Disk-Based Storage for Scalable Video

Ed Chang and Avideh Zakhor, *Member, IEEE*

***Abstract*—In this paper, we consider the placement of scalable video data on single and multiple disks for storage and real-time retrieval. For the single-disk case, we extend the principle of constant frame grouping from constant bit rate (CBR) to variable bit rate (VBR) scalable video data. When the number of admitted users exceeds the server capacity, the rate of data sent to each user is reduced to relieve the disk system overload, offering a graceful degradation in comparison with nonscalable data. We examine the qualities of video reconstructions obtained from a real disk video server and find the scalable video more visually appealing.**

In the VBR case, scalability is also used to improve interactivity by reducing the delay associated with using interactive functions in a predictive admission control environment. Finally, we consider the multiple disk scenario and prove that periodic interleaving results in lower system delay than striping in a video server using round-robin scheduling. We verify the results through detailed simulation of a four-disk array.

***Index Terms*—Data storage, hard disk, scalable coding, server, video.**

I. Introduction

In this paper, we consider storage and retrieval of scalable video data. By scalable, we mean a video sequence coded such that subsets of the full-resolution video bit stream can be decoded to recreate lower-quality or lower-resolution videos. Many applications can take advantage of a scalable compression scheme. For example, in a video-on-demand system, a digital cable television company may wish to provide different customers with different levels of service: a customer with a high-definition television (HDTV) set will want much higher-quality and higher-resolution video than a customer with a small conventional television. The workstation scenario can also take advantage of scalable video. Consider a video server connected to a network of workstations—as users open windows of different sizes to watch video segments, the server must provide videos with different rates and resolutions to accommodate the different users.

In the above applications, the storage of scalable video provides the following benefits. First, scalable compression of video sequences is more storage efficient in that it eliminates the need for storing multiple copies of the video at different rates and resolutions in the server. Second, when too many users request video from a server, the storage of scalable video permits the server to gracefully degrade, i.e., to reduce the bit rates of the videos sent to each user in order to service all users. Without scalable video, the server would not be able to service all of the requests simultaneously.

Keeton and Katz [12] consider the problem of scalable video data layout on parallel disk arrays in a standard file server environment. They examine issues of striping data across multiple disks and evaluate their placement strategies in simulation by measuring average request service times. Chiueh and Katz [9] consider the specific case of storing scalable video coded in a Laplacian or Gaussian pyramid. Their simulations show that the use of scalable video greatly increases the I/O rate and decreases the waiting time as compared to full-rate nonscalable video. Chen *et al.* [6] propose a method of staggering scalable data blocks in order to achieve better load balancing and reduce buffer requirements in a conventional file system.

The use of scalable video in interactive servers has also been studied. We have previously presented a segment skipping scheme to implement pause, reverse scan, and forward scan [1]. Chen *et al.* have modified the segment skipping idea by load balancing disks through offset placement and retrieval methods [5]. In addition, they performed visual tests and showed segment skipping as a viable means of browsing video at different speeds. Paek *et al.* [13] focus on reducing the interactivity delay of segment skipping by trading off disk utilization. An alternative to segment skipping has been proposed by Dey *et al.* [10]; in their system, users who browse video at speed N consume N times the normal playback bandwidth. They use statistical multiplexing to provide statistical quality of service guarantees. In addition, they use scalability to trade off delay in providing service.

Work has also been done on scalability and interactivity for MPEG video. In [2], we presented one method of MPEG frame rearranging to create a scalable bitstream. Chen *et al.* [7], [8] use our frame rearrangement method for variable-speed browsing. They account for the increased bandwidth of browsing users by gracefully degrading the service of other users. Paek *et al.* [13] also propose using our MPEG frame rearrangement method in conjunction with frequency scaling to reduce interactivity delay. Finally, Shenoy and Vin [14] present a video server that allows interactive browsing of MPEG video using a combination of temporal and frequency scaling to reduce disk overhead. In addition, they amortize the remaining disk overhead over an array of disks to minimize the effect on other users.

In previous work [1], we have presented a general strategy for scalable data placement on disk that maximizes the total data transfer rate for an arbitrary distribution of requested data

Manuscript received September 1, 1996; revised January 31, 1997. This paper was recommended by Guest Editors B. Sheu, C.-Y Wu, H.-D. Lin, and M. Ghanbari.

E. Chang was with the University of California, Berkeley, CA 94720 USA. He is currently with the Digital Equipment Corporation, Cambridge Research Lab, Cambridge, MA 02139 USA.

A. Zakhor is with the University of California, Berkeley, CA 94720 USA.

Publisher Item Identifier S 1051-8215(97)05883-7.

rates. That strategy consisted of two main concepts: *constant frame grouping* to order data rate layers within one storage unit on disk, and *periodic interleaving* to arrange the storage units on multiple disks. In this paper, we build on those concepts as follows. We first present our disk model and scalable compression scheme in Section II. In Section III, we extend the principle of constant frame grouping to incorporate variable bit rate (VBR) video data placement on a single disk. In experimental tests, we exploit the scalability to demonstrate storage efficiency and graceful degradation. We consider interactive VBR video in Section IV to show how scalability can be used to reduce delays, and we consider the use of scalable video for true VCR functions in Section V. In Section VI, we prove that the use of periodic interleaving always results in a lower delay than striping across multiple disks for a given number of users. We present our conclusions in Section VII.

II. System Parameters

We begin by defining some basic assumptions about our video server system. 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. In each service round, the disk must therefore perform at least one seek and one read for each user, as we assume users do not batch requests with other users. The server itself is composed of one or more disks and a dual buffer system [3], [16]. In each service round, data is read from disk and sent to one buffer, while previously-read data is sent from the other buffer to each user at the corresponding playback consumption rate. At the end of each service round, the buffers switch roles. Thus, the start delay in the single-disk system is one service round, the amount of time required to fill one buffer. 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.

In Table I we list our system parameters. We model our disk as an ideal device with a constant read rate R_d and seek time T_s. Our scalable video server uses an HP C3325W hard drive, and we measure the raw read rate R_d to be 42 200 Kb/s. To arrive at a seek time estimate, we use conservative bounds for both the disk seek and rotation times to ensure that there is no possibility of disk overload, i.e., a service round in which the time necessary to service all of the users exceeds the round duration. To find the worst case seek time, we assume r requests are evenly spaced across the n_{total} disk tracks such that there is an equal number of tracks between each requested read unit. The reason for doing so is that this has been shown to maximize the total seek time under the SCAN algorithm [16] and as such results in a conservative upper bound for seek time. We show later in this section that we can serve at most $r = 19$ full-rate requests for our chosen service duration; this results in a worst case disk track seek time of about 10 ms [3]. By adding this pessimistic seek time to the worst-case rotation time of 11 ms for a full disk revolution, we obtain the worst case total seek time estimate $T_s = 21$ ms.

TABLE I
System Parameters

Read rate	R_d
Seek time	T_s
Read time for one user	T_r
Service round duration	T_{SR}
Total number of users	U
Data rate requested by user u	R_u
Disk efficiency	E_d
Maximum number of users served	U_{max}
Number of frames grouped in one storage unit	N_g

The video sequence we use is a set of scenes from *Raiders of the Lost Ark*. We concatenate five different scenes from the movie to form a single 2496-frame sequence at 24 source input format (SIF)-sized frames per second. To compress the video, we use a highly-scalable three-dimensional (3-D) subband video compression scheme [15] which results in a set of N_R bit rates shown in Table II; these are the constant rates for constant bit rate (CBR) video compression and the average rates of the corresponding VBR compression. For example, when a user requests video at the rate of 316 Kb/s, the server will send layer 1 at 190 Kb/s, layer 2 at $(253 - 190 = 63)$ Kb/s, and layer 3 at $(316 - 253 = 63)$ Kb/s. At the highest rate, 1330 Kb/s, the video is approximately equivalent to MPEG-1 in both bit rate and reconstructed visual quality.

The choice of service round length is a critical issue, as previous work has shown that it greatly affects the maximum number of users serviced by a disk as well as the amount of buffer memory each user requires [1], [3], [16]. In this paper, we choose our service round duration T_{SR} to minimize the sum of the disk and buffer cost per user [3]. The number of users served and hence T_{SR} depends on the distribution of users' requested video bit rates. In the remainder of this section, we find the "optimal" value of T_{SR} for the case when all users request full rate video; this optimal value is used throughout most of this paper for users requesting full rate video.

We now describe a way to compute the cost associated with a chosen value of T_{SR}. Our video sequence is coded at 24 frames per second (f/s), and we have chosen to use two levels of temporal filtering in the subband codec, resulting in each group-of-pictures (GOP) containing four frames. If we choose not to split any GOP's across disk storage units, our minimum resolution of storage is then one GOP, with a real time duration of one-sixth of a second. To maximize disk utilization, we allow only one seek and one read per user in each service round. Thus, we consider service round durations of multiples of one-sixth of a second.

Given our assumption of one read unit per service round, we find the read time T_r by dividing the amount of full-rate video by the disk read rate: $T_r = R_u T_{\text{SR}}/R_d$. We define disk efficiency to be the ratio of read time to the sum of read and seek times: $E_d \equiv T_r/(T_r + T_s)$.

With a 100% efficient disk, we could serve a number of users equal to the ratio of disk read rate to video bit rate. Otherwise, we multiply this ratio by the disk efficiency to

TABLE II
SCALABLE VIDEO DATA BIT RATES

Rate Index	1	2	3	4	5	6	7	8	9	10	11
Current layer rate (kb/s)	190	63	63	64	126	127	127	127	126	127	190
Total bit rate (kb/s)	190	253	316	380	506	633	760	887	1013	1140	1330

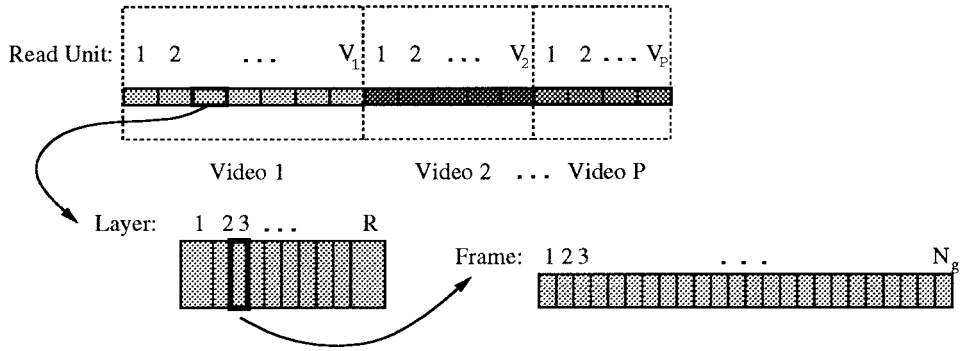


Fig. 1. Constant frame grouping.

find the maximum number of users we can serve: $U_{\max} = \lfloor E_d R_d / R_u \rfloor$.

Note that this expression is equivalent to solving the service time constraint [1], [16] for the number of users U, assuming all users request the same rate of video. This constraint prevents disk overload by limiting the total read and seek times required by all users in one service round to the round duration

$$\sum_{u=1}^{U} \left(\frac{R_u T_{\mathrm{SR}}}{R_d} + T_s \right) < T_{\mathrm{SR}}. \tag{1}$$

The disk cost is found by multiplying the current price of disk storage, \$0.25 per MB, by the amount of data contained in a two-hour movie at the full video bit rate of 1330 Kb/s, and adding the cost of a disk controller, \$200. We calculate the buffer usage by assuming a dual-buffer system. Thus, the buffer per user is twice the amount of data that is served at full rate in T_{SR} seconds. We then assume the price of memory to be \$15 per MB and calculate the buffer cost accordingly.

Table III shows cost as a function of service round duration using the above approach. As seen, the total system cost is minimized at T_{SR} of one second. Once we have chosen the service round duration, the number of frames per service round, N_g, is also fixed at 24. This value, N_g, will be shown to influence the data placement strategy in Section III. Finally, our choice of T_{SR} results in $V = 104$ rounds of video for our 2496-frame sequence.

TABLE III
COST PER FULL-RATE STREAM AS A FUNCTION OF T_{SR}

T_{SR} (seconds)	0.17	0.33	0.67	1.0	2.0	4.0	8.0
Read time (ms)	5.25	10.5	21	31.5	63	126	252
Disk Efficiency	0.20	0.33	0.50	0.60	0.75	0.86	0.92
Users	6	10	15	19	23	27	29
Disk Cost per user (\$)	82.04	49.22	32.82	25.91	21.40	18.23	16.97
Buffer per user (kB)	55.4	110.8	221.7	332.5	665	1330	2660
Buffer Cost per user (\$)	0.81	1.62	3.24	4.87	9.74	19.48	38.96
Total Cost (\$)	82.85	50.84	36.06	30.78	31.14	37.71	55.93

III. DATA PLACEMENT FOR SCALABLE VIDEO

To analyze data placement for scalable video, we first consider the basic case of CBR video. Our primary goal is to maximize the bit rate throughput and number of users serviced simultaneously. Thus, we must maximize the disk efficiency, or equivalently, the percentage of time the disk spends reading data. The principle of constant frame grouping [1] accomplishes this goal by grouping together N_g frames of each layer, as shown in Fig. 1. This strategy allows optimal disk operation for each user in each service round by performing one seek and a contiguous read of the exact amount of data requested.

We now consider the more complex case of VBR video. We have shown in previous work [3] that the use of VBR video 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. Thus, we extend the principle of constant frame grouping to the storage of scalable VBR video.

In general, storage of VBR video is not as straightforward as that of CBR video. For CBR video, the data can be stored and retrieved in constant-sized data blocks without risking jitter free, real-time video delivery [1], [16]. For VBR data, the block sizes to be written to and read from the disk cannot 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). Finally, we can 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. In previous work [3], we have shown that CDL requires too much buffer, but both CTL

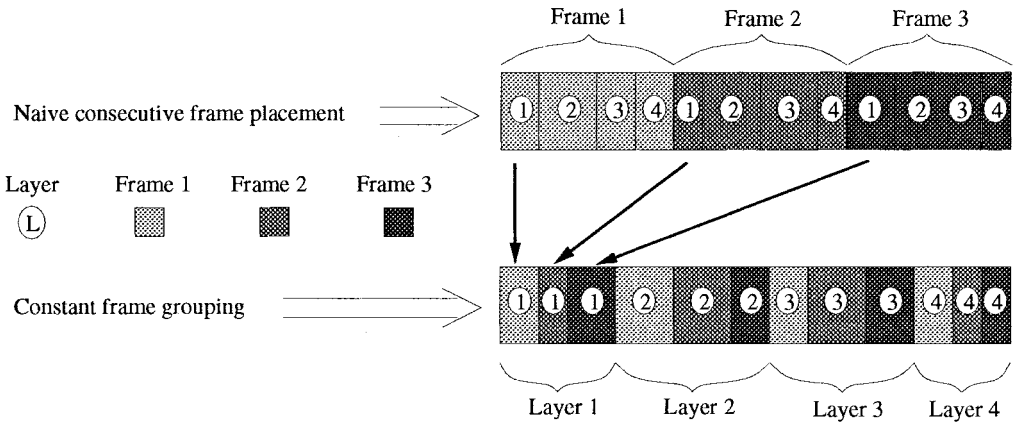


Fig. 2. Scalable CTL data placement, $N_g = 3$, $R = 4$.

and hybrid data placement are viable strategies: CTL is slightly more efficient, while hybrid results in lower fragmentation.

In Section III-A, we consider CTL storage of scalable data. We then consider hybrid data placement of scalable data in Section III-B. Finally, we present experimental results in Section III-C.

A. Storage of Scalable VBR Video Using CTL Data Placement

We now consider CTL data placement for the scalable VBR video sequence generated by the 3-D subband coder described in Section II. In Section III, we defined CTL as a data placement strategy in which stored block sizes are proportional to their corresponding playback bit rates. For CTL data placement, the extension of the constant frame grouping strategy from the CBR case is straightforward. Assuming a constant frame rate, there will be a constant number of frames, N_g, in each block stored on disk. To apply the constant frame grouping strategy, we rearrange the rate layers in each stored block such that N_g frames of each layer are stored together, just as in the CBR case. This is illustrated in Fig. 2 for a CTL data block with an N_g of three and four data layers.

As seen, the data placement is conceptually no different than that of the CBR case as shown in Fig. 1. The only difference is that each rate layer of each frame may be of different size. The different frame sizes may result in service rounds with disk overload [3]. As in the CBR case, disk overloads will result in complete dropouts for nonscalable video but graceful degradation for scalable video. In the VBR case, however, the overload rounds will not happen periodically but instead randomly with a probability distribution based on the number of users simultaneously requesting video [3].

B. Storage of Scalable VBR Video Using Hybrid Data Placement

CTL data placement has been shown to be highly efficient, but it results in high fragmentation for real-time video editing or replacement [3]. One data placement scheme that reduces the fragmentation problem is a hybrid CTL–CDL data placement scheme that stores constant-sized blocks but retrieves a variable number of blocks for each user in each service round [17]. The fragmentation is reduced because the blocks are constant-sized, and varying the number of retrieved blocks greatly reduces the amount of buffer required as compared to a straight CDL system [3].

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 than those of the CTL system; whereas each full-rate user in a CTL system can read hundreds of contiguous 1-Kb disk sectors per round, each full-rate user in the hybrid system reads only a few large noncontiguous blocks of data. The other 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.

For standard nonscalable hybrid data placement and retrieval, we begin by calculating the average bit rate of the entire video sequence to be stored. We then choose a service round duration T_{SR} and assume we read an average of one data block in each service round. Our data block size is then equal to the average bit rate multiplied by T_{SR}. We next calculate the cumulative user data consumption and the number of blocks that are required to be read in each round to keep up with the consumption. Unused data at the end of each round is buffered to be used in the next round.

One approach to storing the hybrid blocks in a scalable format would be to use constant frame grouping on each block, that is to sort the data within each block by resolution. In this case, users would retrieve the same number of blocks regardless of requested bit rate; only the amount read in each block would vary. Since we wish to reduce the total number of seeks, we sort the data within each *set* of blocks read by a user in one service round. For example, if a user retrieves two blocks of data in one round to keep up with the consumption rate, then we sort the data to start with the lowest layer data for both blocks, followed by the next higher layer for both blocks, etc. An example is shown in Fig. 3 for a set of two hybrid data blocks with four data layers.

As in the CTL case, we use an admission control based on the statistics of the video. Thus, overload rounds will

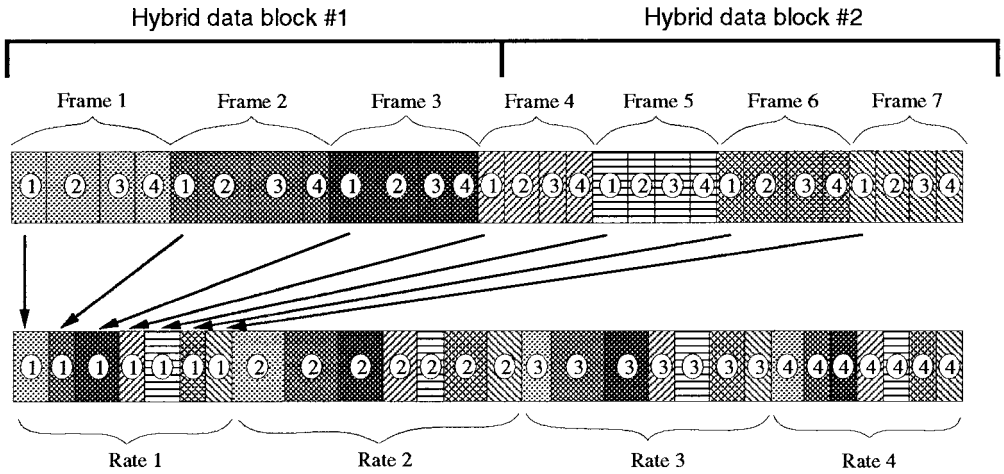


Fig. 3. Scalable hybrid data placement, $N_g = 3$, $R = 4$.

occur randomly depending on the variation in the requested bit traces. In the following section, we experimentally examine the effects of these overloads.

C. Experimental Results

We use our real video server to test the consecutive frame grouping strategy for VBR video. To demonstrate the effects of scalability, we admit a large number of users such that the total time required to service all users may exceed the service round duration. Thus, we require an intelligent disk scheduling algorithm to avoid causing a video delay or jitter to users. The SCAN algorithm assumes that the time required to service all of the users will not exceed the service round duration. By admitting too many users, we no longer provide that guarantee. Therefore, we modify the SCAN algorithm in our video server to first calculate the total read and seek time required by all of the users admitted on the disk, using the worst-case seek time estimate from Section II. If this total exceeds the service round duration, users will have their rates reduced in a uniform fashion in round-robin sequence until the time constraint in (1) is satisfied.

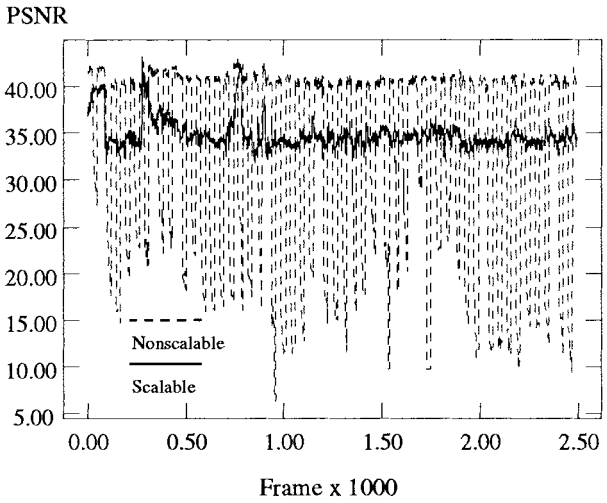
This new data scheduling algorithm handles nonscalable and scalable video differently. For the nonscalable case, users have their rates reduced to zero temporarily to relieve overload. For example, if 11 users are being served, but the server capacity is only ten users, then a user watching video will experience a one-round dropout every ten rounds. For the scalable case, users are scaled down to the next lower rate by dropping the highest layer. Intuitively, users in the scalable video system will experience more drops, but each drop will be of smaller magnitude. We now compare the qualities of the reconstructed scalable and nonscalable video streams. For the nonscalable streams, we assume the video freezes during a dropout, displaying the last transmitted frame for the duration of the service round.

To determine the exact number of users to admit, we use a statistical admission control strategy [3] as follows.

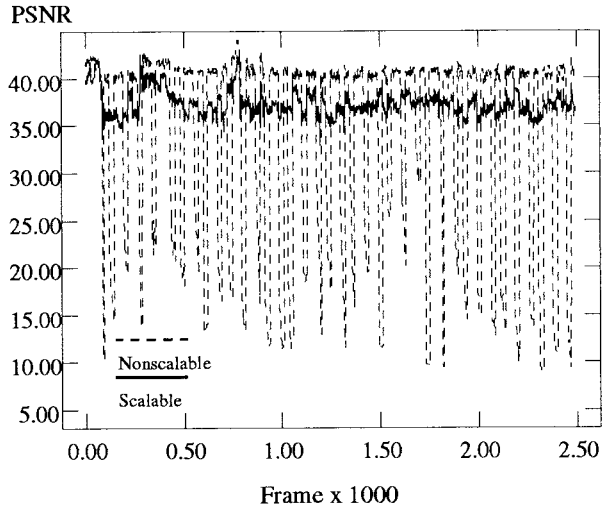
- Assume user i requires a random amount of data with probability density function (pdf) $p_i(x)$. For CTL, this is measured in kilobytes; for hybrid, it is measured in number of hybrid data blocks. The pdf's are assumed to be known, since the server can precisely compute the histograms at the time the videos are stored.
- Assume $U-1$ users on the system, and consider admitting user U.
- Compute $p_{\text{agg}}(x)$, the pdf of the aggregate resource required by U users by convolving their pdf's, $p_{\text{agg}}(x) = p_1(x) * p_2(x) * \cdots * p_U(x)$.
- Integrate the aggregate pdf beyond the disk threshold limit to find the probability of overload, $P_o(U)$. For CTL, the disk threshold limit is $R_d(T_{\text{SR}} - UT_s)$; for hybrid, it is $\lfloor T_{\text{SR}}/(T_s + T_{r,\text{block}}) \rfloor$.
- If $P_o(U)$ exceeds the chosen failure threshold, reject the user; otherwise, admit. For our tests, we choose a failure threshold of 10^{-3}.

We now test the scalable VBR data placement strategies on our real disk video server by increasing the disk threshold limit, using a value of $2T_{\text{SR}}$ in place of T_{SR}. In Fig. 4, we see the effects of scalability on both CTL and hybrid placement. Using nonscalable video, overloads occur approximately every other service round. Scalable video, however, eliminates the dropouts by reducing the peak SNR. In effect, it amortizes the overload over all service rounds to provide a more steady quality of service. The reconstructed video for the scalable case is thus more visually pleasing because the scalable video compression scheme exploits the fact that not all bits are of equal value; each user always receives the most important bits required in each service round. It is interesting to note that the average bit rate each user receives is *lower* in the scalable case because each user requires a seek in every service round. In the nonscalable case, users that are downgraded in any given service round retrieve no data and thus require no disk seeks. Thus, the scalable data results in a lower average bit rate with higher subjective quality.

We find that, as expected, the hybrid data placement is slightly less efficient than CTL due to the increased number of seeks; for the 2 T_{SR} limit, CTL serves 34 full-rate users, and hybrid serves 29. The scalable hybrid placement strategy,



(a)



(b)

Fig. 4. PSNR of VBR video reconstructions: (a) CTL and (b) hybrid.

however, results in less degradation than the scalable CTL strategy. As seen in Fig. 4, the average PSNR drop from the full-rate to the scaled reconstruction is 6 dB for CTL, as compared to 4 dB for hybrid. This occurs because in the hybrid case the number of seeks required by each user falls as the rate is scaled down, improving the disk efficiency and average retrieved video bit rate. For example, in Fig. 3, users who are served rate 1 video will only do one seek to read one block, while users served rates 2, 3, or 4 will do two seeks to read both blocks. In the CTL case, the number of seeks remains constant, at one seek per read. Thus, the hybrid scheme may become more efficient as users are served videos of lower bit rates. To accurately compare the two schemes, we admit the same number of users and measure the total bit rate throughputs. By admitting 18 users to both systems, CTL has a throughput of 23 200 Kb/s, and hybrid has a throughput of 21 700 Kb/s. By admitting 34 users to both systems, CTL has a throughput of 12 000 Kb/s, and hybrid has a throughput of 16 900 Kb/s. The smaller drop in disk efficiency for the hybrid scheme occurs because we apply constant frame grouping across sets of blocks as shown in Fig. 3; if we were to sort each block separately, the total throughput would drop more as in the CTL case. Thus, hybrid data placement shows a throughput advantage in high overload conditions.

IV. Interactivity in Scalable VBR Video

In Section III, we demonstrated the use of scalability in relieving overloads in VBR video. The overloads occurred because we used a statistical admission control algorithm that did not guarantee service. One method of guaranteeing service assuming straightforward playback is to use a predictive data-limit deterministic admission control [3], in which the actual stored video bit traces are used to regulate the number of admitted users. This guarantee fails, however, if we allow the use of interactive functions because the server no longer has an accurate prediction of the future amount of data requested. In this section, we will show a way in which scalable video can be used to reduce the readmission delay of interactive users in the context of predictive admission control.

A. Data-Limit Deterministic Admission Control

Let us begin with a brief description of predictive data-limit admission control as shown in Fig. 5. The basic idea is to precompute the total number of blocks consumed by all users in all future service rounds. Let $N[v(u), r, i]$ be the number of blocks consumed by user u requesting video $v(u)$ at rate r and time index i. Let $D[v(u), r, i]$ be the total amount of data requested from the $N[v(u), r, i]$ blocks. Finally, 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 time required to serve user u in the current service round t is $N[v(u), r, t-t_0(u)]T_s + D[v(u), r, t-t_0(u)]/R_d$. Because the disk system has full knowledge of the current videos requested by all of the users, the total number of blocks and amount of data requested for all future service rounds can be calculated *a priori*. To prevent disk overload at all times, we extend the condition from (1)

$$\sum_u \left\{ N[v(u), r, t-t_0(u)]T_s + \frac{D[v(u), r, t-t_0(u)]}{R_d} \right\} \leq T_{\mathrm{SR}} \qquad \forall t. \tag{2}$$

Using (2), 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. This is shown pictorially in Fig. 5 where the bit trace for the current users is added to that of the new user in order to determine whether disk overload occurs. By using a worst case seek time estimate, we guarantee that a user load which satisfies (2) will not overload the actual disk [3]. However, this guarantee relies on an accurate prediction of the future amount of data requested. If we allow interactive functions, users will invalidate the prediction and possibly cause disk overloads.

We consider the following three types of interactive functions: pause, reverse scan, and forward scan. We assume that these functions occur at the boundaries of service rounds, given round-robin scheduling. During pause, a user does not receive

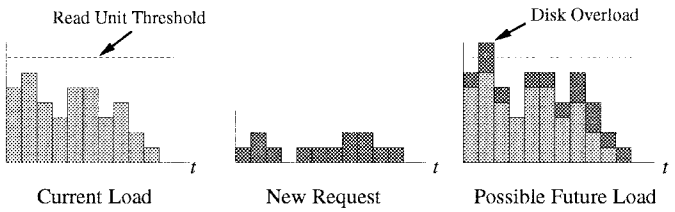


Fig. 5. Data-limit deterministic admission control.

any data from the server and therefore cannot threaten to overload the disk. Upon readmission, however, the temporally shifted request may violate the conditions set by the data-limit admission control and result in disk overload. The reverse and forward scans are implemented by skipping a constant number of segments between each read; for example, a user scanning forward at five times normal playback speed might retrieve the blocks corresponding to time index 2, 7, 12, etc. Since they retrieve data in each round, these scans may cause disk overload during their operation in addition to the those caused after resuming playback.

One strategy of preventing overload caused by interactive functions is to combine bandwidth reservation with delaying the interactive users. For all users in pause or scan, we reserve bandwidth equal to the average bit rate of the requested video trace to approximate the future load. In this way, we will not admit too many new users when some current users choose to pause or scan. In addition, the users who finish their interactive functions must wait for the current disk load to accommodate them before they are readmitted for normal playback; specifically, their readmission must not violate (2). Users in forward or reverse scan are subjected to the same readmission delay as users in pause. However, the scan users experience an additional delay for each round in which their request for data would cause disk overload, since the server delays the request to avoid overload. This does not occur during pause because users do not retrieve data from the server during the length of the pause.

An alternative strategy is to use scalability to reduce these delays by scaling the requests of all users down to a prespecified limit. We specify the rate scaling limit by the two following quality of service (QoS) measures. First, we limit the maximum number of rates that a user may drop from the requested rate in one service round. For instance, with a maximum drop of one rate, a user that requests VBR video at rate index 11 might receive a minimum of video at rate index 10 as given in Table II. Second, we specify a limit on the maximum number of reductions that a user may experience the entire video playback, where a reduction is defined by one service round in which a user is dropped by one rate. If a user drops two rates in one round, it is counted as two reductions. By varying these QoS parameters, we change the delays associated with the interactive functions. Thus, we can achieve a spectrum of tradeoffs between delay and rate scaling.

B. Experimental Results

To measure interactive delays as a function of rate scaling, we require a large number of trials for an accurate statistical sample. In previous work [3], we have shown that we can accurately characterize our real disk video server through simulation. We therefore use simulation in this section to conduct each test over 10^6 service rounds, equivalent to 280 h of real-time operation. We assume users request full-length VBR videos of the randomly phase-shifted *Raiders* sequence at rate index 11, as described in Section II. We choose the reverse and forward scan speeds to be $+5$ and -5, respectively.

Our first experiment is to characterize pause and scan delays for the case of nonscalable data. Without loss of generality, we can fix the request length to be the full length of 104 rounds and arbitrarily choose the pause and scan durations to be ten rounds each. Then the interaction delay will be a function of the number of interactions requested by each user. We define the request frequency to be the fraction of users in pause or scan averaged over time. For example, if all users will request exactly one interaction at some point in playback, the request frequency will be $10/(104+10)$. We choose to test a request frequency of 0.05 for each of the three interactive functions and measure the 99% delay quantile; that is, 1% of the delays exceed the quantile threshold. We measure a threshold of 13, 15, and 16 rounds for pause, reverse, and forward scan, respectively. As expected, the delay for the pause function is lower than for the scan functions because pause does not require bandwidth from the server during its execution. However, the difference is small, indicating that the majority of the delay results from readmission, not from retrieving blocks in scan mode.

We next test the minimum amount of scalability by applying the following two QoS measures. We assume that users will agree to be scaled down by at most one rate, to rate index 10. We then vary the limit on the maximum number of scaledowns acceptable by each user and repeat the previous experiments. For the pause case, we find that allowing just one scaledown per user reduces the delay quantile from 13 to 7. Further increases in the acceptable number of scaledowns reduces the delay further: a delay of four at two scaledowns, and three at four scaledowns. For reverse and forward scans, the results are similar, but each scaledown has less effect, as shown in Table IV.

TABLE IV
INTERACTIVE FUNCTION 99% DELAY QUANTILES, MAX 1 SCALEDOWN/USER

Scaledowns	0	1	2	3	4	5	6	7	8
Reverse scan delay quantile	15	14	11	8	5	4	4	4	4
Forward scan delay quantile	16	14	10	7	5	4	4	4	3

To further increase the delay reduction, we may allow more flexibility in scaling down current user requests. We test this by increasing the scaledown threshold from one to two rates. Specifically, users permit the quality of their video to drop two rates, from rate index 11 to rate index 9. To compare these results against those of the previous experiment using one scaledown, we fix the number of acceptable scaledowns for the pause case at four and for the scan cases at eight. We find that by increasing the scaledown threshold to two rates per user, the delay quantile drops to zero for all three interactive functions.

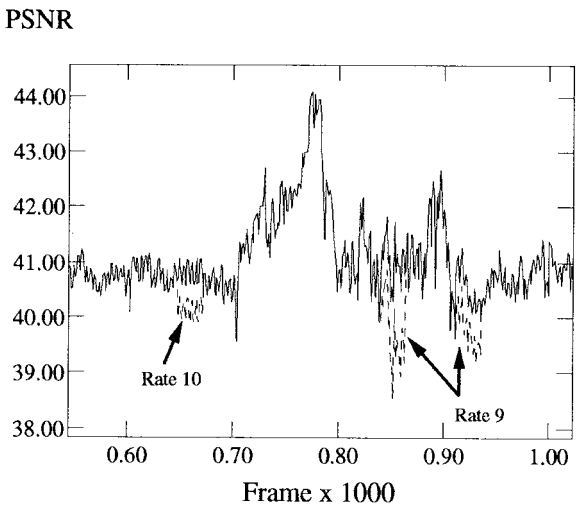


Fig. 6. Effect of scaledowns on PSNR.

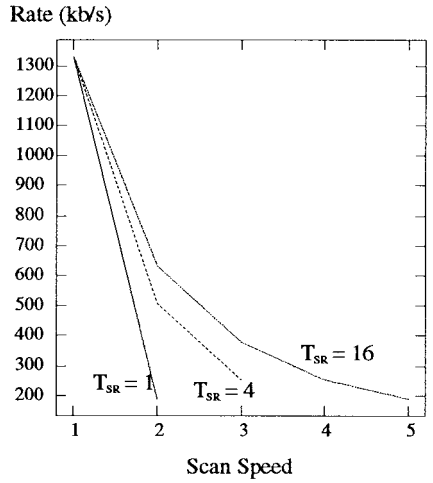


Fig. 7. Maximum bit rates for VCR scanning.

To show the effects of these scaledowns on the perceived reconstructed video, we plot a sample PSNR curve for a dropdown threshold of two, with five dropdowns. As seen in Fig. 6, the dropdowns have little effect on the PSNR. The one-rate dropdown around frame 650 is 0.5 dB down from the requested video, and the two-rate dropdowns around frames 840 and 910 are 2 and 1 dB down, respectively. These drops are difficult to detect and are much less objectionable than delays during the execution of interactive functions.

V. True VCR Functionality

In this paper, we have considered segment skipping as the only means of interactive browse functions. Although forward and reverse segment skipping may be used to browse video, it may be desirable in some applications to implement true VCR-style scanning. This requires the server to retrieve data from all read units over the scanning duration for smooth motion. Without scalability, this results in the scanning user's bandwidth requirement increasing by a factor of the scan speed [10]; however, scalability can be used to relieve the server load in two ways. First, the server may elect to gracefully degrade the video to all of the users as done in Section IV and [7] and [14]. Alternatively, the server may limit the scanning user to the normal playback bandwidth by degrading only the video requested by the scanning user; this prevents scanning users from affecting the quality of service provided to other users. In this scheme, the choice of service round duration T_{SR} determines the available scan speeds. Using the analysis in Section II, we plot the maximum serviced bit rate as a function of the scan speed in Fig. 7, assuming an N_x speed scan requires N_x read units per round. As seen, a large increase in T_{SR} is required to obtain reasonable scan speeds; this suggests that graceful degradation or a bandwidth reservation policy should also be used to relieve the server load for scanning users.

VI. Multiple Disk Video Storage

So far in this work, we have assumed that our video server uses a single disk in a round-robin environment. In this section, however, we consider the issues of interleaving and striping multiple videos on a multiple disks. Many real video servers will require multiple disks, as a 2-h movie coded at the MPEG-1 rate of 1.2 Mb/s requires 1 Gb of disk storage. In addition, by spreading out multiple videos across multiple disks, the potential number of users that can choose any one video is increased. 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. The main benefit of redundancy is robustness in the event of disk failure. In addition, some RAID levels allow the disk array to provide higher aggregate throughput to individual requests. For example, data is interleaved across disks at the bit level in RAID-3 and at the sector level in RAID-4 and RAID-5. Thus, one unit of video data that spans multiple sectors will span multiple disks. As a result of the higher throughput due to simultaneous multiple disk access, individual requests require less time to access [11]; this has in part motivated the use of striped disk arrays for video storage [9], [12]. For video retrieval, the data read from different disks correspond to the same time index of video. In the case of scalable video, the data may correspond to different layers of one read unit [9], [12]. A video server in which data is evenly striped across all disks results in a perfectly load balanced disk array; upon admission, a new user can begin access to any video in the system.

As an alternative to striping, we consider a periodic interleaving technique [1], as shown in Fig. 8. In this technique, consecutive storage units are placed on consecutive disks instead of being placed 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

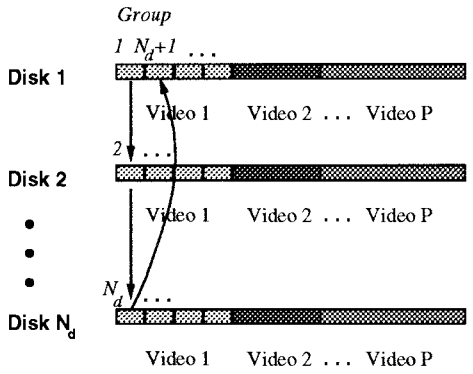


Fig. 8. Periodic interleaving.

the disks operate in a lock-step manner [17]. As an example, in Fig. 8, a user watches video 1 by reading unit 1 from disk 1 in the first round. In 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. Switching across disks does not require additional disk seeks and therefore does not lower the system bit rate throughput.

Although interleaving does not affect the maximum bit rate throughput or number of users that can be serviced, the main benefit to interleaving videos is an increase in 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 $0, 8, 16, \cdots, M \times 8$. Another set of users can access rounds $1, 9, 17, \cdots, (M \times 8) + 1$, etc.

In addition, users can effectively pause or browse videos at different speeds by moving across the disks at a rate other than one per round [1]. In the case that the video server is serving the maximum capacity of users, these functions are limited: pauses must be of duration MN_d rounds, and browsing must be done at a speed of MN_d. This is necessary to keep the interactive users in phase with the other users occupying the full server. However, if the server is not full, there will be more flexibility in the duration and speeds of the interactive functions. This is equivalent to the user leaving the system and requesting readmission starting from a different disk; we show in the following section that this admission delay drops rapidly as the number of available "slots" in the server increases.

Paek *et al.* have shown that striping and interleaving lie at the opposite ends of a spectrum of placing videos across multiple disks [13]. Specifically, they define a segmentation level S that specifies how many segments each read unit is divided into. Each segment is then placed on a separate disk. For example, in our periodic interleaving scheme, the read units are not divided at all, and so $S = 1$. We will also refer to this placement as "fully interleaved." For a striped scheme, each read unit is divided across all N_d disks in a disk array, $S = N_d$. We will also refer to this placement as "fully striped." Paek *et al.* show a tradeoff between admission delay and disk utilization efficiency by varying S [13], but we show in the following section that by fixing the user capacity and considering disk buffer delay, increasing S results in *greater* total system delay.

A. System Delay: Interleaving Versus Striping

In this section, we examine the delay of a user waiting to access a video stored on an array of N_d disks. We assume that the user makes the request at the beginning of a service round and that the disk system has capacity to handle the additional user. There are two components of a multiple-disk system delay: admission delay and disk buffer delay. Admission delay is the time required for the admission control to admit the user to the disk with the requested data. Disk buffer delay is always one round, the time required to fill one buffer in a dual buffer system.

In a perfectly load-balanced disk array, the admission delay will be zero, since we have assumed that the disk array has the capacity to handle an additional user. The admission delay only takes on positive values when there is insufficient capacity on disks containing the requested data; this can only occur in an unbalanced array as follows. For admission control, we must first determine which disks contain the first read unit of the requested video. The user then applies for admission on those disks, and each disk operates its own admission control independently of the other disks. If all of the disks contain enough capacity to handle the additional user, the user is admitted. Otherwise, the user is rejected and applies for admission in the next service round. In an array without perfect load balancing, all of the users switch disks at the end of each service round.

A user who requests data from a fully interleaved system as shown in Fig. 8 will apply for admission at only one disk. Without loss of generality, let us assume that the user requests data from disk 1 at service round 1. To consider the worst case delay scenario, assume that the system only has the capacity for one additional user, consisting of an open "slot" at disk 2 at round 1. Then the user must wait $N_d - 1$ rounds for the current users to cycle through all of the disks and for the slot to appear at disk 1. Thus, the maximum delay for the fully interleaved system is $N_d - 1$ rounds. Paek *et al.* have shown that the maximum admission delay for a system with segmentation level S is $N_d/S - 1$ rounds [13].

In comparing the fully interleaved system with a fully striped system, we see that the worst case total delay of the interleaved system is N_d rounds, while that of the striped system is only one round. However, the worst case scenarios do not show that the total delay is a function of the current user load and system capacity. We now present an analysis to show the counterintuitive result that at any user load, a striped system has *higher* mean delay than an interleaved system, provided they both have the same user capacities.

Consider an array of N_d disks. We will compare two systems: a fully interleaved system with $S = 1$, and a partially striped system with $S = 2$, as the analysis can be extended to compare systems with higher segmentation levels. Assume the disk transfer rate is R_d, and users all request CBR video at a bit rate of R_u.

We begin by computing the maximum number of users served for both disk systems with respect to the service round duration T_{SR}. For the $S = 1$ system, we use (1) to find the maximum number of users we can service on one disk

$$U_{\max,\,S=1} = \frac{T_{\mathrm{SR}}}{T_{\mathrm{read}} + T_s} = \frac{T_{\mathrm{SR}}}{\frac{R_u T_{\mathrm{SR}}}{R_d} + T_s}. \tag{3}$$

Then the total number of users we can serve on N_d disks is

$$U_{\mathrm{total,\,S=1}} = N_d U_{\max,\,S=1} = N_d \frac{T_{\mathrm{SR}}}{\frac{R_u T_{\mathrm{SR}}}{R_d} + T_s}. \tag{4}$$

For the $S = 2$ system, the data is striped such that each user accesses a pair of disks in a service round. Each of the two disks in the pair sends data at a rate of $R_u/2$ to the user for the combined requested bit rate of R_u. Thus, each pair of disks can serve the following number of users:

$$U_{\max,\,S=2} = \frac{T_{\mathrm{SR}}}{\frac{\left(\frac{R_u}{2}\right)T_{\mathrm{SR}}}{R_d} + T_s}. \tag{5}$$

Since there are $N_d/2$ pairs of disks in the array, the total number of users the array can serve is

$$\begin{aligned} U_{\mathrm{total,\,S=2}} &= \left(\frac{N_d}{2}\right) U_{\max,\,S=2} \\ &= \frac{N_d}{2} \frac{T_{\mathrm{SR}}}{\frac{\left(\frac{R_u}{2}\right)T_{\mathrm{SR}}}{R_d} + T_s} \\ &= N_d \frac{T_{\mathrm{SR}}}{\frac{R_u T_{\mathrm{SR}}}{R_d} + 2T_s}. \end{aligned} \tag{6}$$

By comparing (4) and (6), we see that in order to serve the same number of users, the $S = 2$ system must have twice the service round duration T_{SR} of the $S = 1$ system.

Let the actual number of users on the $S = 1$ system be $U \in [0,\, N_d U_{\max,\,S=1})$ and the number of available slots be $N_s \equiv N_d U_{\max,\,S=1} - U$. In the Appendix, we show that the mean admission delay for a user seeking admission to one disk assuming N_s randomly distributed slots on N_d independent disks is $(N_d - 1/N_s + 1)$ rounds. Assuming the service round duration T_{SR} is T seconds, the total delay in seconds for the $S = 1$ system is

$$D_{S=1} = T\left(1 + \frac{N_d - 1}{N_s + 1}\right). \tag{7}$$

We now consider an $S = 2$ system with service round duration $T_{\mathrm{SR}} = 2T$ such that it can serve the same number of users as the $S = 1$ system. Since the $S = 2$ system has $N_d/2$ independent pairs of disks, the admission delay is $[(N_d/2) - 1/N_s + 1]$ rounds. Thus, the total delay of the $S = 2$ system in seconds is

$$\begin{aligned} D_{S=2} &= 2T\left(1 + \frac{\frac{N_d}{2} - 1}{N_s + 1}\right) \\ &= T\left(2 + \frac{N_d - 2}{N_s + 1}\right) \\ &= T\left(1 + \frac{N_d - 1 + N_s}{N_s + 1}\right). \end{aligned} \tag{8}$$

By comparing (7) and (8), we see that the total delay for the $S = 2$ system is always higher than that of the $S = 1$ system for the same user capacity $U_{\mathrm{total},\,S}$ and number of available slots N_s. The same analysis can be extended to compare striping levels $S = 2$ and $S = 4$, $S = 4$ and $S = 8$, etc. The basic conclusion is that delay increases with striping level.

B. Four-Disk Array Example

To verify our theory with simulation, we consider a four-disk array, each disk having the parameters given in Section II. We choose four evenly spaced bit rates to interleave and stripe across the four disks: 253, 506, 759, and 1012 Kb/s. Thus, there are four rate layers, each at 253 Kb/s. We show the data placement patterns in Fig. 9. Different rate layers are shown by different textures, as indicated in the legend at the left, while each time index, corresponding to a user access, is enclosed in dashed lines.[1] For the $S = 1$ system, we use the constant frame grouping principle of Section III to construct data units consisting of all four rate layers. We do no striping and place consecutive data units on consecutive disks. Thus, the user cycles through one disk per round, completing the cycle in four rounds. For the $S = 2$ system, we stripe data across a pair of disks by storing the first two layers on one disk and the other two layers on the other disk, as shown in Fig. 9. For the next time index, we stripe another pair of units across the other pair of disks. Thus, the user cycles through two disks per round, completing the cycle in two rounds. Finally, for the $S = 4$ system, we fully stripe the data across the disks by placing one layer per disk. Thus, the user retrieves data equally from all four disks in one round.

We compare theoretical system delay curves for $S = 1$, $S = 2$, and $S = 4$ in Fig. 10. For each segmentation level S, we plot the total system delay over the range of $[U_{\max,\,S}/2,\, U_{\max,\,S})$ for these values of T_{SR}: $\frac{1}{6}$, $\frac{1}{3}$, $\frac{2}{3}$, 1, 2, 4, 8, 16. As seen, for all T_{SR}, a system with lower segmentation level S can serve more users. Since we are allowed to choose T_{SR} as a parameter of the system, in Fig. 11 we plot the minimum of the entire family of curves generated by $T_{\mathrm{SR}} = M/6$, $M = [1, 2, \cdots, 120]$ seconds. It is clearly seen that a system with lower segmentation level S has a lower delay for the same number of users over the entire range of users.

[1] In striped systems for scalable video, Chen *et al.* have shown the benefits of staggering the data units by rotating the blocks within each time index [6]. This does not affect the performance when all users read full-rate video, as we assume in the current example.

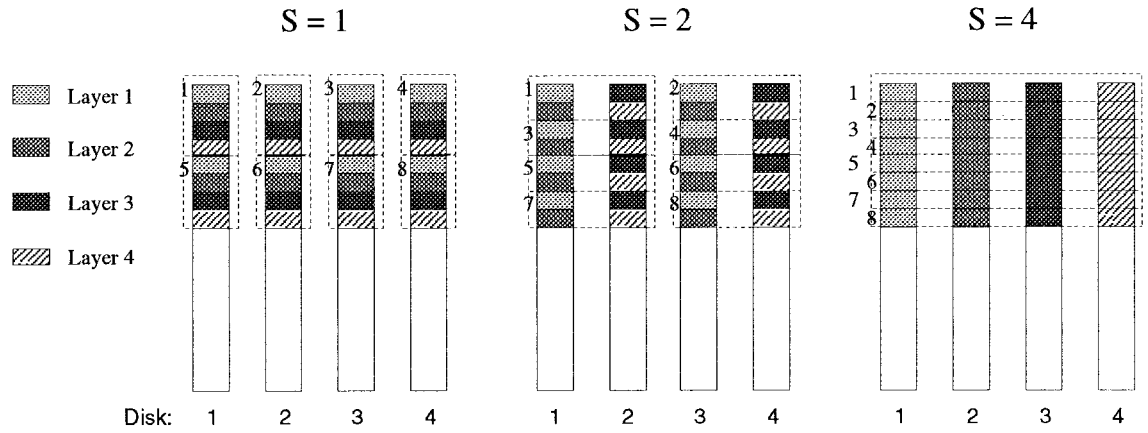


Fig. 9. Data placement for four-disk system.

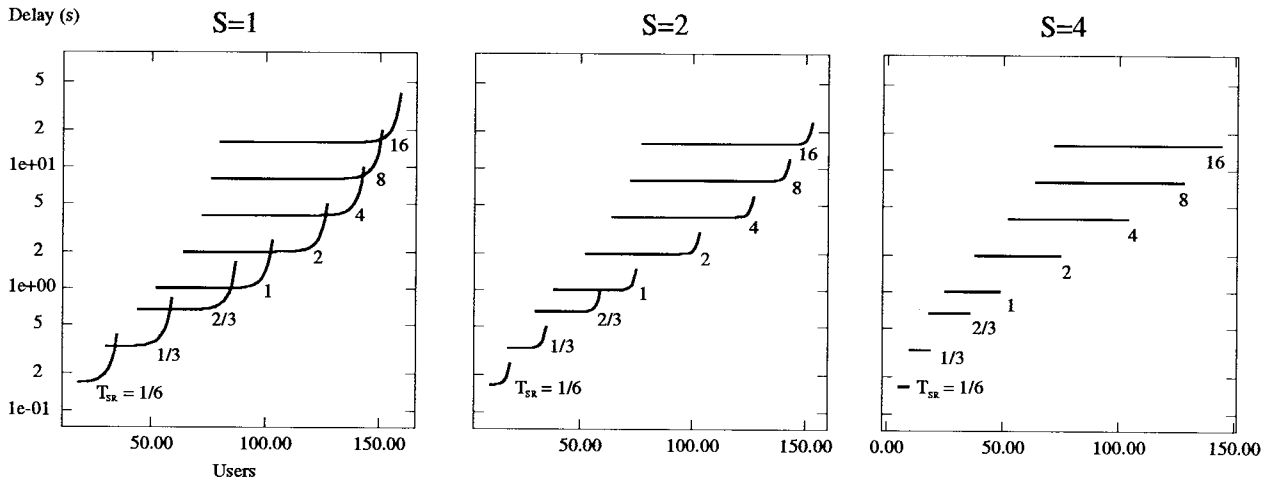


Fig. 10. Theoretical system delay.

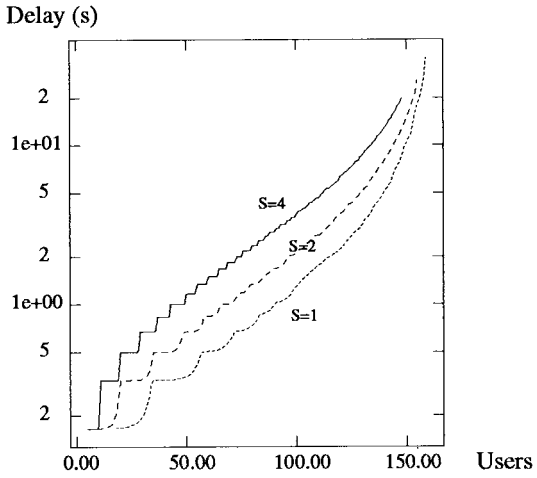


Fig. 11. Theoretical minimum system delay over all T_{SR}.

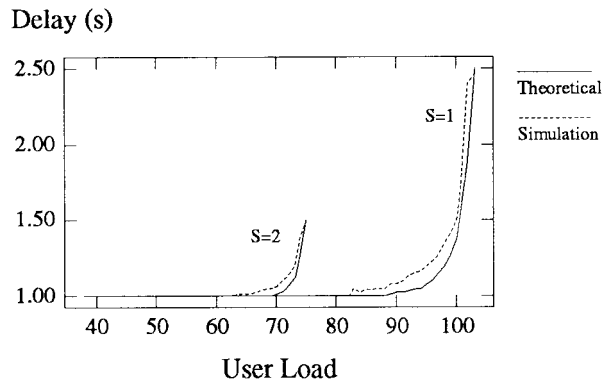


Fig. 12. $S = 1$ and $S = 2$ system delays at $T_{\mathrm{SR}} = 1$.

We use simulations to verify the delays of the $S = 1$ and $S = 2$ systems. The $S = 4$ system is known to have no admission delay and is thus not tested, since the total system delay is just the disk buffer delay of one service round. We choose a service round duration of $T_{\mathrm{SR}} = 1$ s and assume the user request lengths are randomly distributed between 1 s and 2 h. In Fig. 12, we show the theoretical and simulation total delays for the $S = 1$ and $S = 2$ systems over the range of $[U_{\max,S}/2, U_{\max,S})$. As seen, the simulations confirm our theoretical results.

VII. Conclusions

We have proposed two methods of storing VBR scalable video using the principle of constant frame grouping and

implemented our techniques on our real disk video server. Our results demonstrated the advantages of scalability in providing a more flexible range of user rates and also graceful degradation in times of disk overload. For interactive VBR video, we have shown that scalability can be used to greatly reduce the delay arising from the use of pause and segment skipping functions under predictive admission control.

In the multiple-disk case, we have shown that periodic interleaving results in lower system delay than striping in a video server using round-robin scheduling. In addition, the use of periodic interleaving allows limited interactive functions such as pause and segment skipping, but more work remains to be done to improve these functions. Finally, we have not addressed the issue of redundancy in the context of interleaving, and this should be resolved for a commercially viable system.

APPENDIX
MEAN ADMISSION DELAY

In this Appendix, we calculate the mean admission delay for a user seeking admission to one disk assuming N_s randomly distributed slots on N_d independent disks. Slots are defined as the capacity of a disk to handle additional users, i.e., if a disk is currently servicing U users but has a capacity of $U_{\max}$, then it is said to contain $U_{\max} - U$ slots. Without loss of generality, we consider a user requesting admission to disk N_{d-1} out of the set of disks $[0, N_{d-1}]$. Assuming there are $N_s \in [1, U_{\max}]$ slots randomly distributed across the N_d disks, we note that admission occurs when at least one slot exists on disk N_{d-1}. If admission does not occur immediately, then in the next round all users will cycle one disk; current users accessing disk 0 will move to disk 1, users from disk 1 will move to disk 2, etc. The users on disk $N_d - 2$ will move to disk N_{d-1}, and thus any slots that were on disk $N_d - 2$ will now result in admission for the user waiting on disk N_{d-1}.

It is clear from the problem description that admission delay is equal to the number of disks to the closest slot. Let D_s be the number of disks from slot s to disk N_{d-1}; the admission delay is then equal to the minimum of D_s over all s. Since the slots are assumed to be randomly distributed, we approximate the values D_s with continuous random variables Y_s uniformly distributed over the range $[0, N_{d-1}]$. If we define W to be the minimum of Y_s over all s, then the admission delay is approximately equal to W. To find the cumulative distribution function of W, we note that $W \geq w$ if and only if $Y_s \geq w$ for all s. Since the random variables Y_s are assumed to be independent

$$P(W \geq w) = \prod_{s=1}^{N_s} P(Y_s \geq w) = [(N_d - 1) - w]^{N_s}. \quad (9)$$

Then the probability distribution function $F_W(w)$ can be found as follows:

$$\begin{aligned} F_W(w) &= P(W \leq w) \\ &= 1 - P(W \geq w) \\ &= 1 - [(N_d - 1) - w]^{N_s}. \end{aligned} \quad (10)$$

The pdf $p_W(w)$ is equal to the derivative of $F_W(w)$ with respect to w

$$p_W(w) = N_s(N_d - 1 - w)^{N_s - 1}. \quad (11)$$

Then the mean of W can be found through integration by parts

$$E(W) = \int_0^{N_d - 1} w \cdot p_W(w)\, dw \quad (12)$$

$$= \int_0^{N_d - 1} w N_s (N_d - 1 - w)^{N_s - 1}\, dw \quad (13)$$

$$\begin{aligned} &= w(N_d - 1 - w)^{N_s}\Big|_0^{N_d - 1} \\ &\quad - \int_0^{N_d - 1} (N_d - 1 - w)^{N_s}\, dw \\ &= 0 - \frac{(N_d - 1 - w)^{N_s + 1}}{N_s + 1}\Bigg|_0^{N_d - 1} \\ &= \frac{N_d - 1}{N_s + 1}. \end{aligned} \quad (14)$$

Thus, the mean admission delay is approximately equal to $(N_d - 1/N_s + 1)$.

REFERENCES

[1] E. Chang and A. Zakhor, "Scalable video data placement on parallel disk arrays," in *IS&T/SPIE Int. Symp. Electronic Imaging: Science and Technology, Volume 2185: Image and Video Databases II*, San Jose, CA, Feb. 1994, pp. 208–221.

[2] ———, "Variable bit rate MPEG video storage on parallel disk arrays," in *First Int. Workshop on Community Networking Integrated Multimedia Services to the Home*, San Francisco, CA, July 1994, pp. 127–137.

[3] ———, "Cost analyzes for VBR video servers," *IEEE Multimedia*, vol. 4, no. 3, pp. 56–71, Winter 1996.

[4] E. Chang, "Storage and retrieval of compressed video," Ph.D. dissertation, University of California at Berkeley, 1996, http://www-video.eecs.berkeley.edu/ changed/thesis.ps.

[5] M. S. Chen, D. Kandlur, and P. Yu, "Support for fully interactive playout in a disk-array-based video server," in *Proc. ACM Multimedia'94*, San Francisco, CA, Oct. 1994, pp. 391–398.

[6] ———, "Using rate staggering to store scalable video data in a disk-array-based video server," in *Multimedia Computing and Networking 1995, SPIE*, San Jose, CA, Feb. 1995, vol. SPIE-2417, pp. 338–345.

[7] H. J. Chen, A. Krishnamurthy, D. Venkatesh, and T. D. C. Little, "A scalable video-on-demand service for the provision of VCR-like functions," in *Proc. 2nd Int. Conf. Multimedia Computing Systems*, Washington DC, May 1995, pp. 65–72.

[8] H. Chen, T. Little, and D. Venkatesh, "A storage and retrieval technique for scalable delivery of MPEG-encoded video," *J. Parallel and Distributed Comput.*, vol. 30, no. 2, pp. 180–189, Nov. 1995.

[9] T. C. Chiueh and R. Katz, "Multi-resolution video representation for parallel disk arrays," in *Proc. ACM Multimedia'93*, New York, Aug. 1993, pp. 401–409.

[10] J. K. Dey-Sircar, J. D. Salehi, J. F. Kurose, and D. Towsley, "Providing VCR capabilities in large-scale video servers," in *Proc. ACM Multimedia'94*, San Francisco, CA, Oct. 1994, pp. 25–32.

[11] R. Katz, G. Gibson, and D. Patterson, "Disk system architectures for high performance computing," *Proc. IEEE*, vol. 77, pp. 1842–1858, Dec. 1989.

[12] K. Keeton and R. Katz, "The evaluation of video layout strategies on a high-bandwidth file server," in *Fourth Int. Workshop on Network and Operating System Support for Multimedia*, UK, Nov. 1993, pp. 228–239.

[13] S. Paek, P. Bocheck, and S. F. Chang, "Scalable MPEG2 video servers with heterogeneous QoS on parallel disk arrays," in *Fifth Int. Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, Apr. 1995, pp. 363–374.

[14] P. Shenoy and H. Vin, "Efficient support for scan operations in video servers," in *Proc. ACM Multimedia'95*, San Francisco, CA, Nov. 1995, pp. 131–140.

[15] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 3, pp. 572–588, Sept. 1994.

[16] F. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming RAID—A disk array management system for video files," in *Proc. ACM Multimedia'93* Anaheim, CA, Aug. 1993, pp. 393–400.
[17] H. Vin, S. Rao, and P. Goyal, "Optimizing the placement of multimedia objects on disk arrays," in *Proc. Int. Conf. Multimedia Computing and Systems*, Washington, DC, May 1995, pp. 158–165.

Ed Chang received the B.S. degree from Purdue University, W. Lafayette, IN, and the M.S. and Ph.D. degrees from University of California, Berkeley, all in electrical engineering, in 1988, 1990, and 1996, respectively.

He is currently with Digital Equipment Corporation at the Cambridge Research Lab. His research interests are in the areas of video storage, video and image processing, and digital photography.

Dr. Chang was a National Science Foundation Fellow from 1989 to 1992.

Avideh Zakhor (S'87–M'87) received the B.S. degree from the California Institute of Technology, Pasadena, and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, all in electrical engineering, in 1983, 1985, and 1987, respectively.

In 1988, she joined the faculty at University of California, Berkeley, where she is currently an Associate Professor in the EECS Department. Her research interests are in the general area of signal processing and its applications to images and video. She holds four U.S. patents.

Dr. Zakhor was the recipient of the NSF Presidential Young Investigator Award in 1990.