Striping for Interactive Video: Is it Worth it?

Martin Reisslein *
GMD FOKUS
Kaiserin–Augusta–Allee 31
10589 Berlin, Germany
reisslein@fokus.gmd.de
www.fokus.gmd.de/usr/reisslein

Keith W. Ross
Institute Eurecom
2229 Route des Cretes
06904 Sophia–Antipolis, France
ross@eurecom.fr
www.eurecom.fr/~ross

Subin Shrestha
Wharton Computing (WCIT)
University of Pennsylvania
Philadelphia, PA 19104
subin@wharton.upenn.edu

October 1998

Abstract

We study the design of interactive video servers consisting of disk arrays. In order to avoid the hot–spot problem in video servers it is conventional wisdom to stripe the videos over the disk array using Fine Grained Striping or Coarse Grained Striping techniques. Striping, however, increases the seek and rotational overhead, thereby reducing the throughput of the disk array. Our results indicate that the decrease in throughput is substantial when interactive delays are constrained to be less than 1 second. We show that both a high degree of interactivity and high throughput are achieved by using a narrow striping width and replicating the videos according to the user's request pattern. Specifically, we find that striping over two disks gives the highest throughput when a tight 1 second constraint on interactive delays is imposed. We also demonstrate that localized placement (i.e., no striping at all) performs nearly as well when a good estimate of the user request pattern is available.

Keywords: Coarse Grained Striping; Fine Grained Striping; Interactivity; Video Placement; Video Server.

*Corresponding Author: Martin Reisslein, GMD FOKUS, Kaiserin–Augusta–Allee 31, 10589 Berlin, Germany, reisslein@fokus.gmd.de, phone: +49–30–3463–7282, fax: +49–30–3463–8000, http://www.fokus.gmd.de/usr/reisslein

1 Introduction

Consider designing a video server which makes available to its clients 10–100 constant–bit–rate (CBR) encoded videos, with each video being 1 to 200 minutes long. Suppose a design constraint is that all interactive delays — including delays after initial video start–up, after a pause/resume and after a temporal jump — be less than, say, 1 second. Such a constraint gives the user a pleasurable interactive experience with the system. In this paper we address the following question: How do we design such a highly–interactive video server which can accommodate a large number of concurrent video streams at reasonable cost?

One possibility is to endow the server with a huge RAM and to place all the prerecorded videos in the RAM. The high transfer and access rates out of RAM will ensure that the design constraints are met. This pure–RAM solution, once believed to be completely far–fetched, is now a realistic possibility due to the rapid decline in RAM cost in recent years. Nevertheless, if the total amount of prerecorded video content exceeds a few Gigabytes, as it would with a few MPEG–2 full–length movies, the pure RAM solution becomes cost prohibitive, and a disk–array becomes necessary. The per–byte cost of disk remains several orders of magnitude less than the per–byte cost of RAM.

Given that we are going to use a disk array for our video storage, the next issue is how are we going to place the videos on the disk array in order to maximize the number of connections that the server can simultaneously support? The most natural solution is *localized placement*, whereby each video file is contiguously placed on a disk. But if an entire video file is stored on one disk, the number of concurrent accesses to that file is limited by the disk throughput, which leads to the well–documented *hot spot problem*.

The hot–spot problem is that the great majority of the demand is often for a small subset of the stored videos. Because each disk can service only a small number of concurrent streams, much of the demand for the popular videos can go unsatisfied, while the disks housing the less popular videos are under–utilized. As an example, suppose that a video server houses 100 videos, each on its own disk; also suppose that each disk can support up to six concurrent streams. If nearly all the demand is for the five most popular videos, then the server will service only 30 concurrent streams, even though it has the theoretical capacity to service 600 concurrent streams. This example shows that the hot–spot problem, when not properly addressed, can lead to a tremendous reduction in server throughput.

To circumvent the hot–spot problem, it is conventional wisdom to *stripe* the videos across the disks. By striping a video over a subset of the disks, the server can use the throughput of the entire subset to generate streams emanating from the video. In fact, if each video is striped across the entire disk array, then the hot–spot problem vanishes — all demand distributions can be equally accommodated.

Striping has an obvious reliability problem — if one disk fails, then all the video files that are partially contained in the disk become unavailable to the users. (With localized placement a given disk stores portions of fewer videos, so disk failure has less impact on reliability.) The reliability problem that results from striping can be partially mitigated through the use of RAID (redundant array of inexpensive disks) technology.

But for video servers promising a high–degree of interactivity, striping engenders a more subtle performance problem. Specifically, the smaller striping unit increases the relative seek and rotational overhead, thereby decreasing the effective throughput of each disk in the array. As we shall show in this paper, this decrease in throughput can be painfully significant when interactivity constraints on the order of a second or less are present.

For highly–interactive video servers, we show that if striping is used, it should be used with a narrow

striping width. Furthermore, for the same number of disks, a simpler localized placement solution with appropriate replication may perform nearly as well as the striping solution. Thus, for highly–interactive video servers, the conventional wisdom to stripe videos may be flawed.

This paper is organized as follows. In Section 2 we present the disk model and request distribution model. In Section 3 we discuss uniform replication, whereby the same number of copies of each video is stored in the disk array. We consider uniform replication both for localized placement and for striping with a variety of striping widths. We consider both Fine Grained Striping (FGS) and Coarse Grained Striping (CGS). We also present numerical results which indicate that uniform placement can lead to unsatisfactory performance. In Section 4 we consider non–uniform replication, again for localized and striping placement. We show that if the videos are replicated to reflect the user's request pattern, then the server can achieve satisfactory throughput; moreover, localized placement gives satisfactory performance, and performs nearly as well as the best striping solution. In Section 5 we review the existing literature on file placement in video servers. In Section 6 we briefly summarize our findings.

2 The Model

We can highlight the main points most easily by introducing the following simplifying assumptions:

- We assume that all videos are constant–bit–rate (CBR) encoded. For CBR encoding, the quantization scale is dynamically changed to produce a near CBR bit stream. The resulting encoded video is sent to the client at a constant rate. When the client receives the video, it accumulates video in a small buffer and briefly delays playback.
- All videos have been encoded at the same rate, denoted by b.
- All video files are the same size, denoted by B.

In our numerical experiments we use $b = .375$ Mbytes/sec and $B = 3$ Gbytes. We shall assume that each disk has a capacity of 3 Gbytes; thus, each disk can hold exactly one video file.

We investigate two different video placement strategies:

- *Localized placement:* Each video file is contiguously stored within a single disk. If there is sufficient aggregate storage capacity in the disk array, multiple copies of the videos are stored on multiple disks. We consider two video replication strategies: *uniform replication*, whereby each video has the same number of copies; and *non–uniform replication*, whereby the different videos may have different numbers of copies stored in the disk array.
- *Striping placement:* Each video file is striped across a subset of disks in the disk array. If there is sufficient aggregate disk storage, multiple copies of a video may be striped within the array. We again consider two video replication strategies: uniform replication and non–uniform replication.

2.1 Model for User Request Pattern

Throughout our analysis we assume that the user demand for videos varies from video to video. Specifically, if there are M videos with video 1 being the most popular and video M being the least popular, then the probability that the mth most popular video is requested by a user is given by the Zipf distribution [11]:

$$q_m = K/m^{\zeta}, \qquad m = 1, \ldots, M,$$

where

$$K = \frac{1}{1 + 1/2^{\zeta} + \cdots + 1/M^{\zeta}}$$

The Zipf distribution corresponds to a highly–localized user request pattern that has been typical at movie rental stores. Note that the Zipf distribution depends on a parameter $\zeta > 0$. Increasing ζ increases the relative popularity of the most popular videos.

2.2 Disk Model

We assume that each disk consists of single platter side and a single arm. Let D denote the number of disks in the array. We assume that $D \geq M$, so that all the videos can be stored in the disk array.

We assume throughout that the server serves the ongoing video streams in constant–time rounds. During each round the server retrieves a fixed number of bytes for each client. Within a round, the number of bytes retrieved by the server for a client is equal to the number of bytes transmitted to the client. Specifically, with T denoting the round length, within each round and for every stream, the server retrieves a block of video of bT bytes from the disk subsystem and sends to the network bT bytes.

We assume that each disk in the disk array uses the SCAN scheduling algorithm [7]. Specifically, within each round, each disk arm sweeps across its entire platter exactly once with no back tracking. Because we assume the SCAN scheduling algorithm, the overhead incurred within a round for a given disk has the following form

$$\text{disk overhead} = l_{\text{seek}} + Il_{\text{rot}},$$

where I is the number of streams that the disk is servicing. The constant l_{seek} is the maximum seek time of the disk (the time to move the arm from the center to the edge of the platter, which is equal to the time to move the arm from the edge to the center of the platter). The constant l_{rot} is the per–stream latency, which includes the maximum rotation time of the disk and the track–to–track seek time. Table 1 summarizes our disk notation and the nominal values for the disk parameters. The nominal parameters reflect the current performance of high–speed disks [1].

parameter	notation	nominal value
disk size	X	3 Gbytes
disk transfer rate	r	2.5 MBytes/sec
maximum seek time	l_{seek}	20 msec
rotational latency	l_{rot}	10 msec
number of disks in array	D	10–100

Table 1: Nominal values of disk parameters used in numerical studies.

In main memory, the video server allocates to each stream a disk buffer and a network buffer. While the disk array fills the disk buffer, the network drains the network buffer, which has been previously filled. When the network has depleted the disk buffer, the disk buffer becomes the network buffer and vice versa. The roles of the two buffers continue to alternate throughout the life of the stream. Because one such double buffer is required for each stream, the amount of main memory required to support J streams is $2JTb$.

The initial start–up delay as well as the responsiveness to an interactive request (pause/resume or a temporal jump) is typically modeled to be twice the round length, $2T$, when the SCAN algorithm is used. This delay model is based on the worst–case assumption that the request of the user arrives just after the start of a round, say round k, and arrives too late to be scheduled by the SCAN algorithm for round k. The request has to wait for the start of the next round. The request is included in the disk read schedule of round $k+1$ and the requested video data is read into the disk buffer during round $k+1$. The disk buffer of round $k+1$ becomes the network buffer of round $k+2$ and the transmission of the requested video data out of the network buffer starts at the beginning of round $k+2$. Thus, the disk–subsystem introduces a maximum delay of two rounds, i.e., $2T$. We shall assume that the maximum disk–subsystem delay is constrained to .5 sec. Therefore, we use a round length of $T = .25$ sec. Note that the total interactive delay also includes transmission delays as well as client de–smoothing and decoding delays. These additional delays add another .25 sec to .5 sec to the .5 sec disk–subsystem delay, giving a total delay on the order of .75 sec to 1.0 sec. Thus, with a round length of .25 sec the system is able to give the user a pleasurable interactive experience with less than 1 second delay for all interactions.

3 Uniform Replication

In this section we will consider two placement strategies: localized placement and striping placement. If $D > M$, the disk array will contain multiple copies of some or all of the videos, with each copy on a different disk. Throughout this section we assume that the videos are *uniformly replicated*, i.e., D/M is a positive integer and that each video has the same number D/M copies in the disk array.

3.1 Localized Placement

Consider one of the D disks, and suppose that this disk is servicing I ongoing streams. Then within a round this disk will transfer I blocks of data to main memory, with each block consisting of bT bytes. The disk transfers each of these blocks at rate r. Thus the total disk transfer time within a round is IbT/r. The total disk overhead within a round is $l_{\text{seek}} + Il_{\text{rot}}$. Thus the amount of time the disk requires to service the I ongoing streams in a round is $IbT/r + l_{\text{seek}} + Il_{\text{rot}}$. Because the time required to service the I streams in a round must be no greater than the round length itself, we have

$$T \geq l_{\text{seek}} + Il_{\text{rot}} + I\frac{Tb}{r}.$$

Rearranging the terms in the above equation, the maximum number of streams that a disk can support is:

$$I = \left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{r}} \right\rfloor .$$

Because there are D disks, the maximum number of streams that the disk array can service for a given round time T is:

$$J_{\max}^{\text{local}} = D \left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{r}} \right\rfloor . \tag{1}$$

Because the demand for the various videos is typically non–uniform, the disk array will most likely serve substantially less than $J_{\max}^{\text{local}}$ streams. In the worst case, all requests will be for the same video,

in which case the number of streams serviced is

$$J_{\min}^{\text{local}} = \frac{D}{M}\left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{r}} \right\rfloor. \tag{2}$$

3.2 Striping Placement

With *full striping* each video is striped over all of the disks. There are essentially two different striping techniques: Fine Grained Striping (FGS) and Coarse Grained Striping (CGS) [4, 6].

Fine Grained Striping

With Fine Grained Striping each block is segmented into D equal–sized parts, called *stripes*, and each of the disks stores one of the block's stripes. When the server retrieves a block from the disk array, it reads all D stripes of the block in parallel. Let J denote the number of ongoing streams at the server. Consider one of the D disks. Within a round this disk will transfer J stripes to main memory, with each stripe consisting of bT/D bits. The disk transfers each of these stripes at rate r. Thus the total disk transfer time within a round is JbT/rD. Within this same round, the disk overhead is $l_{\text{seek}} + Jl_{\text{rot}}$. Thus the amount of time required to service the J ongoing videos in a round is $JbT/rD + l_{\text{seek}} + Jl_{\text{rot}}$. Because the time required to service the J streams in a round must be no greater than the round length itself, we have

$$T \geq l_{\text{seek}} + Jl_{\text{rot}} + J\frac{Tb}{rD}.$$

It follows from the above inequality that the maximum number of streams the server with FGS can support for a given round time T is

$$J_{\text{FGS}}^{\text{stripe}} = \left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{rD}} \right\rfloor. \tag{3}$$

Note that full striping with FGS can support $J_{\text{FGS}}^{\text{stripe}}$ streams for any request pattern.

A quick comparison of (1) and (3) shows that the localized layout can support a larger number of simultaneous streams. This is because each disk wastes a larger fraction of the round with seeks and rotations when Fine Grained Striping is employed. However, because each video file is spread over all D disks, the the full striping layout can support $J_{\text{FGS}}^{\text{stripe}}$ streams independent of the request pattern.

Coarse Grained Striping

With Coarse Grained Striping (also refered to as Data Interleaving in [5]) each block is stored on a separate disk. The blocks are typically assigned to the disks in a round–robin manner, that is, if block n is stored on disk 1 then block $n+1$ is stored on disk 2, and so on. When the server retrieves a block from the disk array it reads the entire block from one disk. Therefore CGS has less overhead than FGS (recall that with FGS the server has to access D disks to retrieve one block). The drawback of CGS, however, is its large interactive delay. Specifically, with T_{CGS} denoting the round length of the CGS server, the start–up latency and the responsiveness to interactions is typically modeled as $(D+1)T_{\text{CGS}}$. This interactive delay is based on the following worst–case scenario: The user requests video data from a disk, say disk d. The request arrives just after the start of a round, say round k. Even though disk d has a free slot in round k, that is, it has free disk transfer capacity to accommodate the request, the request arrives too late to be scheduled by the SCAN algorithm for round k. The delay model further assumes that none of the other disks in the array has a free slot in round k, that is, all other disks have already exhausted their disk transfer capacity. In CGS video servers with round–robin block assignment

the disk schedules circle around the disks in a round–robin fashion, that is, the disk schedule of disk d in round k is used by disk $d+1$ in round $k+1$, and so on. Therefore, in the considered scenario, disk d does not have a free slot in the next $D-1$ rounds. The next free slot at disk d is in round $k+D$. The requested video data is read into the disk buffer during this round. The transmission out of the network buffer starts at the beginning of round $k+D+1$. Thus, the disk–subsystem introduces a delay of $(D+1)T_{\mathrm{CGS}}$. We assume (as before in the case of localized placement and FGS) that the maximum disk–subsystem delay is constrained to .5 sec. Therefore we use a round length of $T_{\mathrm{CGS}} = .5\mathrm{sec}/(D+1)$. Note that the round length depends on the number of disks in the array; in order to satisfy the imposed constraint of interactive delays the CGS server has to reduce its round length when the number of disks grows. Recall that the FGS scheme satisfies the .5 sec constraint on the maximum disk–subsystem delay with a round length of $T = .25$ sec (independent of the number of disks). For ease of comparison of CGS with FGS we express the round length of CGS in terms of the round length of FGS:

$$T_{\mathrm{CGS}} = \frac{2T}{D+1}. \tag{4}$$

We proceed to analyze the throughput of the CGS server when a tight constraint on interactive delays is imposed. Let J denote the number of ongoing streams at the server and consider one of the D disks. Within a round this disk will transfer J/D blocks to the disk buffer (assuming that the blocks accessed in a round are evenly divided among the disks). Each block consists of bT_{CGS} bits. The disk transfers the blocks at rate r. Thus the total disk transfer time within a round is JbT_{CGS}/rD. Within this same round, the disk overhead is $l_{\mathrm{seek}} + Jl_{\mathrm{rot}}/D$. Since the time required to service the J/D streams in a round must be no greater than the round length itself, we have

$$T_{\mathrm{CGS}} \geq l_{\mathrm{seek}} + Jl_{\mathrm{rot}}/D + J\frac{T_{\mathrm{CGS}}b}{rD}.$$

The maximum number of streams the server with CGS can support for a given round time T_{CGS} is

$$J_{\mathrm{CGS}}^{\mathrm{stripe}} = \left\lfloor \frac{T_{\mathrm{CGS}} - l_{\mathrm{seek}}}{\frac{l_{\mathrm{rot}}}{D} + \frac{T_{\mathrm{CGS}}b}{rD}} \right\rfloor. \tag{5}$$

For ease of comparison with FGS we substitute (4) into (5) and obtain:

$$J_{\mathrm{CGS}}^{\mathrm{stripe}} = \left\lfloor \frac{T - \frac{D+1}{2}l_{\mathrm{seek}}}{\frac{D+1}{2D}l_{\mathrm{rot}} + \frac{Tb}{rD}} \right\rfloor. \tag{6}$$

We provide a detailed numerical comparison of (3) and (6) in Section 3.3. We note here that for the trivial case of one disk, $D = 1$, FGS and CGS can support the same number of streams. We also note that for $D \geq \lceil 2T/l_{\mathrm{seek}} - 1 \rceil$ CGS is not able to support any streams with a constraint of $2T$ on the maximum disk–subsystem delay.

3.3 Group Striping

Now consider *group striping*. In this scheme, we stripe each video file over $W \leq D$ disks. We refer to W as the *striping width*. A little thought shows that in this strategy (i) each disk contains data from W video files and (ii) if $W \leq M$, then the disks can be partitioned into groups of size W such that each copy of a video file is striped within a group. Group striping with $D = 6$ disks, $M = 3$ movies and different striping widths W is illustrated in Figure 1.

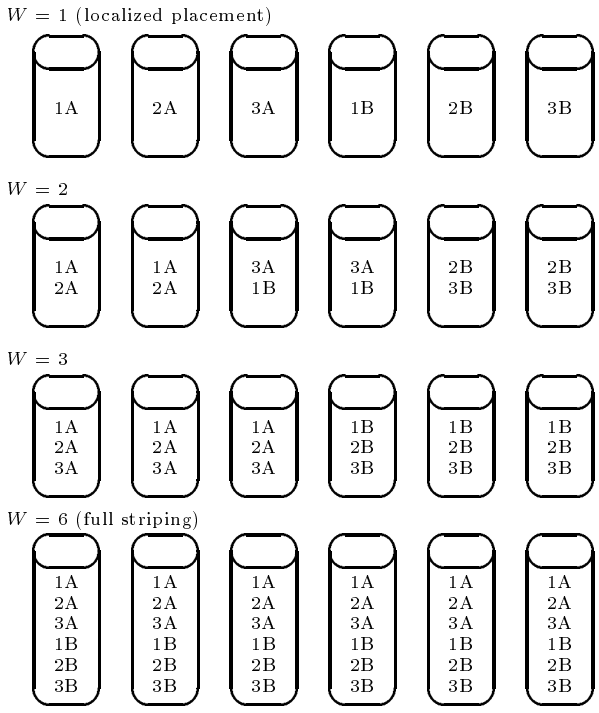

Figure 1: Group striping of $M = 3$ movies (denoted by 1, 2, and 3) over $D = 6$ disks with striping widths of $W = 1$, 2, 3, and 6. We use uniform replication in this example; thus there are $D/M = 2$ copies (denoted by A and B) of each movie stored in the disk array.

We consider group striping both with FGS and CGS. With FGS each block is segmented into W stripes, and each disk in the striping group stores one of the block's stripes. With I denoting the number of streams serviced by a given striping group with FGS, we have

$$T \geq l_{\text{seek}} + Il_{\text{rot}} + I\frac{Tb}{rW}.$$

It follows from the above inequality that the maximum number of streams the striping group can support with FGS for a given round time T is

$$I_{\text{FGS}} = \left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{rW}} \right\rfloor. \tag{7}$$

With CGS the blocks of a video are stored on the W disks in a striping group in a round–robin fashion, with each disk storing an entire block. With a derivation that parallels the analysis of CGS for full striping we obtain for the maximum number of streams a striping group can support with CGS:

$$I_{\text{CGS}} = \left\lfloor \frac{T - \frac{W+1}{2} l_{\text{seek}}}{\frac{W+1}{2W} l_{\text{rot}} + \frac{Tb}{rW}} \right\rfloor. \tag{8}$$

Table 2 shows the number of connections that can be supported by a striping group with FGS and CGS as a function of the width of the striping group. Focusing for now on the FGS results, we see from this table that the maximum number of streams grows sub–linearly with the width of the striping

Width of Striping Group	Maximum Number of Streams with FGS	Maximum Number of Streams with CGS
1	4	4
2	8	8
3	10	10
4	11	12
5	13	14
10	16	15
20	19	5
50	21	0
100	22	0

Table 2: Maximum number of streams that can be supported by an isolated striping group with Fine Grained Striping (FGS) and Coarse Grained Striping (CGS).

group. For example, if we increase the number of disks from 50 to 100, only one more additional stream can be supported! This sublinear growth is due to the increased relative seek and rotational overhead that comes from Fine Grained Striping. We note, however, for the particular system values considered, that increasing the width from 1 to 2 allows for linear growth. Comparing the results for FGS and CGS we see that CGS can support one more stream with $W = 4$ and $W = 5$. This is due to the fact that CGS has less overhead; CGS accesses one disk to retrieve one block whereas FGS accesses W disks. However, the number of streams CGS can support drops off sharply for larger W, and for $W \geq \lceil 2T/l_{\text{seek}} - 1 \rceil = 23$ CGS can not support any streams with a 1 second constraint on interactive delays. This is because the round length of CGS depends on the interactive delay constraint and the number of disks in the striping group, i.e., $T_{\text{CGS}} = .5\text{sec}/(W+1)$. For larger striping widths the round length has to be shortened in order to meet the interactive delay constraint. Once the round length is shorter than the seek latency, l_{seek}, however, there is no time left in the round to transfer video data. In summary, CGS performs slightly better than FGS for moderate striping widths but extremely poorly for large striping widths. We use FGS throughout the rest of this paper.

The maximum number of streams that group striping with FGS can support is

$$J_{\max}^{\text{stripe}}(W) = \left\lfloor \frac{D}{W} \right\rfloor \left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{rW}} \right\rfloor. \tag{9}$$

Note that (9) is maximized at $W = 1$, that is, localized placement beats striping for all striping widths with *respect to the maximum number of possible of streams.*

The worst–case request distribution is such that all requests are for the same video. In this case, the number of videos that can be served by the disk array is

$$J_{\min}^{\text{stripe}}(W) = \left\lfloor \frac{D}{\max(M, W)} \right\rfloor \left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{rW}} \right\rfloor. \tag{10}$$

Note that $J_{\min}^{\text{stripe}}(W)$ is maximized at $W = M$. Thus we see that $J_{\max}^{\text{stripe}}(W)$ decreases with the striping width but $J_{\min}^{\text{stripe}}(W)$ increases with the striping width (for $W \leq M$).

Table 3 gives $J_{\min}^{\text{stripe}}(W)$ and $J_{\max}^{\text{stripe}}(W)$ for $D = 20$ disks and $M = 10$ videos for a range of widths W. It is interesting to note from Table 3 that for small widths there is a large difference between the

minimum and maximum streams, i.e., the number of streams that can be supported with small widths depends largely on the request pattern. However, the request pattern has little influence on the number of streams for the larger striping widths.

Striping Width	Minimum Number of Streams	Maximum Number of Streams
1	8	80
2	16	80
3	20	60
4	22	55
5	26	52
10	32	32
20	19	19

Table 3: Minimum and maximum number of streams that can be supported by a disk array with $D = 20$ disks and $M = 10$ videos.

Let us now suppose that the request pattern follows the Zipf distribution with parameter $\eta = 1$. For a fixed number of disks, D, and a fixed number of videos, M, we now determine the number of streams that can be typically supported by each of the placement schemes. We make this determination using the following simulation experiment. For a given placement strategy and target number of streams, S, we generate S requests from the Zipf distribution and determine the number of requests that can be supported (which will be no greater than S). We repeat the experiment 1000 times, creating $1000 \cdot S$ requests. If 95% of these requests can be supported, we then increment S and repeat the entire procedure. The procedure continues until the 95% criterion is violated.

The S determined in this procedure is the number of customers that a service provider could plan to support at the peak hour on any given day. Specifically, if the provider allows for S requests each evening during the peak hour, it should be able to support about 95% of the first–choice requests over the year.

Table 4 presents the results of this procedure for 10, 20, and 100 disks. We observe that for the case of uniform replication, the striping width $W = 2$ gives significantly more connections than does localized placement ($W = 1$). Further increasing the striping width decreases the number of connections.

Striping Width, W	Number of Streams 10 disks	Number of Streams 20 disks	Number of Streams 100 disks
1	10	25	136
2	21	46	245
5	21	43	225
10	16	32	160

Table 4: Number of streams that can be supported by a disk array with $D = 10$, 20, and 100 disks and $M = 10$ videos for different striping widths W; Zipf request pattern with $\xi = 1$.

4 Non–Uniform Replication

Examining Table 4 we see that the number of streams that can typically be supported is significantly lower than the maximum number of possible streams. For example, for 20 disks the disk subsystem has a capacity of 80 streams, whereas only 46 streams can typically be supported with the optimal striping width ($W = 2$); similarly, for 100 disks the disk subsystem has a capacity of 400 streams, whereas only 245 streams can typically be supported with the optimal striping width ($W = 2$). We are therefore motivated to consider non–uniform replication in order to further increase the number of streams that can typically be supported.

In this section we permit non–uniform replication for both localized and striping placement. Let C_m be the number of copies of video m stored in the disk array. Because the size of each video file equals the capacity of a single disk,

$$\sum_{m=1}^{M} C_m = D.$$

Adapting the theory of the previous section, we obtain the maximum number of video–m streams that can be supported with localized placement:

$$J_{\max}^{\text{local}}(m) = C_m \left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{r}} \right\rfloor.$$

For striping, if $WC_m \leq D$ for all $m = 1, \ldots, M$, then copies of the same video can be striped over disjoint groups of disks, and the maximum number of streams for video–m is

$$J_{\max}^{\text{stripe}}(m; W) = C_m \left\lfloor \frac{T - l_{\text{seek}}}{l_{\text{rot}} + \frac{Tb}{rW}} \right\rfloor.$$

Now let us suppose that the user request pattern for the M videos has a known distribution (perhaps a Zipf distribution with known parameter ζ). We consider replicating the videos such that the replication distribution is approximately equal to the request distribution. In particular, we replicate using the following algorithm:

1. $C_m = \lfloor q_m D \rfloor$, $m = 1, \ldots, M$.
2. If $C_m > D/W$, set $C_m = \lfloor D/W \rfloor$, $m = 1, \ldots, M$.
3. If $C_m = 0$, set $C_m = 1$.
4. Calculate $C = C_1 + \cdots + C_M$.
5. If $C > D$, decrement C_m for the least popular video with $C_m > 1$, then for the next to least popular with $C_m > 1$, etc., until $C = D$.
6. If $C < D$, increment C_m for the most popular video with $C_m < \lfloor D/W \rfloor$, then for the next to most popular video, etc., until $C = D$.

This algorithm ensures that there is at least one copy present for each of the M videos. After determining the number of copies of each video, we must assign the copies to the various groups of disks. For this, we use of the tennis–player–seeding heuristic: we place the most popular videos with

the least popular videos in a group, and we place moderately popular videos with other moderately popular videos in a group.

Table 5 summarizes uniform and Zipf replication for 20 disks. We see from the table that Zipf replication for striping widths of 1 and 2 roughly doubles the number of connections.

W	Uniform Replication	Zipf Replication	Maximum Number of Streams
1	25	52	80
2	46	80	80
5	43	52	52
10	32	32	32

Table 5: Number of streams that can be supported by a disk array with $D = 20$ disks and $M = 10$ videos.

Table 6 summarizes uniform and Zipf replication for 100 disks. We see from this table that Zipf replication greatly increases the typical number of connections for localized placement ($W = 1$) and striping with $W = 2$. Notice that $W = 2$ with Zipf replication increases the number of supported connections from 245 to 400, which is the maximum number of connections the disk subsystem can support. But we now notice that for the larger system with $D = 100$ disks localized placement performs nearly as well as striping with $W = 2$ and significantly better than the larger striping widths ($W = 5$ and $W = 10$).

W	Uniform Replication	Zipf Replication	Maximum Number of Streams
1	136	388	400
2	245	400	400
5	225	260	260
10	160	160	160

Table 6: Number of streams that can be supported by disk array with $D = 100$ disks and $M = 10$ videos.

In summary, if videos are replicated to reflect a known user request pattern, than the number of connections that are typically supported can approach the maximum number of connections possible. Furthermore, the optimal striping width is small, and for large systems localized placement performs nearly as well as does striping with the optimal striping width.

Next, we study the robustness of the non–uniform replication approach to unpredictable changes in the user request distribution. We study how the Zipf replication performs when the *actual* user request distribution differs from the *expected* user request distribution. (The videos are replicated according to the expected user request distribution.) For this study we focus on the video server with $D = 100$ disks and $M = 10$ movies. We assume that the expected user request distribution is the Zipf distribution with parameter $\xi = 1$. We replicate the movies according to this distribution in the disk array. We furthermore assume that the actual user request distribution is the Zipf distribution with the parameter ξ in the interval $[0.25, 3]$. Note that a larger ξ increases the popularity of the most popular movie. With a parameter of $\xi = 1$, for instance, on average 34 % of the requests are for the most popular movie, whereas with $\xi = 2$ on average 65 % of the requests are for the most popular movie. In Figure 2 we

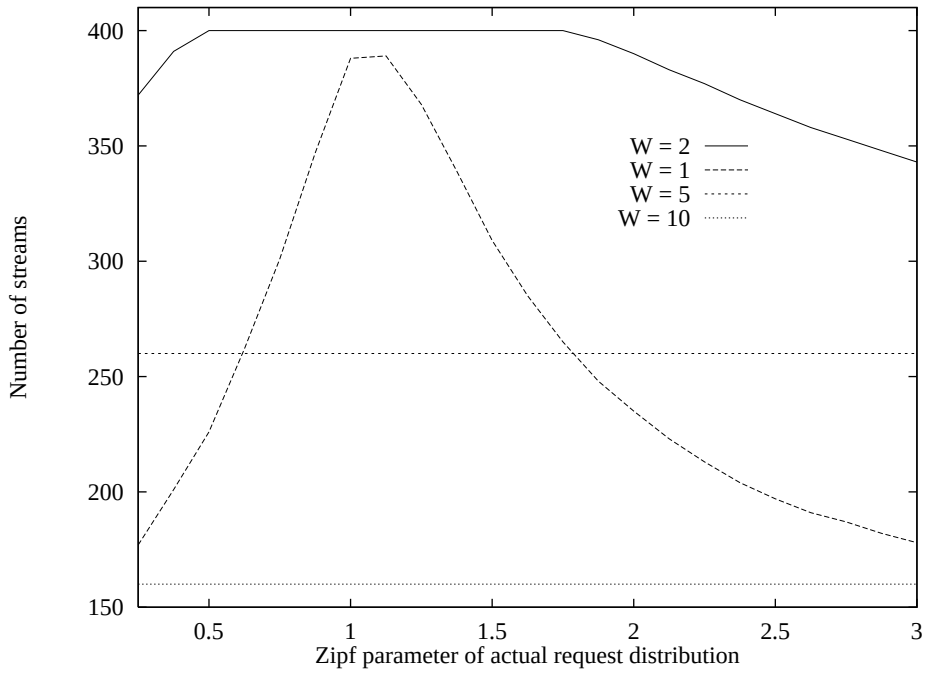

Figure 2: Number of streams as a function of the Zipf parameter ξ of the actual user request distribution. The movies are replicated according to the expected user request distribution, which is assumed to be the Zipf distribution with parameter $\xi = 1$.

plot the number of streams that can be supported as a function of the Zipf parameter ξ of the actual request distribution. We observe from the figure that the server with striping width $W = 2$ can support 400 connections over a wide range of the Zipf parameter. This indicates that a narrow striping width of $W = 2$ gives good robustness, that is, the server can support the maximum number of connections even when there is some uncertainty about the actual user request distribution. With localized placement ($W = 1$) the server can support close to 400 connections when the user request distribution is fairly well known. The number of streams drops off quickly when the actual user request distribution deviates from the expected user request distribution. However, localized placement is still able to support more connections than group striping with $W = 5$ over a relatively wide range of ξ from 0.6 to approximately 1.8. The larger striping widths 5 and 10 give excellent robustness, but the maximum number of streams with these larger striping widths is small.

4.1 Estimating the User Request Pattern

We are advocating that the videos should be replicated in a fashion that reflects the user request pattern. But how can a video service provider determine the user request pattern? Let us propose a partial solution to this problem in the context of movies on demand. Suppose that the service provider has an estimate, $q_1, \ldots, q_M$, of the request pattern for M movies that are currently stored on the server. Further suppose that at the end of the current evening the provider has observed that the demand for the evening was $p_1, \ldots, p_M$. We recommend that the new estimate of the user request pattern be updated according an exponential moving average:

$$q_m = (1 - \alpha)q_m + \alpha p_m,$$

where $0 < \alpha < 1$ is the dampening factor. The choice of an appropriate α is up to the service provider. If the demand distribution evolves rapidly, then α should be relatively large, .2 or higher. If, on the other hand, the demand distribution evolves slowly, with occasional unusual and unpredictable demands, then α should be set to a small value.

The exponential moving average is a reasonable scheme for dynamically estimating the user request pattern when the collection of offered movies does not change. But how should the distribution be modified when a new movie is introduced? To answer this question, movie–on–demand service providers should study the strategies used by VCR tape rental companies, who have had to address a similar problem. Obviously, the new distribution will have to take in account the past success that the movie (or a similar movie) has had in the big–screen movie theaters.

Of course it is also possible to dynamically copy movies that experience unexpected high demand. This is indicated when a particular movie proves to be much more popular than expected early on in the evening. If one disk with that particular movie (assuming localized placement) is still unused, this disk can be used exclusively to produce a new copy. In this case there are no seek or rotational delays involved and the transfer of the movie is limited only by the the disk transfer rate. With the parameters of Table 1 it takes $B/r = 20$ minutes to copy the movie. If all of the disks storing the unexpectedly popular movie are already serving clients, the copy should be made from the disk serving the fewest clients. In this case seek and rotational latencies are incurred and copying the movie takes therefore longer.

5 Related Work

There is a large body of literature that addresses striping and placement strategies for continuous stored media which is complementary to the problems addressed in this paper. The tutorial by Gemmel *et al.* [5] gives a general introduction to the issues involved in video server design.

Gafsi and Biersack [4] study video servers with full striping, i.e., each video is striped over all the disks in the server; group striping is not considered. Gafsi and Biersack introduce Mean Grained Striping (MGS), a striping technique whereby each block is striped over a different subset of the disks. Roughly speaking, MGS is similar to CGS if the blocks are striped over few disks; if the blocks are striped over many disks MGS is similar to FGS. Gafsi and Biersack compare the maximum number of supported streams, the start–up latency and the buffer requirement of the FGS, MGS, and CGS schemes. They find that CGS can support roughly three times as many streams as FGS. However, they also find that the worst–case start–up latency with CGS is approximately five times larger than the start–up latency with FCS. The full–striping video servers studied in [4] have typically start–up latencies of the order of tens of seconds. We have shown in this paper that given a 1 second constraint on interactive delays, FGS and CGS can support about the same number of streams when a small striping width is used. We have also shown that CGS performs extremely poorly for larger striping widths.

Özden *et al.* [6] study FCS and CGS in video servers with full striping. Their focus is on finding the block size that maximizes the maximum number of streams the server can support while concurrently minimizing the server cost. A great deal of attention is devoted to the analysis of the scheduling of user requests in the CGS scheme. Özden *et al.* find that for the same server cost CGS can support roughly three to four times as many streams as FCS. However, start–up latencies and interactive latencies, which are a key design constraint in this paper, are not studied in [6].

Vin *et al.* [10, 8] study striping for VBR media. Their striping technique is similar to CGS in that

each disk stores a fixed–size block. Due to the VBR nature of the stored videos, however, the number of blocks (and hence the number of disks) accessed in a fixed–length round varies. Their model assumes that the movies are uniformly replicated. It also assumes that the user request distribution is uniform, that is, a user is equally likely to request any of the stored movies. Vin *et al.* argue that the number of streams that the disk array can support is limited by the the most heavily loaded disk. They develop an analytical model for the work load of the most heavily loaded disk. This model is used to determine the block size for striping that maximizes the number of supported streams.

Chervenak *et al.* [2] study the performance of video servers in Video on Demand (VoD) systems that do not allow for any interactivity, such as VCR actions, and have a 60 second start–up delay. Their study is restricted to localized placement and full striping; group striping is not considered. Chervenak *et al.* find that for their non–interactive VoD system full striping outperforms localized placement. We have shown in this paper that this results does not hold for highly interactive video servers.

Flynn and Tetzlaff [3, 9] study block assignment schemes for CGS. They consider the round–robin assignment scheme and different permutation based schemes. They investigate the impact of the different assignment schemes on reliability and response time of the server.

6 Conclusion

We have studied the placement of videos on disk arrays of interactive video servers. We have taken a critical look at the conventional wisdom to to use wide striping, (i.e., to stripe the videos over the entire disk array) in order to avoid the hot–spot problem. Our numerical studies based on the parameters of current high performance disks demonstrate that wide striping results in low throughput when tight constraints on interactive delays are imposed. We advocate localized placement (i.e., no striping at all) or striping with a narrow striping width to achieve high throughput in highly interactive video servers. In order to overcome the hot–spot problem we propose to replicate the videos according to an estimate of the user's request pattern. We have outlined how such an estimate can be obtained. We have demonstrated that localized placement is the placement strategy of choice when a good estimate of the user's request pattern can be obtained. Besides giving high throughput and good responsiveness to interactions, localized placement is simple, allows for straightforward disk scheduling and avoids the reliability problems that arise with striping. If only a rough estimate of the user's request pattern can be obtained we recommend to stripe over two disks. We have demonstrated that a striping width of two gives high throughput and responsiveness even when the actual user's request pattern differs significantly from the estimate.

References

[1] H. Boegeholz. Platten–karussell: 263 festplatten im vergleich. *c't magazine fuer computer technik*, (14):150–161, July 1998.

[2] A. L. Chervenak, D. A. Patterson, and R. H. Katz. Choosing the best storage system for video service. In *Proceedings of ACM Multimedia*, 1995.

[3] R. Flynn and W. Tetzlaff. Disk striping and block replication algorithms for video file servers. In *Proceedings of 3rd IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, June 1996. also available as IBM research report RC 20328.

[4] J. Gafsi and E. W. Biersack. Data striping and reliability aspects in distributed video servers. In *Cluster Computing: Networks, Software Tools, and Applications*, 1998. Available at http://www.eurecom.fr/~erbi.

[5] D. J. Gemmel, H. M. Vin, D. D. Kandalur, P. V. Rangan, and L. A. Rowe. Multimedia storage servers: A tutorial. *IEEE MultiMedia*, 28(5):40–49, May 1995.

[6] B. Özden, R. Rastogi, and A. Silberschatz. Disk striping in video server environments. In *Proceedings of IEEE Conference on Multimedia Systems*, pages 580–589, Hiroshima, Japan, June 1996.

[7] A. L. N. Reddy and J. C. Wyllie. I/O issues in a multimedia system. *Computer*, 27(3):69–74, March 1994.

[8] P. J. Shenoy and M. Vin. Efficient striping techniques for multimedia file servers. In *Proceedings of NOSSDAV '97*, pages 25–36, May 1997.

[9] W. Tetzlaff and R. Flynn. Block allocation in video servers for availability and throughput. In *Proceedings of Multimedia Computing and Networking*, January 1996. also available as IBM research report RC 20329.

[10] H. M. Vin, S. S. Rao, and P. Goyal. Optimizing the placement of multimedia objects on disk arrays. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS '95)*, Washington D.C., May 1995.

[11] G. K. Zipf. *Human Behavior and Principle of Least Effort: An Introduction to Human Ecology.* Addison–Wesley, Cambridge, MA, 1949.