Multimedia Systems 5:371–379 (1997)

Multimedia Systems

Using rotational mirrored declustering for replica placement in a disk-array-based video server

Ming-Syan Chen[1], **Hui-I Hsiao**[2], **Chung-Sheng Li**[2], **Philip S. Yu**[2]

[1] Electrical Engineering Department, National Taiwan University, Taipei, Taiwan ROC
[2] IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

Abstract. In a video-on-demand (VOD) environment, disk arrays are often used to support the disk bandwidth requirement. This can pose serious problems on available disk bandwidth upon disk failure. In this paper, we explore the approach of replicating frequently accessed movies to provide high data bandwidth and fault tolerance required in a disk-array-based video server. An isochronous continuous video stream imposes different requirements from a random access pattern on databases or files. Explicitly, we propose a new replica placement method, called *rotational mirrored declustering* (RMD), to support high data availability for disk arrays in a VOD environment. In essence, RMD is similar to the conventional mirrored declustering in that replicas are stored in different disk arrays. However, it is different from the latter in that the replica placements in different disk arrays under RMD are properly rotated. Combining the merits of prior chained and mirrored declustering methods, RMD is particularly suitable for storing multiple movie copies to support VOD applications. To assess the performance of RMD, we conduct a series of experiments by emulating the storage and delivery of movies in a VOD system. Our results show that RMD consistently outperforms the conventional methods in terms of load-balancing and fault-tolerance capability after disk failure, and is deemed a viable approach to supporting replica placement in a disk-array-based video server.

Key words: Video-on-demand – Disk-array-based video server – Replication – Rotational mirrored declustering – Fault-tolerance

1 Introduction

Recent advances on multimedia technologies have created several ventures on both information-providing service and entertainment business. Given the extremely large data size, one of the major challenges to handle multimedia data is to support very high disk bandwidth for video data retrieval. For example, to display HDTV-quality image, it will require a data rate at 2–3 MB per second (even after compression). In general, it is very undesirable to store such a large video in a single disk of two reasons. First, a 100-min HDTV movie will require more than 12-GB storage. Such a large disk is usually expensive. Second, playing a hot (i.e., frequently requested) movie by a single disk will cause a performance bottleneck. In fact, even for playing ordinary MPEG movies, the need to support multiple video streams by a video server also calls for the use of disk arrays. Consequently, it is highly desirable to use disk arrays to handle the storage and retrieval of multimedia data.

In a video-on-demand (VOD) system, multimedia streams are stored on a storage server (the video server) and played out to the user station upon request. A significant amount of effort has been elaborated upon many design aspects of a video server [8, 11–15]. A video server for this purpose is expected not only to concurrently serve many clients (hundreds or more), but also to provide many interactive features for video playout, such as pause/resume, backward play, and fast-forward and fast-backward play, which home viewers have been enjoying from the current VCR systems. Providing interactive features will unavoidably compromise the opportunity of batching requests and increase the system resource required [4, 5]. In addition, it is projected that, similar to the current movie rental business, the movie request in a VOD environment will be highly skewed, i.e., most of the requests are made for viewing a small number of hot movies. These factors, together with the importance of fault tolerance in real-time VOD applications, suggest that replicating certain frequently accessed movies in some disk arrays is a viable approach to providing the VOD service required [2, 3, 6]. It is worth mentioning that data replication in VOD system is employed primarily to support the high I/O bandwidth required for multimedia data. This is in contrast to data replication in OLTP (Online transaction processing) systems that is mainly used to provide fault tolerance. The VOD system considered in this paper is illustrated in Fig. 1, in which the disk-array-based video server is composed of many disk arrays. Each disk array is the unit to store a copy of a movie. If necessary, one movie may be replicated and stored in more than one disk array.

Correspondence to: M.-S. Chen

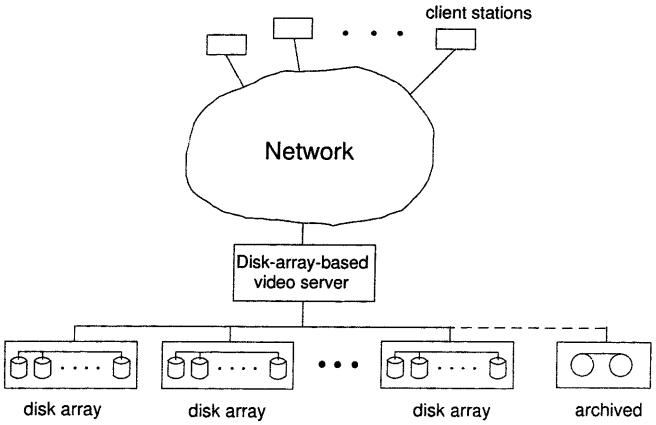


Fig. 1. A disk-array-based video server

To employ the replication approach to VOD applications, the primary issue is to determine the placement of movie copies in disk arrays. Due to the nature of sequential access, it is desirable to decluster each movie copy into blocks and stripe them across a disk array. Basically, there are three replica declustering techniques proposed in the literature to improve fault tolerance, namely, *mirrored declustering*, *chained declustering*, and *interleaved declustering*. These declustering techniques are primarily employed to support random access applications, e.g. database transaction processing [10]. However, as will be explained in Sect. 2.1, the isochronous support for each continuous video stream in a VOD environment imposes different requirements, and hence calls for alternative replica declustering techniques. In view of this and the fact that the fault tolerance provided by the conventional mirrored declustering method is deemed insufficient, we shall propose a new replication method, called *rotational mirrored declustering* (RMD), to support high data availability for disk arrays in VOD environments. In essence, RMD is similar to mirrored declustering in that multiple copies are stored in different disk arrays to increase the supportable throughput. It is, however, different from the latter in that the data placements in different disk arrays under RMD are properly rotated, thereby greatly increasing the degree of fault tolerance. The net effect of the rotation in replica placement is to achieve a different layout on striping for the replica from the original copy, and upon disk failure, the traffic to the failed disk can be spread across all disks in the disk array containing the replica. (This is in contrast to the conventional mirrored declustering, where the traffic to the failed disk can only be redirected to one of the disks in the array containing the replica.) Combining the merits of both chained and mirrored declustering methods, RMD is particularly useful for storing multiple movie copies in a disk-array-based video server to support VOD applications.

To assess the performance of RMD, we shall conduct a series of experiments by emulating the storage and delivery of movies in a VOD system. Specifically, we shall first determine the number of replicas required for each movie, and then properly assign each movie copy to a disk array. Clearly, for a given aggregate disk bandwidth and the access frequency of each movie, one would naturally like to determine the number of replicas for each movie so as to maximize the system throughput, as well as to attain the fault tolerance required. To deal with this, we shall introduce a new parameter, termed *replication threshold*, and utilize it to conduct several sensitivity analyses for various replication scenarios. Replication threshold is defined as the maximal percentage of the aggregate disk array bandwidth that is allowed to be allocated to a single movie. As such, replication threshold is used to determine if a movie is required to be replicated, and then how many copies are required if replication is necessary. By properly varying the value of replication threshold, we can empirically determine the operating point for a given movie access distribution. In addition, it is noted that even with a given replication threshold, similarly to the conventional packing problem, different placement sequence of movies will result in different allocation scenarios, thus leading to different system throughputs and fault tolerance is not yet fully exploited. In view of these, we shall evaluate different policies for movie selection to provide some insights into this aspect. Based on the above model, we conduct several experiments to evaluate the performance of RMD for various replication scenarios. Our results show that RMD consistently outperforms the conventional methods in terms of load balancing and fault tolerance capability after disk failure, and is deemed a viable approach to supporting movie placement in a disk-array-based video server.

This paper is organized as follows. The proposed RMD scheme is described in Sect. 2. The model for our experiment is explained in Sect. 3. Performance studies are conducted and the corresponding results analyzed in Sect. 4. Section 5 summarizes this paper.

Table 1. A double redundant data placement with mirrored declustering

	Disk array 1				Disk array 2			
Disk	0	1	2	3	4	5	6	7
m0 – m3	m0	m1	m2	m3	m0	m1	m2	m3
m4 – m7	m4	m5	m6	m7	m4	m5	m6	m7

2 RMD for replica placement

We shall describe the conventional replica placement methods in Sect. 2.1 and then propose a new placement method, RMD in Sect. 2.2.

2.1 Conventional data placement

There are three basic replica declustering techniques proposed in the literature to improve fault tolerance, i.e., mirrored declustering, chained declustering, and interleaved declustering. These three declustering techniques are primarily employed to support random access applications. In mirrored declustering, the disks are organized into many identical disk arrays, and fault tolerance is then achieved by storing replicated data in these disk arrays. An illustrative example for storing two movie copies in two disk arrays under mirrored declustering is given in Table 1. It can be seen that, while being easy to implement and proper for random access applications, the mirrored declustering is not suitable for sequential access applications, such as VOD, where losing a single disk in a disk array will render the whole striped

Table 2. Chained declustering disk array with double redundancy

	Disk array 1				Disk array 2			
Disk	0	1	2	3	4	5	6	7
1st copy	m0	m1	m2	m3	m4	m5	m6	m7
2nd copy	m3	m0	m1	m2	m7	m4	m5	m6

array useless since the video delivery requires access to each disk in cyclical order repeatedly.

On the other hand, under the chained declustering method, the primary data copy in disk i has a backup copy in disk (i + 1) mod n, where n is the total number of disks employed. An illustrative example for chained declustering is shown in Table 2, where the cluster size is equal to the disk array size, i.e., four. As shown in [9, 10], for random access pattern, the chained declustering method can achieve better fault tolerance and load balancing after failure than the mirrored declustering method. For example, when disk 1 fails under mirrored declustering (Table 1), accesses to m1 and m5 will need to be redirected to disk 5, resulting in a 100% workload increase at disk 5. With chained declustering (Table 2), when disk 1 fails, accesses to m0 and m1 can be evenly redistributed among all remaining disks in the same cluster [10]. Note, however, that, using the chained declustering, it will be costly to dynamically change the number of replicas, since to do that, it would be necessary to change the number of disk arrays to stripe across for the existing copies.

In interleaved declustering, the backup copy of the primary data in a disk is broken up into multiple sub-partitions. Each of the sub-partitions is stored on a different disk within the same disk array – but not on the disk containing the primary data [7]. It can be seen that due to a similar reason as in chained declustering, adding/dropping movie copies dynamically is very costly under the interleaved declustering method. However, in real-time VOD applications, the number of copies stored for a video has to change as the demand for the video varies. With chained declustering and interleaved declustering, applications will need to be interrupted while adding/dropping data copies, thus making these two methods unfavorable for VOD applications. As a result, the isochronous support for each continuous video stream in a VOD system calls for alternative replica-declustering techniques.

2.2 Description of the RMD scheme

To remedy the drawbacks of the conventional mirrored and chained declustering methods, we describe here a new method, RMD, to support movie placement in a VOD system. Combining the merits of both chained and mirrored declustering methods, RMD is particularly useful for storing multiple movie copies in a disk-array-based video server to support VOD applications. Formally, RMD can be formulated as follows. Given a set of rn disks numbered $0, 1, 2, \ldots, rn-1$, where n is the size (number of disks) of a disk array, the i^{th} partition of the j^{th} replica of a movie is placed on the disk numbered $d(i, j)$, where

$$d(i,j) = (i + \lfloor i/n \rfloor \times (j-1)) \bmod n + (j-1) \times n \ .$$

Table 3. A double redundant data placement under RMD

	Disk array 1				Disk array 2			
Disk	0	1	2	3	4	5	6	7
m0–m3	m0	m1	m2	m3	m0	m1	m2	m3
m4–m7	m4	m5	m6	m7	m7	m4	m5	m6

Table 4. A triple redundant data placement under RMD

	Disk array 1				Disk array 2				Disk array 3			
Disk	0	1	2	3	4	5	6	7	8	9	10	11
m0–m3	m0	m1	m2	m3	m0	m1	m2	m3	m0	m1	m2	m3
m4–m7	m4	m5	m6	m7	m7	m4	m5	m6	m6	m7	m4	m5

For example, a double redundant data placement based on RMD is shown in Table 3, where each disk array in the RMD declustering strategy contains a movie copy. Unlike the chained declustering method, RMD allows replica and disks to be added online, while neither having to move data in the existing copy around nor affecting the disk arm movement on existing disks. For example, to add a new movie copy to the existing two copies shown in Table 3, a third data array will be allocated (either from an existing and available pool of disk arrays or adding a new disk array to the system) to the movie. The movie copy can then be loaded from tape to the newly allocated disk array without affecting the applications accessing the existing two movie copies. After the third copy of movie has been loaded, the copy can then be made online for viewing. Table 4 illustrates the data placement with RMD after the third copy has been added. Consequently, users can incrementally add I/O capacity, I/O bandwidth, and the number of replicas without affecting existing services. This feature is particularly useful for VOD applications. Clearly, this feature is not available in the chained declustering method. The capability of supporting streams by the mirrored declustering in the presence of disk failures is shown in Table 5, where the throughput of each disk is denoted by λ. It can be seen that the performance of RMD is superior to that of the mirrored declustering method in the presence of disk failures. By allowing customers to dynamically add or remove movie copies, as well as providing good fault tolerance and load balancing, RMD is deemed a viable approach to supporting movie placement in a disk-array-based video server.

3 Replication and placement

In a VOD system, a video server usually stores hundreds or even thousands of movies. The demand for the movies, however, is usually highly skewed. Newly released movies are likely to attract most of the viewers, while older movies receive very few requests. It has been shown that the access frequency of movie rentals in a particular week in video stores can be characterized by a Zipf distribution function [1]. Figure 2 shows the corresponding Zipf distribution function with parameter 0.2716. It is interesting to see that the hottest (most popular) 10 movies receive about 50% of the total viewing request in the week surveyed. To meet the high demand for the hot movies and to guard against disk failures, hot movies are usually replicated across multiple disk arrays.

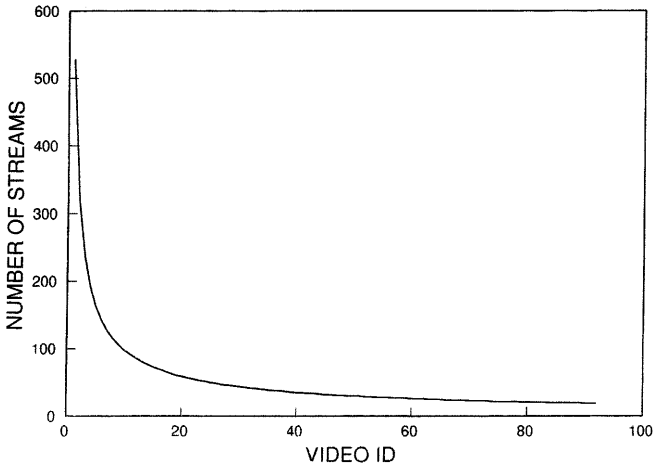


Fig. 2. A Zipf distribution function

Table 5. The capability of supporting multiple streams in the presence of disk failures by mirrored declustering and RMD

	r = 2	r=3	
	single failure	single failure	double failures
mirrored	$n\lambda$	$2n\lambda$	$n\lambda$
RMD	$(2n-1)\lambda$	$(3n-1)\lambda$	$(3n-2)\lambda$

3.1 Replication threshold

For a given disk array with N disks, the number of streams that it can support concurrently is N^*R_{max}, where R_{max} is maximum access rate per disk. If the demand for a movie is greater than N^*R_{max}, the movie will need to be replicated. Even if the demand for a movie is less than N^*R_{max}, it may still be a good business decision to replicate a hot movie to guard against disk failures. Without replication, a disk failure may result in the interruption of all ongoing movies and the loss of revenue. However, full replication of both hot and cold movies is not a good choice either, because this will significantly increase the cost of setting up a video server. To systematically study the effectiveness and efficiency of movie replication, we introduce a new concept of *replication threshold*.

The replication threshold for a movie is defined to be the percentage of total disk array bandwidth (access capacity) allocated to the movie. It is used to determine how many copies of a movie need to be stored in a video server in order to meet the demand. If a movie requires more bandwidth of a disk array to satisfy viewers' demand than it is allocated to, the movie will need to be replicated on multiple disk arrays. For example, if the replication threshold is picked to be 50% (0.5), a movie requiring less than 50% of a disk array bandwidth will not be replicated and a movie requiring more than 50% but less than 100% will be replicated once. Similarly, a movie requiring 125% of the disk array's bandwidth will have three copies stored in three different disk arrays.

3.2 Data placement policies

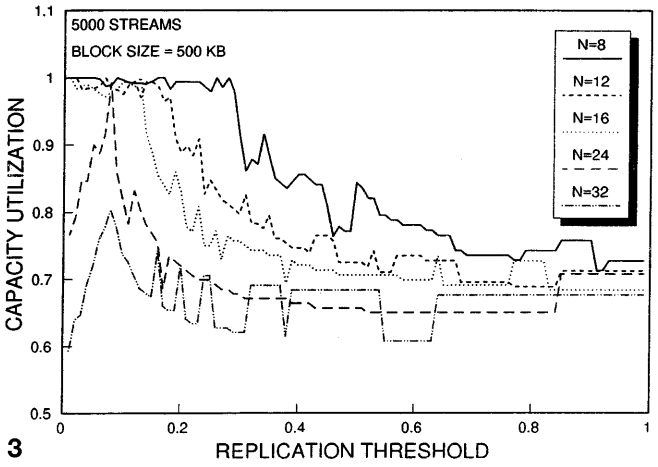
Given a replication threshold, the maximum number of video streams supported (bandwidth or access capacity) by a disk array, and the number of movies with the associated access frequency, the number of disk arrays required for a video server can be determined. After the number of disk arrays and thus the number of disks is determined, the next step is to decide movie placement, i.e., how to decide the order of which movie to be selected first and where (in which disk array) the movie is placed. We consider three policies to determine the order of movie selection: *sequential*, *alternate*, and *random*.

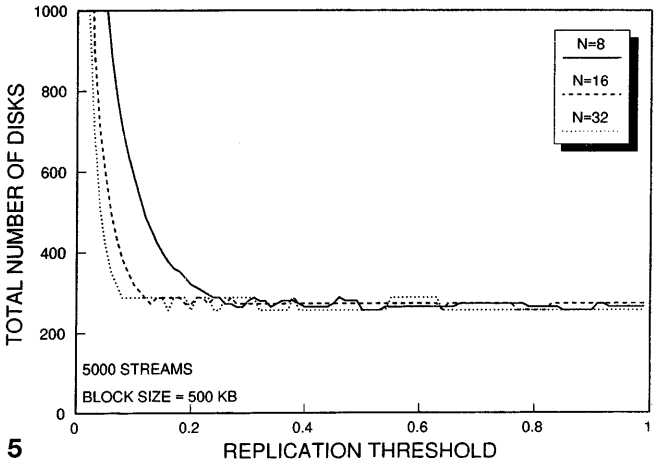
With sequential policy, the hottest movie is selected first and placed onto disk arrays. After all copies of the movie have been placed, the hottest movie in the remaining movies is selected and placed onto the disk array with sufficient capacity, in terms of both bandwidth and disk space. The selection process continues until all movies have been placed onto disks. With alternate policy, hottest and coldest movies are selected and placed onto disk arrays alternately. With random policy, movies are selected at random. When a movie is selected, it is likely that more than one disk array is available for storing the movie copy. There are two well-known schemes for picking a disk array for the movie: *first fit* and *best fit*. With the first-fit scheme, the first disk array having sufficient bandwidth and disk capacity for the movie is used. One variation of the first-fit scheme is the so-called *circular first fit* or *next fit*. In this scheme, the disk array last selected is used as the starting point for finding the next available one. For the best fit scheme, the criterion could be in terms of disk capacity or disk bandwidth. Studies have shown that the best fit scheme will result in fragmentation and is in general not a good choice. The circular first-fit scheme will thus be used in our performance study.

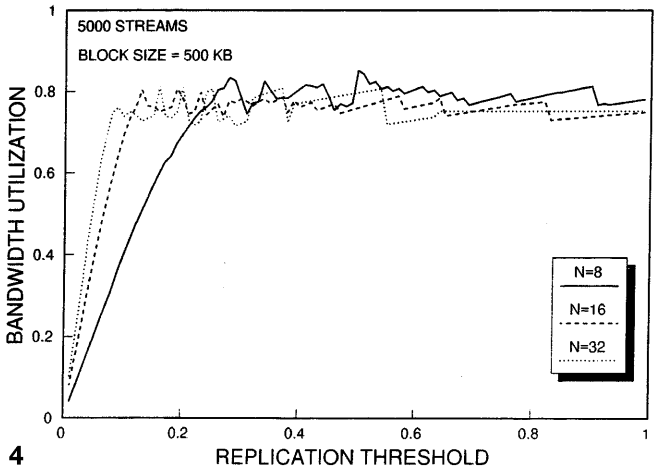
The movie selection and placement procedure used in our performance study can be summarized as follows. A replication threshold is first chosen. Next, a movie is selected, based on each of the three movie selection schemes described above. The number of copies to be stored is then computed, using replication threshold and the access frequency of the movie. Finally, all copies of the selected movie are placed to disk array one by one, using the first-fit, or best fit, algorithm. The movie selection and placement process continues until all movies have been placed on disks. In the case of disk failure, the workload on the failed disk is redirected to another disk array if possible. Since RMD rotates the placement of data partitions across different disk

Table 6. Parameters used

Parameter	Notation	Value
disk array size	N	
all concurrent video streams	L	
video streams for video i	L_i	
total number of videos	M	92
parameter for Zipf distribution	θ	0.271
video length	T	120 min
compressed video data rate	R_b	2 Mb/s
disk capacity	C_d	1 GB
disk read overhead	t_{oh}	25 ms
data block size	D	500 KB
peak disk throughput	R_d	10 MB/s
access rate for a video	R_s	
number of video copies	C	
replication threshold	R_{th}	
max. access rate per disk	R_{max}	







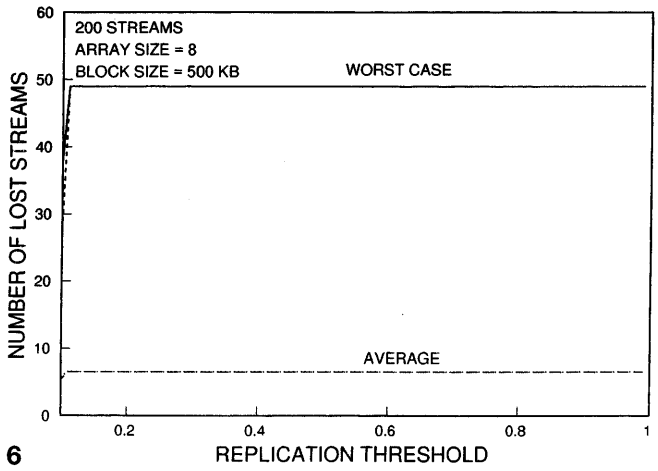


Fig. 3. Average disk capacity utilization vs replication threshold

Fig. 4. Average bandwidth utilization vs replication threshold

Fig. 5. Total number of disks used vs replication threshold

Fig. 6. Number of lost streams vs replication threshold

arrays, its workload redirection procedure after disk failure is essentially the same as that for chained declustering [10]. The redirection process under the mirrored declustering works as follows. When a disk in a disk array fails, the replicated movie with the lowest access frequency (least number of replicas) on the failed array is selected first and the access of that movie to the failed disk is redistributed among all operational replicas. Then, the replicated movie with the lowest access frequency among the remaining movies is selected and its workload redistributed. If a disk array has reached its bandwidth limit, the redistribution process skips that disk array even if a replica exists on the array. Since a hot movie in general has more replicas than a cold movie, finding a disk array storing the same replica and having sufficient bandwidth is easier for a hot movie than for a cold one. In view of this, our workload redistribution process for the mirrored declustering starts with the coldest movie.

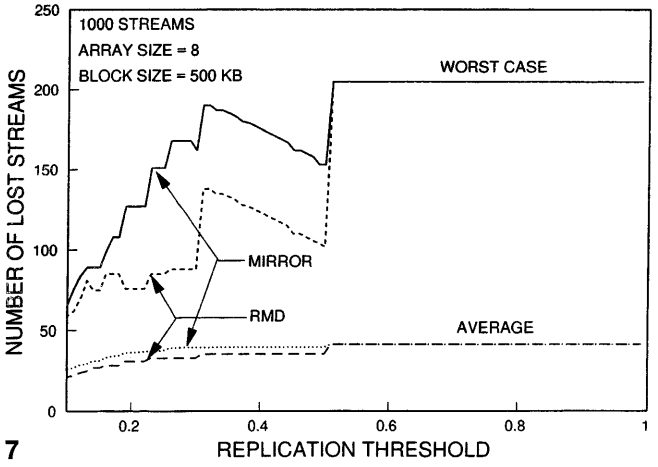
4 Performance results

4.1 System model

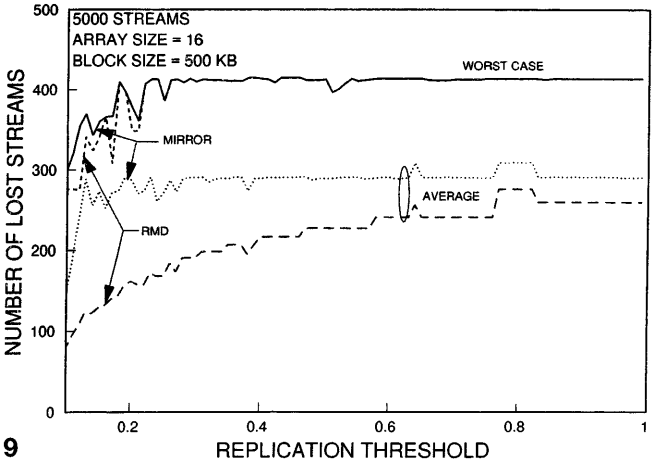
(1) Video order selection

Denote the total number of simultaneous video streams to be carried by the system as L and the total number of videos as M. The number of simultaneous video streams needed to be supported for each video is $L_i = \frac{L*c}{i^{1-\theta}}$, where $c = \frac{1}{\sum_{i=1}^{M}(1/i^{1-\theta})}$ and thus $\sum_{i=1}^{M} L_i = L$. The length of each movie, denoted by T, is assumed to be constant, and the average bit rate is R_b. Hence, the total size occupied by each movie equals $T \times R_b$. Each disk in a disk array has a capacity of C_d. Therefore, a disk array of size N can accommodate up to NC_d/TR_b movies.

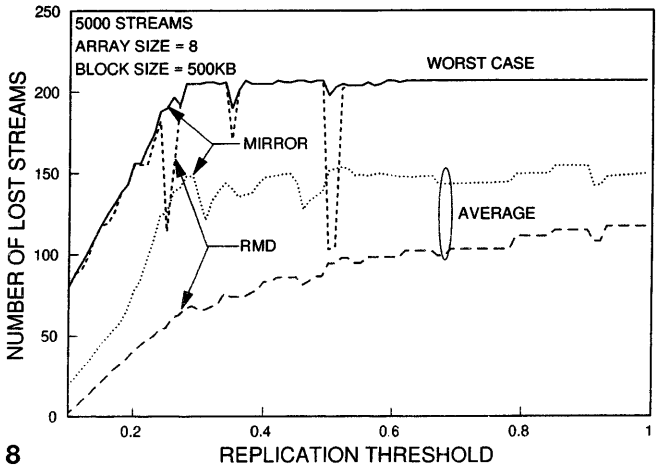
We assume the peak disk bandwidth is R_d, and the overhead, including the disk seek and rotation time associated with each read, is t_{oh}. The maximum access rate per disk equals $R_max = \frac{1}{t_{oh}+\frac{D}{R_d}}$, where D is the data block size on a disk. The access rate required for supporting a video stream is $R_s = R_b/D$.



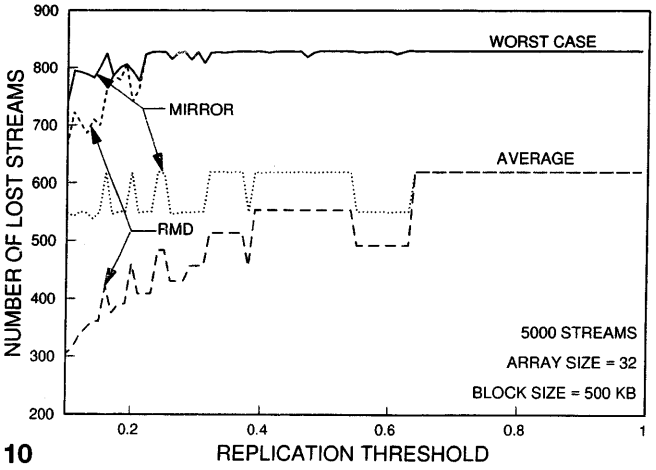
7



9



8



10

Fig. 7. Number of lost streams vs replication threshold

Fig. 8. Number of lost streams vs replication threshold

Fig. 9. Number of lost streams vs replication threshold

Fig. 10. Number of lost streams vs replication threshold

(2) Calculating the number of replicas

The number of copies, C_i for the i^{th} video is determined by the replication threshold, R_{th}, in such a way that $C_i = \lceil \frac{L_i}{R_{th} R_{max}} \rceil$. It can be seen from this equation that a movie will be replicated more times for either a lower replication threshold or a smaller value of R_{max}.

(3) Data placement

For each disk array, the data placement has to observe the following two criteria so that both the maximum available capacity and the maximum bandwidth of each array are not exceeded: $|H_i| \leq \frac{NC_d}{TR_b}$, where H_i is the set of videos that have been placed on the i^{th} disk array, while the operator $|\cdot|$ is the size of the set. Then, $\sum_{j \epsilon H_i} \frac{L_j}{C_j} \leq \frac{NR_{max}}{R_s}$.

4.2 Fault tolerance model

The fault tolerance of each data placement strategy is evaluated by computing the number of lost streams, both worst case and average, when one of the disks in a disk array has failed.

(1) Mirrored declustering

Assuming that the number of lost streams is D_i when a disk in the i^{th} array fails, the number of lost streams in the worst case is $D_{max} = max\{D_i | i = 1, \ldots, U\}$, where U is the total number of arrays used. The average number of lost streams is $E[D] = \frac{1}{U} \sum_{i=1}^{U} D_i$.

(2) RMD

A video stream cannot be supported by the system upon a disk failure, either when there are no backup copies on other disk arrays, or when those disk arrays containing other copies do not have sufficient bandwidth. Specifically, the requirement for the increased traffic on the i^{th} disk array has to observe $\sum_j \epsilon H_i \frac{L_j}{C_j} + \sum_k \epsilon G_i Mk \leq \frac{NR_{max}}{R_s}$, where G_i is the set of the videos that are distributed to the i^{th} disk and M_k is the the number of video streams of the k^{th} video that are allocated to the i^{th} disk array.

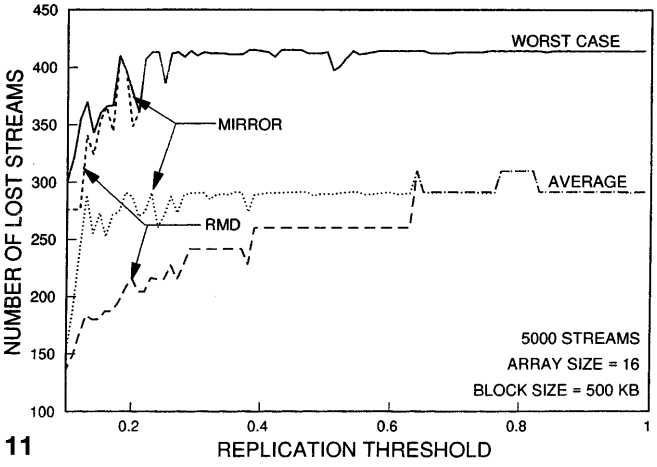
4.3 Numerical examples

Here, several cases are evaluated to illustrate the system and fault tolerance models derived in the previous section. The parameters used are summarized in Table 6. Using the results derived in the previous section, the average disk capacity utilization is shown as a function of the replication threshold, R_{th}, in Fig. 3. The capacity utilization is in general better for smaller R_{th}. For each array size, maximum capacity utilization is reached when R_{th} is below a threshold, T_R. With the parameters used in this study, the maximum. capacity utilization reaches unity for a disk array size less than 16 and R_{th} less than T_R. Figure 3 indicates that T_R is smaller for an array with larger array size. For example, as shown in Fig. 3, T_R is 0.28, 0.17, and 0.14 for N equal to 8, 12, and 16, respectively. Furthermore, the capacity utilization no longer stays flat when the array size is larger than 16 and R_{th} is less than T_R. This is because the large bandwidth needed by the most popular videos requires a large number of disk arrays for very small R_{th}. These arrays could not be filled up when the array size is larger than a certain value (16 in this case), since distinct replicas have to reside on different disk arrays. This situation is more serious for even smaller values of R_{th}, since the capacity utilization decreases drastically after a maximum is reached.

The bandwidth utilization is shown in Fig. 4 as a function of the replication threshold. Similar to the previous figure, there exists a threshold value in R_{th} (i.e., approximately 0.2 in this case), above which point the bandwidth utilization saturates. The maximum achievable bandwidth utilization is independent of the value of R_{th}. The number of disks used for three disk array configurations of different array size is shown in Fig. 5. The total number of disks used for storing the same amount of videos and supporting the same amount of simultaneous video streams increases drastically when R_{th} decreases and becomes less than the threshold point.

We note that the number of disks used is not sensitive to the disk array size except when the value of R_{th} is smaller than the threshold. This is due to the fact that the video server is bandwidth-limited, not capacity-limited. Hence, the additional replicas needed for smaller array sizes only increases the capacity utilization (Fig. 3), but not the number of disks (Fig. 5).

The fault tolerance of the video server for various traffic loads, array configurations and placement algorithms is illustrated in Figs. 6–10. The fault tolerance is measured for both mirroring and RMD in terms of the lost video streams in the presence of a single disk failure. Three types of traffic loads are compared: light (200 streams), medium (1000 streams), and heavy (5000 streams) traffic. Figure 6 shows that, when the traffic load is light, conventional mirroring technique performs roughly the same as the RMD scheme in terms of the number of lost streams when one disk of a disk array fails. This is because very few videos are replicated at this traffic load. At medium traffic load (Fig. 7), the fault tolerance of the two placement schemes begin to show a difference for a replication threshold smaller than 0.5. Under heavy load (Figs. 8–10), the difference in fault tolerance between two placement methods is significant. It is evident that the RMD scheme is effective in reducing the number of



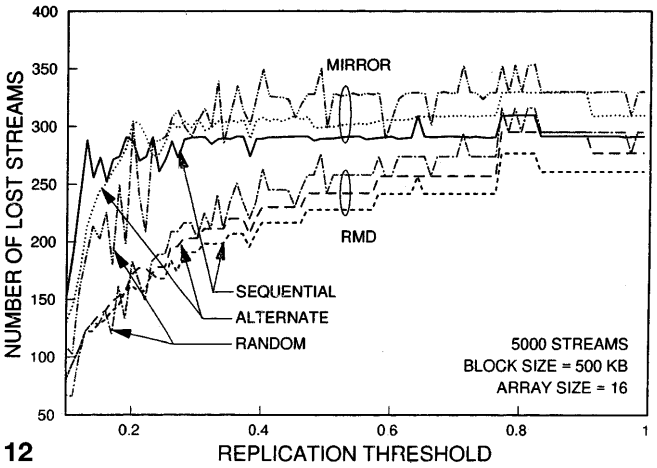


Fig. 11. Number of lost streams vs replication threshold for double disk failures

Fig. 12. Comparison of number of lost streams between three different video placement methods

the lost streams, regardless of the array size. The difference in the number of lost streams between the RMD scheme and the mirroring scheme clearly increases with the array size. This can be explained by the fact that the RMD scheme possesses better fault tolerance only if a video is replicated.

The fault tolerance of the disk array in the presence of two disk failures in a single array is shown in Fig. 11. The difference between the mirror scheme and the RMD scheme becomes smaller as compared to the case of a single disk failure. This is because a single failure will force the streams belonging to the same disk to be redistributed to other disk arrays in the mirroring scheme. Hence, a disk array with additional disk failures does not lose more streams under mirrored declustering.

On the other hand, only those read accesses to the videos with more than three replicas can be redistributed to other arrays using the RMD scheme. This is also an indication of the graceful performance of RMD.

From the results obtained, it can be concluded that the operating range of the replication threshold parameter should be so selected that the maximum fault tolerance is achieved

without using additional disks. The existence of such a value is evident from Fig. 5, as the number of disk used drastically increases when the replication threshold is below 0.28 for an array size of 8, and below 0.14 for an array of size of 16. The disk array is operated in the bandwidth-limited region when the replication threshold is larger than this value, and in the capacity-limited region when the replication threshold is smaller than this value. Even though sequential video selection is used throughout this section, the improvement in fault tolerance is quite insensitive to the video selection process, as shown in Fig. 12.

5 Conclusions

In this paper, we proposed a new data placement method, RMD, to support high data availability for disk arrays. Combining the merits of chained and mirrored declustering methods, RMD is particularly useful for storing multiple movie copies to support VOD applications. To assess the performance of RMD, we have conducted a series of experiments by emulating the storage and supply of movies in a VOD system. Specifically, we determined the number of replicas required for each movie declustering, and then allotted each movie copy to each disk array. We introduced a new parameter, replication threshold, and utilized it to conduct several sensitivity analyses for various replication scenarios. By properly varying the value of the replication threshold, we empirically determined the operating point for a given movie access distribution. In addition, we evaluated three policies for movie selection, hot-to-cold, alternate hot-and-cold and random, to study the robustness of RMD. Our experimental results showed that RMD consistently outperforms the conventional methods in terms of load balancing and fault tolerance capability, and is deemed a viable approach to supporting movie placement in a disk-array-based video server.

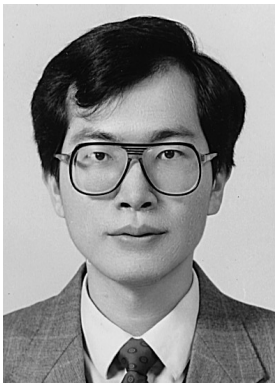
Acknowledgement. Ming-Syan Chen is in part supported by National Science Council, Project No. NSC 87-2213-E-002-009 and Project No. NSC 87-2213-E-002-101, Taiwan, ROC.

References

1. Electronic Engineering Times, March 1993
2. Berson S, Golubchik L, Muntz RR (1995) Fault-tolerant design of multimedia servers. In: Proceedings of ACM SIGMOD, pp 3664–375
3. Burkhardt WA, Menon J (1993) Disk array storage system reliability. In: Proc. of Symposium of Fault-Tolerant Computing, FTCS-23. pp 432–441
4. Chen MS, Kandlur DD (1996) Stream conversion to support interactive playout of videos in a client station. IEEE Multimedia 3(2): ♣–♣
5. Chen MS, Kandlur DD, Yu PS (1995) Storage and retrieval methods to support fully interactive playout in a disk-array-based video server. Multimedia Syst 3(3): 126–135
6. Chen MS, Lee EK, Gibson GA, Katz RH, Patterson DA (1994) RAID: high-performance, reliable secondary storage. ACM Comput Surv 26(2): 145–185
7. Copeland G, Keller T (1989) A comparison of high-availability media recovery techniques. In: Proceedings of ACM SIGMOD, Portland, OR, pp 98–109
8. Dan A, Sitaram D (1995) Online video placement policy based on bandwidth to space. In: Proceedings of ACM SIGMOD, pp 376–385
9. Golubchik L, Lui JCS, Muntz RR (1992) Chained declustering: load balancing and robustness to skew and failure. In: Proceedings of the 2nd International Workshop On Research Issues in Data Engineering: Transaction and Query Processing, Tempe, AZ, pp 89–95
10. Hsiao HI, DeWitt DJ (1990) Chained declustering: a new availability strategy for multiprocessor database machines. In: Proceedings of the 6th Intl Conference on Data Engineering, pp 456–465
11. Kandlur DD, Chen M Shae ZY (1994) Design of a multimedia storage server. In: Proc. IS&T/SPIE Symposium on Electronic Imaging – Conference on high-speed networking and multimedia applications. SPIE
12. Hg RT, Yang J (1994) Maximizing buffer and disk utilizations for news on demand. In: Proceedings of the 20th International Conference on Very Large Databases, pp 451–462
13. Ozden B, Biliris A, Rastogi R, Silberschatz A (1994) A low-cost storage server for movie-on-demand databases. In: Proceedings of the 20th International Conference on Very Large Databases, pp 594–605
14. Rangan PV, Vin HM, Ramanathan S (1992) Designing a multi-user multimedia on-demand service. IEEE Commun Mag 30(7):56–65
15. Vin HM, Rangan PV (1993) Designing a multi-user HDTV storage server. IEEE J Select Areas Commun 11(1): 153–164

MING-SYAN CHENG received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan in 1982, and the M.S. and Ph.D. degrees in Computer, Information and Control Engineering from The University of Michigan, Ann Arbor, MI, in 1985 and 1988 respectively. Dr. Chen is a Professor in Electrical Engineering Department, National Taiwan University. His research interests include database systems, multimedia technologies, and internet software. He had been a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, NY from 1988 to 1996, primarily involved in projects related to parallel databases, multimedia systems, and data mining. Dr. Chen is an editor for IEEE Transaction on Knowledge and Data Engineering. He has published over 70 refereed international journal/conference papers in his research areas, and more than 30 of them appeared in ACM and IEEE journals/transactions. Dr. Chen served as a guest co-editor for IEEE Transaction on Knowledge and Data Engineering on a special issue for data mining in 1996. He is an inventor of many international patents in the areas of interactive video playout, video server design, interconnection network, and concurrency and coherency control protocols. He received the Outstanding Innovation Award from IBM in 1994 for his contribution to parallel transaction design for a major database product, and numerous awards for his inventions and patent applications. Dr. Chen is a member of Association for Computing Machinery.

HUI-I HSIAO joined IBM T.J. Watson Research Center as a research staff member in September 1990. Dr. Hsiao is working on a digital library project in IBM Almaden Research. Before September 1997, he was the manager of the Parallel Databases project at the IBM T.J. Watson Research Center. Working with IBM Toronto, his group was responsible for research, design, and development of DB2 Parallel Edition – IBM's first highly scalable parallel database system on open system platform. His research interest includes parallel database architecture, parallel query and transaction processing strategies, and database performance analysis. Prior to joining IBM, he worked as a software engineer at Nicolet Instrument Corporation where he received a Nicolet Associate Fellow Award. He was a member of the Gamma database machine project at the University of Wisconsin. Dr. Hsiao received a B.S. degree from National Taiwan University and M.S. and Ph.D. degrees in computer science from University of Wisconsin at Madison.



CHUNG-SHENG LI received his B.S.E.E. degree from National Taiwan University, Taiwan, R.O.C. in 1984, and the M.S. and Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley in 1989 and 1991, respectively. He has joined the computer science division of IBM T.J. Watson Research Center as a research staff member since Sept. 1991. His research interests include digital library, optical chip interconnects, optoelectronics, all-optical networks, broadband switching architectures, and high-speed analog/digital VLSI circuit design. He has co-initiated several research activities in IBM on fast tunable receiver for all-optical networks and content-based retrieval in the compressed domain for large image/video databases. He is currently the co-investigator of a satellite image database project funded by NASA. Dr. Li has received a Research Division award from IBM in 1995 for his major contribution to the tunable receiver design for WDMA, and numerous invention and patent application awards. He is serving as the technical editor and feature editor for the IEEE Communication Magazine. He has authored or coauthored more than 60 journal and conference papers and received one of the best paper awards from the IEEE International Conference on Computer Design in 1992. He is a senior member of the IEEE Laser Electro-Optic Society, the Communication Society, the Computer Society, and the Circuit and System Society.

PHILIP S. YU received the B.S. degree in E.E. from National Taiwan University, Taipei, Taiwan, Republic of China, in 1972, the M.S. and Ph.D. degrees in E.E. from Stanford University, in 1976 and 1978, respectively, and the M.B.A. degree from New York University in 1982. Since 1978 he has been with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY. Currently he is manager of the software tools and technique group. His current research interests include database systems, Internet applications, data mining, multimedia systems, transaction and query processing, parallel and distributed systems, disk arrays, computer architecture, performance modelling, and workload analysis. He has published more than 200 papers in refereed journals and conferences, and over 140 research reports and 90 invention disclosures. He holds or has applied for 50 US patents. Dr. Yu is an ACM and IEEE fellow. He was an editor of IEEE Transactions on Knowledge and Data Engineering. In addition to serving as program committee members on various conferences, he has served as the program chair of the 2nd Intl. Workshop on Research Issues on Data Engineering: Transaction and Query Processing and the program co-chair of the 11th Intl. Conference on Data Engineering. He has received several IBM and external honors including Best Paper Award, IBM Outstanding Innovation Awards, Outstanding Technical Achievement Award, Research Division Award, and 19 Invention Achievement Awards.