Cost Analyses for VBR Video Servers

Ed Chang and Avideh Zakhor
University of California, Berkeley, CA 94720
e-mail: changed@robotics.eecs.berkeley.edu, avz@eecs.berkeley.edu

Abstract

In this paper we compare techniques for storage and real-time retrieval of Variable Bit Rate (VBR) video data for multiple simultaneous users. The motivation for considering VBR is that video results in inherently time varying data, and as such, with the same average bit rate, higher quality can be achieved with VBR than with Constant Bit Rate (CBR). We propose and compare the following three classes of VBR data placement and retrieval techniques: Constant Time Length (CTL) places and retrieves data in blocks corresponding to equal playback durations, Constant Data Length (CDL) places and retrieves constant-sized data blocks, and a hybrid solution uses CDL placement but retrieves a variable number of blocks in each service round. We have found that CTL data placement has much lower buffer requirements than CDL but suffers from fragmentation during video editing. We show hybrid placement to have both advantages of high efficiency and low fragmentation.

We also address the issue of admission control policies by comparing statistical and deterministic techniques. "Statistical" admission control uses statistics of the stored data to ensure that the probability of "overload" does not exceed a prespecified threshold. "Deterministic" control uses the actual stored video bit traces to regulate the number of admitted users. We consider two types of deterministic admission control: data-limit and ideal deterministic. Data-limit admission control admits users based on precomputing the total amount of data requested by all users in future service rounds. In contrast, Ideal deterministic admission control not only precomputes the total amount of data requested, but also assumes we have control of data placement at the disk sector level in order to precompute the future seek and rotation times.

We provide a cost/benefit analysis of the above placement/retrieval/admission control techniques and conclude that CTL and hybrid placement/retrieval techniques can reduce the total system cost by up to a factor of 3 in comparison with the strategy of padding the VBR video trace to achieve a constant data rate. For read-only systems, CTL has the lowest cost per user. For writable systems, the hybrid technique achieves a good compromise between low cost and low fragmentation. We find that all forms of deterministic admission control can outperform statistical, but the greatest gain comes from using ideal deterministic admission control. We note, however, that this admission control may be difficult to implement on standard disk controllers. Finally, we have implemented a full disk model simulator that operates 1000 times faster than the real-time disk. Results using the simulator are very close to those measured on the real disk, making the simulator useful for future experiments.

Keywords: cost, VBR, video storage, video servers, disk arrays, admission control, data placement

1 Introduction

Video on demand applications are becoming increasingly more important in entertainment, education and telecommunications industries. Video data placement and admission control for video servers are two of the most important challenges of video on demand applications. In this paper, we investigate storage and real-time retrieval of video data for multiple simultaneous users. We choose a disk as our storage medium, as disk arrays have been shown to provide cost-effective storage [15] and high-bandwidth transfer capabilities and are becoming popular in video on demand systems [9].

For storage efficiency we compress the data before writing to disk. Since there are several different modes of encoding, a major question to address is the type of rate control mechanism, if any, to be applied in generating the compressed bit stream. One option would be to generate a truly VBR, constant quality stream without any rate control, buffers or quantization feedback. Another possibility is to generate a Constant Bit Rate (CBR) stream in which quantization feedback is applied to implement a leaky bucket rate control mechanism in order to avoid buffer overflow or underflow [17, 27].

It is generally believed that video inherently results in variable bit rate data and as such, the main advantage of true VBR over CBR is its true constant quality. As a result, when using CBR, one must choose between low bandwidth and nonuniform quality, or very high bandwidth and uniform quality. In order to maintain uniform quality, the CBR bandwidth is chosen at such a high level so as to ensure the quality remains above a certain threshold at high motion parts of the video sequence. This latter approach results in overallocation of storage resources in storing CBR data. Since it has been shown that typical VBR video may have a peak to mean ratio of 3:1 even averaged over a few seconds [10], it is conceivable to achieve the same quality of video at about 1/3 of the cost, by storing VBR rather than CBR video data on disks. Therefore, the same statistical multiplexing ideas that have traditionally been used in networking applications can be exploited to reduce the cost per stream in VBR video storage applications. There is however, one major difference between networking and storage applications in that unlike the networking applications, in the storage scenario, one can exploit the a priori knowledge of the video bit trace in order to optimize both the data placement algorithms and the admission control algorithms.

The choice of true VBR versus CBR also influences the data units in which the compressed bit stream is stored on the disks. For CBR video, the data can be stored and retrieved in constant-sized data blocks without risking jitter free, real-time video delivery [2, 22]. For VBR data, the block sizes to be written to and read from the disk can not be chosen as easily as for CBR data. The basic issue is whether to store and retrieve data in unequal amounts to conform to the real-time playback duration, or to store and retrieve the data in equal-sized blocks for each user, utilizing buffer memory to provide real-time variable bit rate for playback. We call the first method Constant Time Length (CTL) data placement and the second method Constant Data Length (CDL).

CTL-stored videos are characterized by a unique pattern of variable data block lengths that may be derived from the video bit trace and the real-time duration of the blocks. Because data blocks may be interleaved across many disks [2], the stored blocks may not be contiguous. In this case, replacing or editing the video will result in disk fragmentation problems. To avoid this, we consider a CDL system. However, as we will see in Sect. 3.2, CDL may result in large buffer usage. We therefore consider a hybrid system in which data is stored in CDL blocks, but the number of blocks to be retrieved varies with the playback consumption requirements. This strategy reduces the buffer requirements and will be shown to result in a good compromise between low cost and low fragmentation.

To ensure that the system resources are not overallocated when too many users request data simultaneously, we require admission control algorithms. We consider two main classes of admission control: statistical and deterministic. A statistical admission control exploits the bit rate statistics of the videos on the disk. The advantage of such a strategy over network admission control schemes is that it uses the actual histograms of the data to be read rather than generic video statistics for all possible video sequences. Alternatively, we can apply a deterministic admission control strategy by exploiting the specific knowledge of the bit traces of requested videos rather than their statistics. Since the disk system has full knowledge of the future bit rate traces needed to service all user requests, we can decide if the admission of a new request will cause a system overflow during the length of the request.

Other researchers have considered the issues of data placement and admission control for VBR video servers. Anderson et al. [1] read a variable number of fixed size blocks and use contiguous allocation to avoid extra disk seeks between blocks of one request. The drawback of contiguous allocation is fragmentation as experienced in a CTL scheme. Vin et al. [25] compare the performance of CTL and hybrid systems for a multiple-disk multimedia network server. As for admission control, Vin et al. propose statistical [23] and adaptive [24] policies. Gemmell [12] also considers a variable number of fixed size blocks which may be grouped into sorting sets to reduce disk latencies. Rosario et al. [18, 19] consider pure CDL data placement and describe a deterministic admission control algorithm.

In this paper we provide a cost/benefit analysis of the above placement/retrieval/admission control techniques and conclude that CTL and hybrid placement/retrieval techniques can reduce the total system cost by up to a factor of 3 in comparison with the strategy of padding the VBR video trace to achieve a constant data rate. For read-only systems, CTL has the lowest cost per user. For writable systems, the hybrid technique achieves a good compromise between low cost and low fragmentation. We find that all forms of deterministic admission control can outperform statistical, but the greatest gain comes from using ideal deterministic admission control. We note, however, that this admission control may be difficult to implement on standard disk controllers. Finally, we have implemented a full disk model simulator that operates 1000 times faster than the real-time disk. Results using the simulator are very close to those measured on the real disk, making the simulator useful for future experiments.

In this paper we decouple the issues of data placement/retrieval and admission control as follows. We begin by defining our system metrics and parameters in Sect. 2 and comparing the three data placement and retrieval techniques using a simple disk model and basic statistical admission control in Sect. 3. In Sect. 4 we choose the hybrid data placement strategy to compare admission control techniques using a more explicit disk simulator and an actual disk-based video server. We present our conclusions in Sect. 5.

2 System Metrics and Parameters

To evaluate our strategies, we must specify the operation of our video server system and consider various quality metrics. The periodic nature of video service naturally leads to a round-robin scheduling scheme. We define a service round as the smallest periodic unit of time in which the server sends some data to each user to ensure real-time playback capability. Specifically, we use the common assumption of a dual buffer system [4, 12, 14, 22] in which data is read from disk and sent to the disk buffer, while previously-read data is sent from the network buffer to each user at each user's corresponding playback consumption rate. Data is transferred instantaneously from the disk buffer to the network buffer at the end of each service round. Thus the size of the total system buffer is the sum of the sizes of the disk and network buffers.[1]

Using this dual buffer assumption, we are not constrained to schedule the users on the disk in any given order, other than to ensure every user is scheduled within a given service round. We assume the user requests in each service round are scheduled according to the SCAN algorithm. A more complex grouping strategy can reduce the seek times or amount of buffer required [12, 29] but is beyond the scope of this paper.

We consider the following quality metrics. Our first metric is probability of service round failures. We define a service round failure as any service round in which not all of the admitted users receive their requested data, and we choose a threshold P_{fail} as the upper bound on probability of service round failures. Next, the cost per stream should be as low as possible. This is accomplished by maximizing the throughput, measured in number of simultaneous users, and minimizing the disk and buffer memory cost. We measure the average buffer per user, $B(L)$, by dividing the maximum total system buffer used, B_{total}, by the average number of users served, $U(L)$, for a given request length L.

Our final considerations are start delay and jump delay. Start delay, also known as latency, measures the amount of time between the admission of a user onto the system and the actual delivery of data from the network buffer to the user. The delay between the appearance of a user request and admission is simply a function of the rate of incoming user requests and the average number of users that the system can serve; we do not consider this queueing delay as a part of our latency.[2] Jump delay is defined as the amount of time between the end of a service round that contains a jump request and actual playback of jump destination data.

An important issue throughout this paper is the model we use in our theoretical derivation and simulations for the seek and rotation times. There is an inherent tradeoff between accuracy and complexity of seek models; for instance, if tracks and sector locations of data are taken into account and the actual rotation and movements of the head are modeled precisely, then we would have an accurate, but complex model. On the other hand if the seek and rotation time is assumed to be constant, then we have sacrificed accuracy for the sake of lower complexity used in our simulations.

The approach we have taken in this paper is to propose three different models for the seek time and use different models in different scenarios. For the first one, which we refer to as "typical seek time," we use Vin's assumptions [25] of evenly-spaced requests on the disk for the seek time and one half a disk revolution for the average rotation time. We begin by using techniques from Worthington et al. [28] to measure our HP C3325W disk to estimate a seek time profile, $T_{seek}(n)$, the seek time as a function of n tracks traversed. We further assume that r requests are evenly spaced across the n_{total} disk tracks such that there is an equal number of tracks between each requested block. This results in an equal seek time for each request and a total seek time of $rT_{seek}(\lceil n_{total}/r \rceil)$. It has been shown that evenly-spaced requests maximize the total seek time under the SCAN algorithm [22]. Thus the "typical" seek time esimate combines a conservative seek time estimate with a liberal rotation time estimate.

The second seek time model is the "worst case" and uses evenly-spaced requests on the disk for the seek time similar to the "typical seek time" described above but full disk revolutions for rotation time. Both seek time and rotation time estimates are conservative. The third model uses the actual location of the data on the disk to compute the exact amount of time needed for disk revolutions and head seeks. The first model is used in simulations of Sect. 3 and has been found to be in excellent agreement with experimental data from a real disk. For the results in Sect. 4, however, we have found this model to be inadequate, and as a result, we have resorted to the second and third models.

[1]An alternative architecture would be to distribute the network buffer across the clients receiving video; however, this becomes uneconomical once the number of clients exceeds the simultaneous user capacity of the server. Since we assume that in general, there will be many more clients than can be simultaneously supported on the server, we place the network buffer at the server.

[2]This is commonly known in queueing theory as Little's result [26] and has been empirically confirmed in our tests.

Table 1 lists our system parameters; the first two parameters are characteristics of the physical disk, as measured on our disk drive. The read rate is found by reading 50 MB blocks of data from the disk as a raw device. For our disk we calculate the "typical" estimate of the seek time, T_s, to be 14 ms.

R_v denotes the average coded bit rate of our chosen VBR video sequence, *Star Wars* [10], at 370 kb/s. We choose the service round time T_{SR} to be two seconds, resulting in an average stored block size of 92 kB. The average time to read each block is then T_{read} = 92 kB / 5.03 MB/s = 0.01786 s. Assuming each user performs one seek and one read in each service round, an ideal video server could theoretically service:

$$U_{ideal} = \frac{T_{SR}}{T_s + T_{read}} = \frac{2 \text{ s}}{0.014 \text{ s} + 0.01786 \text{ s}} = 62.77 \text{ users} \qquad (1)$$

We choose an upper bound on the probability of service round failure to be $P_{fail} = 10^{-3}$. Because video is fairly tolerant of dropped frames, one might consider different users requesting different qualities of service, each with different probabilities of failure [3]. However, past work has shown that allowing higher probabilities of overload does not result in a large increase in number of users served [3, 23]. Finally, we test request lengths of $L = 16$ or 3620 service rounds to illustrate the differences between reading short and long video sequences.

Symbol	Description	Value
R_d	disk read rate	5.03 MB/s
T_s	typical disk seek and rotation time	14 ms
R_v	average bit rate of video	374369 b/s
T_{SR}	service round time	2 s
P_{fail}	probability of service failure	10^{-3}
L	request length	16, 3620 SR

Table 1: Experiment System Parameters

3 Data Placement

We now compare the three methods of data placement mentioned in Sect. 1. We examine CTL in Sect 3.1, CDL in Sect. 3.2, and a hybrid strategy in Sect. 3.3. In each case, we calculate the theoretical average number of users served and the buffer requirement, and we verify the results through experiments on a real disk. Finally we compare the total system costs in Sect. 3.4 and discuss interactivity in Sect. 3.5.

3.1 Constant Time Length

In Sect. 1, we defined CTL as a data placement strategy in which stored block sizes are proportional to their corresponding playback bit rates. Because the total amount of data retrieved by all of the users in one service round is variable, the primary concern in CTL data placement and retrieval is the possibility of disk overload. To analyze this probability for the different admission control schemes, we assume that consecutive samples from the same video bit trace are uncorrelated. Under this assumption, we can treat each service round independently to more easily calculate theoretical quality metrics such as probability of disk overload and average number of users admitted.

A straightforward statistical admission control strategy for CTL data is based on analyzing the probability that the total data requested by all of the users in one service round exceeds the disk throughput capacity. We can compute this probability for an arbitrary number of users U watching videos as follows. To have no overload, we must satisfy the condition that the total time required to service all of the users does not exceed T_{SR}. Each user u requires a seek of duration T_s and a read of duration $T_r(u)$. The read times $T_r(u)$ are equal to the ratio of the amounts of requested data $D_r(u)$ and the disk read rate R_d. Our condition is therefore:

$$UT_s + \sum_{u=1}^{U} \frac{D_r(u)}{R_d} \leq T_{SR} \tag{2}$$

$$\sum_{u=1}^{U} D_r(u) \leq R_d(T_{SR} - UT_s) \tag{3}$$

We therefore define the disk read limit as the maximum amount of data that can be read by U users in one service round, $D_{lim}(U) = R_d(T_{SR} - UT_s)$.

Let $p_D(x)$ denote the probability distribution function (pdf) of $D_r(u)$, or equivalently of CTL block sizes; we experimentally estimate $p_D(x)$ by compiling a histogram of all of the CTL block sizes on the system. Since there are U independent users, the pdf of the sum of the instantaneous rates for all U users is obtained by convolving $p_D(x)$ U times with itself.[3] Once the pdf of the instantaneous rate of the U users is computed, the probability of overload, $P_o(U)$, can be found by integrating the resulting curve beyond the threshold limit corresponding to the disk read limit $D_{lim}(U)$. We can thus find $P_o(U)$ for all U, and we define U_{max} to be the maximum U such that $P_o(U) < P_{fail}$. For the parameters in Table 1, $U_{max} = 58$. Since the probability of disk overload is not a function of request length L, the average number of users served is independent of L. We verify the average number of users served on our real disk video server for the same probability of overload within 0.7%.

We compute the buffer usage of this system as follows. For the CTL system, the disk buffer and network buffer are the same size, since the network buffer serves only to delay the contents of the disk buffer for synchronization purposes. The total buffer needed for this system is therefore equivalent to twice the maximum amount of data that can be requested by all of the users in one service round. Assuming there are U_{max} users on the system, the total buffer required is $B_{total} = 2 \times R_d(T_{SR} - U_{max}T_s) = 12238$ kB. The average buffer required by 58 users is therefore 211 kB/user for all L. We have verified this number to within 0.5% experimentally.

The latency of the CTL system is exactly one service round, the amount of time required to fill the disk buffer. The data is then assumed to be transferred to the network buffer instantaneously, where it is available to the user. Interactivity does not pose any problem for the CTL system, as the jump delay is also only one service round. In CTL, users impose the same load on the video server whether they are playing videos forward, backward, scanning by block-skipping, or playing in completely random order, as long as each user retrieves only one block from disk in each service round.

3.2 Constant Data Length

The most commonly assumed method of storing data is to store and retrieve constant sized blocks. In this section, we describe one method of storing CDL blocks for VBR data using a leaky bucket mechanism for buffer control. The principle behind the leaky bucket and the CDL data blocks is to move the data variation from the disks to the buffer. Specifically, the leaky bucket mechanism allows constant rate input from the disk to the buffer and a variable rate output from the buffer to the network.[4] Because the total number of users can overload both the disk throughput and the buffer available, we need to consider both constraints. Disk throughput issues for CDL data placement of CBR video have already been considered in [2] and [22]. For CDL placement of VBR video, the data retrieval from disk is the same: one CDL block per service round per user. We can thus operate the disks at full capacity, and we define $U_{disk} = \lfloor U_{ideal} \rfloor$ to be the number of users that can all read data from the disk in one service round without overloading the disk read bandwidth.

The disk buffer is the same for CDL placement of both CBR and VBR video; it is the total amount of data that can be read from the disk by all users in one service round. The network buffer for VBR video, however, must absorb the variations in the video bit trace for each user. Hence, in the remainder of this section, we consider the two problems of network buffer underflow and overflow.

[3]In previous work [3], we have estimated this distribution with the Central Limit Theorem and Cramer's rule.

[4]Note that the leaky bucket model for our CDL data placement operates in the exact opposite way that a leaky bucket rate control used at the output of a video coder does; in a video coder, there is a variable rate input from the coder to the buffer and a constant rate output from the buffer, resulting in CBR video. Reibman and Haskell present a complete analysis of buffer usage in a leaky bucket system with fixed rate and variable rate channels in [17].

To ensure that the buffer does not underflow and starve the user, the system calculates the amount of prefetch data needed to be read into the buffer. Since the system knows the requested video bit trace a priori, a fixed-size buffer is allocated to each user and the exact amount of data necessary to prevent buffer underflow is prefetched into this buffer.

The buffer overflow problem is more difficult since each of the users will require a variable amount of buffer depending on the part of the video sequence accessed. Similar to the CTL case, we propose a statistical admission control strategy which uses the statistics of the actual data stored on disk. The strategy is based on analyzing the probability that the total buffer state of all of the users exceeds the given system buffer. We can compute this probability for an arbitrary number of users U as follows. We assume each user has the same playback request length, and so the buffer states can be modeled as independent identically distributed random variables with pdf $p(x)$, where x is the instantaneous buffer state. The distribution $p(x)$ is found numerically by compiling a histogram of all buffer states of all possible times t and all possible starting points g in the video sequence or sequences on the disk [4, 5].

As in the CTL case, we compute the probability of overload by convolving $p(x)$ U times and integrating beyond a limit, B_{lim}, which denotes the available buffer size. In the CDL case, however, there are two important differences. The first is that B_{lim} is not specified by the disk parameters; instead it is chosen arbitrarily as the amount of buffer to be added to the disk system. As such, if it is chosen large enough, it can result in zero probability of overflow. The second difference is that the buffer usage histograms and hence probability distributions $p(x)$ are functions of the request length L. Because of this, a real system must either compute or store the buffer histogram for each request length and combine histograms with real-time convolutions, or it must limit a set of acceptable request lengths that may be chosen. An alternative to the above convolution-based approach is to make a simplified estimation using the Central Limit Theorem to approximate the final distribution with a Gaussian curve [3].

We test the system using the *Star Wars* sequence for the two request lengths $L = \{16, 3620\}$. Using the parameters in Table 1, we choose each read unit to be the average size of a 2-second block of the video, 92 kB. To maximize the disk usage, we set the statistical admission control test to allow $U_{disk} = 62$ users on the disk at all times. Experiments on our real disk video server show that we can support this number of users with a probability of overload that is less than the threshold, $P_{fail} = 10^{-3}$. The disk buffer size is simply the amount of data that can be read from disk in one service round, $U_{disk} \times D_{read} = 62 \times 92$ kB $= 5704$ kB. To calculate the theoretical network buffer required, we use the two methods of prediction described above: convolution of the buffer histograms, and normal approximation with a Gaussian curve. We find that the buffer requirements increase greatly with request length. At $L = 16$ rounds, the buffer requirement is 372 kB/user. At the full request length of 3620 rounds, the buffer requirement is 15 MB/user, 72 times the requirement of the CTL system. This disproportionate number results from long-range dependencies in the video sequence [11]; we show in Appendix B that the buffer required by one user is bounded below by the (σ, ρ) curve of the video sequence requested, and our results agree with those obtained by Grossglauser, Keshav, and Tse [13]. Because these buffer amounts exceed the capacity of our video server for *Star Wars*, we use a discrete event simulator to track the buffer usage and verify the theoretical results as shown in Fig. 1. As seen, the simulation results are in close agreement with both the convolution and Gaussian estimation techniques. Although we can admit 4 more users than the CTL system, the increase in buffer requirement overshadows that gain, as we will see in Sect. 3.4.

In addition, we measure the average start delay in the above simulations and find it to be directly proportional to the average network buffer requirements. The average latency for $L = 16$ is 3 service rounds, or six seconds. The average latency for $L = 3620$ is 150 service rounds, or 5 minutes. This delay applies to each user upon beginning a request, but even more objectionably to each user seeking readmission after a jump. Each jump renders the current buffer useless, and a long delay is required to refill the buffer before playback can resume. In [5] we have considered a "burst mode" scheme in which the disk system temporarily devotes all of its resources to loading the pre-fetch data for the user's jump to a new location to minimize the jump delay; however, this results in even higher buffer usage and start delays for non-jumping users.

3.3 Hybrid Data Placement and Retrieval

In Sect. 3.1, we found that CTL data placement services a reasonable number of users with very low buffer usage. However, the disk fragmentation becomes a problem for CTL systems that require on-line editing. The CDL systems overcome the fragmentation problem but use an excessive amount of buffer. We now consider a hybrid system in

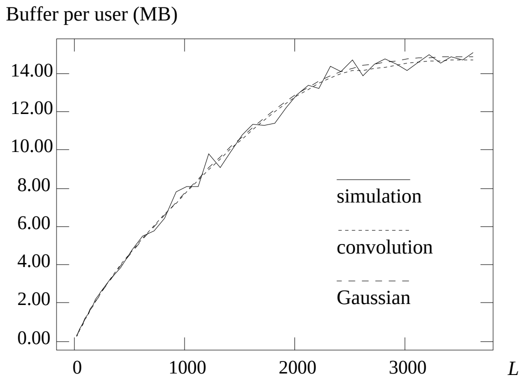

Figure 1: CDL network buffer requirements

which data is written in CDL blocks, but different numbers of blocks are read in each service round according to the user playback rate, corresponding to a coarsely quantized CTL-style read. Vin *et al.* [25] have analyzed the effects of block size on load balancing in a hybrid system. As in the CDL case, we choose our block size to be 92 kB.

There are two main differences between the hybrid system and the CTL system. First, the read units in the hybrid system are much more coarsely quantized; whereas each user in a CTL system can read up to 300 1-kB disk sectors per round, each user in the hybrid system reads zero to three 92-kB blocks of data. The remaining data that is not consumed immediately is stored in the network buffer to prevent the same disk block from being re-read in the next service round. The second difference is that there are no seeks between each sector of a CTL read, whereas there is a seek before each large block of data read for a user of the hybrid system. Specifically, hybrid system users reading multiple blocks in one round must perform multiple seeks to access those blocks. These additional seeks reduce the disk efficiency.

The statistical admission control strategy for the hybrid system is similar to that of the CTL system. We compute the probability that U users will cause a disk overload by using the same service time condition as done in Sect. 3.1. In the hybrid case, however, each disk storage block is the same size and requires one seek; therefore the only variable is the number of blocks read. Let $N(u)$ be equal to the number of blocks read by user u in a given service round, and let $T_{r,block}$ be the time required to read one block. As in Sect. 3.1, we assume a constant seek and rotation time of T_s. Since each block requires its own seek and read time, the service time condition to avoid overload is:

$$\sum_{u=1}^{U} N(u)(T_s + T_{r,block}) \quad \leq \quad T_{SR} \tag{4}$$

Since we do not allow the retrieval of fractional blocks, we define the block limit, N_{lim} as the maximum integer number of blocks that can be read by all users in one service round without exceeding the service time condition.

$$N_{lim} \quad \equiv \quad \left\lfloor \frac{T_{SR}}{T_s + T_{r,block}} \right\rfloor \tag{5}$$

Thus, as long as the total number of blocks requested by U users one service round, $\sum_{u=1}^{U} N(u)$, does not exceed N_{lim}, there will be no overload in that service round. For the parameters in Table 1, $N_{lim} = 62$ blocks.

Let $p_N(x)$ denote the pdf of $N(u)$; we numerically estimate $p_N(x)$ by compiling a histogram of the number of blocks to be retrieved for every service round of the entire video sequence. We show the probability distribution derived from the histogram of our test sequence *Star Wars* in Table 2.

For U users, there may be a non-zero probability that the total number of blocks requested in one service round exceeds the allowable limit, N_{lim}. We calculate this probability as follows. Analogous to the other two data placement schemes, we obtain the pdf of the total number of blocks requested by U users by convolving $p_N(x)$ U

Blocks	0	1	2	3
Probability	.137	.735	.125	.003

Table 2: Histogram of blocks per request in hybrid system

times with itself. Once we compute this pdf, we can find the probability of overload, $P_o(U)$, by integrating the pdf beyond the block limit N_{lim}. We can thus find $P_o(U)$ for all U, and we define U_{max} to be the maximum U such that $P_o(U) < P_{fail}$. For the parameters in Table 1, $U_{max} = 51$.

In a hybrid system, disk block boundaries do not generally correspond to real-time playback data boundaries. Therefore, users who enter the system and begin reading a video sequence must retrieve an additional block of data during the first service round. This additional block guarantees that the total data retrieved from disk is sufficient to begin playback. However, these additional first-round blocks increase the system load, lowering the average number of users served for a given probability of overload as follows. Assuming U users request videos of length L in a given service round, and that the block size is equal to average amount of data required for one service round, the average number of requested blocks is U in the given round. If we further assume that the users are randomly distributed in phase, an average of U/L users will be accessing the first service round of their requests. These first-round requests require an additional block each, and so the average requested total number of blocks is $U + U/L$. Thus the first-round requests increase the number of blocks required by a factor of $(L+1)/L$. If the hybrid server can support an average of X users at $L = \infty$, then the number of users that can be supported for finite L is $XL/(L+1)$, a drop by a factor of $1/(L+1)$. Taking this into account, we revise our earlier estimate of $U_{max} = 51$ to $U_{max} = 48$ for $L = 16$ and $P_o(U) = 10^{-3}$.

We plot the theoretical $P_o(U)$ for both the hybrid and CTL schemes in Figure 2. As seen, for our range of probabilities of overload, the hybrid system serves fewer users due to the extra seek and extra first-round block overheads. Simulations verify the average number of users served and the overload probabilities.

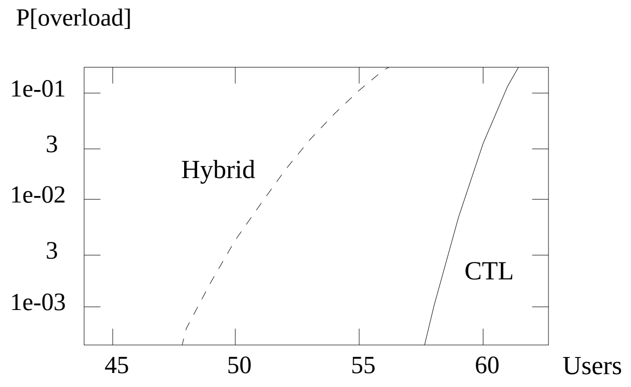

Figure 2: Disk overload probability

We now consider buffer usage and delay for the hybrid system. The blocks are written the same way as in CDL systems, and therefore the disk buffer is the same size: 62 blocks at 92 kB/block = 5704 kB. To estimate the network buffer, we consider its operation. Let D_{block} be the amount of data in each block, 92 kB in our tests. Since users read data from disk in multiples of D_{block} bytes but consume data for playback in any amount, there will be $[0, D_{block})$ bytes remaining in the network buffer after each service round. For example, if the network buffer is initially at a level of $.2D_{block}$ and the user needs to consume $1.3D_{block}$ bytes of data next round, then 2 blocks will be read from the disk. At the end of the service round, these 2 blocks are transferred to the network buffer, bringing the network buffer level to $2.2D_{block}$ bytes. The user then consumes $1.3D_{block}$ bytes in playback, leaving $.9D_{block}$ bytes in the network buffer for the next round.

We therefore model the network buffer level as a random variable uniformly distributed on $[0, D_{block})$ and the amount read from disk as a random variable with the distribution shown in Table 2. Assuming the current levels and amounts read are independent for all of the users, we estimate the probability distribution of the sum of all of the users' disk buffers and network buffers through convolution [5]. For our chosen probability of failure, $P_{fail} = 10^{-3}$, we estimate the required total buffer at 13658 kB. The simulation results differ by less than 2% from this value.

In a hybrid system, blocks are stored in equal-length blocks as in CDL. Therefore, block boundaries will not generally coincide with playback time indices, and a user who begins service will need to prefetch some data to fill the network buffer. In a CDL system, this prefetch greatly increases the latency. For the hybrid system, however, the maximum amount of prefetch is one block. Users read the prefetch block with the other data blocks in the first round to be guaranteed that they have enough data in the buffer to begin immediate playback. Thus the start and jump delays are equivalent to the CTL case, exactly one service round.

3.4 Cost Analysis

We now present a cost analysis for our three data placement strategies. The cost per stream can be divided into two parts: the disk and the buffer. The cost of the disk, C_d, is equal to the cost of the disk controller, C_{dc}, plus the product of the video data size, D_v, and the price of disk storage, $Price_d$. The cost of the buffer, C_b, is given by the product of the total system buffer, B_{total}, and the price of memory, $Price_{mem}$. Dividing the two costs by the average number of users yields the estimates for the costs per stream. We assume the following current prices: $C_{dc} = 200$ dollars, $Price_d = 0.25$ dollars/MB, and $Price_{mem} = 15$ dollars/MB of DRAM. Table 3 summarizes the results.

	Padded VBR	CTL	CDL		Hybrid	
Request Length	any	any	16	3260	16	3260
Disk Cost	15.40	4.84	4.67	4.54	5.82	5.50
Buffer Cost	8.17	3.09	5.44	222.07	4.07	3.95
Total Cost	23.57	7.93	10.11	226.61	9.89	9.45

Table 3: Theoretical: cost per stream in dollars

As seen in Table 3, CDL systems are not cost-effective for storage and retrieval of long sequences. The buffer requirements are too costly, and we find the delays to be too long for interactive use [5]. The only advantage of CDL over CTL is that CDL has no fragmentation problems associated with rewriting videos on disk.

Hybrid schemes, however, eliminate the fragmentation problems while retaining a relatively low cost per stream. The hybrid system costs about 19-24% more than the CTL system; thus we can recommend the CTL system as the least expensive in read-only situations. However, the increase in cost in using a hybrid system may be acceptable in situations in which videos may need to be edited or overwritten on the disk.

3.5 Interactivity

Another consideration in choosing a data placement strategy for video servers is interactivity. CTL data placement is best suited for interactive playback, as there are no penalties for playing blocks in non-consecutive order. As long as each user requests only one CTL block per service round, the maximum number of users served and the average buffer per user does not change. CDL data placement, on the other hand, results in very poor system performance for interactive use, as seen in the long start and jump delays in Sect. 3.2. We have examined a "burst-mode" CDL system [5] which reduces the delay after users jump to different parts of the video, but the buffer requirements increase substantially beyond the baseline CDL usage.

Hybrid systems show reasonable performance in interactive situations. However, they suffer a slight performance penalty because the playback consumption data boundaries do not generally correspond to disk block boundaries. During straightforward playback, these mismatches are absorbed by the network buffer as explained in Sect. 3.3. For users that first enter the system, an extra block must be read to prefetch data for the network buffer; this results in the $1/(L+1)$ loss in average number of users as described in Sect. 3.3. However, a network buffer prefetch also applies to users that interrupt normal playback to jump to a different part of the sequence. Let P_j be the probability of a jump per user per service round and U be the average number of users served. The average number of users requesting jumps in one service round will then be UP_j. These jump requests require an additional block each, and so the average requested total number of blocks is $U + UP_j$. Thus the jump requests increase the number of blocks required by a factor of $1 + P_j$. If the hybrid server can support an average of X non-jumping users, then the number of users that can be supported for non-zero P_j is $X/(1 + P_j)$, a drop by a factor of $P_j/(P_j + 1)$.

4 Admission Control and Disk Models

In Sect. 3 we applied a simple statistical admission control that uses statistics of the stored data to compute a hard limit on the number of users in the system. This user threshold guarantees that the probability of disk or buffer overload does not exceed a prespecified threshold. However, the server has access to more information than just the statistics of the stored bit traces. Specifically, the bit traces themselves are known, and the server can use these traces to examine each incoming request on an individual basis. This form of deterministic admission control has been previously proposed [4, 24, 16, 18, 19] but not theoretically analyzed.

To fully understand the implications and effects of deterministic admission control, we consider the following two types. *Data-limit* admission control admits users based on precomputing the total amount of data requested by all users in future service rounds. In contrast, *ideal* deterministic admission control not only precomputes the total amount of data requested, but also assumes we have control of data placement at the disk sector level in order to precompute the future seek and rotation times. To simulate these types of deterministic admission control, we use a a full disk model [20] and verify the results using a video server constructed on an HP 9000 725/100 workstation with an HP C3325W hard drive.

Thus we have a total of three admission control strategies: statistical, data-limit deterministic, and ideal deterministic. In Sect. 3 we have examined statistical admission control for each of our data placement strategies. In this section we examine deterministic admission control for the hybrid data placement since it achieves a compromise between cost and fragmentation. In doing so, we fix the request length L at 16 service rounds. We consider data-limit deterministic admission control in Sect. 4.1, and ideal deterministic in Sect. 4.2. We compare results and costs in Sect. 4.3.

4.1 Data-Limit Deterministic Admission Control

We now present a data-limit deterministic admission control strategy that accepts users based on a priori knowledge of the bit rate traces of the requested video sequences. We assume a constant seek and rotation time T_s as in the previous sections. This scheme is different from the statistical admission control of Sect. 3 in that instead of using the *statistics* of all the videos stored on a server, it uses the actual bit trace of the particular video that is being served. As can be expected, this scheme is more complex to implement than statistical admission control. However, as we will see, it can potentially achieve zero probability of overload and in general results in a lower cost per stream.

Let $N(v(u), i)$ be the number of blocks consumed by user u requesting video $v(u)$ at time index i. Let t be the current service round index and $t_0(u)$ be the service round in which user u was admitted to the disk. Then the amount of data requested by user u in the current service round t is $N(v(u), t - t_0(u))D_{block}$. Because the disk system has full knowledge of the current videos requested by all of the users, the total number of blocks requested for all future service rounds can be calculated a priori. Specifically, the total number of blocks required by the disk to service all of the users at service round t is $\sum_u N(v(u), t - t_0(u))$. To prevent disk overload at all times, we extend the condition from Eq. (4):

$$\sum_u N(v(u), t - t_0(u)) \quad \leq \quad \frac{T_{SR}}{T_s + T_{r,block}} \qquad \forall\ t \tag{6}$$

At any service round t, the limit on the maximum number of blocks read to avoid overload is given in the right side of Eq. (6). Since we do not allow the retrieval of fractional blocks, we can instead use N_{lim} as defined in Eq. (5) to be the upper bound on the number of read blocks. Because the disk system can exactly predict the occurrences of disk overload for a given set of users, the admission control calculates the hypothetical future disk usage given the addition of a new request. If the addition would result in disk overload for any service round t, the request is denied access; otherwise it is admitted.[5] This operation is illustrated in Fig. 3. The first graph shows the total amount of data requested as a function of time. We assume that the users are randomly distributed in time so that the total amount of data falls as a function of time. For example, at $t = L/2$ service rounds, we expect half of the users to have completed their requests and exited the system, leaving the current load at half of the $t = 0$ load. The second graph shows the bit rate trace of the next user request. The admission control adds the two graphs, arriving at the third graph. We see that the addition causes one of the future service rounds to overflow, and so we deny admission to the user. The user waits until the next service round before applying for admission again. This process repeats until the admission control verifies that no future rounds will be overloaded and then admits the user.

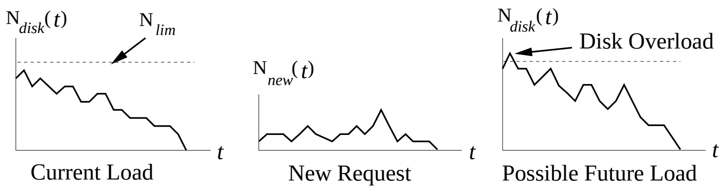

Figure 3: Data-limit deterministic admission control

In the statistical admission control policy in Sect. 3.3, we use T_s from Table 1 to estimate the average seek and rotation time. Using this value of T_s in Eq. (5) results in $N_{lim} = 62$. If the seek and rotation times did not vary from the average value, this data-limit admission control would guarantee zero disk overload at this value of N_{lim}. However, when tested on the real disk video server, we measure a 10% probability of overload due to the seek and rotation time variation. Clearly we must use a more conservative approach to approximating the seek and rotation time.

We propose a worst case scenario by assuming a conservative worst case seek bound as explained in Sect. 2, and a worst case rotation bound equal to the time required for a full disk revolution. This results in a reduction in N_{lim} in Eq. (5) from 62 to 53 blocks and guarantees a zero probability of overload. Experiments on a real disk system show that $N_{lim} = 53$ blocks results in an average of 49.5 users. However, this admission control may be unnecessarily restrictive – in the same real disk system experiments, we find that we can achieve zero overload in 10^5 trials with $N_{lim} = 58$, resulting in an average of 54.1 users served.

We see that by changing the seek and rotation time estimate T_s, we change the block limit N_{lim}. As the block limit increases, so does the average number of users served. In Fig. 4 we plot the measured probability of overload as a function of the number of users served and compare the results with those obtained using statistical admission control. These experiments are conducted on our real disk system. As seen, for all tested proabilities of overload, the data-limit deterministic admission control admits more users than the statistical for a given probability of overload. The difference grows larger as we reduce the probability of overload. It would be interesting to test lower probabilities of overload, but long tests are infeasible due to the real-time nature of the video server.

To perform longer tests, we simulate our video server on a full disk model based on work by Ruemmler and Wilkes [20]. We use our measured disk parameters to calculate a layout pattern by translating the video bit trace blocks into sectors and tracks on the disk simulator. Finally, we trace the execution of the simulator, adding the appropriate seek, head switch, and rotation times. We assume the scheduling policy to be the SCAN algorithm. We do not model secondary effects such as slipped defective sectors and thermal recalibration. The simulator operates about 1000 times faster than real time, and our tests show that it predicts the number of users within 2% of the measured real disk value as seen in Fig. 4. The difference in probability of overload between the real disk and simulator at any given U can be attributed to the large slope of the overload curve. For our test parameters, a shift in the disk seek time profile of one ms results in a change of one user served. Since our accuracy in measuring read and seek times is limited to a one ms resolution, the error between our simulator and the actual disk can result from an imperfect estimate of the seek time profile or other disk parameters.

As seen in Fig. 4, the simulator shows that the data-limit deterministic admission control admits 20% more users than the statistical for a 10^{-5} probability of overload. We can also theoretically predict the average number of

[5]Alternatively, the system may elect to maximize disk usage by allowing some overloads to users with a lower quality of service. Distribution of overloads among users is an interesting resource allocation problem and has been partially addressed in [3].

users for the data-limit admission control within 2% of the real disk value using the convolution techniques shown in Appendix A.

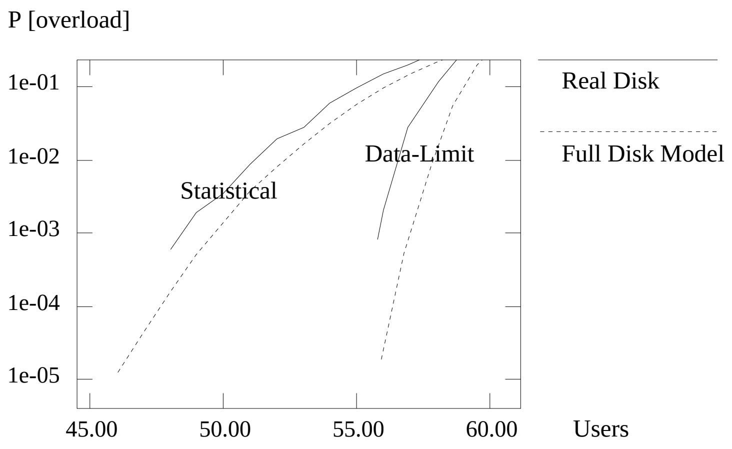

Figure 4: Comparisons between statistical and data-limit deterministic admission control

We now consider delays for data-limit deterministic admission control. The start delay is the same as for statistical admission control; users receive data exactly one service round after being admitted to the system. The jump delay, however, increases because users are not guaranteed readmission after switching videos or video starting points. The admission control grants access to a user based on the vector of future number of blocks requested in each service round; a change in this vector may cause a delay while the admission control checks each upcoming service round for a round in which readmission will satisfy Eq. 6. Both our real disk experiments and full disk model simulations show that the jump delay increases from that of statistical admission control by only 5-7% using our system parameters in Table 1 and varying the jump probability P_j from 0.1% to 10% per user per service round. We conclude that the use of data-limit deterministic admission control does not significantly degrade the interactivity potential for our video server.

4.2 Ideal Deterministic Admission Control

In statistical admission control, we assume a constant read time and constant seek time, and we admit a given number of users based on our theoretical estimates of the probability of disk overload. By using a data-limit deterministic admission control, we can reduce the probability of overload for a given number of users. However, to guarantee zero overload using that admission control, we must reduce the average number of users served by assuming a conservative bound for the total seek and rotation time. We would like to eliminate this conservative bound by extending the future load calculations to include seek and rotation times. We call this ideal deterministic admission control.

Unfortunately, ideal deterministic admission control is not possible to implement on our real disk video server, as our system uses a SCSI-2 interface, which does not allow track and sector-level data manipulation. We are therefore left with the full disk simulator which suffers no such limitations. That simulator results in an average of 59.7 users served under ideal deterministic admission control. This user load is 21% higher than the conservative data-limit deterministic admission control. As for jump delay, our simulator shows an increase of 7-10% above the jump delay of a statistical admission control system using the same test conditions as in Sect. 4.1. We conclude that neither data-limit nor ideal deterministic admission control significantly hinder interactive use.

4.3 Cost Analysis

We now present a cost analysis for our admission control strategies. All tests are done for $L = 16$, as we have found that costs do not vary significantly as a function of L for CTL and hybrid systems [5]. In Table 4, we present the system costs based on experiments using our real disk video server. For comparison, we examine the same system costs based on tests run on our full disk model simulator in Table 5. All of the systems may be classified into one of two categories: overload-prone or zero-overload. The overload-prone systems result in a positive probability of disk overload as a function of the number of users admitted. The zero-overload systems guarantee that the disk will never be overloaded.

In Table 4a, we show the system costs of the overload-prone systems as tested on the real disk for $P_{fail} = 10^{-3}$. As seen, the data-limit deterministic admission control significantly reduces the hybrid system cost compared to statistical admission control. We have found similar gains by applying the same admission control to CTL systems [5], making them the best solution for read-only systems. While CTL results in lower cost than hybrid, the fragmentation problem makes the CTL strategy unsuitable for general read and write applications. The CTL and hybrid systems cost less than the Padded VBR by a factor of 2.5 to 3.

In Table 4b, we examine the costs of the zero-overload systems. For the Padded VBR system, the cost does not increase significantly from the overload-prone case because the buffer cost per stream is fixed. For the hybrid system, however, both the disk and buffer costs increase, resulting in a 16% increase in cost from the corresponding overload-prone system. The difference in cost between the two systems is still significant, a factor of 2.5.

Table 5 shows that for all of the systems tested in Table 4, the full disk model simulator yields very similar results. In addition, the full disk model allows us to test ideal deterministic admission control which is not possible on our real disk video server. As seen, for about the same cost, one could use the ideal deterministic admission control to achieve zero overload, or one could use data-limit admission control and accept an overload probability of 10^{-3}. The actual choice will depend on the cost and feasibility of using a disk system with track and sector level manipulation of data.

	Padded	CTL	Hybrid	Hybrid
Admission	Fixed U	Stat	Stat	Data-Limit
Disk Cost	15.96	4.88	5.82	5.06
Buffer Cost	8.17	3.09	3.93	3.57
Total Cost	24.13	7.95	9.75	8.63

	Padded	Hybrid
Admission	Fixed U	Data-Limit
Disk Cost	16.62	5.68
Buffer Cost	8.17	4.31
Total Cost	24.79	9.97

Table 4: Real disk results: cost per stream for a) $P_{fail} = 10^{-3}$, b) zero-overload

	Padded	CTL	Hybrid	Hybrid
Admission	Fixed U	Stat	Stat	Data-Limit
Disk Cost	15.41	4.75	5.68	4.94
Buffer Cost	8.17	3.09	4.23	3.72
Total Cost	23.58	7.84	9.91	8.66

	Padded	Hybrid	Hybrid
Admission	Fixed U	Data-Limit	Ideal Deter
Disk Cost	16.62	5.71	4.71
Buffer Cost	8.17	4.35	4.17
Total Cost	24.79	10.06	8.88

Table 5: Full disk model results: cost per stream for a) $P_{fail} = 10^{-3}$, b) zero-overload

5 Conclusions

The main contributions of our work have been as follows. First, we have shown that CDL is an infeasible data placement strategy for long VBR video sequences because of extremely high buffer usage. Between CTL and hybrid solutions, one must choose between lower cost in the CTL case or ease of editing videos in the hybrid case. Second, we have performed an exhaustive comparison between statistical and two types of deterministic admission control, using a video server implemented on an actual disk to verify our results. We find that all forms of deterministic admission control can outperform statistical, but the greatest gain comes from using an ideal control to account for disk seek and rotation times. We note, however, that ideal deterministic admission control may be difficult to implement on standard disk controllers. Finally, we have implemented a full disk model simulator. Results using the simulator are very close to those measured on the real disk, making the simulator useful for future experiments.

The cost of using deterministic admission control is very small compared to the disk and buffer storage costs. Implementing deterministic admission control requires maintaining a current system load memory as shown in Figure 3. The size of that memory is equal to the number of service rounds multiplied by the bytes per service round. As for the computation power required, each new admission request requires as many integer additions as there are service rounds in the new request. For a two-hour movie with two-second service rounds, the amount of memory required is only 14 kB per disk, assuming 4-byte integers, and we only need 3600 integer additions for each incoming user in a 2-second service round. Our full disk simulator with deterministic admission control runs 1000 times faster than the real disk video server, showing that computation time is not a bottleneck.

6 Acknowledgements

This work is sponsored by NSF grant MIP-9057466, ONR grant N00014-92-J-1732, AFOSR grant F49620-93-1-0370, University of California MICRO award 95-169, and gifts from HP, Rockwell, and Philips. We thank John Wilkes and David Tse for useful discussions.

A Appendix A: Users served under data-limit deterministic admission control

In this section we calculate the theoretical average number of users for the hybrid system using data-limit deterministic admission control. In Sect. 3.3, we defined $N(u)$ to be the number of blocks of data requested by user u in one service round, $p_N(x)$ to be the pdf of $N(u)$, and N_{lim} to be the maximum number of blocks that may be read by all users in one service round. We found the probability of disk overload for the statistical admission control system by convolving the $p_N(x)$ U times with itself and integrating the resulting pdf beyond the block limit N_{lim}.

For the deterministic case, however, we must note that for each user index u, the distribution of total requested data is guaranteed not to exceed N_{lim} at the previous user index $u-1$. Let $S(u)$ be the sum of $N(1)\ldots N(u)$. Upon admitting the first user, the pdf of the number of blocks requested in one service round is simply $p_{S(1)}(x) = p_N(x)$.

In order to admit a second user under data-limit deterministic admission control, the number of blocks requested by the first user must not exceed N_{lim}. If $S'(1)$ denotes the number of blocks requested by the user given that $S(1) < N_{lim}$, then its pdf is given by:

$$p_{S'(1)}(x) = p_{S(1)}(x|S(1) \leq N_{lim}) = \begin{cases} \frac{p_{S(1)}(x)}{P[S(1)\leq N_{lim}]} & x \leq N_{lim} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Thus the pdf of the total number of blocks requested by two users is equal to the convolution of the pdf given in Eq. 7 and the pdf of an independent second user, $p_N(x)$:

$$p_{S(2)}(x) = p_{S'(1)}(x) * p_N(x) \tag{8}$$

This series of operations can be extended by induction to u users to yield:

$$\begin{aligned} p_{S'(u-1)}(x) &= \begin{cases} \frac{p_{S(u-1)}(x)}{P[S(u-1)\leq N_{lim}]} & x \leq N_{lim} \\ 0 & \text{otherwise} \end{cases} && (9) \\ p_{S(u)}(x) &= p_{S'(u-1)}(x) * p_N(x) && (10) \end{aligned}$$

For the data-limit deterministic admission control system, we define $P_o(u)$ as the probability that user u is rejected given that user $u-1$ was accepted, in one service round:

$$P_o(u) = P[S(u) > N_{lim}] \tag{11}$$

We can now estimate the average number of users served on the system as follows. Assuming independent service rounds, the probability of admitting the entire request length of a user is equal to the product of the probabilities of admitting a user at each service round separately.

$$\begin{aligned} \text{Prob[admit entire reqlen L]} &= \prod_{t=0}^{L-1} \text{Prob[admit at round } t] \\ &= \prod_{t=0}^{L-1} (1 - \text{Prob[reject at round } t]) \end{aligned} \tag{12}$$

The probability that a request is rejected at round t is equal to $P_o(u)$ given u users on the system at round t. Given our assumption that all users request video of the same length L, and since the trials are uniformly distributed in time, the number of users on the disk will on average fall linearly with respect to time. This is illustrated in Fig. 3 where the load has a fairly linear drop. Because of this drop, it is much more likely that a user will be rejected because of the current load at $t = 0$ rather than for a large t. We account for this by using a linear interpolation of the function $P_o(u)$:

$$P_{o,interp}(x) = P_o(\lfloor x \rfloor) + [P_o(\lceil x \rceil) - P_o(\lfloor x \rfloor)]\,(x - \lfloor x \rfloor) \tag{13}$$

Then the probability of admitting a request of length L at user load u is:

$$P_L(u) = \prod_{t=0}^{L-1} \left(1 - P_{o,interp}\left(\frac{L-t}{L} u\right)\right) \tag{14}$$

The probabilities $p_L(u)$ are the probabilities of admitting the entire request length L of user u given that user $u-1$ was admitted. We can then calculate the probability of admitting at least u users admitted as follows:

$$P_{least}(1) = P_L(1) \tag{15}$$
$$P_{least}(2) = P_L(2) \times P_L(1) \tag{16}$$
$$P_{least}(u) = \prod_{i=1}^{u} P_L(i) \tag{17}$$

Finally, we calculate the probabilities of admitting exactly u users by taking differences:

$$\begin{aligned} P_{exact}(u) &= P_{least}(u) \times P(\text{reject } u+1 \mid \text{accept at least } u) \\ &= P_{least}(u) \times [1 - P_L(u+1)] \\ &= P_{least}(u) - P_{least}(u+1) \end{aligned} \tag{18}$$
$$U_{avg} = \sum_u u P_{exact}(u) \tag{19}$$

B Appendix B: (σ, ρ) curve as a lower bound for maximum CDL buffer usage

In this section we assume a VBR video of finite duration t_{len} characterized by $D(t)$, the amount of data consumed by a user at service round index t. For the CDL system, let R_b be the disk transfer rate of one block of data to one user, measured in bytes per service round. Then assuming zero prefetch data, the buffer state $B(t)$, is found by subtracting the VBR data sent to the network from the CDL data read from the disk:

$$B(t) = \sum_{\tau=0}^{t} [R_b - D(\tau)] \tag{20}$$

If $B(t)$ becomes negative for any t, the buffer will underflow for that request. To prevent this, a buffer prefetch B_{pre} is computed by finding the most negative value of $B(t)$ over the request length and rounding up to an integral number of disk read blocks:

$$B_{pre} \;=\; \begin{cases} \lceil -\min\limits_t \dfrac{B(t)}{R_b} \rceil R_b & \min\limits_t \dfrac{B(t)}{R_b} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

The resulting buffer state after adding the prefetch data is then guaranteed not to underflow:

$$B'(t) = B_{pre} + \sum_{\tau=0}^{t} [R_b - D(\tau)] \geq 0 \tag{22}$$

For the discrete-time case, the VBR data trace $D(t)$ has a (σ, ρ) function defined as follows [6, 7]:

$$\sigma = \max_{s \leq t} \sum_{\tau=s}^{t} [D(\tau) - \rho] \tag{23}$$

ρ is the constant rate at which the buffer is drained, equivalent to R_b in our CDL system.

$$\begin{aligned} \sigma &= \max_{s \leq t} \sum_{\tau=s}^{t} [D(\tau) - R_b] & (24) \\ &= \max_{s \leq t} \left[\sum_{\tau=0}^{t} [D(\tau) - R_b] - \sum_{\tau=0}^{s} [D(\tau) - R_b] \right] & (25) \\ &= \max_{t} \sum_{\tau=0}^{t} [D(\tau) - R_b] - \min_{s} \sum_{\tau=0}^{s} [D(\tau) - R_b], \qquad s \leq t & (26) \end{aligned}$$

The maximum amount of buffer needed by one user in a CDL system is $\max\limits_t B'(t)$. We now show that the elements in $\max\limits_t B'(t)$ are bounded below by those in σ. For the first element:

$$\max_t B_{pre} \geq \lceil -\min_t \frac{B(t)}{R_b} \rceil R_b = \lceil \max_t \frac{-B(t)}{R_b} \rceil R_b \geq \max_t [-B(t)] = \max_t \sum_{\tau=0}^{t} [D(\tau) - R_b] \tag{27}$$

For the second element:

$$\max_t \sum_{\tau=0}^{t} [R_b - D(\tau)] \geq \max_{s \leq t} \sum_{\tau=0}^{s} [R_b - D(\tau)] = -\min_{s \leq t} \sum_{\tau=0}^{s} [D(\tau) - R_b] \tag{28}$$

Therefore $\max\limits_t B'(t) \geq \sigma$.

References

[1] D. Anderson, Y. Osawa, and R. Govindan, "A File System for Continuous Media," *ACM Transactions on Computer Systems*, vol. 10, no. 4, Nov. 1992, pp. 311-337

[2] E. Chang and A. Zakhor, "Scalable Video Data Placement on Parallel Disk Arrays," *IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology, Volume 2185: Image and Video Databases II*, San Jose, February 1994, pp. 208-221

[3] E. Chang and A. Zakhor, "Variable Bit Rate MPEG Video Storage on Parallel Disk Arrays," *First International Workshop on Community Networking Integrated Multimedia Services to the Home*, San Francisco, July 1994, pp. 127-137

[4] E. Chang and A. Zakhor, "Admissions Control and Data Placement for VBR Video Servers," *First IEEE International Conference on Image Processing*, Austin, November 1994, pp. I:278-282

[5] E. Chang, "Scalable and Variable Bit Rate Video Compression and Storage," Ph.D. thesis, University of California at Berkeley, 1996

[6] S. Chong, S. Q. Li, "(σ, ρ)-Characterization Based Connection Control for Guaranteed Services in High Speed Networks," *IEEE INFOCOM '95*, Boston, April 1995, pp. 835-844

[7] R. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Transactions on Information Theory*, Jan. 1991, v. 37, no. 1, pp. 114-131

[8] R. Cruz, "A Calculus for Network Delay, Part II: Network Analysis," *IEEE Transactions on Information Theory*, Jan. 1991, v. 37, no. 1, pp. 132-141

[9] T. Costlow, "Disk arrays spark video-on-demand," *Electronic Engineering Times*, June 20, 1994, pp. 51-55

[10] M. Garrett, "Contributions Toward Real-Time Services on Packet Switched Networks," Ph.D. thesis, Columbia University, New York, NY, 1993

[11] M. Garrett, "Analysis, Modeling, and Generation of Self-Similar VBR Video Traffic," ACM SigComm, London, Sept. 1994

[12] D. J. Gemmell and J. Han, "Multimedia network file servers: multichannel delay-sensitive data retrieval," *Multimedia Systems* (1994), pp. I:240-252

[13] M. Grossglauser, S. Keshav, and D. Tse, "RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic," *ACM SIGCOMM '95*

[14] D. Kandlur, M. S. Chen, Z. Y. Shae, "Design of a multimedia storage server," *IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology, Volume 2188*, San Jose, February 1994, pp. 164-178

[15] R. Katz, G. Gibson, and D. Patterson, "Disk System Architectures for High Performance Computing," *Proceedings of the IEEE*, Dec. 1989, v. 77, no. 12, pp. 1842-1858

[16] G. Neufeld, D. Makaroff, and N. Hutchinson, "The Design of a Variable Bit Rate Continuous Media Server," *Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video,* Durham, NH, April 1995, pp. 375-378

[17] A. Reibman and B. Haskell, "Constraints on Variable Bit-Rate Video for ATM Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, Dec. 1992, v. 2, no. 4, pp. 361-372

[18] J. M. del Rosario and G. Fox, "Constant Bit Rate Network Transmission of Variable Bit Rate Continuous Media in Video-On-Demand Servers," NPAC Technical Report SCCS-677, Syracuse University, Dec. 1994

[19] J. M. del Rosario, M. Podgorny, and G. Fox, "m-Frame granular transport and buffer requirements for VBR encoded media in VOD servers," NPAC Technical Report SCCS-733, Syracuse University, March 1995

[20] C. Ruemmler and J. Wilkes, "An Introduction to Disk Drive Modelling," *Computer*, March 1994, v. 27, no. 3, pp. 17-28

[21] S. Stoller and J. DeTreville, "Storage Replication and Layout in Video-on-Demand Servers," *Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video,* Durham, NH, April 1995, pp. 351-363

[22] F. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming RAID – A Disk Array Management System for Video Files," *First ACM International Conference on Multimedia*, Anaheim, CA, Aug. 1993

[23] H. Vin, P. Goyal, A. Goyal, and A. Goyal, "A Statistical Admission Control Algorithm for Multimedia Servers," *Proceedings of the ACM Multimedia '94*, San Francisco, CA, Oct. 1994, pp. 33-40

[24] H. Vin, A. Goyal, P. Goyal, "Algorithms for Designing Multimedia Servers," *Computer Communications*, March 1995, v. 18, no. 3, pp. 192-203

[25] H. Vin, S. Rao, P. Goyal, "Optimizing the Placement of Multimedia Objects on Disk Arrays," *Proceedings of the International Conference on Multimedia Computing and Systems*, Washington, DC, May 1995, pp. 158-165

[26] J. Walrand, *Queueing Networks*, Prentice Hall, 1988

[27] A. Wong, C. T. Chen, D. J. Le Gall, F. C. Jeng, K. M. Uz, "MCPIC: A Video Coding Algorithm for Transmission and Storage Applications," *IEEE Communications Magazine*, November 1990, pp. 24-32

[28] B. L. Worthington, G. R. Ganger, Y. N. Patt, J. Wilkes, "On-Line Extraction of SCSI Disk Drive Parameters," *ACM SIGMETRICS '95*, Ottawa, Ontario, Canada, 1995, pp. 146-156

[29] P. Yu, M. Chen, and D. Kandlur, "Design and Analysis of a Grouped Sweeping Scheme for Multimedia Storage Management," *Proc. Third International Workshop on Network and Operating System Support for Digital Audio and Video*, Nov. 1992, San Diego