# High performance digital video servers: storage and retrieval of compressed scalable video

*Seungyup Paek and Shih-Fu Chang*
Department of Electrical Engineering
Columbia University
New York, N.Y. 10027-6699, U.S.A.

**Abstract.**Developments in the fields of computing, computer networks and digital video compression technology have led to the emergence of unprecedented forms of video communications. In the future, it is envisioned that users will be able to connect to a massive number of distributed *video servers*, from which users will be able to select and receive high quality video and audio. High performance video servers will store a large number of compressed digital videos and allow multiple concurrent clients to connect and retrieve a video from the collection of videos. High performance video servers must provide guaranteed *Quality of Service* for each video stream that is being supported. The video server has to retrieve multiple video streams from the storage system and transmit the video data into the computer network. There has been a great deal of research and development in the past few years on the various issues related to high performance video servers. In this chapter, we provide an overview and snapshot of some of the main research areas that have been worked on recently. We critically evaluate and summarize the main research results. In particular, we focus on the interrelated issues of digital video compression, storage and retrieval for video servers.

# Contents

# 1 Introduction

Developments in the fields of computing, computer networks and digital video compression technology have led to the emergence of unprecedented forms of video communications. Advances in computer networks have led to broadband networks that can handle much higher data rates with greater reliability. Advances in state of the art digital video compression technologies have greatly reduced the data rate and storage requirements of digital video, together with the added flexibility of *scalable resolutions*. Advances in computing have led to multimedia workstations for the home that can provide high quality video and audio. One of the forms of video communication that are enabled by and dependent on the confluence of these key technologies is Video on Demand (VoD).

In VoD, multiple users will be able to connect to remote digital video libraries and view videos 'on-demand'. It is envisioned that broadband computer networks will allow users to connect to a massive number of distributed 'video servers', from which users will be able to select and receive high quality video and audio. One of the critical computing systems of VoD is the *video server*. High performance video servers will store a large number of compressed digital videos and allow multiple concurrent clients to connect over the computer network to retrieve a video from the collection of videos.

High performance video servers not only have to store and archive a large number of compressed digital videos, but must also provide guaranteed Quality of Service for each video stream that is being supported. The video server has to retrieve multiple video streams from the storage system and transmit the video data into the computer network.

There has been a great deal of research and development in the past few years on the various issues related to high performance video servers. There have been many proposals for different algorithms to maximize the utilization of computing resources, and also many proposals for the architecture of high performance video servers. In this work, we provide an overview and snapshot of the main research areas that have been worked on recently and describe why each area has been considered to be important. We critically evaluate and summarize the main research results that have been presented. In particular, we focus on the interrelated issues of digital video compression, storage, retrieval and network transmission in video servers.

# 2 Compressed MPEG video

High performance video servers store a large number of compressed digital videos and allow multiple concurrent clients to connect over the computer network to retrieve a video from the collection of videos. Due to the extremely large storage and bandwidth requirements of digital videos, it is very important to compress the video data for cost effective storage and transmission in video servers. Video compression technology addresses the problem of how to reduce the data rate and storage requirements of digital videos, without significantly reducing the visual quality of videos.

In the last few decades, image and video compression has been a dominant topic in the image and video processing community, and it will undoubtedly continue to be a critical technology for the future of emerging multimedia applications [18]. This is because video sequences can require very large bandwidths and storage when represented in digital form. This can be demonstrated by a simple example.

Consider an image with 720x480 pixels. If the picture is in color, 3 bytes can be used for each pixel i.e. one byte for each color component (R, G, B) of a pixel. This means that each picture is 1037 KBytes. If a video sequence is comprised of a sequence of such images at 24 frames per second, the video sequence will have a data rate of 200 Mbps. For a one hour video, the storage requirement for uncompressed video is approximately 90GB. At current magnetic disk technologies, this corresponds to about 44 hard disks to store a single video. Clearly, it is not cost effective to store uncompressed digital videos with current technologies. With state of the art video compression, the above uncompressed video sequence with a data rate of 200 Mbps can be compressed to data rates as low as 4 Mbps. Therefore, the storage requirement for compressed video will be approximately 2GB.

In order to understand video server technologies, it is important to understand video compression technologies. In this section we briefly overview the MPEG (Motion Picture Experts Group) video compression standard, which is being adopted as a world wide standard for compressed digital video.

The MPEG-1 standard is a video compression standard for digital storage media applications. The MPEG-2 standard is a follow on to MPEG-1 and is intended primarily for higher bit rates, larger picture sizes, and interlaced video frames. The MPEG-2 standard builds directly upon MPEG-1, and is a relatively straight forward extension of MPEG-1. However, MPEG-2 provides a different set of capabilities, including advanced techniques for HDTV (High Definition Television). In particular, MPEG-2 provides a set of scalable extensions to provide video with multiple resolutions. This will be discussed further below.

The MPEG-1 and MPEG-2 standards are actually comprised of several parts. The video part of the MPEG standards is aimed at the compression of video sequences. The audio part is aimed at the compression of audio data. The systems part deals with issues such as the multiplexing of multiple audio

and video streams, and the synchronization between different streams. In this section, we briefly overview the video part of the MPEG standards. An in depth coverage of the MPEG standards can be found in [22]. The overview presented here is based on [22].

MPEG video compression is based on both *inter-frame* and *intra-frame* techniques. Inter-frame techniques refer to compression techniques in which information in adjacent frames of a video sequence are used to compress a given frame. Intra-frame techniques refer to compression techniques in which a frame is compressed independent of information in any other frames. Inter-frame techniques are very effective in video compression because there is a great deal of redundancy between adjacent frames i.e. adjacent frames of a video generally have very little variation. Consider a scene in a film in which two people are talking. If we look at the frames of the video, we will see that adjacent frames are almost identical, with minor variations in the expression on the faces of the people talking. Therefore, it is not necessary to compress all the information in each frame of the video independently of each other. Intuitively, we can see that significant compression can be achieved by only encoding the variations between successive frames of a video.

We will first describe the syntax of MPEG compressed video bitstreams. The outermost layer of an MPEG video bitstream is the video sequence layer. The video sequence layer is divided into consecutive groups of pictures (gop), as shown in Figure 1. Each gop is composed of pictures that are either I, P, or B pictures. I pictures are coded independently with intra-frame techniques. P,B pictures are compressed by coding the difference between the picture and reference pictures that are either I or P pictures. P (predictive-coded) pictures are coded by using information from temporally preceding I,P pictures. B (bidirectionally predictive-coded) pictures obtain information from the nearest preceding and/or following I,P pictures.

The basic building block of each MPEG picture is the macroblock. Each macroblock is composed of 4 8x8 blocks of luminance samples in addition to two 8x8 blocks of chrominance samples (one for Cb and one for Cr). An MPEG picture is not simply a sequence of macroblocks but is composed of consecutive slices, where each slice is a contiguous sequence of macroblocks. We will now briefly describe the main techniques used for MPEG video compression.

*Discrete Cosine Transform.* The Discrete Cosine Transform (DCT) algorithm in MPEG video is the basis of both intra-frame and inter-frame coding. The DCT has properties that simplify the coding model. Basically, the DCT decomposes a block of image data into a weighted sum of spatial frequencies. For example, in MPEG, an 8x8 block of pixels is represented as a weighted sum of 64 two dimensional spatial frequencies. If only low frequency DCT coefficients are non-zero, the data in the block varies slowly. If high frequency coefficients are present (non-zero), the block intensity changes rapidly from pixel to pixel within the 8x8 block.

*Quantization.* In MPEG, the DCT is computed for a block of 8x8 pixels. It is

desirable to represent coefficients for high spatial frequencies with less precision. This is referred to as quantization. DCT coefficients are quantized by dividing it by a non-zero positive integer called the quantization value, followed by rounding the quotient. The bigger the quantization value, the lower the precision of a quantized DCT coefficient. Lower precision coefficients can be transmitted to a decoder with fewer bits. MPEG uses larger quantization values for higher spatial frequencies. This allows the encoder to selectively discard higher spatial frequencies.

A macroblock is composed of 4 8x8 blocks of luminance samples and two 8x8 blocks for each of two chrominance samples. Chrominance samples represent color in terms of the presence or absence of red and blue for a given luminance intensity. 8x8 blocks of data are the basic units of data processed by the DCT. A lower resolution is used for the chrominance blocks since the human eye resolves high spatial frequencies in luminance better than chrominance. This sub-sampling also contributes significantly to the compression.

The DCT has several advantages from the point of view of data compression. For intra-frame coding, coefficients have nearly complete decorrelation. Therefore, the coefficients can be coded independently. In inter-frame coding, the difference between the current picture and a picture already transmitted is coded. The DCT does not really improve decorrelation. However, the main compression gain is mostly by visually weighted quantization.

*Motion compensation.* If there is motion in a video sequence, better compression is obtained by coding differences relative to areas that are shifted with respect to an area being coded. The process of determining motion vectors in an encoder is called motion estimation. Motion vectors describing the direction and amount of motion of macroblocks are transmitted to decoders.

In MPEG, the quantized DCT coefficients are coded losslesly. The DCT coefficients are organized in a zig zag scan order, in which the order approximately orders coefficients in ascending spatial frequency. Visually weighted quantization strongly deemphasizes higher spatial frequencies. Therefore only few lower frequency coefficients are non-zero in a typical transformation.

## 2.1 Variable bit rate and constant bit rate MPEG video

The variable bit rate of MPEG2 video is dependent on the encoding structure of the MPEG2 coding algorithm. In the MPEG2 digital video technology, compression is achieved by the combination of techniques such as the discrete cosine transformation (DCT), variable length codes, quantization of DCT coefficients, motion estimation and motion compensated inter-frame prediction. MPEG2 has a buffer control mechanism in which the quantization parameter can be varied adaptively in order to achieve a constant average bit rate of the compressed video. The disadvantage of this mechanism is that the subjective visual quality will be variable, since the quantization parameter is continually varied. An alternative is to maintain a constant quantization parameter during the encoding of video. This results in variable bit rate video, in which the amount of data to represent different time scales of video (macroblock, slice, frame, group of pictures etc.) are variable. Figure 2 shows the data trace of a variable bit rate encoded MPEG-2 video sequence.

## 2.2 Scalable MPEG video

Compared to simulcast coding, scalable coding schemes can provide multiple levels of video with a minimal cost of extra bandwidth or storage capacity. In scalable video coding, subsets of the full resolution bitstream are used to obtain subsets of the full resolution video [12]. Scalable video will be used in advanced computer networks to support heterogeneous clients. Mobile wireless clients may only have computing resources to receive the lowest layer of video, while high performance workstations will request all the scalable layers of video.

The MPEG2 standard allows a combination of spatial, SNR (signal-to-noise ratio) and temporal scalability for up to three layer coding of video sequences. In one possible hybrid, three layer scalable coding scheme, the base layer provides the initial resolution of video. The spatial enhancement layer enables the upsampling and hence increase in frame size of the base layer. Finally, the SNR enhancement layer increases the visual quality of the (base+spatial enhancement) layers of video. In another scheme for MPEG-2 video, three layer temporally scalable video is achieved as follows. The lowest scalable layer is comprised of the I frames of a video (I layer). The P frames (P layer) enable an increase in the temporal resolution of the I frame layer. Finally, the B frames (B frames) increase the temporal resolution of the I+P layers. We refer to this as IPB scalable video. In this scheme, scalabilty is inherently provided by the MPEG-2 encoding structure.

Figure 1: MPEG Group of pictures

Figure 2: MPEG-2 variable bit rate trace data

# 3 Inter-disk data placement of constant bit rate video

In this section we focus on the storage system of a video server. Specifically, we will consider storage systems that are based on a parallel array of independent magnetic disks. The way in which video data is stored on the magnetic disk systems can have a significant impact on the performance of a video server. We will refer to the way in which data is stored on an array of disks as the data placement scheme.

In this section, data placement refers to inter-disk data placement, and not intra-disk data placement. Inter-disk data placement refers to how video data is distributed across multiple disks, whereas intra-disk data placement refers to how data is stored within a single disk.

Given that a set of videos have to be stored on a set of disks, data placement schemes are important in achieving load balancing in video servers. Consider a simple data placement strategy in which an entire video is stored on a single disk. The advantage of this scheme is that it is simple. The disadvantage is the lack of load balancing. If all users connecting to a video server request a video that is stored on one disk, the disk storing the popular requested video will be fully utilized. However, all the other disks will remain idle. If the data for the popular video was distributed over all the disks, the combined throughput of all the disks could have been used to allow more clients to connect to the server to view the videos.

In this section we overview the research in the placement of video data on a parallel array of disks. We will show how the performance of a video server depends on the data placement scheme.

In the data placement of videos on an array of disks, we will show that there is a basic trade-off between the worst case interactivity delay and the utilization of disks. For the guaranteed retrieval of video data, if the data placement scheme maximizes the utilization of the disk systems, the worst case interactivity delay is shown to be at a maximum. Conversely, if a data placement scheme minimizes the worst case interactivity delay, the utilization of the disk system is at a minimum. In relation to the data placement scheme, we also consider how scalable video can improve the performance of video servers.

Research on scalable video data placement in which the utilization of the disk system is maximized is presented in [6]. However, the proposed scheme has a large worst case start up and interactivity delay. In [19], a multiresolution video data placement scheme is presented in which the interactivity delay is minimized, but in which the utilization of the disks is low. In contrast to the data placement schemes presented in [6, 19] a data placement scheme is presented in [25] in which different videos can have a range of interactivity and hence disk utilization performance. On one end of the spectrum, the utilization of the disks is maximized (hence increasing the number of concurrent video

streams). On the other end, the maximum interactivity delay is minimized. The flexibility of this strategy is that different videos can operate at different points of this performance spectrum to provide a range of interactivity QoS. This is in contrast to the schemes presented in [6, 19], in which the performance is at the extreme points of the performance spectrum. It is also shown how the performance of a video server supporting scalable video can be improved for the proposed data placement strategy.

This section is organized as follows. We first present the system model of the video server that we will consider. Then, two extreme strategies for the placement of video data on a parallel array of disks is compared. These schemes show the tradeoff between the worst case interactivity delay and the utilization of disks. For each scheme, advantages and disadvantages are compared. Finally, a flexible strategy for the placement of video data on a parallel array of disks is presented. It is also shown how scalable video can improve the overall utilization and interactivity performance of a video server based on the proposed data placement strategy.

## 3.1 System model

We consider a parallel array of disks each connected in parallel to a memory system. The details of magnetic disk systems can be found in [10]. Consider the operation of a single disk. When a request for an I/O operation is received at a disk, two types of overhead are incurred before data can be transferred from the disk to the memory. These are as follows: the time it takes for the head to move to the appropriate cylinder (referred to as the seek time), and the time it takes for the first sector to appear under it (referred to as the rotation latency). Following this overhead, the transfer of the data begins. The transfer time for a request is a function of the total data requested.

Each disk is assumed to use the SCAN disk head scheduling algorithm. [9] covers disk head scheduling algorithms in detail. In the SCAN scheduling algorithm, the scanning cycle consists of two phases. During the first cycle, the head scans the disk from the inner most track to the outer most track. While scanning the disk, data blocks belonging to different streams are read from the disk. Upon reaching the outer most track, the head is returned to the initial position.

The disks of a video server are assumed to operate on a cycle. For every cycle of the video server, each disk completes one complete SCAN cycle. In this section, we assume that videos have constant bit rate. Therefore, in each cycle, one retrieval block of video data has to be retrieved for every video stream. The data placement scheme determines how each retrieval block of a video is stored on a parallel array of disks. If each of the retrieval blocks are stored on single disks, a higher utilization efficiency can be achieved for the disks. This is because for each disk seek overhead, more data is retrieved. However, a larger buffer size will be required. Each video stream is serviced in a round robin

fashion during each cycle. The retrieval block is a fixed number of frames that are referred to as a group of frames (gof).

For an approximate analysis of the utilization of a single disk using the SCAN disk head scheduling algorithm, several assumptions are made. Firstly, any stream accessed during the first phase will add to the total retrieval cycle the maximum rotational latency, the data reading time and the minimum seek time. Secondly, since the retrieval cycle consists of two phases of head movement, we add two maximum seek delays to the total cycle time.

It is shown that utilization of a single disk system is as follows:

$$\rho = \frac{R_p \cdot S_{max}}{N_d \cdot R_d} \tag{1}$$

($S_{max}$ is the maximum number of video streams that the parallel array of disks can support, $R_p$ is the video playback rate and $R_d$ is the maximum disk transfer rate). As shown in [25], $S_{max}$ can be derived from the following equation, where $T_{cycle}$ is the round robin cycle time, $T_{sx}$ is maximum seek time, $T_{sm}$ is the minimum seek time and $T_{rx}$ is the maximum rotation latency:

$$T_{cycle} = S_{max} \cdot \cdot (\frac{T_{cycle} \cdot R_p}{N_d \cdot R_d} + T_{rx} + T_{sm}) + 2 \cdot \cdot T_{sx} \tag{2}$$

*Interactivity QoS*

In advanced digital video systems of the future, we reconsider the commonly accepted notions of interactivity. The goal for interactivity in the video server is not to 'simulate' VCR functions exactly but to achieve effective search mechanisms while efficiently utilizing the limited resources of a video server. We propose that the critical functions of interactivity that are required for video servers are location of specific scenes and multiple rate 'scanning' of video segments (fast/slow forward/reverse).

## 3.2 Balanced placement

This scheme has the lowest interactivity delay, however, the utilization of each disk is low. The interactivity delay (defined in Figure 3) of this scheme is one cycle. For example, if a user pauses the playback of a video stream and after some time requests that the video stream be resumed, the video stream would be able to resume in the following cycle.

In this scheme, each group of frames (gof) of each resolution of video is divided into $N_d$ equal segments and placed over all $N_d$ disks. In [19] a similar data placement strategy is presented, in which the full resolution gof is segmented to $N_d$ segments. For the balanced placement strategy, since data in each gof is distributed over all disks, the data retrieved in each disk for each disk seek is small. Therefore, the disk utilization is low.

### 3.3 Periodic placement

This scheme [6] represents the opposite side of the interactivity QoS. This scheme maximizes disk utilization, however the worst case access and interactivity delay will be shown to be $N_d$ cycles. This is in contrast to the balanced data placement scheme, in which the delay is always one cycle.

For each video, consecutive gof are placed on consecutive disks in a round robin fashion. For every cycle, one gof is retrieved for every video stream connected to the video server. Each gof is retrieved from a single disk during a cycle (compared to multiple disks in the balanced placement scheme). If a single disk can support n gof retrievals in one cycle, then $N_d$ disks support $(N_d \cdot n)$ video streams concurrently. The observation is made that for a video stream starting retrieval of video data at cycle $r$, the video stream accesses a different single disk during each cycle. However, the video stream accesses the same single disk during each cycle as all video streams with start cycles in the following set:

$$\{r_i, i = 1, 2, ...N_s | (r_i \text{ modulo } N_d = r \text{ modulo } N_d)\} \tag{3}$$

$r_i$ denotes the start cycle of video stream $i$ and we assume the first gof of all videos are stored on the same disk. This observation shows that for all video streams connected to the video server, we can group the video streams into $N_d$ video stream sets.

All video streams in a video stream set retrieve data from the same disk during any given cycle (Figure 4). It can be shown that the worst-case interactivity delay for a video stream is $N_d$ cycles for any of the equivalent interactivity functions. To prove this, we first note that each of the interactive functions are equivalent in that a specific required gof must be retrieved from the array of disks. The number of video streams being serviced on the disk that contains the required gof is the number of video streams in the video stream set that is accessing the disk during a particular cycle. The required gof cannot be accessed until a video stream set that can accommodate a new video stream is accessing the appropriate disk. Since the total number of sets are $N_d$, the maximum access delay before retrieval is $N_d$. We can also show that the scan granularity for this scheme is $N_d$. It has been shown that for regular playback, a video stream $j$ accesses consecutive disks to retrieve consecutive gof. If video stream $j$ requires a forward scan while only utilizing the resources reserved in its video stream set, we can show that the scan granularity is $g = N_d + 1$.

### 3.4 Multiple segmentation placement of scalable video

This scheme [25] is flexible in that it allows videos to take on a range of maximum interactive delay and scan granularity values. It is shown that decreasing interactivity delay can be achieved at the cost of decreasing utilization efficiency. Therefore, there is a design range for the placement of video data. The scheme

presented here uses different degrees of segmentation of gof blocks for the placement of gof blocks across a parallel array of disks. We first present the Multiple segmentation (MS) scheme:

1. For a parallel array of $N_d$ disks, define $(\log_2 N_d) + 1$ segmentation levels: $S = S_i = 2^i, i = 0, 1, ... \log_2 N_d$

2. For a given segmentation level $S$, divide each gof into $S$ equal segments.

3. For a given segmentation level $S$, specify $(N_d/S)$ sets of disks.

4. For each video sequence, the consecutive retrieval blocks (gof) which were each divided into S equal segments are stored on consecutive sets of disks as in xx.

Balanced placement is a special case of multiple segmentation with $S = 8$, while periodic placement is a special case with $S = 1$. It is shown that increasing the segmentation $S$ reduces the maximum interactivity delay at the price of utilization efficiency. For a segmentation level $S$, a video stream accesses $S$ disks during each cycle.

Extending the structure of video stream sets, we develop the structure of component video stream sets. For a parallel array of $N_d$ disks, we define $N_d$ component video stream sets. For a video $j$ that is stored with segmentation level $S$, we say that $S$ component video streams are required for a single video stream of video $j$. Therefore, resources are reserved on $S$ component video stream sets for the retrieval of a single video stream for video $j$. Figure 5 shows which component video stream sets are used for the retrieval of a given video stream at cycle $r$ stored with segmentation level $S$.

All component video streams in a video stream set retrieve data from the same disk during any given cycle. Based on this, it can easily be shown that the worst-case interactivity delay for a video stored with segmentation level $S$ is $N_d/S$ cycles for any of the equivalent interactivity functions.

Suppose that a video $j$ is stored with segmentation level $S = 2$ on an $N_d = 8$ disk array. Consider a video stream that has reserved resources on component video stream sets $(0, 4)$, with all other video stream sets exhausted by other video streams. Assume a request for a gof stored on disks $(0, 4)$ is made during cycle $r$, and the desired start gof is stored on the disks which have just been accessed during cycle $r$. The delay before the desired gof can be accessed from the appropriate disks is $N_d/S = 4$.

In summary, a video stored with segmentation level $S$ requires $S$ component video streams. If resources are reserved on S component video stream sets, a maximum of $N_d/S$ cycles are required before a given set of $S$ component video stream sets has accessed all disks. Using a similar analysis, we can also show that the scan granularity for this scheme is $N_d/S$.

This scheme has advantages in flexibility in that videos with high interactivity delay tolerance can be stored with a smaller segmentation level, and videos requiring low interactivity delay are stored with higher segmentation levels. Multiple levels of segmentation can be supported on the same array of disks in a video server to provide a range of interactivity QoS.

The proposed multiple segmentation scheme can be used to segment each gof of each layer of scalable video into $S$ segments. The specific value of $S$ to use is a design parameter that can be chosen by the system designer for each video in the video server, depending on its access requirements. We now consider how to divide each gof of each resolution (layer) of video into $S$ segments. Each gof of each layer consists of a sequence of I, B, P frames of MPEG2 video. There are two basic ways to segment the gof of a given layer, as shown in Figure 6. Using method 1, we see that if one segment is not retrieved, all frames of a gof will be affected. Method 2 is clearly a better option. Furthermore, based on method 2, we may group together frames of the same type (I, P, B) before segmentation. In this way, we assign the highest priority to segments containing I frames, intermediate priority to segments containing P frames, and the lowest priority to segments containing B frames. Segmentation does not occur exactly at frame boundaries. Each segment has an associated priority, and the priorities can be used in the video server scheduler to selectively drop segments to achieve graceful degradation in the case of congestion. In addition, further granularity in interactive scan functions can be easily achieved by skipping B and/or P frames. In many real time applications or near real time applications for which fast responses are critical, lower layers may be segmented with a higher level (method 3, Figure 6) so that lower layers can be retrieved with shorter delays for a high degree of interactivity. This can be used for progressive retrieval in which lower layers are displayed before full resolution layers are fully retrieved.

*Admission Control Framework Based on Multiple Segmentation.*

The multiple segmentation placement strategy has a simple admission control framework. It was shown that each incoming video stream can be decomposed into a number of component video streams. Higher segmentation levels require more component video streams for a single video stream. Admission control at the video server is an operation at the call establishment level for a video stream request at a video server. Given an incoming request with a specific QoS requirement, the admission control must decide to accept or reject the call. The policy has to determine if the request can be serviced by the video server while maintaining heterogeneous QoS requirements of all video streams already connected to the video server. The challenge is to maximize the utilization of the video server resources while ensuring heterogeneous QoS requirements of connected video streams. For a parallel array of $N_d$ disks, we define $N_d$ component video stream sets (CVSS). All component streams in a given CVSS retrieve video data from the same single disk during a given cycle. The component video streams in a CVSS are said to be connected to the same logical disk. The CVSS simplification provides a strategy for admission control in the video server. In the video server, we maintain a single CVSS admission control table. For each incoming video stream, we update the corresponding CVSS entries accordingly. Note that depending on the resolution of the video stream, we calculate whether the incoming video stream can be supported on the each logical disk associated with each CVSS. All logical disks are assumed

to be identical with the same disk characteristics.

## 3.5 Fault tolerant video storage

In very large scale video servers, a very important issue is that of fault tolerance [2]. A large scale video server can potentially have 1000 (1 GB) disks. This would provide enough storage for approximately 300 MPEG-2 movies at 4.5 Mbps or 900 MPEG-1 movies at 1.5 Mbps. Assuming a bandwidth of 4 Mbytes per second, 1000 disk drives provide enough bandwidth to support approximately 6500 concurrent MPEG-2 users or 20000 MPEG-1 users.

Although a single disk can be fairly reliable, given such a large number of disks, the aggregate rate of disk failures can be too high. The *Mean Time To Failure (MTTF)*of a single disk is on the order of 300000 hours. Therefore, the MTTF of some disk in a 1000 disk system is on the order of 300 hours (approx. 12 days).

In large scale video servers, all the video data may be stored on tape drives, with a subset being stored on the disks. Therefore, data loss on disks can always be recovered. However, a disk failure can result in the interruption of requests in progress.

If a video being retrieved from disks and transmitted into the network is stored on the failed disk, this can lead to degraded video quality at the client. In large scale video servers, as shown in the previous section, it is likely that each video will be striped over multiple disks in order to improve the load balancing in video servers. In that case, multiple videos can have a portion of their data stored on a failed disk. This means that a single disk failure can lead to degraded video quality for multiple videos.

Once a disk has failed, one option is to restore the data for multiple videos that were stored on the failed disk from the tape archive onto a new disk. This can be a very slow process, since data for multiple videos has to be retrieved from robotic tape archives onto the disk. Therefore, without some form of fault tolerance, such a system is not likely to be acceptable.

Reliability and availability can be improved by using a fraction of disk space to store redundant information. Typically parity schemes and mirroring schemes are used for this purpose. For example, in a disk system with 5 disks, 4 disks may be used to store actual data, while a fifth disk is used to store parity information. In this example, four fifths of the total storage space is used to store data, and four fifths of the disk bandwidth is used to retrieve data from disks.

In [2] two observations are stated for the design of fault tolerant video servers.

*Observation 1*: One should not mix data blocks of different objects in the same parity group. If this observation is violated, there may not be enough disk bandwidth to reconstruct data on the fly in the event of a disk failure.

*Observation 2*: To avoid degradation in video quality when a failure occurs, the first fragment in a parity group cannot be scheduled for transmission over

the network until the entire parity group has been read from the disk.

We note that in the unlikely event of two disks failing in the same parity group, a rebuild from tape drives would have to be performed. This is referred to as a *catastrophic failure*.

The goal of fault tolerant video servers is to achieve very low probabilities of catastrophic failures with a minimum of increase in disk storage, disk bandwidth and buffer requirements.

There is another serious type of system failure which is called *degradation of service* [2]. This occurs when there is insufficient available disk bandwidth, due to failures, to continue delivering all active requests. One way this can happen is if observation 1 is violated. For example, suppose that a parity group contains fragments of video $x$ and video $y$. If video $x$ was already being retrieved, then disk bandwidths have alredy been reserved for its retrieval. However, if video $y$ was not being retrieved, then there may not be enough disk bandwidth available for its retrieval when a disk failure occurs.

The Streaming RAID (Redundant Array of Independent Disks) scheme was first proposed in [30]. For fault tolerance, disks are grouped into fixed sized clusters of $C$ disks, each with $C-1$ disks and one parity disk. The set of data blocks, one per data disk, and a parity block on the parity disk form a *parity group*. The parity block is the bitwise exlcusive or of the data blocks. Each video is striped over all the data disks. The sequence of parity groups associated with an object are allocated in a round-robin fashion over all of the clusters. For example, parity groups one, two, three are stored on clusters one, two, three respectively. For each active stream, a single parity group is read from a single cluster in a single cycle and delivered to the network in the subsequent cycle.

With this scheme, in the event of a disk failure, the missing data can be reconstructed by a parity computation. For each active stream, an entire parity group is read from a cluster in each cycle. If a disk failure has occurred on that cluster, the parity block is also read from the parity disk and the lost data is reconstructed *on the fly*.

The Streaming RAID scheme achieves fault tolerance of up to one disk per cluster. If more than one disk fails in a single cluster, then there is a catastrophic failure. In this case, data has to be rebuild on a new disk from tape archives.

Streaming RAID achieves fault tolerance at the cost of disk bandwidth and storage, since a portion of the disk resources are used for redundant parity information.

Figure 3: Interactivity QoS

Figure 4: Periodic data placement

Figure 5: Component video stream sets

Figure 6: Segmentation of MPEG-2 scalable video

# 4 Buffer replacement algorithms

Disks are a primary form of storage for computer systems. In disk based computer systems, a *buffer cache* is used to reduce the number of disk I/O. When a request is made for data, the operating system first checks to see if the data is available in a memory cache. In this way, disk I/O can be reduced. A *buffer manager* is responsible for buffer replacement to free memory to accomodate new data blocks from disk. Each time data is retrieved from disk, the buffer manager has to decide which data in the buffer cache to remove in order to store the newly retrieved data. The *buffer replacement algorithm* has the goal of reducing the total number of cache misses. An optimal buffer replacement achieves the lowest number of cache misses, and hence the lowest number of disk I/O. In general the optimal algorithm is unachievable as it requires exact future knowledge of data requests. Most disk systems use approximation algorithms such as *Least Recently Used* (LRU) and *Most Recently Used* (MRU). However, it can be shown that both algorithms yield poor performance for continous media data such as video.

Access to continous media data requires rate guarantees so that clients can meet their timing constraints. To ensure rate guarantees, buffer space and disk bandwidth are reserved for each client on a video server. In this section, we will assume that a global buffer cache is used in which data can be shared among all the clients. The rationale for using a cache is to reduce disk I/O.

Reducing disk I/O is important due to the following reason. Video servers need to handle continous media data as well as conventional data. Continous media data requires a high I/O rate. Furthermore, continous media data requires the I/O bandwidth to be reserved throughout the duration of playback to meet the real time transfer rate requirement. The reservation tends to last a long time. Reserving large portions of disk bandwidth for long durations can result in drastic degradation in performance of accesses to other data types.

The lower the buffer cache miss ratio for continous media data, the higher the performance for conventional data accesses will be in a multimedia storage system.

In this section, we will present a buffer replacement algorithm for video servers and then compare the proposed algorithm with the LRU, MRU and optimal buffer replacement algorithms. The proposed algorithm is shown to have significantly better performance than LRU and MRU.

## 4.1 System model

A video server or multimedia storage system can store various types of data. Clients access data either in *real time mode* or *non-real time mode*. In real time mode, clients can retrieve data at a guaranteed rate. In non-real time mode, no rate guarantees are provided to clients.

A multimedia storage system must ensure that rate guarantees for clients

| Buffer manager tasks |
|---|
| 1. Add the buffers containing data blocks which were consumed in the last service cycle to the free buffer pool. |
| 2. Determine which data blocks need to be retrieved from disk in the current service cycle. |
| 3. Determine if a block that has to be fetched is already in the buffers. |
| 4. Allocate buffers from the set of free buffers for those data blocks that need to be retrieved from disks. |
| 5. Issue disk I/O to retrieve the needed data blocks from disks into allocated buffers. |

Table 1: Buffer manager tasks

accessing data in real time mode are met without yielding poor performance for clients accessing data in non-real time mode. We will assume that real time clients access data in read only mode.

We also assume that each real time client must specify the rate at which data transfer must occur when it first requests data. For constant bit rate video, real time clients access data at the rate at which the data was encoded. To provide rate guarantees, the video server performs admission control and resource reservation.

We assume the system retrieves data blocks from disks into buffer space and flushes data from the buffer space to disks periodically in service cycles. The maximum length of a service cycle will be denoted $T$.

Finally, we assume clients can interactively control data transfer by pause, resume and jump.

The buffer cache space is managed as a buffer cache consisting of $n_b$ buffers, each of size $d$. Each buffer can either be free or used. We will refer to all the free buffers as the free buffer pool.

At the begining of each service cycle, the storage system scans each client to determine which data blocks were consumed by the client and which data blocks need to be fetched from disk. Table 1 shows the steps that accomplish this task.

The algorithm that decides which free block should be allocated for a block that needs to be pre-fetched is called the *buffer replacement algorithm*. The content of a free buffer is either valid or invalid. We assume the replacement algorithm first uses all invalid free buffers.

In the following, we will present a buffer replacement algorithm for video servers and then compare the proposed algorithm with the LRU, MRU and optimal buffer replacement algorithms.

## 4.2 BASIC buffer replacement algorithm

The LRU algorithm selects the buffer which contains the buffer that is used least recently. The MRU algorithm selects the buffer which contains the block that is used most recently. The optimal algorithm selects the buffer that contains the block that will not be referenced for the longest time. This cannot be acheived in practice, but can be implemented for comparison purposes in simulation.

In order to describe the BASIC buffer replacement algorithm presented in [23], we first define a *progressing client* as a client that is in a state other than pause.

When a buffer is to be allocated, BASIC selects the buffer which contains a block that would not be accessed for the longest period of time by the existing progressing clients if each client consumed data at a specified rate from the moment on.

If there are buffers which contain blocks that would not be accessed by the existing clients, the block with the highest offset rate ratio is selected as the victim. For example, the tenth block of a video with a rate of $r$=1.5 Mbps in a system with buffer size $d$=32kB has an offset-rate ratio of 9x32kB/1.5Mbps=1.536sec.

In [23], the BASIC algorithm is compared with LRU, MRU and the optimal algorithm in the case where there are only two clients accessing the same continous media file continously at a rate of r.

The length of a file is denoted as $l$ blocks. The distance between two clients is denoted *dist* blocks. We assume that any two consecutive service cycles are $T$ units apart and $d = T \cdot r$ holds. If there is one client and initially the buffer cache does not contain any of the data blocks that the client will access, then the number of disk I/O is $l$.

In [23], for the restricted scenario mentioned, it is shown that LRU will always yield miss ratios higher than BASIC. It is also shown that even if clients are not relatively close to each other, BASIC will reduce cache misses periodically. For this scenario, BASIC and MRU are equivalent to the optimal algorithm. However, it can be shown that as the number of clients increases, MRU results in significantly more cache misses than BASIC.

## 4.3 Comparison of buffer replacement algorithms

In [23], it was shown that conventional buffer replacement algorihtms such as LRU and MRU perform poorly for servers supporting continous media data. The commonly used LRU and MRU buffer cache replacement algorithms do not reduce disk I/O significantly when used in this domain. If all clients access videos continously, LRU yields miss ratios in the range of 99 percent and 88 percent.

The new buffer replacement algorithm called the BASIC algorithm was shown to reduce cache misses up to 30 percent compared to LRU and MRU, when all clients access the same continous media data. It is shown that the new

algorithms only have at most about 3 percent increase in cache misses compared to the optimal algorithm if videos are sufficiently long.

As such, the proposed algorithm is a very suitable candidate for buffer replacement scheme in storage systems with continous media data.

# 5 Interval caching

Traditional buffer management policies employed by various software systems (e.g. operating systems) are based upon the concept of a hot set of data. Various buffer management policies such as LRU use different mechanisms to identify the hot set and to retain it in the buffer. In the previous section, we presented the BASIC buffer replacement algorithm as an improvement to the LRU and MRU policies. From extensive simulations, the BASIC algorithm is shown to improve the performance of a video server by reducing the cache miss probability.

In the previous work, it was assumed that buffer and disk bandwidth resources would be reserved for each stream requiring real time retrieval. The goal of the BASIC buffer replacement algorithm was to try to reduce the disk I/O as much as possible so that the performance of non-real time disk I/O will be improved. Note that even with the BASIC algorithm to reduce disk I/O for continous media, a fixed disk bandwidth must still be reserved for each real time request. The BASIC algorithm only tries to reduce the disk I/O as much as possible, but does not provide any guarantees (deterministic or statistical) about how much the performance will be improved.

In general, buffer cache policies that operate at the block level such as BASIC and LRU do improve performance. However the improvement in performance is unpredictable and the server does not guarantee any specific performance improvement.

In this section, we propose a buffer management policy called the *interval caching policy*. The interval caching policy is based on the following observation. For any two consecutive requests for the same video, the later stream can read the data brought into the buffer by the earlier stream if the data is retained in the buffer until it is read by the later stream. With a large number of concurrent streams, it is possible to choose the streams to be retained so as to maximize the number of streams that are read from the buffer rather than from the disk, thereby reducing disk I/O.

The buffer cache is used to store the interval between subsequent requests of the same video while providing guaranteed continous retrieval. The policy identifies certain streams and temporarily buffers the pages brought in by those streams. We examine the efficacy of this technique to reduce disk overload and hence to increase the capacity of the video server. It is shown that the interval caching policy makes it cost effective to use memory to buffer video streams even with a uniform access pattern to all movies.

## 5.1 System model

The system model relevant to this section is shown in Figure 7, which shows the various software components of a video server. The buffer space is divided into a number of blocks $m$ and the blocks containing data are in the buffer pool while the rest of the blocks are in the free pool. During normal operation, for

each data stream the disk manager initiates a disk I/O in anticipation and reads data for that stream to a free buffer block.

It does this by acquiring a free block, inserting it into the buffer pool and starting the disk I/O to read data into a block. The communication manager initiates the communication I/O process and sends the previous data block of a stream stored in the buffer to the client process.

Data brought in by a stream can be re-used by other closely following streams, if sufficient buffer space is available to retain the data blocks in the buffer. The completion of the communication I/O process then invokes the buffer manager to decide whether the block should be returned to the free pool or retained in the buffer pool for re-use by other streams.

The blocks in the buffer pool are thus divided into in-blocks that are in the process of having video data read into them, out-blocks from which data is being transmitted to one or more clients and data-blocks that are being retained for reuse by other clients.

## 5.2 Interval caching policy

The main idea of the interval caching policy is to choose the consecutive pairs to be buffered so as to maximize the number of streams served from the buffer. The policy orders all consecutive pairs in terms of increasing buffer requirements an then allocates buffers to as many of the consecutive pairs as possible. The buffer requirement of a consecutive pair depends on the time interval between the two streams and the compression method used.

The main idea is illustrated in Figure 8. The small arrows marked by $S11$ through $S31$ represent the pointers corresponding to the various playback streams on the movies $1, 2, 3$. Two streams $S_i$ and $S_j$ are defined as consecutive if $S_j$ is the stream that next reads the data blocks that have just been read by $S_i$. In Figure 8, $(S_{11}, S_{12})$, $(S_{12}, S_{13})$ and $(S_{13}, S_{14})$ form three consecutive pairs for movie 1.

## 5.3 Comparison of interval caching policy with static policy

The effectiveness of the interval caching policy is studied by modelling a server with various amounts of buffer sizes delivering MPEG-1 videos. Since the effectiveness of the buffering policy depends on the distribution of access to the movies, two distributions are used in the simulations. The first distribution is based on the empirical data on video rentals in various video stores during one particular week. The second distribution is a uniform distribution over all the movies. In the simulations, the interval caching policy is compared to that of the policy of statically buffering the most popular movies.

In [15, 14] it is shown that the performance of the interval caching policy is superior to that of the static policy of buffering the most popular movies, using

the number of streams buffered as the performance metric. For the same amount of buffer, the interval caching policy buffers more streams than the static policy. The relative difference depends on the actual distribution of the accessed over the movies. For a uniform distribution, the interval caching policy buffered ten times as many streams as the static policy, while under a skewed distribution, the interval caching policy buffered twice as many streams. In addition, the interval caching policy automatically adapts to the changes in the access distribution always providing superior performance, whereas the performance of the static policy may be affected severely by the changes in the access rates.

Figure 7: Interval caching system model

Figure 8: Interval caching policy

# 6 Batching

In the future, as the cost of magnetic disk storage and memory continue to drop and broadband (gigabit) network infrastructures become widespread, large scale Video-on-Demand systems will become cost effective. In such VoD systems, large databases of movies will be stored in a set of centralized servers. Geographically distributed clients will request movies from the centrallized video servers.

For a video server to support a video stream, it is necessary for CPU, memory buffer, disk bandwidth and network interface bandwidth to be reserved at the video server. Therefore, there is a hard limit on the number of streams that can be supported concurrently by a video server.

In such a VoD system, a new movie stream can be started to satisfy each request. Alternatively, requests for movies can be *batched* together so that a single stream will satisfy requests for the same video. In this way, the video server can increase the number of supported clients. In a video server supporting batching, the same video stream can be multicast to all the clients in a given batch group.

In this section, we will present research on batch scheduling policies. The batch scheduling policies determine which set of playback requests should be batched at any given time at a video server [13].

## 6.1 System model

In this section we describe the system model relevant to the batch scheduling policies.

*Playback requests* The playback requests for movies from different clients are assumed to be independent of each other and will arrive at random time intervals at the video server.

*Video access frequencies* Given that a video server stores a set of videos, we assume that access to movies are non-uniform. Some movies are more popular than others. Based on the rental statistics from video rental stores, the access frequencies to various movies are characterized by a Zipf distribution with parameter 0.27. In a Zipf distribution, if the movies are sorted by access frequencies, then the access frequency for the $i$th movie is given by $f_i = c/i^{(1-\theta)}$, where $\theta$ is the parameter for the distribution and c is a normalization constant.

*Customer reneging behavior* The batch scheduling policy can depend on the user behavior. Once a client requests a certain video from a video server, the amount of time a user will wait before deciding to leave may not be known in advance. In the following, we will assume that the reneging time $R$ of each client is a random variable with an exponential distribution.

For the exponential reneging time distribution, the probability that a customer leaves at any moment is independent of the amount of time the client has been in the system. We further assume that all clients for all movies have

the same mean reneging time. Therefore, all customers in the queue are equally likely to remain in the queue.

## 6.2 Proposed policies

We first present two orthogonal classes of policies that select a movie for batching, given that there are a set of requests at a video server [13]. Afterwards, we will summarize the results of the simulations to compare the performance of these two policies.

*First- come-first-served (FCFS) policy*

In this policy, the requests for all movies join a single queue which we call the *requests queue*. Each customer can leave the queue independently of others if it has to wait too long. This is dependent on the reneging time of each client. Once server capacity for delivering a stream becomes available, the client at the front of the requests queue is served. All customer requests for the same movie are also satisfied by the same stream. This is a fair policy since it selects a movie independent of the identity of a movie.

*Maximum queue length (MQL) policy*

In this policy, requests for each movie join a separate queue. The movie with the maximum queue length is selected when resources are available. One drawback of this policy is that it may only choose hot movies since there are very few requests for cold movies within a short time. This may considerably increase the reneging probability of the requests for cold movies, causing an increase in unfairness.

## 6.3 Comparison of batch scheduling policies

We now summarize the results of the simulations presented in [13] which were run to compare the performance of these two policies.

*Reneging probability*

The reneging probability was found to be lower under MQL than FCFS. This is due to the memoryless property of the exponential distribution. MQL selects the movie with the largest number of waiting clients. Since the reneging probability of all clients are the same, MQL minimizes the overall reneging probability. This may not hold if the reneging probability of waiting clients depend on the amount of time it has been waiting in the system.

*Average waiting time of accepted clients*

The average waiting time of accepted clients was found to be lower for MQL than FCFS. This is because in FCFS, a playback stream can be used to serve a single client request while many requests for a popular video are waiting. However, the difference in average waiting times was found to be greatly reduced as the server capacity increases.

*Fairness*

Based on the Zipf distribution, the set of all movies can be classified into bins, such that movies in each bin receive roughly the same number of requests. For example, the movies can be divided into 10 bins such that roughly 10 percent of the requests fall into each bin. The measure used for fairness is defined as follows

$$U = \frac{\sum_{i=1}^{M}(r_i - r_{av})^2}{M} \tag{4}$$

In the above equation, $M$ is the number of bins, $r_i$ is the reneging probability of the $i$th bin movies and $r_{av}$ is the average reneging probability over all bins. In the simulation, it is found that FCFS is more fair. This can be understood intuitively because FCFS Dtreats all the movies the same.

*Complexity*

FCFS is easier to implement than MQL since little state information required.

## 6.4 Summary

In general, batching is more effective in larger servers since there are more available requests for batching. We presented two batch scheduling policies proposed in [13] that differ in their choice of a movie to be played when server capacity becomes available. Since analytical modeling is complex, simulations were performed to compare the performance of the scheduling policies. Various policies trade off various performance objectives. The performance objectives we looked at were reneging probability, average waiting time, and fairness.

The FCFS policy always serves the longest waiting client and thereby ensures fairness. It is also easier to implement. MQL serves the requests for the movie that has the largest number of waiting clients. The policy attempts to maximize the number of clients that are served at the expense of unfairness. In the simple case of assuming exponential distributions for the reneging times, MQL was found to always have lower reneging probabilities. However, if the reneging probability depends on the amount of waiting i.e. if it is not memoryless, MQL may not even perform as well as the FCFS policy. For large scale video servers, the average waiting time was found to be very close (less than 5 percent). Therefore, overall, FCFS is preferred over MQL.

# 7 Retrieval scheduling and resource reservation of variable bit rate video

In section 2, it was shown that state of the art digital video compression introduced by MPEG-2 can produce bursty, variable bit rate (VBR) video. Figure 2 shows trace data for MPEG-2 VBR video data. The video sequence is from the movie 'Forrest Gump'. VBR video can provide several advantages over constant bit rate (CBR) video, including consistent video quality and lower encoder complexity. However, the bursty nature of VBR compressed video complicates the design of real time systems such as video servers, in contrast to the simpler case of CBR video.

Video servers operate in cycles, and during each cycle time, video data is retrieved by a *disk retrieval scheduler* from the disk system to memory for each stream that is supported. The disk retrieval scheduler determines how much data should be retrieved from the disk system to memory during each cycle, for each stream that the video server supports. Video data has to be retrieved to memory before it can be transmitted into the network.

For CBR data, the disk retrieval schedule is simple. Let us say the data rate of all videos are 4.0 Mbps, and let us assume that a cycle is 0.5 sec. Then, during each cycle, 2.0 Mbits of video data is retrieved from the disk system to the memory for each stream. In the case of VBR video, the data rate is constantly changing. For VBR video, it is not clear what the best way for data to be retrieved for each stream should be.

In *Constant Time (CT) retrieval* [7, 31], data corresponding to a constant time is retrieved for each VBR stream during each cycle. Let $v(s)$ be the video that a stream $s$ is retrieving. For continous, lossless retrieval, it is necessary to reserve a disk bandwidth equal to the peak data rate of video for stream $s$.

In *Constant Data (CD) retrieval* [7, 1], a constant amount of data is retrieved for each VBR stream during each cycle of operation of a video server. A disk bandwidth equal to the average data rate of video $v(s)$ is reserved for a stream $s$. This retrieval schedule will be shown to require a certain amount of pre-fetch data to be retrieved from disk before transmission into the network can begin.

These retrieval schedules will be discussed in more depth in this section. It will be shown that CD and CT retrieval are inflexible and do not fully utilize the bandwidth and memory resources of a video server in maximizing the number of supported VBR streams.

The Minimal Resource (MR) retrieval schedule and its associated resource reservation algorithm can be shown to effectively overcome the limitations of CD and CT retrieval. The MR retrieval approach fully utilizes the video server resources to maximize the number of supported video streams.

In MR retrieval, a range of disk bandwidths can be reserved for the retrieval of VBR data. This is in contrast to CD retrieval in which a disk bandwidth equal to the average data rate of a VBR video is reserved. This is also in

contrast to CT retrieval in which a disk bandwidth equal to the peak data rate of a VBR video is reserved. However, it is shown that each disk bandwidth reservation requires a certain amount of pre-fetch buffer reservation. The MR schedule minimizes the amount of buffer that is required for each disk bandwidth reservation.

We present performance evaluations based on simulations using MPEG2 trace data. It is found that MR retrieval dramatically improves the performance of video servers compared to CT or CD retrieval. For a video server configuration with 4 disks and a memory resource of 120 MBytes, the MR retrieval approach supports 50 percent more video streams than approaches based on CT retrieval. For the same configuration MR retrieval supports 275 percent more video streams than approaches based on CD retrieval.

A key point is that MR retrieval always has better or equal performance over CD or CT retrieval, irrespective of the particular video server resource configurations. The increase in complexity for MR retrieval is also shown to be small. Furthermore, The MR retrieval schedule and associated resource reservation algorithms are flexible enough to be implemented on general purpose computers. The MR retrieval approach does not depend on any special video data layout strategies on disks, and is directly applicable to video servers that are based on general fault tolerant storage architectures (e.g. RAID-3 Redundant Array of Independent Disks[10].)

Compared to simulcast coding, scalable coding schemes can provide multiple levels of video with a minimal cost of extra bandwidth or storage capacity. In scalable video coding, subsets of the full resolution bitstream are used to obtain subsets of the full resolution video. We show how scalable video can improve the performance of a video server when used with the MR retrieval schedule.

## 7.1 System model

In this section we describe the video server system model relevant to this section. The system model is shown in Figure 9. The video server has a fault tolerant disk array for the storage of video data, a memory resource for pre-fetch buffers, and a network interface for transmission into the computer network.

The disk retrieval scheduler has a cyclic operation [2, 24]. Each cycle, the disk retrieval scheduler retrieves data for multiple video streams from the disk system to the memory. The *network transmission scheduler* also has a cyclic operation, although the cycle time of the network transmission scheduler will typically be much smaller than the cycle time of the disk retrieval scheduler. Much like the disk retrieval scheduler, the network transmission scheduler transmits data for multiple video streams into the network during each cycle.

The video data is interleaved over all the disks of the array. During each cycle, the disk heads of each disk in the array complete one cycle of a SCAN disk head schedule, as described in 3. The network is assumed to accommodate the peak bandwidth of all video streams and introduces zero delay and zero

| Notation | Description |
|---|---|
| r(t) | Data rate of stored video |
| $t_n$ | Start of network transmission |
| $t_r$ | Start of disk retrieval |
| $T$ | Duration of video |
| $o(t)$ | Cumulative data transmitted out of memory into network (output) |
| $i(t)$ | Cumulative data retrieved from disk into memory (input) |
| $m$ | Buffer memory reserved for retrieval |
| $b$ | Disk bandwidth reserved for retrieval |

Table 2: Retrieval scheduling notation

jitter. Recently, research has been done on the network transmission schedule for VBR video in which limited network bandwidth, network delay and network jitter are being considered [21, 28].

The video server supports completely interactive viewing and continous, lossless retrieval. Video servers supporting completely interactive viewing allow viewers to pause and resume playback at any time during a viewing session. Playback can also resume at any point of a video. Video servers supporting continous, lossless retrieval provision resources so that once transmission of a portion of a video has started, no delay is introduced at the server i.e. the transmission never stops until all the video has been transmitted. Note that this does not mean that there can be no delay *before* transmission begins. This is an important distinction, and will be discussed more below.

Each stream supported by a video server has a fixed bandwidth and fixed buffer reserved for the entire duration of interactive retrieval. There are no renegotiations of resources for each stream during a viewing session. This is a key simplifying assumption.

## 7.2 Retrieval constraints

In the following sections, although the operation of a video server is based on discrete time cycles, we will use continous time notation to clearly convey the central ideas. Table 2 defines the notation we will use in the following sections.

In the following, we make an important assumption about the network transmission schedule as follows

$$o(t) = \begin{cases} 0 & \text{for } t < t_n \\ \int_0^t r(t - t_n)\, dt & \text{for } t \geq t_n \end{cases} \tag{5}$$

The retrieval constraints for the retrieval of a video are as follows:
1) $i(t) \geq o(t)$ (Continous retrieval constraint)
2) $i(t) - o(t) \leq m$ (Buffer constraint)
3) $\frac{di(t)}{dt} \leq b$ (Disk bandwidth constraint)

### 7.3 Constant time retrieval

Constant time (CT) retrieval retrieves data from disk to memory according to the video data rate. This scheme is described by the equation

$$i(t) = \begin{cases} 0 & \text{for } t < t_r \\ \int_0^t r(t - t_r)\, dt & \text{for } t \geq t_r \end{cases} \tag{6}$$

The cumulative input data and cumulative output data are equivalent at any given time. In an actual system, the network transmission scheduler waits one cycle time $t_c$ after disk retrieval begins before it can start transmission into the network. Therefore $o(t) = i(t - t_c)$.

The delay of one cycle exists because we assume that the video server uses a double buffer scheme for retrieval and transmission (Figure 10). During each cycle, data is read from the disk into the disk read buffers for each stream. Each disk executes one SCAN cycle. Concurrently, data is written from the network write buffers into the network interface card (NIC) for each stream. At the end of each cycle, the contents of the disk read buffers are written onto the network write buffers, and the process repeats. No pre-fetch buffer is required for CT retrieval. In the first cycle, the disk read buffer is being filled, while the network write buffer is empty. At the end of the first cycle, the contents of the disk read buffer are copied onto the network write buffer. Therefore, network transmission can only begin after a delay of one cycle. This delay is different from the pre-fetch delay mentioned in the following sections and is common to all the retrieval schedules. We shall ignore this delay in the following sections. The pre-fetch delay for CT retrieval is zero.

For each stream, a disk bandwidth of $b$ equal to the peak data rate of the video being retrieved must be reserved for the entire duration of an interactive viewing session:

$$b = \max\{r(t), 0 \leq t \leq T\} \tag{7}$$

In the following sections, we ignore the disk read buffers and network write buffers and only consider the pre-fetch buffers. For CT retrieval, for each stream, the memory requirement for pre-fetch buffers is zero.

For the MPEG-2 VBR video shown in Figure 2, CT retrieval has to reserve a bandwidth of 14 Mbps for the entire duration of retrieval. The buffer requirement for a video server operating at 0.5sec cycle time is 1.75 MB.

In [31], CT retrieval is the basis for the admission control scheme in multimedia servers. The primary contribution of the work is that statistical service guarantees are provided to all streams. In other words, for each stream, a continous retrieval is guaranteed to a fixed percentage of the video data. It is proposed that a certain percentage of video data can have the continuity requirement violated without significantly affecting the quality of the video. This

leads to an improvement in the utilization of the server. New clients are admitted for service as long as the statistical estimate of the aggregate data rate requirement (rather than the peak data rate requirement) can be met.

## 7.4 Constant data retrieval

In constant data (CD) retrieval, a bandwidth of $b$ Mbps equal to the average data rate of a video is reserved. The reserved bandwidth is typically much smaller than in CT retrieval, in which the peak bandwidth is reserved. This scheme is described by the equation

$$i(t) = \begin{cases} 0 & \text{for } t < t_r \\ b \cdot (t - t_r) & \text{for } t \geq t_r \end{cases} \tag{8}$$

CD retrieval retrieves a fixed amount of data during each cycle(Figure 11, Figure 12). This differs from CT retrieval, in which a variable amount of data is retrieved in each cycle.

In this scheme, data has to be pre-fetched to ensure that continous, lossless retrieval of the video is guaranteed. This can occur because the amount of data transmitted during each cycle is variable, while the amount of data retrieved from the disk system is constant during each time cycle. Since a pre-fetch data has to be retrieved, there is a pre-fetch delay associated with CD retrieval. The worst case pre-fetch delay can be determined for stored video because the entire trace is known a-priori:

$$t_n - t_r = \frac{p}{b} \tag{9}$$

$$p = \max\{o(t_n + \Delta) - b \cdot \Delta\} \tag{10}$$

$$0 \leq \Delta \leq T \tag{11}$$

For the MPEG-2 VBR video shown in Figure 2, CD retrieval reserves a bandwidth of 3.8 Mbps for the entire duration of retrieval. The buffer requirement for a video server operating at 0.5sec cycle time is found to be 50 MB.

In [7], CD retrieval is the basis for both a deterministic and statistical admission control scheme. The retrieval scheme is referred to as Constant Data Length (CDL) retrieval. In [1], two retrieval schemes called traditional CDL and generalized CDL (GCDL) are presented. The traditional CDL scheme described is actually very different from the CDL scheme of [7]. In the traditional CDL scheme of [7], a constant amount of data is retrieved from the disk for a video stream in the first disk cycle. In the second cycle, the same constant amount of data is retrieved only if it is required to prevent buffer underflow. Otherwise, no data is retrieved. This process repeats throughout the retrieval. Therefore, each retrieval cycle is either an idle or active round. Although a constant data

amount is retrieved during an active round, the overall retrieval can be considered to be variable bit rate. This is different from [1], in which the overall retrieval is constant bit rate i.e. there are no idle rounds. In [1], the GCDL scheme is an extension of the traditional CDL scheme in which the retrieval round can be different for different video streams and which are a multiple of the disk cycle. This is shown to reduce the buffer requirements compared to traditional CDL.

## 7.5 Minimal resource retrieval

In this section, we present a Minimal Resource (MR) retrieval schedule [27] for continous, lossless retrieval of VBR video. MR retrieval is similar to CD and CT retrieval in that the retrieval alternates between intervals of constant time retrieval and constant data retrieval. However, it differs in that a range of bandwidths can be reserved for the retrieval. CD retrieval requires that a bandwidth equal to the average data rate is reserved, while CT retrieval requires a bandwidth reservation equal to the peak data rate.

If the bandwidth reserved for retrieval of VBR video is less than the peak data rate, then data has to be pre-fetched to ensure continous, lossless retrieval. Therefore, a buffer for pre-fetch data is required. In order to minimize the buffer requirement, data should be pre-fetched just-in-time. For the retrieval of VBR video, MR retrieval minimizes the worst case buffer requirement that is required for a given disk bandwidth reservation.

We first describe the MR retrieval schedule and then discuss its properties and then compare it with CD and CT retrieval. As before, let $t_n$ be the start of network transmission of a stream. The accumulated data output from the memory to network is $o(t)$. The bandwidth reserved for the retrieval is $b$. We define the following function that we will use in describing MR retrieval:

$$a(\alpha, \beta) = b \cdot (t - \alpha) + \beta \tag{12}$$

We also use two new variables $t_b$, $t_e$ to mark the beginning and end of each maximum retrieval interval. The determination of the MR retrieval schedule of a stored video is described in Table 3 and is shown graphically in Figure 13 and Figure 14.

MR retrieval is defined by the maximum retrieval intervals. Figure 13 shows $i(t)$ for MR retrieval. If the time $t$ falls inside any of the maximum retrieval intervals, the retrieval rate is at the maximum bandwidth $b$. Otherwise, the retrieval amount is equal to the data rate. The buffer status at time $t$ is $m(t) = i(t) - o(t)$. The worst case pre-fetch delay is:

$$d = \frac{\max\{m(t)\}}{b} \tag{13}$$

$$t_n \leq t \leq t_n + T \tag{14}$$

| Algorithm to determine MR retrieval schedule |
|---|
| 1. Set $t_e = t_n + T$ |
| 2. Decrease $t_e$ until $\frac{do(t_e)}{dt} \geq b$ |
| 3. Find the intersection of $a(t_e, o(t_e))$ with $o(t)$ |
| 4. Let $t_b \leq t_e$ equal the intersection point |
| 5. Mark the interval $[t_b, t_e]$ as a maximum retrieval interval |
| 6. If $t_b \leq t_n$ then stop |
| 7. Set $t_e = t_b$ and return to step 2 |

Table 3: Determination of MR retrieval schedule

Proof of MR retrieval optimality MR retrieval can be shown to minimize the worst case pre-fetch buffer requirement for a given bandwidth reservation for the continous, lossless retrieval of a video. We can prove this by showing that MR retrieval is based on just-in-time retrieval. Consider the first maximum retrieval interval $[t_b, t_e]$. In MR retrieval, the retrieval rate during a maximum retrieval interval is $b$. We define a small time interval $\Delta$. Consider the start of retrieval to be delayed to $t_b + \Delta$.

In this case, it can be seen that even if the retrieval is at the maximum retrieval rate of $b$, the continous retrieval constraint will be violated (Figure 15). Therefore, $t_b$ is the latest time at which pre-fetch of data can start if continous retrieval is to be guaranteed up to $t_e$.

Consider the start of retrieval to start earlier at $t_b - \Delta$. It can be seen that the buffer requirement will increase for any possible retrieval schedule as data is retrieved earlier than required. This analysis can be done iteratively for all the maximum retrieval intervals. Therefore, since MR retrieval is based on just-in-time retrieval, it minimizes the buffer requirement while satisfying the constraints for continous, lossless retrieval.

## 7.6 Buffer-bandwidth resource relation

For a given bandwidth reservation, MR retrieval minimizes the worst case pre-fetch buffer that is necessary for continous, lossless retrieval. It can also be shown that increasing the reserved disk bandwidth will reduce the worst case buffer requirement. Therefore, MR retrieval leads to a buffer-bandwidth resource relation for the retrieval of a video. Figure 16 shows the buffer-bandwidth resource relation for the MPEG2 encoded video trace data shown in Figure 2. This relation shows the worst case pre-fetch buffer reservation that is necessary for a given disk bandwidth reservation. From this relation, the corresponding worst case pre-fetch delay can be found. If the worst case buffer requirement for the retrieval of a given video is $m$, then the worst case pre-fetch delay is $m/b$, where $b$ is the corresponding reserved bandwidth.

For the interactive viewing of videos, we introduce the concept of a Pre-

fetch Delay Tolerance Quality of Service (PDT QoS). A PDT QoS is specified for each stream that a video server supports, and it specifies the worst case pre-fetch delay that can be tolerated during interactive viewing. It is shown that a PDT QoS specified for a stream $s$ that retrieves video $v(s)$ is equivalent to placing a lower bound on the bandwidth that can be reserved for stream $s$. The lower bound for the bandwidth can be determined from the buffer-bandwidth resource relation of video $v(s)$.

## 7.7 Comparison of retrieval schedules

The primary strength of MR retrieval is the flexibility to optimally trade bandwidth and buffer. This is captured in the buffer-bandwidth resource relation. While CD and CT retrieval are each represented by a single point on the resource relation for the stored video, MR retrieval can operate at multiple operating points on the resource relation. MR retrieval can set any bandwidth reservation. As the bandwidth reservation is reduced, it is necessary to increase the reserved buffer. Consider a video server that uses MR retrieval. We present two cases to demonstrate the advantage of using MR retrieval over CD or CT retrieval.

*Case 1.*

Assume that initially, each stream has a bandwidth reservation equal to the peak data rate of the video being retrieved. Assume that the total bandwidth reserved for all streams is equal to the total bandwidth of the video server. Assume that a large buffer memory exists in the video server. Using CT retrieval, no more streams can be supported by the video server because of the bandwidth limitation. In MR retrieval, if all viewers can tolerate a pre-fetch delay, the bandwidth reserved for all the streams can be substantially reduced from the peak bandwidth. Reducing the reserved bandwidth for each stream requires an increase in pre-fetch buffer requirements for each stream, if continous, lossless retrieval is to be guaranteed. In this way, memory resources can be utilized to alleviate the I/O bandwidth bottleneck. By reducing the total bandwidth reserved for all the streams, the video server can potentially increase the number of streams that are supported.

*Case 2.*

Assume that initially each stream has a bandwidth reservation equal to the average data rate of the video being retrieved. Each stream has a pre-fetch buffer requirement. Assume that the total buffer memory reserved for all streams is equal to the total memory resource of the video server. However, assume that total bandwidth reserved for all streams is less than the total disk bandwidth of the video server. Using CD retrieval, no more streams can be supported by the video server because of the memory limitation. In MR retrieval, by increasing the bandwidth reserved for each stream, the memory requirement for each stream can be substantially reduced. In this way, the bandwidth resources can be utilized to alleviate the memory bottleneck of a video server. By reducing

| Notation for resource reservation algorithm | |
|---|---|
| $M$ | Total number of streams |
| $k = 1, ...M$ | Stream index |
| $B_I$ | Bandwidth increment |
| $b_k \cdot B_I, b_k = 1, 2, ...$ | Stream bandwidth reservation |
| $p_k$ | Lower bound for stream bandwidth reservation |
| $q_k$ | Peak bandwidth of video accessed by stream $k$ |
| $R_k(b_k)$ | Stream buffer-bandwidth resource relation |
| $d_k$ | Stream PDT QoS |
| $C_M$ | System memory resource constraint |
| $C_B$ | System disk bandwidth resource constraint |
| $\overline{B} = (b_1, ...b_M)$ | Reservation vector for all streams |
| $S$ | Reservation vectors with $p_k \leq b_k, k = 1, ...M$ |

Table 4: Notation for resource reservation

the total memory reserved for all the streams, the video server can potentially support more streams.

## 7.8 Resource reservation

In the previous section we developed and presented the MR retrieval schedule. It was found that the worst case memory buffer requirement for the retrieval of a VBR video decreases as the bandwidth reservation increases. It was seen that MR leads to a buffer-bandwidth resource relation. In this section, we develop the resource reservation algorithm based on MR retrieval for multiple streams in a video server.

In a video server, data for multiple, concurrent streams is retrieved from the disk system to memory and then transmitted into the network. The video server has resources of disk bandwidth and memory that have to be shared amongst all streams. If MR is used for the retrieval of each stream, the important question remains as to what buffer and bandwidth reservations should be made for each stream i.e. the operating point on the resource relation of the video retrieved by each stream must be determined.

For each incoming stream, the reservations should be made to maximize the number of streams that can be supported by a video server while guaranteeing the continous, lossless retrieval and PDT QoS of each stream. Before describing the reservation problem, we present some definitions in Table 4.

The lower bound on the stream bandwidth reservation is determined as follows:

$$p_k = \min b_k \tag{15}$$

Subject to:

$$\frac{R_k(b_k)}{b_k} \leq d_k \tag{16}$$

where $\frac{R_k(b_k)}{b_k}$ is the pre-fetch delay.

Our objective in a video server is to maximize the number of streams that can be supported. Therefore, we formulate the resource reservation problem as follows:

For a given set of streams, determine if there exists a reservation vector $\overline{B}$ for all streams that satisfies the following constraints:

$$M_T = R_k(b_k) \leq C_M \tag{17}$$

$$B_T = b_k \leq C_B \tag{18}$$

$$\overline{B} \subset S \tag{19}$$

The constraints are the memory, disk bandwidth and PDT QoS constraints respectively.

If the reservation vector exists, the set of streams can be supported by a video server, otherwise the set of streams cannot be supported. Note that in an actual system, the computation of the total bandwidth reservation is not as simple as above, since the SCAN disk head schedule is assumed. The actual computation based on simplifying assumptions is given in [25]. The resource reservation problem above can be shown to be equivalent to the following two step algorithm:

1) Find a reservation vector which is the solution to the following constrained minimization problem: $\min M_T$ subject to $B_T$, $\overline{B} \subset S$

2) If $M_T \leq C_M$ (system memory constraint), the set of all streams can be supported by the system, otherwise the set of streams cannot be supported.

Let $A$ be the set of all reservation vectors that meet both the PDT QoS constraint and the system bandwidth constraint specified above. The non-linear minimization problem gives us an optimal reservation vector which is the reservation vector in $A$ that minimizes the total system memory requirement $M_T$. If $M_T$ is greater than the system memory constraint, then there can be no reservation vector in $A$ that will also meet the memory resource constraint. This is because the optimal reservation vector is the vector in $A$ that minimizes the memory requirement. This means that there can be no reservation vector that meets all system resource constraints and the PDT QoS constraint. Therefore the two step algorithm is equivalent to the optimal resource reservation algorithm.

In [25], both a fast optimal solution and a fast heuristic solution to the resource reservation problem are developed and presented. The algorithms are not presented here.

## 7.9 Progressive display of scalable video

In section 2 we described scalable MPEG video. Scalable video can improve the performance of a video server that uses the MR retrieval schedule and its associated resource reservation algorithm.

In the *progressive display* of scalable video for interactive viewing [27], a progressively increasing PDT QoS is specified for the progressively higher scalable layers of a video. Each scalable layer is considered as an independent video. This is in constrast to non-progressive display of scalable video in which a single PDT QoS is specified for a full resolution video.

In progressive display, the pre-fetch data for all the scalable layers are retrieved simultaneously since each layer is considered to be an independent video. At any given time, a video is transmitted only with all the scalable layers for which the pre-fetch data have been fully retrieved. For example, suppose that transmission is to be resumed after an interactive function. If enough time has elapsed only for the pre-fetch data of the lowest scalable layer to have been retrieved, only the lowest scalable layer is transmitted. If enough time has elapsed for the pre-fetch data of the first two layers to have been retrieved, the first two scalable layers are transmitted.

Progressive display of scalable video improves the performance of a video server. For scalable video, let the lowest layer of video have a PDT QoS of l sec. The higher layers will have PDT QoS values larger than l sec. In non-progressive display of scalable video, to achieve the same degree of interactivity, all the scalable layers have the same PDT QoS value of l sec.

There are various ways for scalable video data to be placed on disks. In this research, we assumed that each scalable layer is stored separately as an independent 'video' and that each layer is interleaved over all disks.

## 7.10 Performance evaluations of retrieval schedules

This section has a subset of the performance evaluations presented in [27]. For the performance evaluation presented here, the disk performance characteristics of two disk systems are as shown in Table 5. Disk system 1 has the disk performance characteristics of a current magnetic disk. In disk system 2, the performance parameters were improved by a factor of two to project the performance characteristics of the next generation of magnetic disk systems. For performance evaluation, trace data for MPEG2 scalable and non-scalable video was obtained using Columbia University's full-profile, standard-conforming MPEG2 software encoder/decoder [32].

In the following simulations, the video server receives requests for videos from clients. Each new request specifies a certain video which is stored on the video server, for which there exists a resource relation. Each request also has an associated PDT QoS. For each new request, the video server determines if it can accept the request or not. If the video server can accept the new request, a

| Parameter | Disk system 1 | Disk system 2 |
|---|---|---|
| Disk cycle time/sec | 0.5 | 0.5 |
| Max. rotation latency/ms | 14.2 | 7.1 |
| Max. seek latency/ms | 18.0 | 9.0 |
| Min. seek latecy/ms | 1.5 | 0.75 |
| Max. disk transfer rate/Mbps | 60.0 | 120.0 |
| No. of disks in array | 4 | 4 |

Table 5: Disk performance parameters

stream is established for the new request. For each simulation the total number of video streams that can be supported by the video server is found for a given set of available video server resources. The simulations find the total number of admissible streams to the video server system as the on-board memory resource is increased, while maintaining a fixed disk bandwidth resource.

*Comparison of MR and CT retrieval*

Figure 18 and Figure 19 compares the performance of MR and CT retrieval scheduling. Each line corresponds to a single simulation. In each simulation, all streams are accessing the same MPEG-2 VBR video which has the trace data shown in Figure 2. Also, in each simulation, each stream specifies the same PDT QoS value. For disk system 1, if a video server has 120 MB, MR retrieval supports 50 percent more streams than CT retrieval, if users can tolerate a pre-fetch delay of 10.0 sec.

In the case of MR retrieval scheduling, we used the heuristic resource reservation algorithm to maximize the number of streams that can be supported concurrently by a video server. In the case of CT retrieval, the resource reservation algorithm is based on the fact that each stream requires a bandwidth reservation equal to the peak data rate of the requested video. Figure 18 and Figure 19 show the total number of admissible video streams at the video server system as the total memory resource is increased, while keeping the disk system the same.

For MR retrieval, we can see that the number of streams that can be supported increases as the video server memory resources are increased. For continous, lossless retrieval in interactive viewing, the resource reservation algorithm based on MR retrieval guarantees that no other retrieval schedule can support more video streams for a given set of video server resources.

It can be seen that CT retrieval cannot take advantage of any increase in the memory resource of a video server. The advantage of the CT retrieval schedule is that the PDT QoS is always zero. This does not mean that the total delay that the client experiences before receiving its requested video is zero. However, the pre-fetch delay is zero. It can be seen that the performance of this scheme is the same as MR retrieval in which clients specify a PDT QoS of zero.

*Comparison of MR and CD retrieval*

Figure 20 compares the performance of MR and CD retrieval scheduling. Each line corresponds to a single simulation run. In each simulation all streams are accessing the same MPEG-2 VBR video which has the trace data shown in Figure 2. Also, in each simulation, each stream specifies the same PDT QoS value. For disk system 1, if a video server has 150 MB, MR retrieval supports 275 percent more video streams than CD retrieval. CD retrieval is memory bound.

In the case of MR retrieval scheduling, we used the fast heuristic resource reservation algorithm to maximize the number of streams that can be supported concurrently by a video server. In the case of CD retrieval, the resource reservation algorithm is based on two facts. Firstly, each stream requires a bandwidth reservation equal to the average data rate of the requested video. Secondly, there is a fixed memory requirement for the retrieval of the video. Figure 20 shows the total number of admissible streams at the video server system as the on-board memory resource is increased, while keeping the disk bandwidth the same.

We can see that the number of streams supported by CD retrieval is generally much lower than MR retrieval. This scheme is essentially memory bound. The bandwidth is not fully utilized since the memory requirements are the limiting factor in the resource reservation.

Figure 9: Video server architecture

Figure 10: Constant Time retrieval

Figure 11: Constant Data retrieval

Figure 12: CD cumulative data analysis

Figure 13: Determination of MR retrieval schedule

Figure 14: MR cumulative data analysis

Figure 15: Proof of MR optimality

Figure 16: Resource relation for MR retrieval

Figure 17: Resource reservation framework

Figure 18: Performance evaluation

Figure 19: Performance evaluation

Figure 20: Performance evaluation

# 8 Conclusions

There is no doubt that video servers will be a critical component of the information technologies of the future. Currently many industries are investing research and development teams into developing the next generation of high performance video servers for the Internet, intranet and broadcast markets.

In the past few years, there has been a great deal of excitement about video servers for Video-on-Demand and interactive video. Many people (both technical and non-technical) were led to believe that the technology for such systems were just around the corner. Various companies proposed a model of VoD in which a large scale, centrallized server would store massive amounts of video information. Consumers around the country and even around the world would connect to this video server to select from the massive number of video titles and receive it *on demand*. Just as a ball park figure, by very large scale, newspaper articles, technology white papers and research papers mentioned VoD systems in which video servers store on the order of 1000 videos and support on the order of 1000 users.

Naturally, it was in the economic interests of companies striving to develop such systems to try to convince investors that this particular model of VoD was indeed about to explode on the market. When companies failed to deliver on their promises, there was a resultant skepticism about VoD. The companies found out that the technology was not at a point where large scale, centrallized video servers could not be developed cost effectively. Furthermore, the network infrastructure to support relatively high bandwidth applications to the home (such as Video-on-Demand) are currently not available on a wide scale. There was no market for VoD at the prices that the companies could offer. Consumers were unwilling to pay large sums of money to have a limited selection of videos, all of which (and much much more) could be borrowed at a fraction of the price from their local video rental store.

At this point it would be extremely short sighted to say that VoD is finished. Despite the current skepticism about interactve video, there is no doubt that VoD will indeed become a reality. How can such a strong claim be made in the face of disappointing efforts to develop VoD? Such a strong claim can be made for several reasons.

Firstly, magnetic disk, robotic tape archive, RAM memory, computer network bandwidth costs are continuing to drop rapidly. Furthermore, there is rapid progress in optical networks that will eventually lead to gigabit and terabit networks. It is just a matter of time before VoD will indeed explode on the market. Then, consumers will indeed have access to an array of high quality visual content. The era in which consumers passively accepted whatever content broadcast companies chose to distribute will come to an end.

Secondly, although it is currently economically infeasible to develop very large scale centrallized servers, the technology is definitely ready for small scale video servers, especially for *intranet* environments, in which it is much easier to

guarantee $QoS$ end to end. It is easy to imagine that there will be a proliferation of small scale video servers on University and corporate based intranets. As the global network infrastructure improves, it is conceivable that video servers on intranets will eventually be interconnected. At that point, the *network-connectivity gain* will result in users on the global network having access to massive amounts of visual information.

The *network-connectivity gain* is as follows. Consider the global network to have $N_s$ video servers, each storing $N_v$ videos. This means that each user on the network can potentially have access to $N_s \cdot N_v$ videos. We will refer to this as the *network-connectivity gain*. As an example, consider each video server to store 10 videos. If there are 10000 video servers on the global network, each user has access to 100000 videos. This is the advantage that results from the connectivity of computer networks.

It is not necessary to stick to the model of a video server that has to store massive amounts of information. The aggregate of massive numbers of video servers, each of which may only store a small number of videos will lead to unprecedented amounts of visual information available on the global network.

It is clear that the development of video servers will be a key component of the *information technology revolution*. Video server development will be intimately dependent on the costs of components such as RAM memory and disk systems. Video server development will also be dependent on the extent that $QoS$ and seamlesss connectivity can be provided in a cost effective way by computer networks. Video servers will likely first be developed for the intranet and broadcast environment. In the midst of all of the above dynamically varying factors and industry trends, research in video servers will be critical in guiding the development and evolution of cost effective and high performance video servers.

# References


[1] E. Biersack, F. Thiesse, and C. Bernhardt. *Constant Data Length Retrieval for Video Servers with VBR Streams.* Proceedings of the International Conference on Multimedia Computing and Systems. June 1996.

[2] Steven Berson, Leana Golubchik, Richard R. Muntz. *Fault Tolerant Design of Multimedia Servers.* ACM SIGMOD '95. 1995.

[3] William J. Bolosky, Joseph S. Barrera, III, Richard P. Draves, Robert P. Fitzgerald, Garth A. Gibson, Michael B. Jones, Steven P. Levi, Nathan P. Myhrvold, Richard F. Rashid (Microsoft Corp). *The Tiger Video Fileserver.* NOSSDAV 96. 1996.

[4] Alan J. Chaney, Ian D. Wilson and Andrew Hopper (Olivetti Research Labs). *The Design and Implementation of a RAID-3 Multimedia File Server.* NOSSDAV 95. 1995.

[5] S.- F. Chang, A. Eleftheriadis, D. Anastassiou. *Development of Columbia's Video On Demand Test bed.* Image Communication Journal, Special issue on Video on Demand and Interactive TV. 1996.

[6] Ed Chang and Avideh Zakhor. *Scalable Video Data Placement on Parallel Disk Arrays.* SPIE Symposium on Imaging Technology, San Jose. 1994.

[7] E. Chang and A. Zakhor. *Admissions Control and Data Placement for VBR Video Servers.* IEEE International Conference on Image Processing, 1994.

[8] Ed Chang and Avideh Zakhor. *Cost Analyses for VBR Video Servers.* IEEE Multimedia. 1996.

[9] M.S. Chen, D.D. Kandlur, P.S. Yu. *Optimization of Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams.* ACM Multimedia '93. 1993.

[10] Peter. M. Chen, Edward. K. Lee, Garth. A. Gibson, Randy. H. Katz, David. A. Patterson. *RAID: High-Performance, Reliable Secondary Storage.* Submitted to ACM Computing Surveys. Also UCB//CSD-93-7 78. 1993.

[11] Ann L. Chervenak, David A. Patterson, Randy H. Katz. *Choosing the Best Storage System for Video Service.* Proceedings of ACM Multimedia '95. October, 1995.

[12] Tihao Chiang and Dimitris Anastassiou. *Hierarchical Coding of Digital Television.* IEEE Communications Magazine. May 1994.

[13] Asit Dan, Dinkar Sitaram and Perwez Shahabuddin. *Scheduling Policies for an On-Demand Video Server with Batching.* Proceedings of ACM Multimedia.

[14] Asit Dan, Daniel Dias, Rajat Mukherjee, Dinkar Sitaram, Renu Tewari. *Buffering and Caching in Large Scale Video Servers.* Proceedings of COMPCON 1995.

[15] Asit Dan, Dinkar Sitaram. *Buffer Management Policy for an On-Demand Video Server.* IBM Research Report RC 19347, IBM T.J. Watson Research Center, Yorktown Heights, N.Y., January 1994.

[16] R. Flynn and W. Tetzlaff (IBM T.J. Watson Research Center, Yorktown Heights, N.Y.). *Disk Striping and Block Replication Algorithms for Video File Servers* Proceedings of the International Conference on Multimedia Computing and Systems. June 1996.

[17] D. J. Gemmell and Jiawei Han. *Delay Sensitive Multimedia on Disks.* IEEE Multimedia Magazine. 1994.

[18] Didier Le Gall. *Digital Multimedia Systems.* Communications of the ACM. Vol. 34, No. 4. April 1994.

[19] K. Keeton, R. Katz *Evaluating data layout strategies.* ACM Multimedia Systems Journal. 1996.

[20] Cliff Martin, P.S. Narayanan, Banu Ozden, Rajeev Rastogi, Avi Silberschatz. *The Fellini Multimedia Storage Server.* Bell Laboratories, Murray Hill, N.J. Technical report. 1996.

[21] Jean M. McManus, Keith W. Ross. *Video on Demand over ATM: Constant Rate Transmission and Transport.* IEEE Journal on Selected Areas in Communications, Vol. 14, No. 6. August 1996.

[22] Joan L. Mitchell, William B. Pennebaker, Chad. E. Fogg, and Didier Le Gall. *MPEG Video Compression Standard.* Digital Multimedia Standards Series. Chapman and Hall Publishers, 1997.

[23] B. Ozden, R. Rastogi, A. Silberschatz (Bell Laboratories, Murray Hill, N.J.). *Buffer Replacement Algorithms for Multimedia Storage Systems.* Proceedings of the International Conference on Multimedia Computing and Systems, pages 172-180, June 1996.

[24] B. Ozden, R. Rastogi, A. Silberschatz (Bell Laboratories, Murray Hill, NJ). *Disk Striping in Video Server Environments.* Proceedings of the International Conference on Multimedia Computing and Systems. June 1996.

[25] Seungyup Paek, Paul Bocheck and Shih-Fu Chang. *Scalable MPEG2 Video Servers with Heterogeneous QoS on Parallel Disk Arrays.* NOSSDAV '95. April, 1995.

[26] Seungyup Paek and Shih-Fu Chang. *Video Server Retrieval Scheduling for Variable Bit Rate Scalable Video.* Proceedings of the International Conference on Multimedia Computing and Systems. 1996.

[27] Seungyup Paek and Shih-Fu Chang. *Video Server Retrieval Scheduling and Resource Reservation for Variable Bit Rate Scalable Video.* Submitted to IEEE Transactions on Circuits and Systems for Video Technology, 1997.

[28] James D. Salehi, Zhi-Li Zhang, James F. Kurose, and Don Towsley *Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing* Proceeding of ACM Sigmetrics. May, 1996.

[29] R. Tewari, R. Mukherjee, D. M. Dias, and H. M. Vin. *Design and Performance Tradeoffs in Clustered Video Servers.* Proceedings of the International Conference on Multimedia Computing and Systems. 1996.

[30] Fouad A. Tobagi, Joseph Pang, Randall Baird, Mark Gang (Starworks). *Streaming RAID- A Disk Array Management System for Video Files.* Proceedings of ACM Multimedia 93. 1993.

[31] H. Vin, P. Goyal, A. Goyal(1), and A. Goyal(2). *A Statistical Admission Control Algorithm for Multimedia Servers.* ACM Multimedia '94, San Francisco, USA, 1994.

[32] Y. Yu *Columbia MPEG Software Release 6.5 User's Manual.* Technical report of Image and Advanced Television Laboratory, Columbia University. 1994.