(\frac{1}{f(1+f)})-(\frac{1}{f}+1)(\frac{l-B}{2n})=\eta_1 \tag{5.6}$$

$$\textit{For case } 2: \quad Vc \leq l-k(1-\frac{1}{f})-(\frac{1}{f})(\frac{l-B}{2n})=\eta_2 \tag{5.7}$$

We will now consider the unconditioning of V_c in Equation (5.5) term by term.

1. **Case 1**

   For $V_c \leq \eta_1$, that means when the viewer resumes at the maximum distance of $\frac{l-B}{2n}$ from the end of the partition, a merge to 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 Equation (5.3) instead. Thus the equation follows.

$$T_1=(\frac{n}{l-B})(\frac{1}{l})\left[\int_{Vc=0}^{\eta_1}\int_{\Delta=k}^{\frac{l-B}{2n}}[\frac{1}{f}(\Delta-k)+\frac{k}{1+f}]\,d\Delta dVc + \int_{\eta_1}^{l-(\frac{k}{1+f}+k)}\int_{\Delta=k}^{L_1}[\frac{1}{f}(\Delta-k)+\frac{k}{1+f}]\,d\Delta dVc\right] \tag{5.8}$$

2. **Case 2**

   The equation given below is derived by the same manner as in case 1 with reference to Equation (5.4) and range of V_c defined in Equation (5.7).

$$T_2 = (\frac{n}{l-B})(\frac{1}{l}) \left[\int_{Vc=0}^{\eta_2} \int_{\Delta=k}^{\frac{l-B}{2n}} [\frac{1}{f}(\Delta - k) + k] \, d\Delta dVc \right. \\ \left. + \int_{V_c=\eta_2}^{l-k} \int_{\Delta=k}^{L_2} [\frac{1}{f}(\Delta - k) + k] \, d\Delta dVc \right] \tag{5.9}$$

3. **Case 3**

   The following Equation is calculated with respect to Equations (5.1),(5.2), (5.8) and (5.9). Making use of the upper bound of Δ in each equation and uncondition 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.

$$\begin{aligned} T_3 &= (\frac{n}{l-B})(\frac{1}{l}) \left[\int_{Vc=\eta_1}^{\eta_2} (l - Vc)(\frac{l-B}{n} - \frac{l-B}{2n} - L_1) \, dVc \right. \\ &+ \int_{Vc=\eta_2}^{l-(\frac{k}{1+f}+k)} (l - Vc)(\frac{l-B}{n} - L_1 - L_2) \, dVc \\ &+ \int_{Vc=l-(\frac{k}{1+f}+k)}^{l-k} (l - Vc)(\frac{l-B}{n} - \frac{l-Vc}{2} - L_2) \, dVc \\ &+ \left. \int_{Vc=l-k}^{l} (l - Vc)(\frac{l-B}{n} - (l - Vc) - \frac{l-Vc}{2}) \, dVc \right] \end{aligned} \tag{5.10}$$

The average time to release the I/O stream for case 3 is given by Equation (5.10). The first two terms are derived with reference to Equations (5.8) and (5.9), the last two terms are derived with reference to the second term in Equation (5.1) and (5.2).

Therefore, the average time to release the I/O stream in this scheme, μ_{hold}, is given by the following equation.

$$\mu_{hold} = T_{1k} + T_{2k} + T_1 + T_2 + T_3 \tag{5.11}$$

5.3 Verification

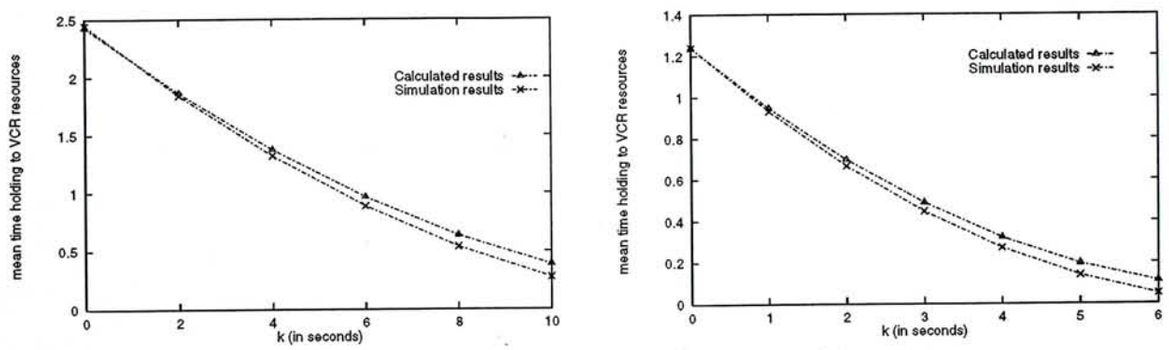
In the previous section, we have derived the equations for calculating the value of μ_{hold} for scheme 2. These equations help to calculate the overall time the viewers will hold on to the VCR resources before they can join back to the partitions. In this section, we will verify our derivation using 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

Table 5.1: Parameters of the two popular movies for simulation

We carry out our simulation using two popular movies 1 and 2, with lengths 60 and 90 minutes, and the maximum time that the viewers can wait for the start of the movies are 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 5 minutes and 2 minutes respectively. Let the performance requirement of P^* of each movie be equal to 0.5. We apply our the mathematical model to calculate the amount of buffer and I/O resources for normal playback and they are summarized in Table 5.1.

The arrival rate of viewers into system, λ, is $15/min$ and the simulation is run for $50,000$ viewers. The average time where the fallen out viewers hold on to the allocated VCR resources (μ_{hold}) is calculated for different values of threshold value, k. The respective values of μ_{hold} obtained by simulation and our derivation of both movies are plotted in Figure 5.4.



(a) movie 1 (b) movie 2

Figure 5.4: Simulation and calculated results of μ_{hold} for different values of threshold, k

From Figure 5.4, we can see that the calculated values of μ_{hold} match closely with the simulation results, which verify our derivation. Yet the values of μ_{hold} obtained from simulation are smaller than that calculated, which can be explained as follows. In our derivation, we try to calculate the time taken for the viewers to join back the existing partitions. However, in the simulation, there exist also other fallen out

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

Chapter 6

Applications to System sizing

In this chapter, we illustrate how we apply the mathematical model described in section 4 and the $M/M/m$ queue 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 while the second category of resources is for serving VCR requests, and is modeled by an $M/M/m$ queue. As mentioned before, the provision of VCR functions will inevitably increase system resource requirement, especially in a system using data sharing techniques like batching and buffering because viewers may fall out of a batch upon resume. Therefore, in managing system resources in a VOD system, our goal is to optimize the performance of a VOD system by reducing the usage of system resources while providing an acceptable level of QoS to the viewers. In other words, our goal is to design a cost-effective system, such that the overall cost of resources, including resources for normal playback as well as servicing VCR requests, can be reduced. We measure the system cost in terms of the amount of buffer space and I/O streams needed to ensure a level of performance. Thus, the overall cost of resources includes the costs for memory buffers per minute of a movie and one I/O stream, expressed as C_b and C_n respectively.

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 probability of 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 of greater than $(1 - P^*)$. Therefore, if we can design our system such

that the probability of 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. On the other hand, a high probability of hit would ensure the release of I/O resources upon resume, which will avoid the exhaustion of reserved resources for servicing VCR requests. In other words, if there is no chance of releasing I/O resources back to the system pool, each VCR request will consume one I/O resource until the viewer finishes the movie. Then more VCR requests simply imply more resources will be held by the viewers until they finish watching the movie. These resources can be used to service future VCR requests or requests for normal playback of non-popular movies. However, to ensure a high probability of hit requires more resources to be reserved for normal playback, which also implies a higher cost. Therefore, it is very important to determine the optimal probability of hit which will yield a system with the lowest overall cost. In the remaining chapter, we will show how to determine the cost of resources for normal playback and that for servicing VCR requests respectively.

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 $n = 1, 2, ...\frac{l}{w}$). The respective buffer size, B, for each n can be obtained by Equation (4.2). We can then determine the cost of resources for normal playback, C_{PB}, for each (B, n) pair, by the equation, $C_b\ B + C_n\ n$, 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 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}).

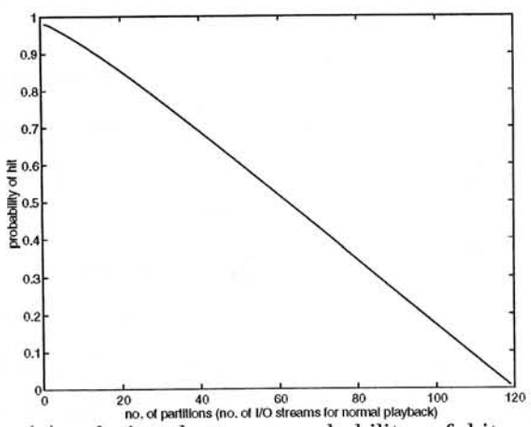
Example 1 *If we have a movie of length 60 minutes, and the maximum time a viewer is willing to wait before a movie starts is 30 seconds. Assume that the time the viewers*

spend in VCR functions follows an exponential distribution with mean equal to 1 minute of movie time. By applying our mathematical model, the respective probability of hit is plotted in Figure 6.1a. We further 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 is assumed to be 4 Mbps, and the cost of 1 MB of main memory is assumed to be \$24. We calculate C_b and C_n as follows.

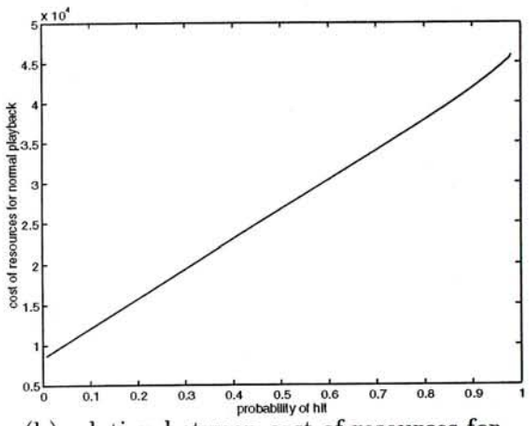
$$C_b = \frac{60s * 4Mbps}{8} * \$24 = \$720$$

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

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 10 times as expensive as one I/O stream, i.e., $\varphi \approx 10$. The cost for normal playback is plotted in Figure 6.1b.



(a) relation between probability of hit and no. of I/O streams required for normal playback



(b) relation between cost of resources for normal playback and probability of hit

Figure 6.1: calculating the resources for normal playback

From Figure 6.1a, we can see that decreasing the number of partitions will increase the probability of hit. More buffer space is utilized to substitute 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 hit. However, we can see that the cost of resources for normal playback also increases with the probability of hit from Figure 6.1b. So, it may seem that adopting a lower probability of hit will yield a lower system cost. We will show, by the next example, 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 optimal 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 Equations (3.7) - (3.9). Therefore in determining the amount of resources needed for servicing VCR requests, we have to define the performance level first. The amount of resources required will also be affected by the arrival rate of VCR requests, λ_v, and mean service time of VCR request, μ_v respectively. Yet from Equation (3.3), μ_v also depends on the average time a viewer spends in doing VCR functions, μ_{VCR}, and the probability of hit. All these variants will result in different amount of memory buffers, B_v, and I/O streams, m, needed to service VCR requests. Therefore 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 for the resources needed for servicing VCR requests.

Example 2 *Consider the system in Example 1. The system will make use of adaptive piggybacking [GLM96] only to facilitate the fallen out viewers to join in the partitions again. In other words, we will adopt Scheme* 2 *but the threshold value, k is set to* 0. *We assume that the average waiting time for a viewer's VCR request to be serviced does not exceed 3 seconds, i.e. the average time a viewer has to wait for the display of the VCR-version of the movie (in case of FF/RW) after he issues the VCR request is 3 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 every* 4.5 *min, and this is considered a very high VCR rate. But we will show later in this chapter the effect of altering different VCR arrival rates. Figure 6.2 shows the relationship between the cost of resources for servicing VCR requests and the probability of hit.*

From Figure 6.2, we can see that the cost of resources decreases with the increase of probability of hit, which is the opposite compared with Figure 6.1b. A lower probability of hit will increase the average time a viewer spends in holding on to the additional resources, which implies 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 and that for servicing VCR requests, i.e. the overall system cost, before we can determine how much resources are to be allocated for normal playback and for servicing VCR requests.

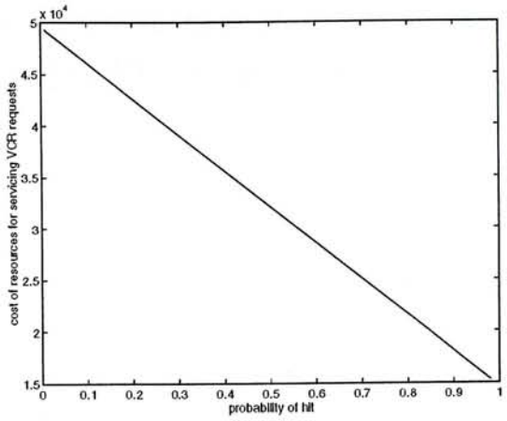


Figure 6.2: cost of resources for servicing VCR requests

6.3 Overall system cost

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

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

Step 1: Generate all the PB-tuples by finding all the possible configurations for normal playback by applying our mathematical model presented in Chapter 4 and calculate the respective hit probabilities and cost.

Step 2: Obtain μ_{hold} for each of the configurations generated in step 1 using Equation (5.11). With reference to the hit probability in each PB-tuple obtained in Step 1, we can obtain the corresponding μ_v by Equation (3.3). We can then find out a corresponding VCR-tuple by applying an $M/M/m$ queue.

Step 3: For each pair of PB-tuple and VCR-tuple, we 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 given by the tuple pair with the lowest cost.

Example 3 *Consider the system in Example 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 Figure 6.3.*

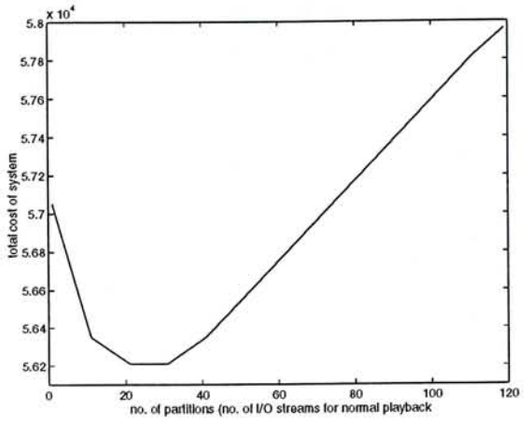


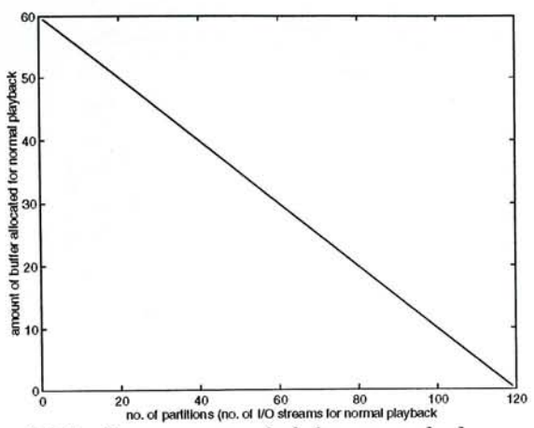
Figure 6.3: Total cost of system

In this figure we can see that the overall cost of the system decreases initially for smaller number of I/O streams, and increases towards 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 Figure 6.1a, the probability of hit lies approximately within the range of $0.75 - 0.85$. To illustrate the respective amount of I/O streams and buffer needed for normal playback and VCR control, we further plot out these amounts against the number of I/O streams used in normal playback in Figure 6.4. From Figure 6.4a, the amount of buffer space needed falls within the range from 44.5 *min* to 49.5 *min*, and from Figure 6.4b, the number of I/O streams needed falls within the range of $280 - 330$.

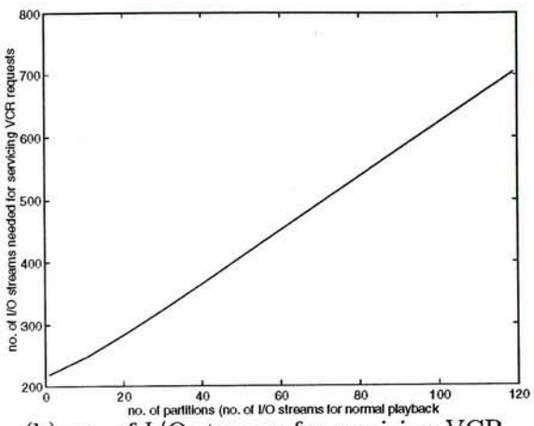
This example shows that when determining the right amount of resources to be allocated for normal playback, we should also consider the cost of resources for VCR requests. The allocation of corresponding resources should yield a system with the lowest cost while satisfying a predefined performance level.

6.4 Comparison

In this section, we demonstrate the effectiveness of our system by running several experiments to determine the overall system cost. Firstly, we incorporate Scheme 1 and Scheme 2 respectively into our normal playback model respectively, and their total system cost is plotted against different arrival rates of VCR requests, while other



(a) buffer space needed for normal playback



(b) no. of I/O streams for servicing VCR requests

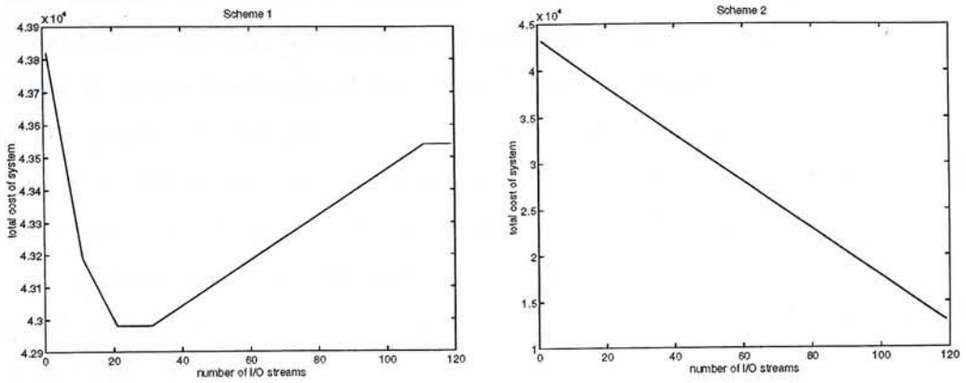
Figure 6.4: break down of respective amount of resources needed in servicing normal playback and VCR requests

parameters remain the same as those described in the previous examples. Secondly, we examine two other systems, namely a system using I/O streams only for normal playback (refer as *pure I/O* hereafter) and a system using buffer space only for normal playback (refer as *pure buffer* hereafter). These are the two extreme cases where only I/O streams or memory buffer is used for normal playback. We aim to show that a balance between I/O streams and memory buffer will yield a system with a lower cost.

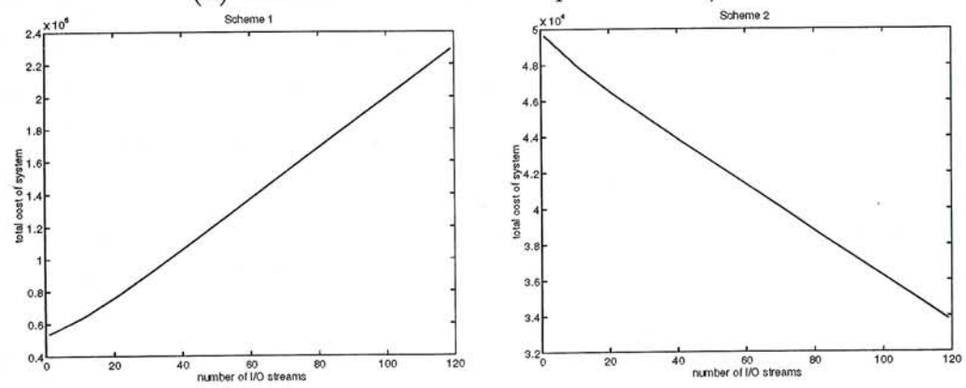
6.4.1 Scheme 1 vs. Scheme 2

Experiment 1 *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 normal playback request is* $15/min$*, so that the expected number of viewers in the system is given by* $15 \times 60 = 900$*. We plot out the overall system cost of each scheme against the number of I/O streams needed for normal playback with the VCR arrival rates as* $15/min$*,* $100/min$*,* $200/min$ *and* $400/min$ *in Figure 6.5a, b, c, d respectively.*

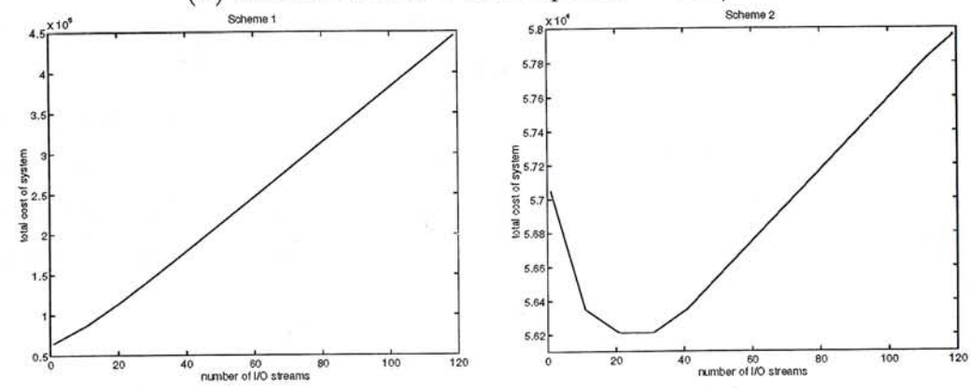
From the figures, we can see that different arrival rates will yield a system with the lowest cost occurring at different positions on the cost curve. For Scheme 1, except for very low VCR arrival rates, the minimum cost is always located at the smallest value of I/O stream, that is, with the largest 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



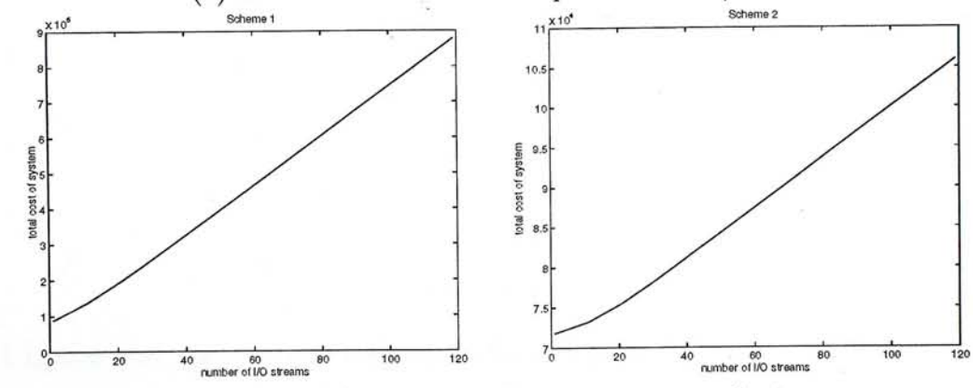
(a) arrival rate of VCR requests = $15/min$



(b) arrival rate of VCR requests = $100/min$



(c) arrival rate of VCR requests = $200/min$



(d) arrival rate of VCR requests = $400/min$

Figure 6.5: Total cost of system for Scheme 1 and Scheme 2 for different VCR arrival rates

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 hit can be increased. By increasing the probability of 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 reason 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 stream, that is, when the least amount of buffer space is used for normal playback. Low arrival rate of VCR requests means not many viewers will be issuing VCR requests. Even if the viewers fall out of a partition upon resume, Scheme 2 can ensure the viewers to join back the partition to release the allocated resource at 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 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 the system of Scheme 2 is much more smaller than that of Scheme 1. To further demonstrate the effectiveness of Scheme 2, we plot out the percentage save in the total system cost by using Scheme 2 compared with Scheme 1. The percentage save by Scheme 2 for different arrival rates, as shown in Figure 6.6 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 compared to Scheme 1 ranges approximately from 13% to as high as 70%.

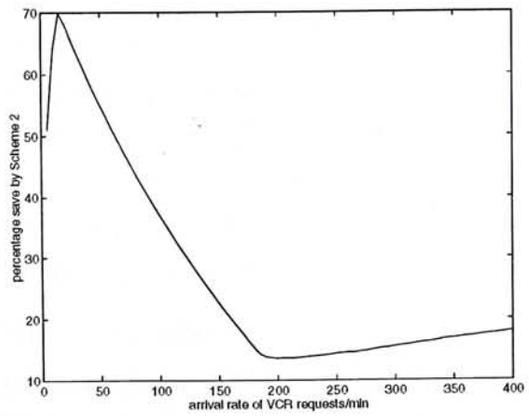


Figure 6.6: percentage save by Scheme 2 over Scheme 1 in system cost

6.4.2 Scheme 2 vs. pure I/O & pure buffer

In this section, we will compare the performance of our system with that of a system using pure I/O or pure buffer for normal playback. These two systems are briefly described below.

- **Pure I/O method**

  In this method, every viewer initiating a normal playback request of the movie is allocated an individual I/O stream. Therefore if the arrival rate for normal playback is λ, the expected number of I/O streams needed, n, is equal to the expected number of viewers in the system, i.e., λl. For the provision of VCR functionality, we separate into two cases because how the VCR-version of the movie is displayed to the viewers is implementation-dependent.

  - *VCR arrival rate dependent*

    When a viewer issues a VCR request of fast-forward or rewind, the viewer has to switch to consume the reserved VCR I/O resources instead of using the normal playback I/O resource to display the VCR-version of the movie (phase 1). In this case, we assume that there is a VCR-version of the movie stored in the disk and a switch to another I/O stream is necessary. Switching to another I/O stream has the advantage that the I/O scheduling is simplified, where we can use common scheduling routine to service normal playback and VCR control. When the viewer resumes from the VCR request, he can always switch back to the individual normal playback I/O stream dedicated to him to continue watching the movie. In other words, the dedicated resources reserved for handling VCR requests can be released immediately, resulting in $\mu_{hold} = 0$. In this case, with reference to Equation (3.3), we have $\mu_v = \mu_{VCR}$. The total number of I/O streams needed is equal to $\lambda l + m$, where m is obtained by applying the $M/M/m$ model. The total cost of the system is calculated by summing up the cost of the total number of I/O streams needed[1].

  - *VCR arrival rate independent*

    We assume that the allocated I/O stream for normal playback can be used to read the VCR-version of the movie e.g. by reading selected frames from the disk. The advantage of which is that less storage is required because we need not store the VCR-version of the movie in the disk. Besides, each

[1] We assume that the I/O streams for normal playback and VCR control are of the same cost

viewer makes use of his own I/O stream for both normal playback and VCR control. However, such implementation will increase the scheduling complexity in servicing VCR requests, because the reading of frames need to be re-scheduled so as to read the required selected frames from the disk. In addition, the selection of frames requires significant additional system resources as the individual frames have to be indexed to facilitate selective retrieval. The total number of I/O streams needed is equal to λl only and is independent of the VCR arrival rate.

Note that the cost in both cases depends on the arrival rate of normal playback requests, λ.

- **Pure buffer method**

  In this method, the whole movie is put in the buffer. Viewers join in by consuming the movie frames from the buffer. We assume that an infinite number of viewers can access the movie frames from the buffer at the same time. Besides, as in pure I/O method, depending on the implementation of how to display the VCR-version of the movie, we split into two cases:

  - *VCR arrival rate dependent*

    The viewer has to switch to consume the reserved VCR I/O resources to access the VCR-version of the movie. As in the pure I/O dependent case, it simplifies the scheduling problem yet increases the storage requirement for storing the VCR-version of the movie. Notice that by buffering the whole movie in the buffer, whenever the viewer resumes from the VCR request, the allocated I/O stream can be immediately released, also resulting in $\mu_{hold} = 0$. Therefore, we have $\mu_v = \mu_{VCR}$ as before. The number of I/O streams needed to be reserved for handling VCR requests is again obtained by applying the $M/M/m$ model as in the pure I/O method, thus dependent on the VCR arrival rate, λ_v. The total cost is calculated by summing up the cost of buffering the whole movie and that of m additional I/O streams for handling VCR request.

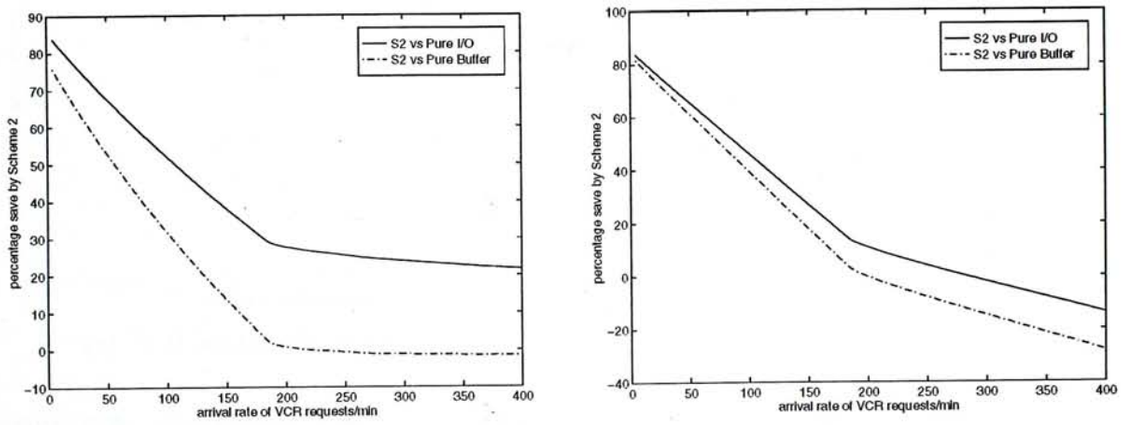
  - *VCR arrival rate independent*

    The VCR-version of the movie is stored in another buffer, from which the interactive viewers fetch the movie frames. In this case, they need not consume I/O resources for servicing their VCR requests and when they resume, they can always fetch the movie blocks for their normal playback from the original buffer. As fast forward or fast rewind can be seen as

displaying the movie at x times of the normal playback rate, we assume that the size of buffer needed to buffer the VCR-version of the movie is given by $\frac{l}{x}$. The total cost is calculated by summing up the cost of buffering the whole movie and the VCR-version, i.e. $l + \frac{l}{x}$.

Note that the cost of buffer for normal playback is not affected by the arrival rate of normal playback request because of our assumption that infinite number of viewers can access the movie frames from the buffer.

Experiment 2 *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 pure I/O. The rate of fast-forward or rewind is assumed to be 3 times faster than that of normal playback. The percentage save in system cost by Scheme 2 for the case where the total cost is dependent on VCR arrival rate is plotted against different VCR arrival rates and is shown in Figure 6.7a, while the independent case is plotted in Figure 6.7b. The percentage save is calculated by the difference between the minimum cost of Scheme 2 and that of pure I/O and pure buffer respectively.*



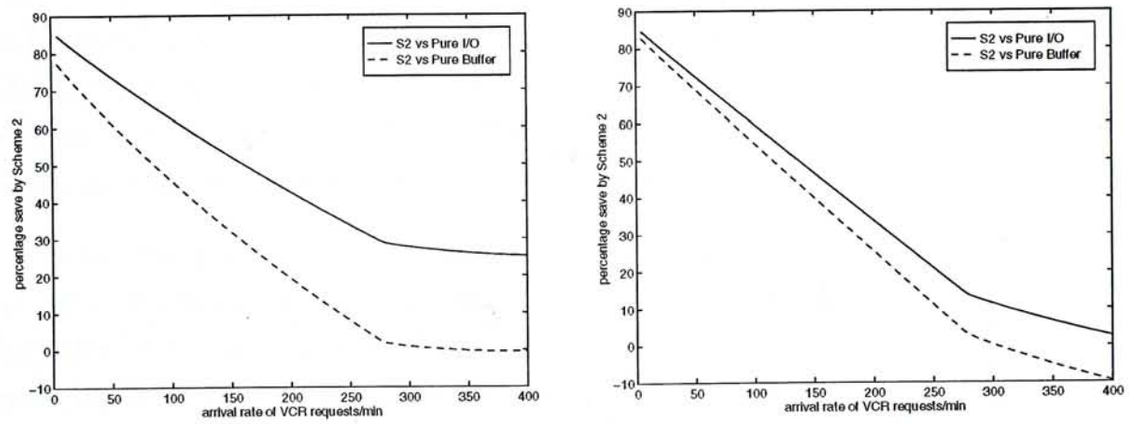
(a) VCR arrival rate dependent

(b) VCR arrival rate independent

Figure 6.7: percentage save by Scheme 2 over pure I/O and pure buffer in system cost

As shown in Figure 6.7a, 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% and also as high as 83%. For pure buffer, Scheme 2 also has a percentage save up to 75%, although after a VCR arrival rate of around $220/min$, the 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 Figure 6.7b, the cost curve also decreases rapidly at first from around 80% 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 pure buffer method perform better than Scheme 2 when the VCR arrival rate increases to a certain extent. The percentage becomes negative when the VCR arrival rate increases up to around $200/min$ and $290/min$ for pure buffer and pure I/O respectively. Besides, 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 increases. On the contrary, as the cost of pure I/O and pure buffer is independent of the VCR arrival rate, which implies that the cost is fixed, thereby resulting in the continuous decrease in the percentage save. We also plot out the same graphs for two more system settings, where the movie is of length 90 *min* and 120 *min*, and they are shown in Figure 6.8 and 6.9 respectively. From these figures, we can see that the percentage save by Scheme 2 is even more significant as the movie length increases.

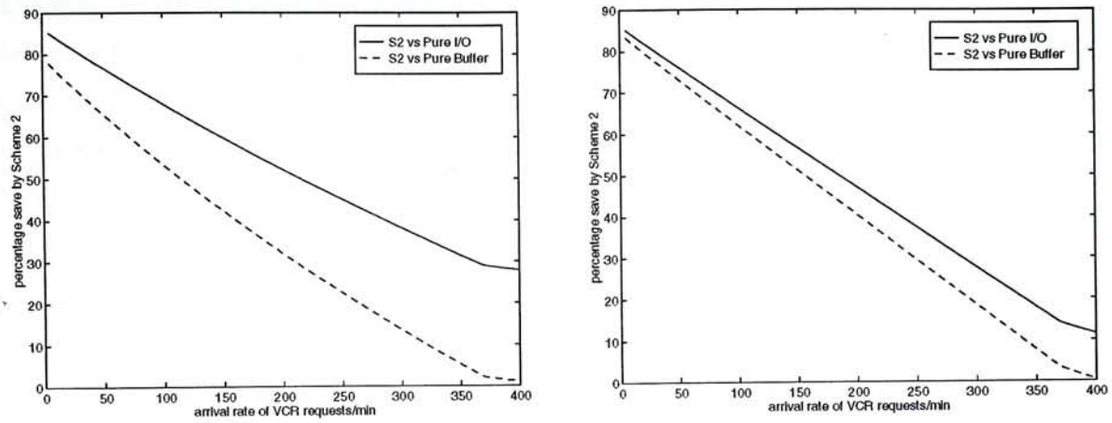


(a) VCR arrival rate dependent

(b) VCR arrival rate independent

Figure 6.8: percentage save by Scheme 2 over pure I/O and pure buffer in system cost for a movie of length 90 *min*

The percentage save 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, and it becomes dominated in the case of Scheme 2. In spite of that, we can claim that Scheme 2 outperforms the pure I/O and pure buffer method 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 save by Scheme 2 is positive, i.e. at most $200/min$



(a) VCR arrival rate dependent (b) VCR arrival rate independent

Figure 6.9: percentage save by Scheme 2 over pure I/O and pure buffer in system cost for a movie of length 120 *min*

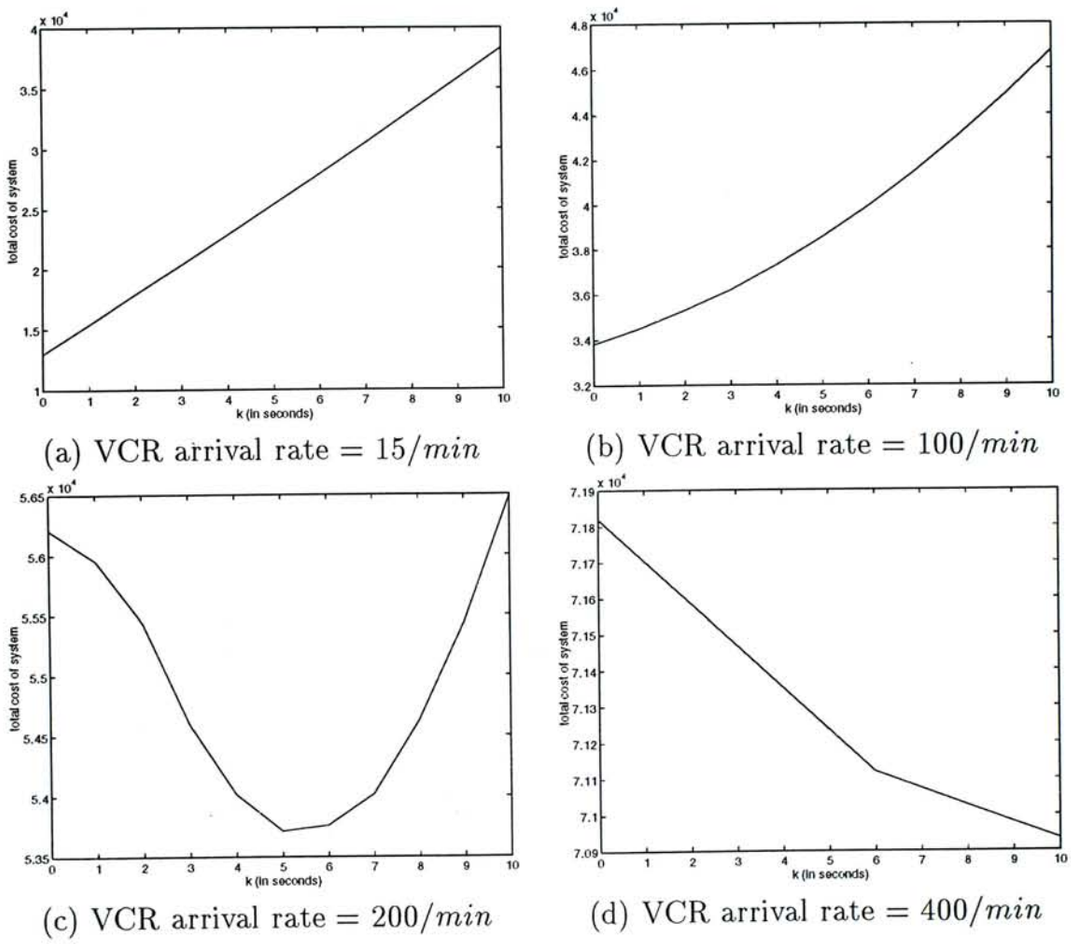
and the expected number of viewers in the system is 900 in this example. That means approximate $\frac{1}{4}$ of the viewers are doing VCR functions per minute or the viewers are issuing VCR requests once every 4.5 minutes on average, and this is a very high VCR rate for viewing movies. In a VOD system, relatively less time will be spent in doing VCR functions as compared with normal playback, for a movie of length 60 minutes, the range of VCR arrival rate is far less that $200/min$. In this region, we can have a substantial reduction in the cost of resources by using Scheme 2.

Besides, the percentage save against pure I/O method is more than that against pure buffer method, which means 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 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 Different values of k

In the previous experiments, we find out the overall system cost of our proposed system by adopting Scheme 2 with $k = 0$, that is, we only make use of adaptive piggybacking to facilitate the viewers to join the partitions. However, as mentioned before, the use of additional buffer can further facilitate the fallen out viewers to join back to the

partitions. In this experiment, we plot out the overall system cost against different values of k for different VCR arrival rates in Figure 6.10. For different values of k, we find out their corresponding values of μ_{hold}, and calculate the lowest cost of supporting such a system. We plot out the total cost of the system with different values of k by varying the VCR arrival rate.



(a) VCR arrival rate = $15/min$ (b) VCR arrival rate = $100/min$

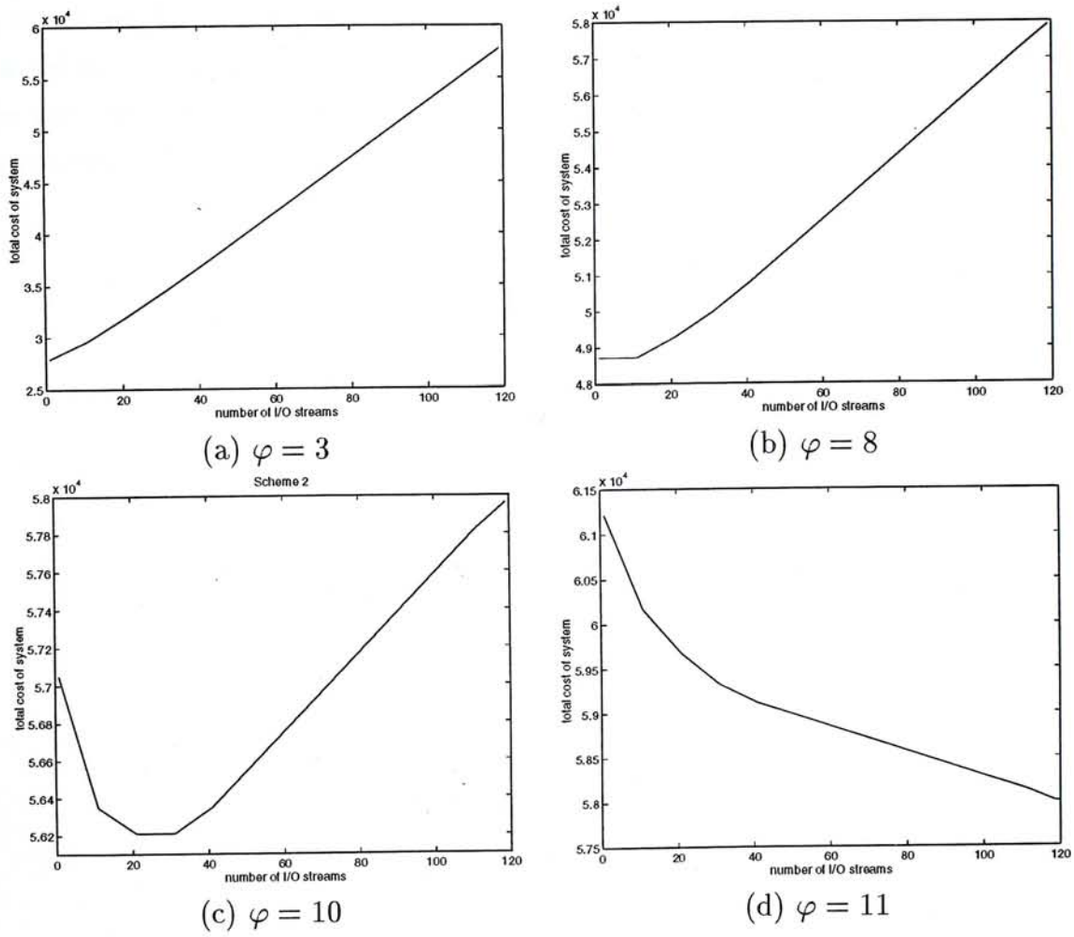
(c) VCR arrival rate = $200/min$ (d) VCR arrival rate = $400/min$

Figure 6.10: The total system cost for different values of k

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 back the partitions, which implies 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. The use of additional buffer space cannot be compensated 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 Different values of φ



(a) $\varphi = 3$ (b) $\varphi = 8$

(c) $\varphi = 10$ (d) $\varphi = 11$

Figure 6.11: Overall system cost vs number of I/O streams for different values of φ

In the previous experiments, we have assumed that the cost of memory buffer per minute is approximately 10 times that of one I/O stream. However, in future technology, where cost of memory may decrease faster than that of I/O bandwidth, the minimum cost will occur at some other positions on the cost curve, as illustrated

in Figure 6.11. In this experiment, we vary φ as $3, 8, 10$ and 11. Figures 6.11(a) and 6.11(b) illustrate the situation where the cost of the memory decreases faster than the cost of I/O bandwidth while Figures 6.11(d) illustrates the 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 buffer for large values of φ, but more memory buffers should be used for smaller values of φ. For smaller values of φ, it means the cost of memory buffer is closely comparable to the cost of I/O stream and use of buffer can not only increase the probability of hit but also cause a reduction in the amount of VCR resources to be reserved. Recall that in normal playback, we are using w minutes of buffer to trade off 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 the optimal system configuration, our model helps system designers to make the right system sizing decisions and at the same time, meet the performance requirements.

Chapter 7

Conclusions

There are various research results on how to design new techniques for the efficient use of resources in a VOD system. The scalability of VOD systems can be improved by incorporating data sharing techniques such as batching, buffering and adaptive piggybacking in servicing normal playback requests. Although these techniques can greatly reduce the total I/O resource requirements, they do not work very well in the presence of interactive VCR functions. The viewers are initially grouped together to share the same set of resources for servicing their normal playback requests. However, as the behavior of viewers in issuing VCR requests is unpredictable and they can resume at any instant of the movie, the viewers may not be able to join in any existing groups of viewers to share the same resources for continuing the normal playback. These fallen out viewers will require additional resources to service their normal playback request, in other words, the benefits achieved by applying the data sharing techniques are lost. Therefore it is very important to have an efficient scheme on handling VCR requests. Besides, determining the right amount of resources to be allocated is crucial for optimizing the performance of VOD systems so as to maintain all the benefits of data sharing techniques.

In this paper, we try to solve two problems. Firstly, we solve the system sizing problem by determining the right amount of resources in the following way. We separate the resources into two categories, the first of which is for servicing normal playback requests while the second of which is for servicing VCR requests. The normal playback model is based on static partitioning, where the buffer is divided into partitions, each serviced by an individual I/O stream. We propose a very general mathematical model to determine the amount of resources required for supporting normal playback by observing the following idea: 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 con-

sumed once the VCR operation is finished. Therefore, the normal playback resources are calculated by ensuring a certain probability of hit when the viewer resumes from VCR requests. This probability will guarantee that the chance for the viewers to hold on to the resources can be reduced to a certain level. However, it is still not desirable for the fallen out viewers to hold on to the resources for a long time. Secondly, we propose a scheme for handling the fallen out viewers so as to help restoring the benefits lost in providing interactive VCR functions. This is achieved by using adaptive piggybacking and buffering to facilitate the viewers to join in the existing partitions again so that the hold on resources can be released as soon as possible. Thus, based on the time a viewer spends in utilizing the additional resources, the amount of resources needed for providing a satisfactory service is calculated by applying an $M/M/m$ queue.

Efficient resource management and proper system sizing is very important because without an accurate model and a good system sizing method, we cannot determine the right amount of resources to be reserved so as to design a cost-effective VOD system. Although the amount of resources will also depend on the arrival rate of normal playback requests and that of VCR requests, our mathematical model, together with the proposed scheme, can be applied to solve the system sizing problem of a VOD system. The amount of resources calculated ensures that the service is provided to satisfy a given performance level, such as the maximum waiting time for starting or resuming the movie is within the viewers' tolerance level. We find the optimal distribution of resources for normal playback and servicing VCR requests so as to reduce the overall cost. We show by experiments that different arrival rates of VCR requests will affect the overall cost and the distribution of resources for servicing normal playback requests and VCR requests.

Bibliography

[AWY96] C. Aggarwal, J. Wolf, and P. Yu. On Optimal Piggyback Merging Policies for Video-on-Demand Systems. In *ACM SIGMETRICS Conference*, pages 200–209, May 1996.

[BP94] Milind M. Buddhikot and G. Parulkar. Distributed Data Layout, Scheduling and Playout Control in a Large Scale Multimedia Storage Server. Technical report, Washington University, July 1994. Technical Report WUCS-94-33.

[CKLV95] H.J. Chen, A. Krishnamurthy, T.D.C. 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.

[CKY95] 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*, pages 126–135, 1995.

[DDM+95] A. Dan, D.M. Dias, R. Mukherjee, D. Sitaram, and R. Tewari. Buffering and Caching in Large-Scale Video Servers. *IEEE CompCon*, 1995.

[DS94] A. Dan and D. Sitaram. Buffer Management Policy for an On-Demand Video Server. *IBM Research Report RC 19347*, January 1994.

[DS96] A. Dan and D. Sitaram. A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads. In *MMCN96*, San Jose, CA, January 1996.

[DSS94] 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.

[DSST94] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley. Channel Allocation under Batching and VCR Control in Movie-On-Demand Servers. *IBM Research Report RC 19588*, 1994.

[DVV94] D. Deloddere, W. Verbiest, and H. Verhille. Interactive video on demand. *IEEE Communication Magazine*, 32(5):82–88, 1994.

[FR94] C. Federighi and Lawrence A. Rowe. A Distributed Hierarchical Storage Manager for a Video-on-Demand System. In *Storage and Retrieval for Image and Video Databases II, IS&T/SPIE*, San Jose, CA, February 1994.

[GLM95] L. Golubchik, John C.S. Lui, and R. Muntz. I/O Stream Sharing for Continuous Media Systems. In *Bulletin of the Technical Committee on Data Engineering*, 1995.

[GLM96] L. Golubchik, J.C.S. 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.

[KRT94] M. Kamath, K. Ramamritham, and D. Towsley. Buffer Management for Continuous Media Sharing in Multimedia Database Systems. Technical report, University of Massachusetts, February 1994. Technical Report 94-11.

[KSKT94] Jayanta K.D., J.D. Salehi, J.F. Kurose, and D. Towsley. Providing VCR Capabilities in Large-Scale Video Servers. In *Proc. ACM Intl. Conf. on Multimedia*, pages 25–32, San Francisco, October 1994.

[LL95] S.W. Lau and John C.S. Lui. A Novel Video-On-Demand Storage Architecture for Supporting Constant Frame Rate with Variable Bit Rate Retrieval. In *Proceedings of the Fifth International Conference on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, April 1995.

[LL96] John C.S. Lui and K.W. Law. Load Balancing and VCR Functionalities Support via Subband Coding Techniques. In *Collected Abstracts from the 6th Initernational Workshop on Network and Operating System Suport for Digital Audio and Video*, pages 77–80. April 1996.

[LL97] Wanjiun Liao and Victor O.K. Li. The Split and Merge (SAM) Protocol for Interactive Video-on-Demand Systems. In *Proc. Intl. Conference on Computer Communications*, pages 1351–1358, Kobe, Japan, April 1997.

[LLG97a] S.W. Lau, J.C.S. Lui, and L. Golubchik. Merging Video Streams in a Multimedia Storage Server: Complexity and Heuristics. *accepted for publication in ACM Journal of Multimedia Systems*, 1997.

[LLG97b] Mary Y.Y. Leung, John C.S. Lui, and Leana Golubchik. Buffer and I/O Resource Pre-allocation for Implementing Batching and Buffering Techniques for Video-on-Demand Systems. In *Proc. Intl. Conference on Data Engineering*, Birmingham, England, April 1997.

[MN95] D.J. Makaroff and R.T. Ng. Buffer sharing schemes for continuous-media systems. Technical Report TR-95-01, Department of Computer Science, University of British Columbia, 1995.

[NY96] R.T. Ng and J. Yang. An analysis of buffer sharing and prefetching techniques for multimedia systems. *Multimedia Systems*, pages 55–69, 1996.

[OBRS94] B. Ozden, A. Biliris, R. Rastogi, and A. Silberschatz. A low-cost storage server for movie on demand databases. In *Proceedings of the 20th VLDB Conference*, pages 594–605, Santiago, Chile, 1994.

[ORS95] B. Ozden, R. Rastogi, and A. Silberschatz. Demand Paging for Video-on-Demand Servers. In *Proc. of Intl. Conf. on Multimedia Computing and Systems*, Washington, D.C., May 1995.

[RZ95] D. Rotem and J.L. Zhao. Buffer Management for Video Database Systems. In *Proc. Intl. Conference on Data Engineering*, pages 439–448, Taipei, Taiwan, March 1995.

[TPBG93] F.A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID - A Disk Array Management System For Video Files. In *Proceedings of ACM Multimedia'93*, pages 393–399, Anaheim,CA, 1993.

[YWS95] 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 Journal of Multimedia Systems*, 3(4):137–150, 1995.

Appendix A

Appendix

The derivation of the mathematical model for pause and rewind is very similar to the derivation in Chapter 4 in the context of fast forwarding. Except that, using V_f as a reference point in the derivation, we will use V_l, since in the case of pause and rewind the viewer in question is "catching up" with the viewers lagging behind him rather than those ahead of him.

A.1 Rewind

Let the pdf of rewind be $g(x)$, where $x \in [0, l]$. As before, a hit can belong to one of two events, a hit_w or a hit_o.

A.1.1 Hits occurring within the same partition (hit_w)

In this case, a viewer can rewind until the last viewer, at position V_l, catches up with him. Therefore, we have

$$P(hit_w|RW, V_c, V_l) = \int_0^{\gamma\Delta_w} g(x)dx \quad \text{where} \quad \Delta_w = V_c - V_l \tag{A.1}$$

- Unconditioning on V_l

  The unconditioning on V_l can be divided into two cases: (1) V_c is larger than $\frac{B}{n}$ and thus it is possible to catch up the last viewer in the same partition regardless of his position, i.e., $V_l = V_c - \frac{B}{n}$ and (2) V_c is smaller than $\frac{B}{n}$, and thus the last viewer's position, V_l must be equal to 0. Given these two cases, we have

$$P_1(hit_w|RW, V_c) = \int_{V_l=V_c-\frac{B}{n}}^{V_c} P(hit_w|RW, V_c, V_l)P(V_l)dV_l \tag{A.2}$$

$$P_2(hit_w|RW, V_c) = \int_{V_l=0}^{V_c} P(hit_w|RW, V_c, V_l)P(V_l)dV_l \quad (A.3)$$

Using the assumption that the arrival process is Poisson, the quantity $P(V_l)$ in the two cases above is equal to $\frac{1}{B/n}$ and $\frac{1}{V_c}$, respectively.

- Unconditioning on V_c

  The second case above handles those viewers whose position, V_c, is smaller than $\frac{B}{n}$. Thus, we uncondition on V_c where $\frac{B}{n} \leq V_c \leq l$ and $0 \leq V_c \leq \frac{B}{n}$ for the first and second cases, respectively.

$$P_1(hit_w|RW) = \int_{V_c=\frac{B}{n}}^{l} P_1(hit_w|RW, V_c)P(V_c)dV_c \quad (A.4)$$

$$P_2(hit_w|RW) = \int_{V_c=0}^{\frac{B}{n}} P_1(hit_w|RW, V_c)P(V_c)dV_c \quad (A.5)$$

  $P(V_c)$ is equal to $\frac{1}{l}$ as in Chapter 4. Combining the two cases, the probability of a hit_w after a rewind request, $P(hit_w|RW)$, is computed by summing Equations (A.4) & (A.5).

A.1.2 Jump to other partitions (hit_o)

For a viewer to resume at the i^{th} partition *behind* the current partition, he must rewind at least long enough to catch up to the first viewer, at position V_{f_i}. A hit_o after a rewind can be divided into two cases: (1) a complete hit_o and (2) a partial hit_o, as in the case of fast-forwarding.

- complete hit_o

  As in Chapter 4, we define Δ_{jump_f} and Δ_{jump_l} as the shortest and the longest duration a viewer must rewind to have a complete hit_o at the i^{th} partition behind the current partition as follows.

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

  $P_c(hit_o^i|RW, V_c, V_l)$, the probability of a complete hit_o at the i^{th} partition, given that the viewer is at position V_c and the last viewer is at position V_l is calculated

as follows.

$$P_c(hit_o^i|RW, V_c, V_l) = \int_{\gamma\Delta_{jump_f}}^{\gamma\Delta_{jump_l}} g(x)dx \tag{A.6}$$

- partial hit_o

  The partition which contains the beginning frames of the movie is termed as the first partition. For the cases where V_f in this partition is smaller than the partition size, $\frac{B}{n}$, the difference between V_f and V_l will be smaller than $\frac{B}{n}$, i.e., V_l is at the 0^{th} minute. In this case, the viewer can rewind for at most γV_c minutes before he reaches the 0^{th} minute (where γ is defined in Chapter 4). Therefore, $P_p(hit_o^i|RW, V_c, V_f)$, the probability of a partial hit_o at the i^{th} partition behind the current partition, given that the viewer is at position V_c and that the last viewer is at V_l, is calculated as follows.

$$P_p(hit_o^i|RW, V_c, V_l) = \int_{\gamma\Delta_{jump_f}}^{\gamma V_c} g(x)dx \tag{A.7}$$

- Unconditioning on V_l

  When unconditioning on V_l, we consider a complete hit_o and a partial hit_o separately.

  - complete hit_o

    case a: able to catch up to viewers at positions V_l and V_f in the i^{th} partition behind the current partition, i.e., $V_l = V_c - \frac{B}{n}$

$$P_1(hit_o^i|RW, V_c) = \int_{V_l=V_c-\frac{B}{n}}^{V_c} P_c(hit_o^i|RW, V_c, V_l)P(V_l)dV_l \tag{A.8}$$

    case b: able to catch up with a viewer at the position with the smallest possible value of V_l

    In this case, the smallest value of V_l must be equal to $\frac{il}{n}$. Then,

$$P_2(hit_o^i|RW, V_c) = \int_{V_l=\frac{il}{n}}^{V_c} P_c(hit_o^i|RW, V_c, V_l)P(V_l)dV_l \tag{A.9}$$

  - partial hit_o

    The first equation below calculates the probability of a partial hit_o for those viewers which also have a complete hit for $\frac{il}{n} \leq V_l \leq V_c$. The second equation below corresponds to those viewers which only have a partial hit_o for all

values of V_l.

$$P_3(hit_o^i|RW, V_c) = \int_{V_l=Vc-\frac{B}{n}}^{\frac{il}{n}} P_p(hit_o^i|RW, V_c, V_l)P(V_l)dV_l \quad \text{(A.10)}$$

$$P_4(hit_o^i|RW, V_c) = \int_{V_l=\frac{il-B}{n}}^{V_c} P_p(hit_o^i|RW, V_c, V_l)P(V_l)dV_l \quad \text{(A.11)}$$

- Unconditioning on V_c

$$P_1(hit_o^i|RW) = \int_{V_c=\frac{il+B}{n}}^{l} P_1(hit_o^i|RW, V_c)\ P(V_c)dV_c \quad \text{(A.12)}$$

$$P_2(hit_o^i|RW) = \int_{V_c=\frac{il}{n}}^{\frac{il+B}{n}} P_2(hit_o^i|RW, V_c)\ P(V_c)dV_c \quad \text{(A.13)}$$

$$P_3(hit_o^i|RW) = \int_{V_c=\frac{il}{n}}^{\frac{il+B}{n}} P_3(hit_o^i|RW, V_c)\ P(V_c)dV_c \quad \text{(A.14)}$$

$$P_4(hit_o^i|RW) = \int_{V_c=\frac{il-B}{n}}^{\frac{il}{n}} P_4(hit_o^i|RW, V_c)\ P(V_c)dV_c \quad \text{(A.15)}$$

The probability of a hit_o^i for RW, $P(hit_o^i|RW)$, is obtained by summing Equations (A.12) through (A.15).

The number of partitions through which a viewer can rewind is equal to $n-1$. Furthermore, during a rewind operation, it is possible to catch up to any viewer which exists when the rewind request is issued. In contrast to fast-forward, some viewers may have finished the movie before a catch-up can be made. Therefore the probability of a hit, given that the VCR request is a rewind is:

$$P(hit|RW) = P(hit_w|RW) + \sum_{i=1}^{n-1} P(hit_o^i|RW) \quad \text{(A.16)}$$

A.2 Pause

The derivation for the pause operation is the same as that for the rewind operation, except that the γ factor in Equations (A.1), (A.6), and (A.7) is equal to 1. Therefore, we do not repeat the derivation here.

CUHK Libraries
003589437