Cost Analyses for VBR Video Servers

Ed Chang and Avideh Zakhor
University of California, Berkeley

Comparing techniques for storage and real-time retrieval of variable-bit-rate video data for multiple simultaneous users, we conclude that constant time length and hybrid techniques can greatly reduce the total system cost. Of three admission control techniques, data-limit admission control offers a moderate gain for a moderate increase in implementation complexity. We also applied our data placement and admission control strategies to an interleaved disk array.

Video data placement and admission control for video servers pose two of the most important challenges facing video-on-demand applications. In the following discussion, we investigate storage and real-time retrieval of video data for multiple simultaneous users. We chose a disk as our storage medium because disk arrays provide cost-effective storage and high-bandwidth transfer capabilities, and because of their growing popularity in VOD systems.

For storage efficiency we compress the data before writing to disk. Given several different modes of encoding, we must decide the type of rate control mechanism, if any, to apply in generating the compressed bit stream. One option generates a truly variable-bit-rate (VBR), constant-quality stream without any rate control, buffers, or quantization feedback. Another possibility generates 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.

Presumably, video inherently results in variable-bit-rate data. Hence, the main advantage of true VBR over CBR is its true constant quality. As a result, when using CBR, you must choose between low bit rates and nonuniform quality or very high bit rates and uniform quality. To maintain uniform quality, you must choose the CBR bit rate at a level high enough to ensure the quality remains above a certain threshold during high-motion parts of the video sequence. Unfortunately, this latter approach results in over-allocation of storage resources in storing CBR data.

Since typical VBR video may have a peak to mean ratio of 3:1 even averaged over a few seconds, we could conceivably achieve the same video quality at about one-third the cost by storing VBR rather than CBR video data on disks. Therefore, the same statistical multiplexing ideas used in networking applications can reduce the cost per stream in VBR video storage applications.

However, one major difference distinguishes networking and storage applications: Unlike the networking applications, the storage scenario exploits the a priori knowledge of the video bit trace to optimize both the data placement algorithms and the admission control algorithms. This knowledge can be stored in memory as an index table, as it comprises a small fraction of the size of stored VBR video data.

The choice of true VBR versus CBR also influences the data units used to store the compressed bit stream on the disks. For CBR video, the data can be stored and retrieved in constant-sized data units without endangering jitter-free, real-time video delivery.[1,2] For VBR data, the unit sizes written to and read from the disk are not 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 units for each user, using buffer memory to provide real-time variable bit rate for playback.

We call the first method *constant time length* (CTL) and the second method *constant data length* (CDL). We also examine a third method, "hybrid," in which we store CDL units but retrieve a variable number of units for each user in each round. Each user in the hybrid scheme retrieves a few constant-length units, and each read unit requires a separate disk seek. Each user in the CTL scheme, on the other hand, retrieves exactly one variable-length unit consisting of many contiguous disk blocks.

CTL-stored videos are characterized by a unique pattern of variable data unit lengths derivable from the video bit trace and the real-time duration of the units. Because data units may interleave across many disks (as shown later), the stored units might not be contiguous. In this case, replacing or editing the video results in disk fragmentation problems. To avoid this, we consider a CDL system. However, as we will see later, CDL might result in large buffer usage, hence the motivation for the hybrid system. That system stores

1070-986X/96/$5.00 © 1996 IEEE

data in CDL units, but the number of units retrieved varies with the playback consumption requirements. This strategy reduces buffer requirements and results in a good compromise between low cost and low fragmentation.

To avoid over-allocating system resources when too many users request data simultaneously, we require admission control algorithms. We consider two main classes of admission control: *statistical* and *deterministic*. Statistical admission control exploits the bit-rate statistics of the videos on the disk. This strategy has the advantage over network admission control schemes of using 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. By storing the bit-rate traces in index tables, the disk system has full knowledge of the future traces needed to service all user requests. Thus it 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. Vin et al.[3] compared the performance of CTL and hybrid systems for a multiple-disk multimedia network server. For admission control, Vin et al. proposed statistical[4] and adaptive[5] policies. Gemmell[6] also considered a variable number of fixed size units, which can be grouped into sorting sets to reduce disk latencies.

In this article, we provide a cost/benefit analysis of the above placement, retrieval, and admission control techniques and conclude that CTL and hybrid placement and retrieval techniques can reduce the total system cost by up to a factor of three in comparison with the strategy of padding the VBR video trace to achieve a constant data rate. We will show that 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 ideal deterministic admission control offers the greatest gain but might be difficult to implement on standard disk controllers. Data-limit admission control, on the other hand, offers a moderate gain for a moderate increase in implementation complexity. Finally, we implemented a full disk model simulator that operates a thousand times faster than the real-time disk. Results using the simulator came very close to those measured on the real disk, making the simulator useful for experiments.

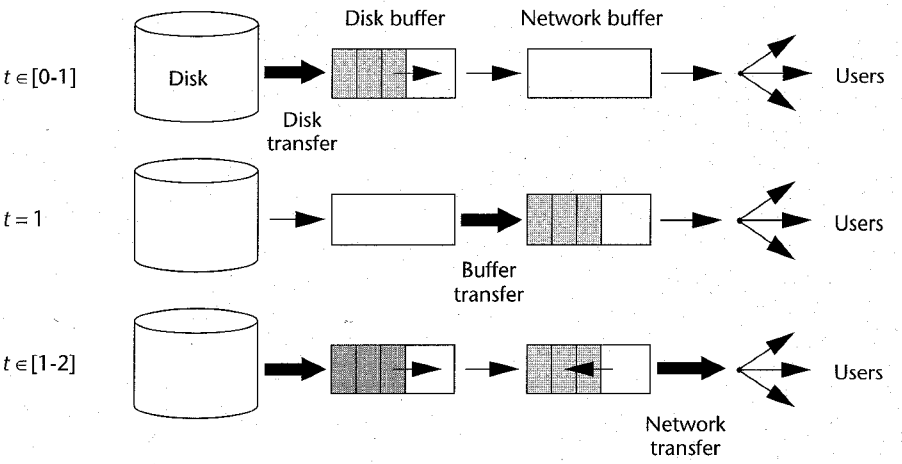


Figure 1. Dual-buffer operation.

System operation and metrics

To evaluate our strategies, we must specify how our video server system operates 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. Figure 1 illustrates the operation of the dual-buffer system.[2,6]

During the first service round, $t \in [0\text{-}1]$, the disk transfers data to the disk buffer, as shown by the heavy arrow marked "disk transfer." At the end of the round, $t = 1$, data is transferred instantaneously from the disk buffer to the network buffer; we call this stage the "buffer transfer."

During the second service round, $t \in [1\text{-}2]$, another disk transfer occurs while the network buffer performs a "network transfer," sending its data to the users. Subsequent rounds proceed in the same manner. Thus the total system buffer is the sum of the disk and network buffers. (An alternative architecture distributes 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 many more clients than the server can simultaneously support, we place the network buffer at the server.)

Using this dual-buffer assumption, we do not have to schedule users in any given order; we simply ensure that every user gets scheduled within a given service round. We assume the user requests in

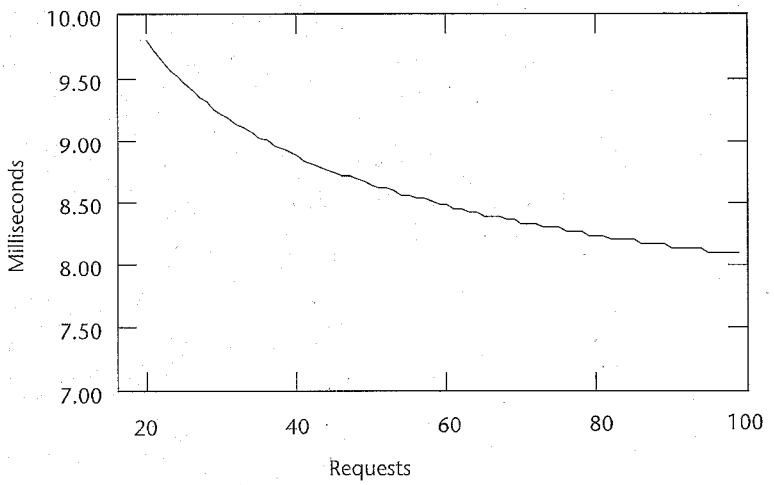


Figure 2. Seek time for equally spaced requests using the Scan algorithm.

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,[6] but lies beyond the scope of this article.

Seek-time models

We now consider the disk operation in more detail. Assume each disk request takes the form of a "seek" operation followed by a "read" operation. We find that we can model the disk read rate, R_d, as a constant by keeping our data on the fastest zone of the disk, which comprises 70 percent of the storage capacity. We measure R_d to be 5.03 Mbytes per second on our HP C3325W disk.

The seek time, however, proves more difficult to model and remains an important issue throughout this article. An inherent trade-off exists between accuracy and complexity of seek models. For instance, if we took into account track and sector locations of data and precisely modeled the actual rotation and movements of the head, then we would have an accurate but complex model. On the other hand, if we assume a constant seek and rotation time, then we have sacrificed accuracy for the lower complexity used in our simulations.

We propose three different models for the seek time and use them in different scenarios. For the first, which we refer to as "typical seek time," we use techniques from Worthington et al.[7] on our HP C3325W disk to estimate $T_{seek}(n)$, 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 an equal number of tracks falls between each requested read unit. Doing so maximizes the total seek time under the Scan algorithm[2] and results in a conservative seek time estimate of $T_{seek}(\lceil n_{total}/r \rceil)$ for each request.

We graph this seek time as a function of the number of requests r in Figure 2 for our HP C3325W disk. As we will see later, we serve about 50 to 60 users for our chosen parameters, resulting in a pessimistic disk seek time of 8.5 ms. If we assume an average half-revolution time of 5.5 ms, we can add the pessimistic seek and average rotation times to obtain the "typical seek time" estimate T_s = 14 ms. As seen in Figure 2, the actual disk seek time is relatively insensitive to the number of users, so we keep this typical seek time estimate throughout the discussion.

Our second model, the "worst case" estimate, uses evenly-spaced requests on the disk for the seek time but full disk revolutions for the rotation time. For our disk, this results in an estimate of 8.5 ms + 11 ms = 19.5 ms. Since both seek and rotation time estimates are conservative, the worst-case estimate provides an upper bound to the total time.

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.

We used the typical seek time model in simulations of data placement and obtained excellent agreement with experimental data from a real disk. For admission control, however, we found this model inadequate and resorted to the worst-case and actual location models.

System parameters and metrics

Table 1 lists our system parameters. The first two parameters represent characteristics of the physical disk, as measured on our disk drive using the techniques described above. The last three represent characteristics of the video. R_v denotes the average coded bit rate of our chosen VBR sequence, *Star Wars*.

To examine both short and long video sequences, we tested request lengths of 30 seconds or two hours. We chose an upper bound on the probability of service-round failure to be one per half hour. This means that in a fully loaded video server, on average one user per half hour will experience a glitch.

Because video tolerates dropped frames fairly well, we might consider different users requesting different qualities of service, each with different probabilities of failure. However, past work has shown that allowing higher probabilities of overload does not substantially increase the number of users served.[8]

Now consider the following quality metrics. First, the cost per stream should be as low as pos-

sible. The cost per stream divides into two parts: the disk and the buffer. The cost of the disk, C_d, equals 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, equals 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 cost per stream. Table 2 lists current prices and the size of our chosen VBR video sequence, *Star Wars*.

We also consider start delay and jump delay. Start delay, also known as latency, measures the amount of time between a user's admission onto the system and the actual delivery of data from the network buffer to the user. The delay between 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 queuing delay as part of our latency. (Commonly known in queuing theory as Little's result, this function has been empirically confirmed in our tests.) Jump delay is the amount of time between the end of a service round that contains a jump request and actual playback of jump destination data.

Data placement

In comparing the three methods of data placement, we first describe the simple statistical admission control strategy. We then examine CTL, CDL, and a hybrid strategy. In each case, we calculated the theoretical average number of users served and the buffer requirement, and verified the results through experiments on a real disk. Finally we compare the total system costs and discuss interactivity.

Statistical admission control

To limit disk and buffer usage during VBR video retrieval, we use an admission control algorithm to regulate the number of users. The simplest method, statistical admission control,[4,8,9] admits users up to a predetermined threshold. We derive this threshold by using *statistics* of the stored data to estimate a probability of disk or buffer overload. Later we consider more complex deterministic admission control algorithms that use a priori knowledge of the actual bit traces in index tables stored in memory.

The data rate variation of our VBR bit trace shows up in different ways for each of the data placement strategies. For CTL, the read unit sizes vary; for CDL, the network buffer size varies; and for hybrid, the number of retrieved units varies.

Table 1. System parameters for our experiment.

Symbol	Description	Value
R_d	Disk read rate	5.03 Mbytes/s
T_s	"Typical" disk seek and rotation time	14 ms
R_v	Average bit rate of video	374,369 bits/s
L	Request length	30 s, 2 hrs
P_{fail}	Probability of service failure	1 per half hour

Table 2. Cost parameters.

Symbol	Description	Value
C_{dc}	Cost of disk controller	$200
$Price_d$	Price of disk storage	$0.25/Mbyte
D_v	Data size of video sequence, *Star Wars*	325 Mbytes
$Price_{mem}$	Price of DRAM	$15/Mbyte

Since each strategy is characterized by a different random variable, we describe a general statistical admission control strategy as follows:

- Assume user i requires a resource (such as disk throughput or buffer usage) with probability density function (PDF) $p_i(x)$. The PDFs are assumed known, since the server can precisely compute the histograms at the time the videos are stored.
- Assume $U-1$ users on the system, and consider admitting user U.
- Compute $p_{agg}(x)$, the PDF of the aggregate resource required by U users, by convolving their PDFs. (In previous work,[8] we estimated this function reasonably well with the Central Limit Theorem and Cramer's rule.)
- Integrate the aggregate PDF beyond a given threshold limit to find the probability of overload, $P_o(U)$.
- If $P_o(U)$ exceeds the chosen failure threshold, reject the user; otherwise, admit.

Constant time length

Earlier, we defined CTL as a data placement strategy in which stored read unit sizes are proportional to their corresponding playback bit rates. Figure 3 on the next page shows two service rounds of disk operation with four users; as seen, the read unit sizes vary between rounds and between users. Using round-robin scheduling, the disk fills the

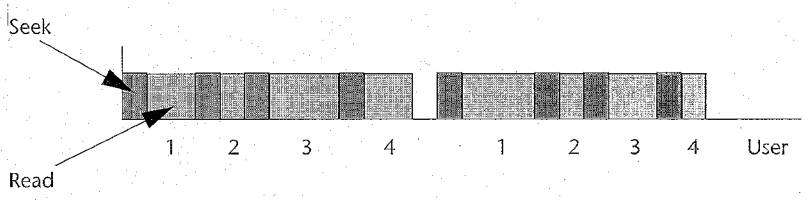


Figure 3. CTL disk operation.

disk buffer with data for each user once per round.

The amount of data sent to any one user in one round exactly equals the amount the user will consume during one round of video playback, rounded up to the next 1-kilobyte disk block. The extra data from the rounding, which is discarded, amounts to less than 1 percent of the data read from disk. Since the amounts consumed by each user in each round equal the amounts read from disk, Figure 1 clearly shows that the network transfer equals the disk transfer in the previous round. Hence we chose disk and network buffers of the same size.

Because the total amount of data retrieved by all users in one service round varies, disk overload becomes a primary concern in CTL data placement and retrieval. A straightforward statistical admission control strategy for CTL data relies on analyzing the probability that the total data requested by all users in one service round exceeds the disk throughput capacity.

Table 3. Theoretical cost per stream for CTL as a function of T_{SR}.

T_{SR} (seconds)	0.5	1.0	1.5	2.0	2.5	3.0	4.0	6.0
Users	25	40	50	58	63	67	74	82
Disk cost (dollars)	11.24	7.03	5.62	4.85	4.46	4.19	3.80	3.43
Buffer cost (dollars)	0.90	1.66	2.41	3.09	3.88	4.64	6.05	8.91
Total cost (dollars)	12.14	8.69	8.03	7.94	8.34	8.83	9.85	12.34

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 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 $T_r(u)$ equal the ratio of the amounts of requested data $D_r(u)$ and the disk read rate R_d. Thus the no-overload condition is given by

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

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

We therefore define the disk read limit as the maximum amount of data that can be read by U users in one service round, $D_{lim}(U) = R_d (T_{SR} - UT_s)$. By definition, in statistical admission control the server does not keep track of individual traces of the stored videos; rather it keeps their histograms in a table. Let the PDF of the video trace requested by the ith user be $p_i(x)$. Then we can state the resulting statistical admission control algorithm for the Uth user given $U-1$ users on the system as follows:

1. Compute the PDF of the aggregate video requested by U users, $p_{agg}(x) = p_1(x) * p_2(x) * \ldots * p_U(x)$.

2. Integrate $p_{agg}(x)$ beyond $R_d (T_{SR} - UT_s)$.

3. If the quantity computed in step 2 exceeds P_{fail}, reject the Uth user, otherwise admit.

In this procedure, the service-round duration T_{SR} determines both the read unit sizes and the maximum number of users served. We therefore choose a service-round duration T_{SR} that minimizes the total system cost per user as follows. The *Star Wars* sequence is stored at 24 frames per second, with 12 frames per group of pictures (GOP). To ensure that no GOPs will be split across service rounds for CTL data placement, we consider various values of T_{SR} in step sizes of 0.5 seconds. We then use the prices in Table 2 to calculate the total costs per stream in Table 3. As you can see, the lowest cost per stream occurs for $T_{SR} = 2$ seconds, although the cost varies by only 10 percent for $T_{SR} \in [1, 3]$ seconds.

To show how we arrived at the entries in Table 3, we use the minimum cost point as an example. Figure 4 shows the histogram of CTL read unit sizes at $T_{SR} = 2$ seconds. We use this histogram as the PDF of each user's video in step 1 of the admission control algorithm above. For steps 2 and 3, we use the parameters in Table 1 and vary the number of users U. We find that the maximum number of users each reading the movie *Star Wars* from the disk is $U = 58$. We verify the average number of users served on our real disk video server for the same probability of overload to within 0.7 percent. We know the disk cost is $281; thus the disk cost per user is $281/58 = $4.85.

The disk buffer must accommodate a disk transfer of the threshold size given above. As mentioned, the disk buffer and network buffer are the

same size, so the total buffer needed for this system is twice the maximum amount of data that the disk can send to all users in one service round. Assuming U_{max} users on the system, the total buffer required is $B_{total} = 2 \times R_d (T_{SR} - U_{max} T_s) = 2 \times 5{,}151$ Kbytes/s $\times$ (2s – 58 $\times$ 0.014s) = 12,238 Kbytes. The average buffer required by 58 users therefore measures 211 Kbytes per user. We have verified this number experimentally to within 0.5 percent on our HP C3325W disk. At our assumed memory price of $15 per Mbyte, the buffer cost per user is $15 $\times$ 211 Kbytes divided by 1,024 Kbytes/Mbyte = $3.09, for a total system cost of $7.94.

The CTL system latency equals exactly one service round, the amount of time required to fill the disk buffer. The data then undergoes a buffer transfer to the network buffer, as shown in Figure 1, where it immediately becomes available to the user. Interactivity does not pose a problem for the CTL system, as the jump delay also takes only one service round. In CTL, users impose the same load on the video server whether they play videos forward or backward, scan by skipping read units, or play in completely random order, as long as each user retrieves only one variable-length unit from disk in each service round.

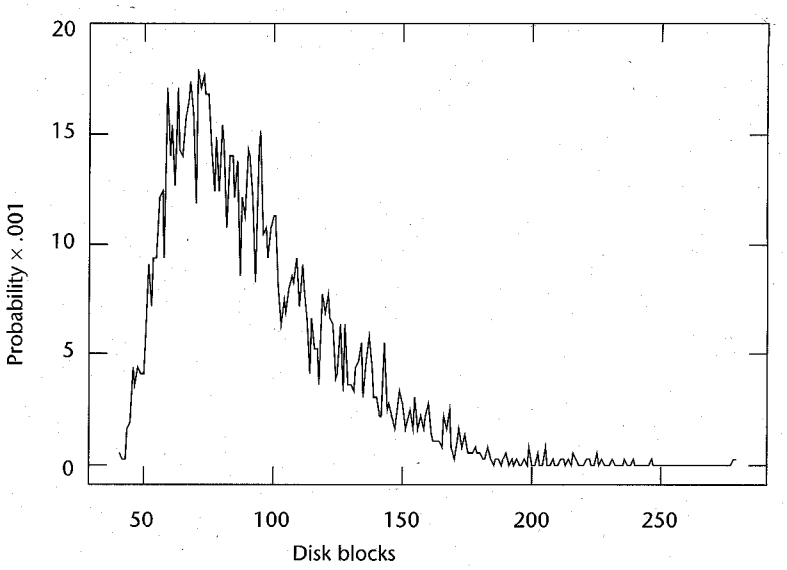
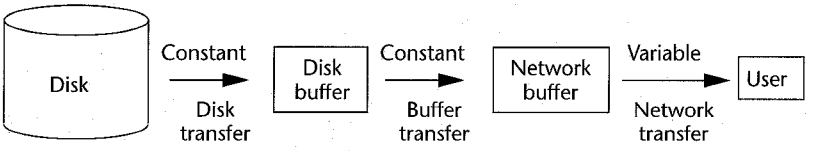


Figure 4. Histogram of CTL unit sizes.



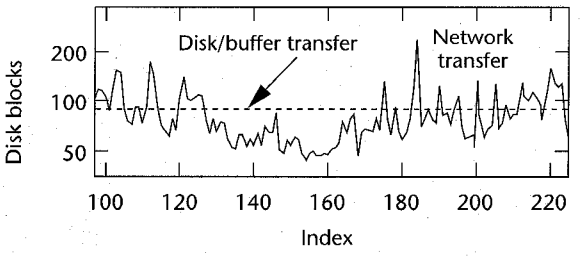


Figure 5. CDL operation.

Constant data length

The most common method of storing video data stores and retrieves constant-sized read units. One method of storing CDL read units for VBR data uses a leaky bucket mechanism for buffer control, as shown in Figure 5. The principle involves moving the data variation from the disks to the buffer. Specifically, the leaky bucket mechanism allows constant rate input from the disk, one read unit per user per round, and a variable rate output to the network. (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. A video coder has variable rate input from the coder to the buffer and constant rate output from the buffer, resulting in CBR video.)

As Figure 5 illustrates, the network buffer smoothes out the variations in the user-requested playback rate to make it compatible with the constant disk transfer rate. As a result, we must ensure that it does not underflow or overflow.

Because the total number of users can overload both the disk throughput and the buffers available, we need to consider three constraints: disk throughput, disk buffer, and network buffer. Disk throughput issues for CDL data placement of CBR video have already been analyzed.[1,2] For CDL placement of VBR video, data retrieval from disk is the same: one CDL unit per service round per user. We can thus operate the disks at full capacity. We define

$$U_{disk} = \left\lfloor \frac{T_{SR}}{T_S + T_{read}} \right\rfloor$$

as 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 size is the same for CDL placement of both CBR and VBR video—the total amount of data that all users can read from the disk in one service round. Given the fixed disk read bandwidth, we can choose the size of this

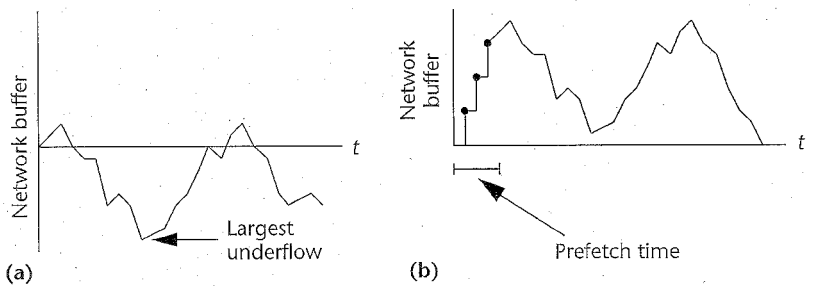


Figure 6. CDL buffer (a) without prefetch and (b) with prefetch.

buffer to avoid overflow. The network buffer for VBR video, however, must absorb the variations in the video bit trace for each user. We thus consider the two problems of network buffer underflow and overflow.

If the user started consuming video the same round the network buffer started receiving data from the disk buffer, the total consumption might exceed the total input to the network buffer. This results in network buffer underflow, as shown in Figure 6a. To ensure that the network buffer does not underflow and starve the user, the system begins reading video from disk for a "prefetch" period before the user begins watching video. During this time, the network buffer fills at a constant rate but does not drain, as shown in Figure 6b. Once the network buffer level is high enough to prevent underflow, it begins to transfer data to the user at the variable playback rate.

To calculate the number of prefetch rounds required for a given video, we find the largest underflow over the length of the video (as shown in Figure 6a) and divide by the CDL data unit size. We assume the system has done this calculation for each video at the time of video storage and therefore knows the number of prefetch rounds of each video a priori.

The buffer overflow problem proves more difficult, since each user will require a variable amount of buffer depending on the video sequence accessed. To implement statistical admission control for CDL, the server uses a table of buffer state histograms to analyze the probability of buffer overflow. Specifically, let $p_i(x)$ denote the PDF of the amount of network buffer needed by the video requested by user i. We set our threshold limit as the total amount of memory we will install on the video server for the network buffer for all users. The resulting statistical admission control algorithm for the Uth user given $U-1$ users on the system can be stated as follows:

1. Compute the PDF of the aggregate buffer states of videos requested by U users, $p_{agg}(x) = p_1(x) * p_2(x) *\ldots * p_U(x)$.
2. Integrate $p_{agg}(x)$ beyond the threshold limit.
3. If the quantity computed in step 2 exceeds P_{fail}, reject the Uth user, otherwise admit.

As in the CTL case, we compute the probability of overload by convolving $p(x)$ U times and integrating beyond a threshold limit of the available network buffer size. The CDL case, however, has two important differences. First, the threshold limit is not specified by the disk parameters; instead it is chosen arbitrarily as the amount of buffer to be added to the disk system. If large enough, it can result in zero probability of overflow. Second, the buffer usage histograms and hence probability densities $p(x)$ are functions of the request length L.

We tested the system using the *Star Wars* sequence for the request lengths of 30 seconds and 2 hours. For the test, we chose an arbitrary service round duration T_{SR} of 2 seconds. (Later, we show that total system cost is more a function of the video request length than the service round length.) To avoid cumulative buffer underflow or overflow, we set the CDL read unit size, D_{read}, to equal the average of two seconds of video, 92 Kbytes. The disk read time for each CDL unit is thus 92 Kbytes/(5.03 Mbytes/s × 1,024 Kbytes/Mbyte) = 0.01786 seconds. To maximize disk use, we set the statistical admission control test to allow

$$U_{disk} = \left\lfloor \frac{T_{SR}}{T_s + T_{read}} \right\rfloor = \frac{2}{0.014 + 0.01786} = 62$$

users on the disk at all times.

The disk buffer size simply equals the amount of data that all users can read from disk in one service round, $U_{disk} \times D_{read} = 62 \times 92$ Kbytes = 5,704 Kbytes. To calculate the theoretical network buffer required, we convolve the buffer histograms as described above. We find that the buffer requirements increase greatly with request length. At L = 30 seconds, the buffer requirement is 372 Kbytes per user. At the full request length of two hours, the buffer requirement is 15 Mbytes per user—72 times the requirement of the CTL system. This disproportionate number results from long-range dependencies in the video sequence. We have shown that the buffer required by one user is

bounded below by the (σ, ρ) curve[10] of the video sequence requested.[11]

Because these buffer amounts exceed the capacity of our video server for *Star Wars*, we use a discrete event simulator to track buffer use and verify the theoretical results. As you can see in Figure 7, the simulation results agree closely with the convolution. (We have also used the Central Limit Theorem to approximate buffer use and found the results in close agreement.[11]) Although the disk can serve four more users than the CTL system, the increase in buffer requirement overshadows that gain, as we will show in the section on cost analysis.

Earlier we defined the start delay as 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. User systems in the CDL scheme prefetch data at a constant rate of one block per round to fill the network buffer, preventing underflow. Thus the average additional start delay is directly proportional to the average amount of prefetch.

In our simulation with user request length L = 30 seconds, we find an average prefetch of 2.06 rounds, equivalent to 2.06 × 92 Kbytes per round = 190 Kbytes of data or 2.06 × 2 seconds per round = 4.12 seconds of delay. In our test with L = 2 hours, we found an average prefetch of 127 rounds of data, equivalent to 11,684 Kbytes of data or 4.2 minutes of delay. These delays come on top of the one-round delay used to fill the disk buffer, as shown in Figure 1. The total delays apply to each user upon beginning a request and—even more objectionably—to each user seeking readmission after a jump. Each jump renders the current buffer useless, and before playback can resume, a long delay occurs while refilling the buffer.

Hybrid data placement and retrieval

The final data placement scheme we consider consists of a hybrid system in which data is written in CDL units, but each user retrieves a different number of units in each service round corresponding to the VBR video playback rate. The hybrid system resembles CTL, but with two main differences.

First, the read units in the hybrid system are much more coarsely quantized. Whereas each user in a CTL system can read hundreds of contiguous 1-Kbyte disk blocks per round, each user in the hybrid system reads zero to a few noncontiguous large units of data. The remaining data not consumed by the end of the service round is stored in the network buffer to prevent the same hybrid unit from being reread in the next service round.

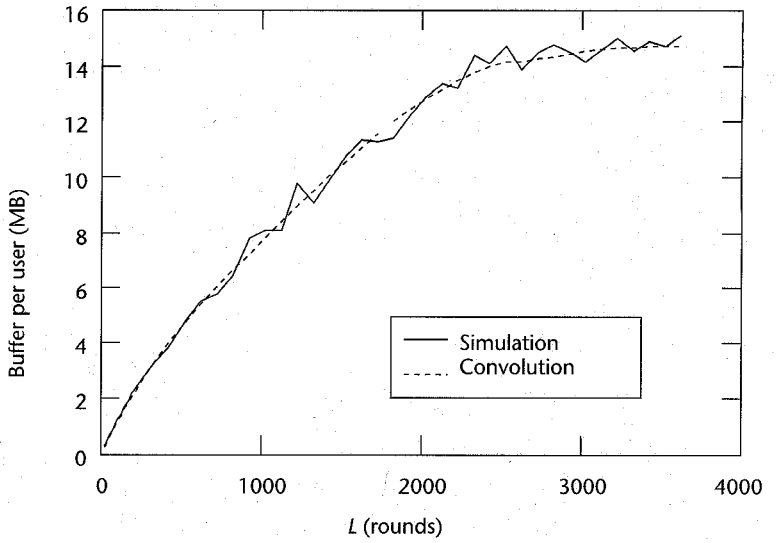


Figure 7. CDL network buffer requirements.

Second, no seeks take place between each disk block of a CTL read unit, whereas a seek occurs before each unit of data read for a hybrid system user. Specifically, hybrid system users reading multiple units in one round must perform multiple seeks to access those units. These additional seeks reduce disk efficiency.

Because the total number of read units retrieved by all users in one service round varies, the possibility of disk overload becomes the primary concern in the hybrid system. To avoid disk overload, the total time required to service all users must not exceed T_{SR}. Each read unit requires a seek of duration T_s and a read of duration T_r. If we define $N(u)$ as the number of units read by user u, the no-overload condition is

$$\sum_{u=1}^{U} N(u)\left(T_s + T_r\right) \le T_{SR}$$

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

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

In statistical admission control, the server keeps a table of histograms of the number of read

Table 4. Theoretical hybrid system costs.

TSR	Hybrid Read Size (× 23 Kbytes)							
(× 0.5 s)	1	2	3	4	5	6	7	8
1	15.64	12.09	11.38	11.88	12.38	12.38	12.75	13.34
2	15.52	10.74	9.77	9.82	10.21	10.64	10.97	11.46
3	16.14	11.12	9.93	9.67	9.88	10.35	10.70	11.17
4	16.82	11.70	10.39	10.12	10.10	10.35	10.80	11.28
5	17.52	12.37	11.10	10.67	10.72	10.86	11.11	11.58
6	18.18	13.14	11.81	11.34	11.30	11.40	11.65	12.01
7	18.88	13.83	12.50	12.10	11.94	12.07	12.23	12.57
8	19.68	14.56	13.19	12.74	12.66	12.71	12.92	13.16

Table 5. Histogram of 92-Kbyte units per request in hybrid system.

Read Units	0	1	2	3
Probability	0.2885	0.6782	0.0329	0.0004

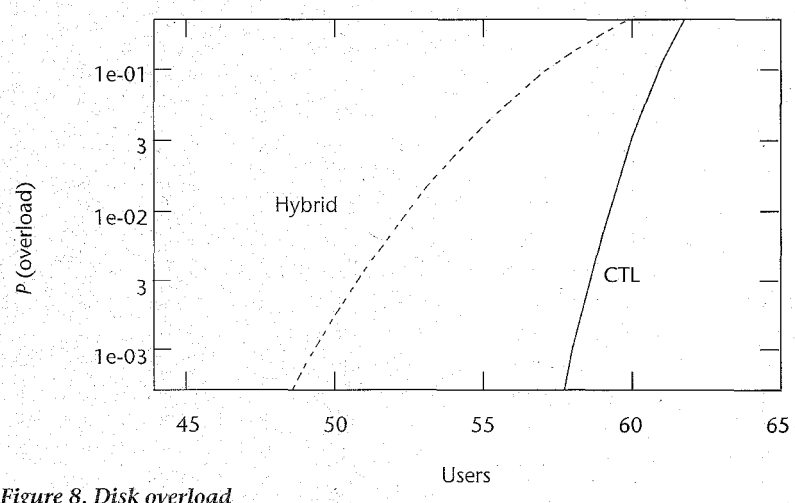


Figure 8. Disk overload probability.

units requested for each stored video. Let the PDF of the number of units requested by the ith user be $p_i(x)$. Then the resulting statistical admission control algorithm for the Uth user given $U - 1$ users on the system follows:

1. Compute the PDF of the aggregate number of units requested by U users, $p_{agg}(x) = p_1(x) * p_2(x) * \ldots * p_U(x)$.
2. Integrate $p_{agg}(x)$ beyond N_{lim}.
3. If the quantity computed in step 2 exceeds P_{fail}, reject the Uth user, otherwise admit.

In the procedure above, you can choose both the service-round duration and the hybrid read unit sizes independently. We varied T_{SR} in steps of 0.5 seconds and the hybrid read unit sizes in steps of 23 Kbytes, the average size of 1 GOP with playback duration of 0.5 seconds. As seen in Table 4, the minimum-cost operating point is $T_{SR} = 1.5$ seconds, hybrid read unit size = 92 Kbytes.

To show how we arrived at the entries in Table 4, we use the minimum-cost operating point as an example. Just as we showed a histogram of CTL read unit sizes in Figure 4, we compiled a histogram of the number of hybrid units retrieved by one user per round in Table 5. As expected, the average number of units per request equals 0.75, the ratio of the service-round duration to the average read unit playback duration.

Using the parameters of Table 1 and a 92-Kbyte read unit size as described above, we found a read time of 17.86 ms and a seek time of 14 ms, for a total read unit access time of 31.86 ms. Dividing the service-round duration of 1.5 seconds by this read unit access time results in a maximum of 47.08 units to retrieve in one service round. Since we do not allow retrieval of fractional units, we set a threshold N_{lim} of 47 units for our statistical admission control. Using the admission control algorithm described above, we found we can serve 49 users at our chosen P_{fail} of one per half hour. We used the disk cost of $281 to obtain an average disk cost per user of $281/49 = $5.73.

Compared to the lowest-cost CTL system, the hybrid system has a higher disk cost due to the extra disk seeks required. We plot the theoretical disk overload probability $P_o(U)$ for both the hybrid and CTL schemes in Figure 8. As you can see, the hybrid system serves fewer users for all tested probabilities of overload.

Now consider the hybrid system's buffer use. The size of the disk buffer equals the amount of data that can be read from disk in one round, 47 units × 92 Kbytes/unit = 4,324 Kbytes. The network buffer equals this amount plus the leftover data of all users from the previous round, given as follows. Let D_{unit} be the amount of data in each unit, 92 Kbytes in our tests. Since users read data from disk in multiples of D_{unit} bytes but consume data for playback in any amount, $[0, D_{unit})$ bytes will remain in the network buffer after each service round.

For example, if the network buffer is initially at a level of 0.2 D_{unit} and the user needs to consume

1.3 D_{unit} bytes of data in the next round, then 2 units will be read from the disk. At the end of the service round, these 2 units are transferred to the network buffer, bringing the network buffer level to 2.2 D_{unit} bytes. The user then consumes 1.3 D_{unit} bytes in playback, leaving 0.9 D_{unit} bytes in the network buffer for the next round. Thus the leftover data is at most 1 D_{unit} = 92 Kbytes per user. Assuming the maximum of 49 users, the network buffer must account for 49 units × 92 Kbytes/unit = 4,508 Kbytes. Thus we set the network buffer size to 4,324 + 4,508 Kbytes = 8,832 Kbytes. The total system buffer is then the sum of the disk and network buffers: 4,324 + 8,832 Kbytes = 13,156 Kbytes. The buffer per user is 13,156/49 = 268.5 Mbytes. At $15 per Mbyte, this amounts to $3.93. Thus the total cost per user is $5.73 + $3.93 = $9.67 with rounding.

One more factor reduces the efficiency of the hybrid system at short request lengths. In a hybrid system, read unit 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 unit of data during the first service round. This additional unit guarantees that the total data retrieved from disk is sufficient to begin playback. However, these additional first-round units increase the system load, lowering the average number of users served for a given probability of overload by a factor of $1/(L+1)$.[11] This has the net effect of increasing the system cost per user by the same factor. Taking this into account, the cost for our hybrid system at L = 20 rounds is $9.67 × (21/20) = $10.15.

In comparison with CDL, a hybrid system needs to do a much smaller prefetch to ensure that the network buffer does not underflow. In a CDL system, this prefetch greatly increases the latency. For the hybrid system, however, the maximum amount of prefetch is one read unit. Users read the prefetch unit with the other data units in the first round to guarantee that they have enough data in the buffer to begin immediate playback. Thus the start and jump delays match the CTL case—exactly one service round.

Cost analysis

Table 6 summarizes the characteristics and theoretical results of each data placement scheme. The first column lists the schemes. The second and third columns describe the read unit lengths and the number of read units retrieved per round. CTL is the only scheme with variable-length read units and thus subject to disk fragmentation. Hybrid is the only scheme that may retrieve more than one block per round, resulting in lower disk efficiency, as seen in the number of users that can be served by the disk (columns 4 and 6). Columns 5 and 7 show the buffer required per user, and it is clear that CDL requires far too much buffer for the full-length sequence.

Table 6. Data placement schemes summarized.

	Read Unit		L = 30 seconds		L = 2 hours	
	Lengths	Number	U	Buffer	U	Buffer
CTL	Variable	1	58	211 Kbytes	58	211 Kbytes
CDL	Constant	1	60	372 Kbytes	62	15 Mbytes
Hybrid	Constant	0-3	46.7	281 Kbytes	49	268 Kbytes

Table 7. Theoretical cost per stream in dollars of the data placement schemes.

	Padded	CTL	CDL		Hybrid	
L	Any	Any	30 s	2 h	30 s	2 h
Disk $	15.40	4.84	4.67	4.54	6.02	5.73
Buffer $	8.17	3.09	5.44	222.07	4.13	3.93
Total $	23.57	7.93	10.11	226.61	10.15	9.67

To get a cost analysis for our three VBR data placement strategies, we used the prices from Table 2 to compute the disk and buffer costs, and compared them against the costs of padding the VBR sequence. To pad the VBR sequence, begin with CTL data trace discretized into T_{SR} = 2-second intervals, find the maximum of the CTL trace, and zero-pad each data unit to that size.

Table 7 summarizes the cost comparisons. You can see that CDL systems are not cost-effective for storage and retrieval of long sequences. The buffer requirements are too high and the delays too long for interactive use. The only advantage of CDL over CTL is that CDL has no fragmentation problems associated with rewriting videos on disk. Hybrid schemes eliminate the fragmentation problems while retaining a relatively low cost per stream. Nonetheless, because the hybrid system costs 22 to 28 percent more than the CTL system, we can recommend the CTL system as the least expensive in read-only situations. However, the cost increase of using a hybrid system might be acceptable when videos need to be edited or overwritten on the disk.

Interactivity

When choosing a data placement strategy for video servers, we must also consider interactivity.

Figure 9. Data-limit deterministic admission control.

CTL data placement works best for interactive playback, having no penalties for playing units in nonconsecutive order. As long as each user requests only one CTL unit 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. We have examined a "burst mode" CDL system[11] that 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 unit boundaries. Thus each user requesting data after a jump will need to retrieve an extra unit to prefetch data for the network buffer. This results in a small loss in the average number of users served.

Admission control and disk models

In the previous section, we applied a simple statistical admission control that uses histograms 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 can be stored in an index table, and the server can use these traces to examine each incoming request on an individual basis. We call this *deterministic* admission control.

Here we explore two types of deterministic admission control. The first, data-limit, uses only the bit trace information and the typical seek time model. The second, ideal, uses the bit trace information along with the disk track and sector placement information to compute the actual seek times. We assume hybrid data placement and request length $L = 30$ seconds. We use a full disk model[12] for simulation and verify the results using a video server constructed on an HP 9000 725/100 workstation with an HP C3325W hard drive.

Data-limit deterministic admission control

Our data-limit deterministic admission control strategy accepts users based on a priori knowledge of the bit rate traces of the requested video sequences. We assume a typical seek and rotation time T_s, resulting in the read unit threshold N_{lim} computed earlier. We describe the data-limit algorithm using Figure 9 as follows:

- The server keeps track of the current load, defined as the total units requested by current users for all time.
- When a new user requests admission, the pattern of future requested read units is added to the current load to result in a possible future load.
- If the possible future load exceeds the read unit threshold, the user is denied access; otherwise the user is admitted. The system may also 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 already been partially addressed.[8]

In our implementation, a user who is not admitted will wait on the queue to try again at the next service round, as the current load will have shifted one round to the left. We repeat the above cycle for the entire duration of our test.

Experimental results. We found that under the typical seek time assumption, we can theoretically read a maximum of 47 hybrid units. However, when tested on the real disk video server with the *Star Wars* sequence, we measured a 20-percent probability of overload due to the seek and rotation time variation. If we assume a more conservative "worst case" seek and rotation time estimate, explained earlier, the threshold number of units read in one service round drops from 47 to 40. This guarantees no overload on the real disk video server but reduces the average number of users served from 57.9 to 48.9.

Thus by changing the disk seek and rotation time estimate T_s, we change N_{lim}, the number of

hybrid units that can be read in one service round, and hence the average number of users served. By using different models of T_s, we can achieve a spectrum of operating points, trading off number of users with disk overload probability. Figure 10 shows this spectrum, plotting the measured probability of overload as a function of the number of users served on the real disk video server.

We computed the probabilities by looking at 10^5 trials and measuring zero overloads for an N_{lim} of 44; we found that we can serve 53.9 users at this operating point. For comparison, we also tested statistical admission control on the video server. For all tested probabilities of overload, the data-limit admission control admits more users than the statistical. 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 simulated our video server on a full disk model based on work by Ruemmler and Wilkes.[12] We used our measured disk parameters to calculate a layout pattern by translating the video read units into sectors and tracks on the disk simulator. Finally, we traced the execution of the simulator, adding the appropriate seek, head switch, and rotation times. We did not model secondary effects such as slipped defective sectors and thermal recalibration. The simulator operates about 1,000 times faster than real time, and our tests show that it predicts the number of users within 2 percent of the measured real disk value, as seen in Figure 10.

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 1-ms shift in the disk seek time profile results in a change of one user served. Since our accuracy in measuring read and seek times is limited to a 1-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.

The simulator shows that the data-limit deterministic admission control admits 20 percent more users than the statistical one for a 10^{-5} probability of overload. We can also theoretically predict the average number of users under data-limit admission control within 2 percent of the real disk value using convolution techniques.[11]

Delay considerations. The start delay for data-limit deterministic admission control 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 a user access based on the vector of future number of units requested in each service round; a change in this vector may cause a delay while the admission control checks each upcoming service round for one in which readmission will not cause disk overload. Both our real disk experiments and full disk model simulations show that the average jump delay increases from that of statistical admission control by at most 7 percent using our system parameters in Table 1 and varying the jump probability P_j from 0.1 percent to 10 percent per user per service round. We have shown that we can reduce this delay using scalable video.[13]

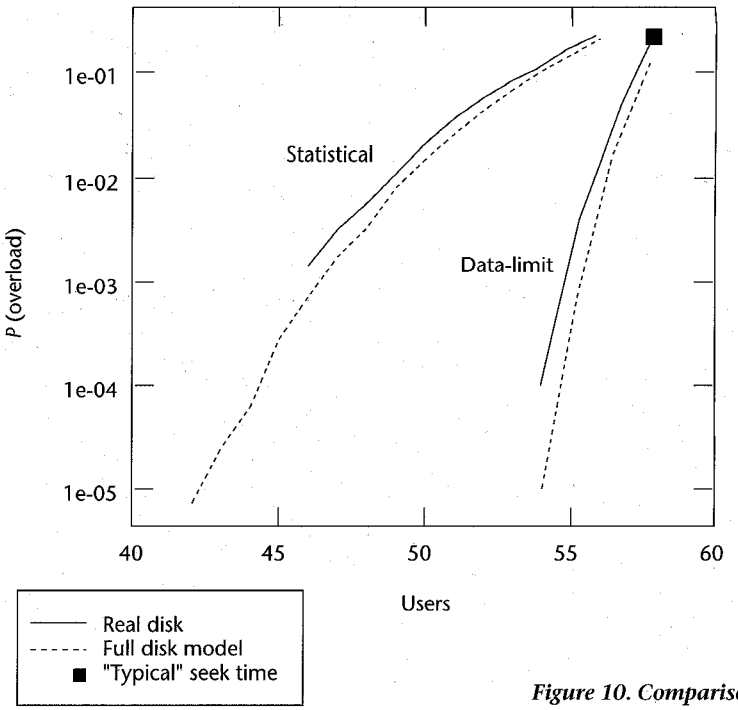


Figure 10. Comparisons between statistical and data-limit deterministic admission control.

Ideal deterministic admission control

In statistical admission control, we assume a constant read rate 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

Table 8. Summary of admission control strategies.

Admission Control	Operation	Implementation Complexity	Users Admitted P_{full} = 1 per half hour	Users Admitted Zero Overload
Statistical	Threshold number of users	Simple	46.2	N/A
Data-limit	Sum bit traces of read units	Moderate	55.3	49.0
Ideal	Sum disk rotation and seek times	Difficult	N/A	58.0

Table 9. Real disk results: cost per stream for P_{full} = 1 per half hour.

	Padded	CTL	Hybrid	Hybrid
Admission	Fixed U	Statistical	Statistical	Data-limit
Disk cost	15.96	4.88	6.16	5.14
Buffer cost	8.17	3.09	4.22	3.52
Total cost	24.13	7.95	10.38	8.66

Table 10. Real disk results: cost per stream for zero overload.

	Padded	Hybrid
Admission	Fixed U	Data-limit
Disk cost	16.62	5.75
Buffer cost	8.17	3.94
Total cost	24.79	9.69

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, we cannot implement ideal deterministic admission control 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 therefore use the full disk simulator, which suffers no such limitations. The simulator results in an average of 58.0 users served under ideal deterministic admission control, 25 percent higher than the load under conservative data-limit deterministic admission control. As for average jump delay, our simulator shows an increase of at most 10 percent above the jump delay of a statistical admission control system using the test conditions described earlier. We conclude that neither data-limit nor ideal deterministic admission control significantly hinder interactive use.

Cost analysis

Table 8 summarizes the operation and characteristics of each admission control scheme. We measured the number of users served on our full disk simulator, as we could not implement ideal deterministic admission control on the real disk video server. The "N/A" entries indicate that we cannot guarantee zero overload with statistical admission control and that ideal deterministic admission control never results in disk overload.

We now present a cost analysis for our admission control strategies. In Table 9 and Table 10 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 11 and Table 12. All of the systems may be classified as 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.

Table 9 shows the system costs of the overload-prone systems as tested on the real disk for P_{full} of 1 per half hour. 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,[11] 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 10, 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-percent cost increase from the corresponding overload-prone system. The cost difference between the two systems is still significant, a factor of 2.5.

Table 11 and Table 12 show that for all of the systems tested in Table 9 and Table 10, the full

Table 11. Full disk model results: cost per stream for P_{lat} = 1 per half hour.

	Padded	CTL	Hybrid	Hybrid
Admission	Fixed U	Statistical	Statistical	Data-limit
Disk cost	15.41	4.75	6.08	5.08
Buffer cost	8.17	3.09	4.17	3.48
Total cost	23.58	7.84	10.25	8.56

Table 12. Full disk model results: cost per stream for zero overload.

	Padded	Hybrid	Hybrid
Admission	Fixed U	Data-limit	Ideal
Disk cost	16.62	5.74	4.84
Buffer cost	8.17	3.94	3.32
Total cost	24.79	9.67	8.16

disk model simulator yields very similar results. In addition, the full disk model allows us to test ideal deterministic admission control. For about the same cost, the ideal deterministic admission control achieves zero overload, whereas data-limit admission control yields 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 data manipulation.

Multiple-disk VBR video storage

In this work, we have assumed that our video server uses a single disk in a round-robin environment. We suggest here possible ways to extend our single-disk results to multiple disks. Note that this discussion is speculative and tentative, requiring further theoretical and experimental verifications.

Many real video servers will require multiple disks because a two-hour movie coded at the MPEG-1 rate of 1.2 Mbytes/s requires 1 Gbyte of disk storage. In addition, spreading out multiple videos across multiple disks increases the potential number of users that can choose any one video. This is useful in video-on-demand applications in which select videos are requested far more often than others. If each disk can service U users, and there are N_d disks, then an ideal placement strategy would partition a popular video across the disks such that UN_d users could access the video simultaneously under a wide range of request patterns.

In the case of redundant arrays of inexpensive disks (RAID), data is typically striped across multiple disks, resulting in one user accessing many disks simultaneously. Redundancy provides robustness in the event of disk failure. Also, a scheme in which data is evenly striped across all disks results in a perfectly load-balanced disk array; upon admission, a new user can access any video in the system. One method of extending our VBR video data placement and admission control strategies in the striped case is to model the entire disk array as a single disk with an aggregate disk throughput and seek time. However, it is unclear how much performance we would lose by using such a high-level approach; the combination of redundancy and storage of VBR video remains open for investigation. In addition, we have shown that striping limits the number of users and increases the starting delay at any given user load.[13]

As an alternative to striping, consider a periodic interleaving technique,[1] shown in Figure 11. This technique places consecutive storage units on consecutive disks instead of on the same disk. Thus each user accesses only one disk in a service round to minimize the number of total disk seeks required. Service rounds on different disks are the same duration and synchronized with each other, so the disks operate in a lock-step manner.[3] As an example, in Figure 11 a user watches video 1 by reading unit 1 from disk 1 in the first round. In

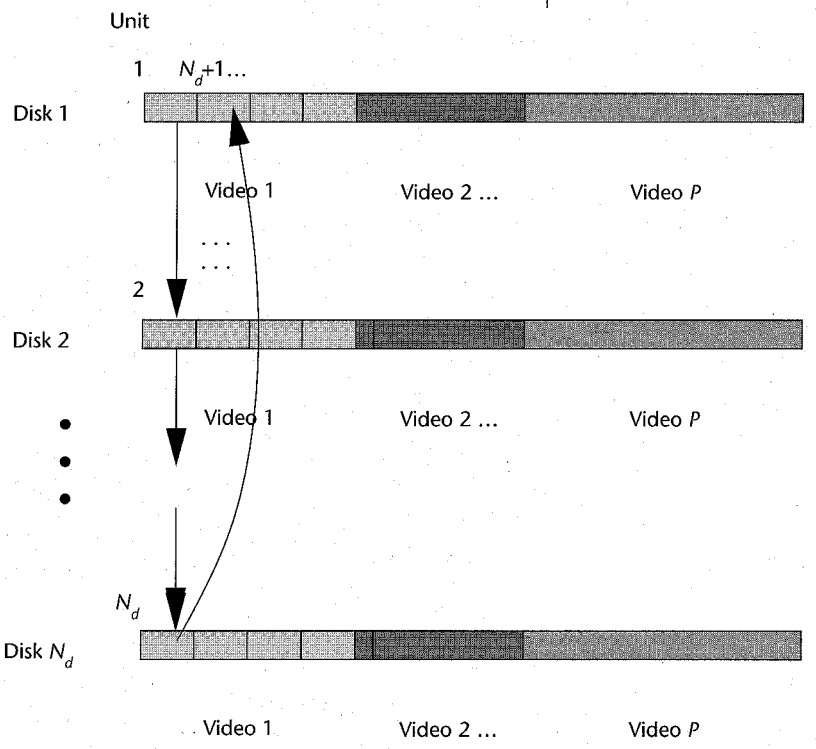


Figure 11. Periodic interleaving.

the second round, the user retrieves unit 2 from disk 2, and so on. After the user has retrieved N_d units in this manner by cycling through the entire disk array, the user reads unit $N_d + 1$ from disk 1 in round $N_d + 1$. This pattern of access is shown by the arrows in the figure.

To interleave CTL or CDL data units, we simply place one unit per disk and cycle through all N_d disks repeatedly until the end of the video. For the case of hybrid data placement, we note that each user retrieves a variable number of units per round. This lets us choose between placing one unit per disk so that each user accesses a variable number of disks per round, or placing a variable number of units per disk so that each user accesses one disk per round. Vin et al. have analyzed load balancing issues in the hybrid case.[3]

One possible admission control algorithm in an interleaved disk system follows. Consider the current users on the system divided into N_d sets, such that all of the users in one set access the same disk in the same round. For a new user requesting admission at the beginning of a round, we first determine which of the N_d sets of users it wants to join. For example, consider an interleaved disk array with CTL data placement. Assume a set of $U - 1$ current users accessing disk 1 at round 1; we denote these users as being in set 1. Suppose user U applies for admission on disk 1 at the start of round 1. The admission control algorithm considers only the statistics or bit traces of the videos accessed by the set 1 users. For statistical admission control, it convolves the PDF of user U's requested video with the aggregate PDF of the set 1 users' videos. By aggregate PDF, we mean the PDFs of the entire video traces requested by set 1 users on all disks. This is because all of the users will cycle through all of the disks during playback. For deterministic admission control, the algorithm sums the data trace of user U with the traces of the current $U - 1$ users.

Although interleaving does not affect the maximum bit rate throughput or number of users that can be serviced, interleaving videos increases the flexibility of user requests. Assuming again that each disk can service U users, UN_d users can access a single popular video with the condition that at most U users can be in phase, modulo N_d. For example, if $N_d = 8$, then at most U users can access rounds 0, 8, 16, ... $M \times 8$. Another set of users can access rounds 1, 9, 17, ... $(M \times 8) + 1$, and so forth. This offers a limited set of interactive functions such as pause and skipping multiples of N_d segments,[1] but more work remains to improve these functions. Finally, we have not addressed redundancy in the context of interleaving, and this must be resolved for a commercially viable system.

Conclusions

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, we must choose between lower cost and ease of editing videos.

We 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 found 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.

We have implemented a full disk model simulator with results 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 9. The size of that memory equals 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 Kbytes per disk, assuming 4-byte integers, and we only need 3,600 integer additions for each incoming user in a two-second service round. Our full disk simulator with deterministic admission control runs 1,000 times faster than the real disk video server, showing that computation time is not a bottleneck.

We briefly discussed methods of extending our single-disk results to multiple disks and have considered how to apply our data placement and admission control strategies to an interleaved disk array. However, work remains to be done on multiple-disk VBR video storage, particularly in the areas of striping and redundancy. **MM**

Acknowledgments

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.

References

1. E. Chang and A. Zakhor, "Scalable Video Data Placement on Parallel Disk Arrays," *Storage and Retrieval for Image and Video Databases II* (Proc. SPIE 2185), SPIE, Bellingham, Wash., 1994, pp. 208-221.
2. F. Tobagi et al., "Streaming RAID—A Disk Array Management System for Video Files," *First ACM Int'l Conf. on Multimedia*, ACM, New York, 1993, pp. 393-400.
3. H. Vin, S. Rao, and P. Goyal, "Optimizing the Placement of Multimedia Objects on Disk Arrays," *Proc. Int'l Conf. on Multimedia Computing and Systems*, IEEE Press, Piscataway, N.J., 1995, pp. 158-165.
4. H. Vin et al., "A Statistical Admission Control Algorithm for Multimedia Servers," *Proc. ACM Multimedia 94*, ACM, New York, 1994, pp. 33-40.
5. H. Vin, A. Goyal, and P. Goyal, "Algorithms for Designing Multimedia Servers," *Computer Communications*, Mar. 1995, Vol. 18, No. 3, pp. 192-203.
6. D.J. Gemmell and J. Han, "Multimedia Network File Servers: Multichannel Delay-Sensitive Data Retrieval," *Multimedia Systems 1994*, Springer-Verlag, Heidelberg, Germany, pp. I:240-252.
7. B. L. Worthington et al., "On-Line Extraction of SCSI Disk Drive Parameters," *ACM Sigmetrics 95*, ACM, New York, 1995, pp. 146-156.
8. E. Chang and A. Zakhor, "Variable Bit Rate MPEG Video Storage on Parallel Disk Arrays," *1st Int'l Workshop on Community Networking: Integrated Multimedia Services to the Home*, IEEE Press, Piscataway, N.J., 1994, pp. 127-137.
9. E. Chang and A. Zakhor, "Admissions Control and Data Placement for VBR Video Servers," *1st IEEE Int'l Conf. on Image Processing*, IEEE Press, Piscataway, N.J., 1994, pp. I:278-282.
10. R. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Trans. on Information Theory*, Vol. 37, No. 1, Jan. 1991, pp. 114-131.
11. E. Chang, "Storage and Retrieval of Compressed Video," PhD dissertation, Univ. of California, Berkeley, 1996, http://www-video.eecs.berkeley.edu/~changed/thesis.ps.
12. C. Ruemmler and J. Wilkes, "An Introduction to Disk Drive Modeling," *Computer*, Vol. 27, No. 3, Mar. 1994, pp. 17-28.
13. E. Chang and A. Zakhor, "Disk-Based Storage for Scalable Video," submitted to *IEEE Trans. on Circuits and Systems for Video Technology*, special issue on multimedia, 1996.
14. D. Taubman and A. Zakhor, "Multirate 3-D Subband Coding of Video," *IEEE Trans. on Image Processing*, Vol. 3, No. 5, Sept. 1994, pp. 572-588.

Ed Chang received a BS degree in electrical engineering from Purdue University in 1988 and MS and PhD degrees in electrical engineering from the University of California, Berkeley, in 1990 and 1996, respectively. He was a National Science Foundation Fellow from 1989 to 1992. He currently works at Digital Equipment Corporation's Cambridge Research Lab. His research interests include video storage, video and image processing, and digital photography.

Avideh Zakhor received a BS degree from California Institute of Technology in 1983 and MS and PhD degrees from Massachusetts Institute of Technology in 1985 and 1987, all in electrical engineering. She is currently an associate professor of electrical engineering and computer science at the University of California, Berkeley. Her research interests include signal processing and its applications to images and video. She holds four US patents and received the NSF Presidential Young Investigator Award in 1990.

Contact Chang at Digital Equipment Corp., Cambridge Research Lab, One Kendall Square, Bldg. 700, Cambridge, MA 02139, e-mail changed@crl.dec.com.