Proceedings of Multimedia Computing and Networking 1997 (MMCN97), San Jose, CA, February 1997

A Fault Tolerant Video Server Using Combined Raid 5 and Mirroring

Ernst W. BIERSACK, Christoph BERNHARDT
Institut Eurécom, 2229 Route des Crêtes,
06904 Sophia-Antipolis — France
Phone: +33 4 93002611
FAX: +33 4 93002627
email: erbi@eurecom.fr

ABSTRACT

Video servers must use large disk arrays to provide the huge amount of storage capacity and bandwidth needed. As the number of disk drives increases, the probability of a video server failure increases too. We propose a redundancy scheme that uses both RAID 5 techniques and mirroring to make a video server tolerant against all single disk failures. Our approach provides a unified framework for the use of RAID 5 and mirroring to achieve fault tolerance at the lowest additional cost possible, while guaranteeing 100% service availability even when operating with a failed disk.

Keywords: Video server, disk storage, reliability, fault-tolerance, RAID.

1. INTRODUCTION

In this paper, we propose a fault tolerant storage configuration for video servers. Many new multimedia applications will be based on video servers storing a possibly large number of videos and concurrently serving 1000s of users. Due to the nature of stored video, already moderately large video servers will have large amounts of storage. Most of this storage will be disk based and disk arrays are believed to be the main storage component of video servers.

Combining many disks, possibly hundreds of them, into disk arrays is challenging with respect to the reliability and availability of the data stored on the disks. Even though modern disks have *Mean Times To Failure* (MTTF) of several hundred thousands of hours, a disk array will have a much lower mean time to failure, defined as the time until **any** of the disks of the array fails. Therefore, disk arrays must be built with failure tolerance in mind and RAID [2] are implementing fault tolerance for disk arrays.

The application of RAID for disk arrays in video servers is not straight forward due to the real-time requirements of clients in a video server. The typical client in a video server reads large files in a sequential fashion. It is of utmost importance that the client's requests are serviced according to deadlines imposed by the continuous nature of the video data even if the video server is operating with a failed disk. The RAID organizations suitable for video servers can only guarantee this continued operation if they are operated at only 50% utilization in perfect condition since a disk failure will in general double the bandwidth needs with an unchanged workload.

We propose a method that combines a standard RAID 5 disk array organization with mirroring (replication) of the most popular videos. By exploiting the expected biased request distribution for videos stored on a video server, we can guarantee service for all clients of the server even with a failed disk. The overhead for our scheme is clearly less than the 50% required for a plain RAID 5 array.

2. Assumptions Concerning the Configuration of a Video Server

2.1 Striping

If the I/O bandwidth of single disk is not sufficient anymore to service the required number of clients, a number of disk can be aggregated into a so called *disk array.* Theoretically, the I/O capacity of a disk array scales linearly with the number of disks in the array. For practical consideration, the available I/O bandwidth of a disk array mainly depends on how the data is distributed over the disks of an array.

The easiest way of taking advantage of the increased I/O capacity of multiple disks is to simply replicate the data, so that all disks store exactly the same information. Using this method, the retrieval of information from an array is only slightly more complicated than in the single disk case. Only when a new client starts up, a decision must be made which disk has enough I/O bandwidth available to service a new client. The client is then assigned to any of these available disks, and continues as in the single disk case. The number of clients that can be supported scales linearly with the number of disks. However, such a scheme wastes large amount of disk space by extensively replicating information.

Other schemes try to avoid wasting storage by using *striping* to distribute the data. Data to be stored on the disk array is partitioned into equal-sized blocks called *striping blocks*. Each of these striping blocks is stored entirely on one of the disks of the array. Subsequent striping blocks are stored on *subsequent*[1] disks of the array. A number of striping blocks that spans all disks of the array exactly once is called a *striping group*. The striping blocks used to compute the parity block needed to reconstruct a missing striping block together with that parity block form a *parity group.*

The I/O performance of a disk array depends largely on the choice of the striping block and the request sizes for data stored on the array. We will only consider read-performance which has a large impact on the operation of a video server that is used in applications like VOD. There are two basic types of striped disk arrays, *byte(bit)-interleaved* and *block-interleaved* arrays.

Bit/Byte-interleaved disk arrays employ a striping block size of 1 bit or 1 byte, respectively. Consequently, the size of a striping group is rather small and therefore, almost every access to the disk array will access all disks concurrently to retrieve the requested data. This organization allows to have a very large bandwidth for a single requests, that scales essentially with the number of disks in the array. On the other hand care must be taken to keep the actual disk accesses as efficient as possible and to reduce seek and latency overhead. The later requires that a block retrieved from a particular disk must have a certain size, preferably close to the data stored on a single cylinder of the disk. For a video server build upon such an array this implies very large request sizes and thus large buffers, since for each client the data buffer that must be provided is proportional to the number of disks and the size of the data block retrieved from an individual disk.

In block-interleaved arrays the situation is different. If the striping block size is chosen close to the amount of data that fits on a cylinder of a disk, good disk efficiency can be reached for striping block sized requests. If data is requested in such blocks, only one disk is used to satisfy the request provided that the offset of the block is chosen to be on a striping block boundary. Unlike for the bit/byte-interleaved array, a single request will not see the accumulated bandwidth of all disks in the array. However, this is also not necessary, since the bandwidth needed for a single client is typically much smaller than the I/O bandwidth of even a single disk. The big savings is in the buffer that is required. Since each client only accesses a single disk at any time, the number of disks in the array does not directly influence the amount of buffering necessary. Instead, every client will need one or two striping block sized buffers, depending on the disk scheduling used.

Considering the strong and weak points of both array organizations, the block-interleaved array is much better suited for the application in a video server.

2.2 Wide Striping

Our video server is assumed to use *wide striping* to store videos on its disks, where a video is striped over **all** available disks on a video server. Such a configuration has the inherent advantage of balancing the load equally over all disks independent of the request distribution. This is contrary to so called *narrow striping* where the disks of a video server are partitioned

[1] Two disks are defined to be “subsequent” according to a certain ordering criterion.

into *disk groups*. In narrow striped video servers, all disk groups are independent, i.e. a video that is stored on a disk group is striped only over the disks belonging to it. As a consequence, different disk groups can experience a different load during operation of the video server because the total popularity of videos stored on each disk group might differ.

Our video server will operate with wide striping and a parity group size that equals the number of disks in the array minus a single parity disk for RAID 5. A disadvantage of a wide striped array is that it can only tolerate a single disk failure before it loses data. This is usually not a problem with a small arrays (about 10 disks), but the larger the number of disks in the array, the higher the probability of a double failure. When using wide striping, we assume that we only have to deal with single disk failure, i.e. a failed disk is replaced before a second disk can fail. Later in the paper, we will discuss how to survive multiple disk failures in large disk arrays by making the parity group size smaller.

2.3 Request Distribution

In the following, we assume that the popularity distribution of the videos stored on a video server follows Zipf's law. We also assume that we know the popularity of a video before it is stored on disk.

The Zipf distribution with parameter ϕ is computed as follows: p_i the probability that the video v_i is chosen is $c/i^{1-\phi}$, where $c = 1/H_{n_{video}}^{(1-\phi)}$ and $H_N^{(s)} = \sum_{i=1}^{N} i^{-s}$ is the N-th Harmonic Number of order s [4]. The parameter ϕ can be varied between 0 and 1 and determines how large the bias of the resulting distribution is. Small ϕ lead to a larger bias, for $\phi = 1$, the distribution is equal to the uniform distribution.

We assume without loss of generality that the videos are ordered with respect to their popularity, so that $f_{Zipf}(i, \phi)$ is the probability that a new client request references video i A small value for ϕ results in a more biased distribution, i.e. most requests are for only a small subset of all videos. A value of $\phi = 1$ results in a uniform distribution where all videos are equally popular. Most real world video servers are assumed to experience a request distribution that follows best a Zipf distribution with a parameter ϕ much smaller than 1 and experiments showed that $\phi = 0.271$ is a good value [1].

2.4 Storage-Limited vs. Bandwidth-Limited Video Servers

Two critical resources in a video server are *storage volume* and the *I/O bandwidth* for accessing the storage. The former decides how many different videos can be stored on a video server, the latter determines how many independent videos streams can be serviced by a video server[2].

If we assume that we only have disk storage in our video server, both resources are tightly coupled to each other over the number and type of disk drives used. The storage consists of a possibly large number of disk drives and each drive comes with a certain storage capacity adding to the storage volume of the server and with a certain I/O bandwidth adding to the total I/O bandwidth of the server.

A video server is defined to be

- *storage-limited* if the number of disk drives that make up its storage is determined by the number of videos that must be stored on the server.
- *bandwidth-limited* if the number of drives is determined by the number of video streams that must be supported concurrently.

As a special case, a video server can be balanced with respect to its storage volume and its I/O capacity, if the number of disks required to store the demanded number of videos is the same as the number of disks needed to provide the I/O bandwidth for the number of concurrent video streams that the server must deliver.In general, a video server will not be balanced, thus it has either spare storage volume or spare I/O capacity.

[2] We focus on independent video streams instead of clients, since using different service types possibly many clients can be serviced by the same video stream (broadcast, or Near Video On Demand [5]). Therefore, only the number of individual streams directly depends on the available storage I/O bandwidth. The number of clients on the other hand also depends on the type of service that is provided to the clients.

We will see later that our modified RAID scheme adapts to both cases and will yield a cost-effective video server configuration.

3. COMBINED RAID 5 & MIRRORING (R5M)

3.1 Parity vs. Replication

To protect against failure, one can trade-off additional I/O bandwidth against additional storage volume. RAID 5 requires a small amount of additional storage volume for each video to protect against failure. But in failure mode, the required I/O bandwidth for **each client** will double.

For popular videos requested by many clients it is also possible to entirely replicate (mirror) the video, thereby doubling the **storage volume** required for the video**.** This allows to avoid that the I/O bandwidth will double in case of failure for **each client** requesting this video [3]. For the following discussion, we will use the parameters in Table 1 that are chosen similar to the parameters used in [10].

Table 1: Symbols used

Parameter		Default Value
$n_{clients}$	Number of concurrent clients	500
n_{videos}	number of videos stored on the server	100
r_{disk}	effective bandwidth of a single disk[a]	32 Mbit/s
r_{video}	playout rate of a video	4 Mbit/s
τ_{video}	duration of a video	2 h
n_{disks}	number of disks required without any redundancy	
$n_{totalDisks}$	total number of disks for a given threshold	
n_{RAID}	additional number of disks for RAID functionality	1
n_{REPL}	additional number of disks for replication	
r_{RAID}	additional disk bandwidth required for RAID storage in failure mode	
r_{REPL}	additional disk bandwidth required for replicated storage in failure mode	
V_{disk}	storage size of a single disk	4000 MByte
ϕ	parameter of the Zipf distribution	0.271
$f_{Zipf}(i, \phi)$	popularity of video i for a Zipf distribution with parameter ϕ	
n_{hot}	number of clients that request hot movies	
n_{cold}	number of clients requesting cold movies	

a. by effective bandwidth we denote the bandwidth that results after deducting all possible kinds of overhead incurred during the retrieval of data blocks from the raw data rate of the disk, i.e. the number of possible clients can be easily computed by dividing the effective bandwidth by the bandwidth of a single video.

3.2 Parity combined with Replication (RAID 5 & Mirroring)

In the following we assume that the popularity distribution of the videos stored on a video server follows Zipf's law. We also assume that we know the popularity of a video before it is stored on disk. With this knowledge we will introduce two classes of videos, the so called *hot* and *cold* classes. Videos are classified to fall into one of these classes and stored according to their classification. Our scheme, where videos in the *cold* class are stored in standard RAID 5 fashion, whereas *hot* videos are mirrored, is called **R5M**. All videos stored on our video server are equal with respect to their size and playout rate.We assume without loss of generality that videos are ordered with respect to their popularity with the most popular video having the index 1 and the least popular having the highest index. We define a threshold h separating *hot* from *cold* videos, i.e. all videos with indices $1, \ldots, h$ are put into the *hot* class, all remaining into the *cold* class.

Next, we want to know, looking at a video server running long enough to have close to its maximum number of concurrent clients[3], how many clients are watching videos in each of the two classes. For a given threshold h and a given popularity distribution of the videos (we use again Zipf's distribution) (EQ 1) and (EQ 2) give the expected number of clients in each class for a fully loaded video server:

$$n_{hot} = \left(\sum_{i=1}^{h} f_{Zipf}(i, \phi) \right) n_{clients} \qquad \text{(EQ 1)}$$

$$n_{cold} = \left(\sum_{i=h+1}^{n_{videos}} f_{Zipf}(i, \phi) \right) n_{clients} \quad . \qquad \text{(EQ 2)}$$

In our scheme R5M, we propose different redundancy measures for the two classes of videos. For the videos in the *cold* class, we still propose the standard RAID 5 parity scheme. On the other hand, for *hot* videos, we propose to do full replication of the videos.

In case of a disk failure, the reconstruction process will differ for clients watching videos in different classes, when they try to access data from the failed disk. Clients watching a *cold* video must have their data reconstructed using reverse parity computation, requiring the retrieval of the full striping group that the missing block belongs to. As before for a standard RAID 5, this will double the load for **each client** watching a ***cold*** video for each disk in the array.

For the retrieval of *hot* videos on the other hand, one will rely on **replicated** information as replacement for the lost data on the failed disk. For this scheme to work, each striping block belonging to a *hot* video is replicated and stored on another disk of the array. Care must be taken, that the replicates for blocks of a specific disk are not stored on one single disk, but are distributed in a random fashion over all remaining disks of the array. If this was not the case, the fault of one disk would immediately overload the disk storing the replicated blocks for the failed disk. If replicated blocks are randomly distributed the load increase for reading replicates will be uniformly spread over all surviving disks of the array. Algorithms for pseudo-random distributions of replicated blocks can be found in [10].

Both storage schemes, RAID 5 and replication, require additional storage and, in failure mode, additional I/O bandwidth that we will now compute.

We will start by computing the number of disks that are required without providing any fault tolerance, thus without having redundancy:

$$n_{disks} = \max \left\{ \overbrace{\left\lceil \frac{n_{videos} \cdot r_{video} \cdot \tau_{video} \cdot 3600}{8 \cdot V_{disk}} \right\rceil}^{\text{storage}} , \underbrace{\left\lceil \frac{n_{clients} \cdot r_{video}}{r_{disk}} \right\rceil}_{\text{bandwidth}} \right\} \qquad \text{(EQ 3)}$$

[3] We are not interested in the transient phase after a video server becomes operational. We are only focusing on its operation close to its capacity limit, since this is where resource bottlenecks become noticable.

The two terms in (EQ 3) correspond to the two resources provided by disk storage: storage and I/O bandwidth. A video server can be either *storage limited* or *bandwidth limited* depending on the characteristics of disks, videos, and the number of concurrent clients. For our example parameters the storage term amounts to a total of 80 disks, whereas the bandwidth term requires a total of 63 disks. Thus, our example server is clearly storage limited and $n_{disks} = 80$.

In a next step, we will compute the additional storage required for both redundancy schemes. In the RAID 5 case, only one additional parity disk is added, thus $n_{RAID} = 1$ for now. The computation of the additional storage due to replication is more complicated, since we must take the threshold h into account, which directly determines the number of videos that must be replicated:

$$n_{REPL} = \frac{h \cdot r_{video} \cdot \tau_{video} \cdot 3600}{8 \cdot V_{disk}} \quad \text{(EQ 4)}$$

We now consider the increased I/O bandwidth that is required if the array is operating in failure mode. The bandwidth required for *cold* videos doubles in failure mode. The additional bandwidth is thus

$$r_{RAID} = n_{cold} \cdot r_{video} \quad \text{(EQ 5)}$$

The additional bandwidth required for *hot* movies on the other hand is what is required to read the replicated blocks for *hot* clients accessing the failed disk. Given that the ratio between clients in the *hot* and clients in the *cold* class that access the same disk is the same for all disks in the array, we can compute the additional bandwidth required for replicated videos in the failure case

$$r_{REPL} = \frac{n_{hot}}{n_{clients}} r_{disk} \quad \text{(EQ 6)}$$

To compute the total number of disks that is required for a given threshold h, we must consider whether our video server is bandwidth limited or storage limited for the threshold, thus

$$n_{totalDisks} = \max\left(\left\lceil \frac{n_{videos} \cdot r_{video} \cdot \tau_{video} \cdot 3600}{8 \cdot V_{disk}} + n_{RAID} + n_{REPL} \right\rceil, \left\lceil \frac{n_{clients} \cdot r_{video} + (r_{RAID} + r_{REPL})}{r_{disk}} \right\rceil\right) \quad \text{(EQ 7)}$$

In order to make a storage solution cost efficient, the number of disks should be minimal. In Fig. 1, the number of disks depending on the threshold h is shown (for different Zipf parameters). The number of disk reaches a minimum at h_{opt}, where the resulting fault tolerant server changes from being bandwidth limited (left of h_{opt}) to being storage limited (right of h_{opt}). For less biased distributions ($\phi = 0.271, 0.5$), h_{opt} is larger than in the more biased case ($\phi = 0$). Moreover, the number of disks that is required at h_{opt} is larger for larger ϕ. Both effects were to be expected. With a less biased distribution clients are more uniformly distributed, i.e. to have many clients use replicated videos in failure mode, the threshold must be chosen larger compared to a more strongly biased distribution. A larger threshold results in more replicated videos, and since we know that at h_{opt} the server is both storage and bandwidth limited, the number of total disks must be larger as well. In Fig. 2, we show again the curves for the number of disks that is required for a fault tolerant server configuration. The upper two curves ($n_{clients} = 1000, \phi = 0, 0.271$) depict server configurations that are bandwidth limited in their non-fault tolerant version. The lower curves ($n_{clients} = 500, \phi = 0, 0.271$) show server configurations, that were originally storage limited. Each single curve consists of two distinct segments, one extending from $h = 0$ to h_{opt}, the other from h_{opt} to the highest possible threshold ($h = 100$). If h is chosen smaller than h_{opt} the resulting fault tolerant server will be bandwidth limited, because only very few videos are replicated and in case of a failure many clients will need double I/O bandwidth since they use the RAID 5 mechanism to restore the data from the missing disk. If h_{opt} is chosen, the resulting server is balanced, i.e. it is bandwidth and at the same time storage limited. Finally, a h larger than h_{opt} results in a storage limited server, where many videos are replicated, even such videos that are only rarely requested. It can be noted that the number of disks required increases linearly in this second segment. This is simply due to the fact that the server is already storage limited and with each additional replication the necessary storage is increased by the amount needed to store an additional video.

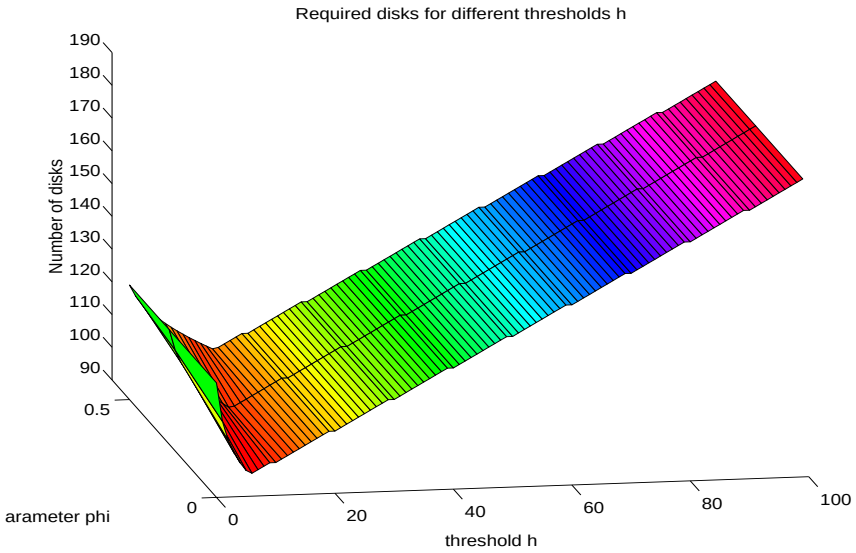

Figure 1. Number of disks required for different thresholds (100 videos, 500 clients, $\phi = 0, 0.271, 0.5$)

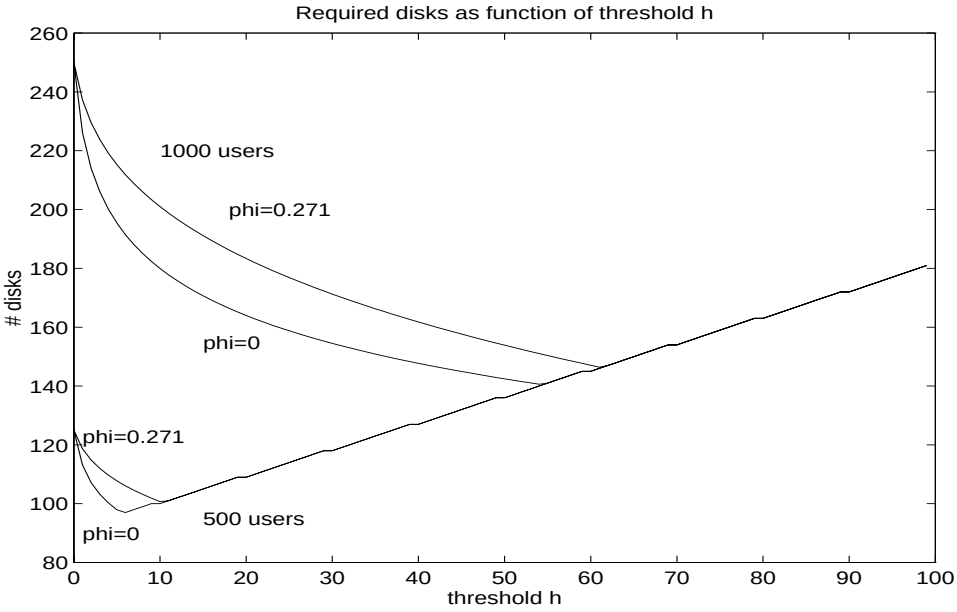

Figure 2. Required disks for storage/bandwidth limited server configurations ($n_{videos} = 100$)

.

Table 2: Overhead of different redundancy schemes

$n_{clients}$	$n_{totalDisks}$							
	no redundancy		**RAID 5** $h = 0$		**R5M** $h = h_{opt}$		**Mirror** $h = 100$	
	#	**%**	**#**	**%**	**#**	**%**	**#**	**%**
500	90	100	125	139	97	108	180	200
1000	125	100	250	200	141	113	180	144

The figure shows also the influence of different values for ϕ. Larger values result in a larger optimal threshold and thus more overall disks. This is due to the fact that for larger values of ϕ more videos must be replicated to convert a given amount of clients from accessing RAID storage to accessing replicated storage in case of a failure. The larger ϕ, the smaller the slope of the curves for values of $h < h_{opt}$, because the savings in bandwidth reduction due to the replication is smaller. Therefore the first, bandwidth limited, segment of the curves falls slower with h and intersects later with the second, storage depending, segment. Since this second segment only depends on h and is independent from the request distribution and thus from ϕ, the number of total disks required for an optimal server configuration increases with larger values ϕ.

3.3 R5M for Different Server Configurations

This section will show how R5M adapts the optimal threshold h_{opt} to different server configurations with respect to the number of clients that the server can support and the number of videos that are stored on the server.

Fig. 3 shows the number of disks required for a given number of videos and users for a standard, non-fault tolerant server and for the fault tolerant server employing R5M. For the non-fault tolerant server (Fig. 3a), the plane shows a clear bend

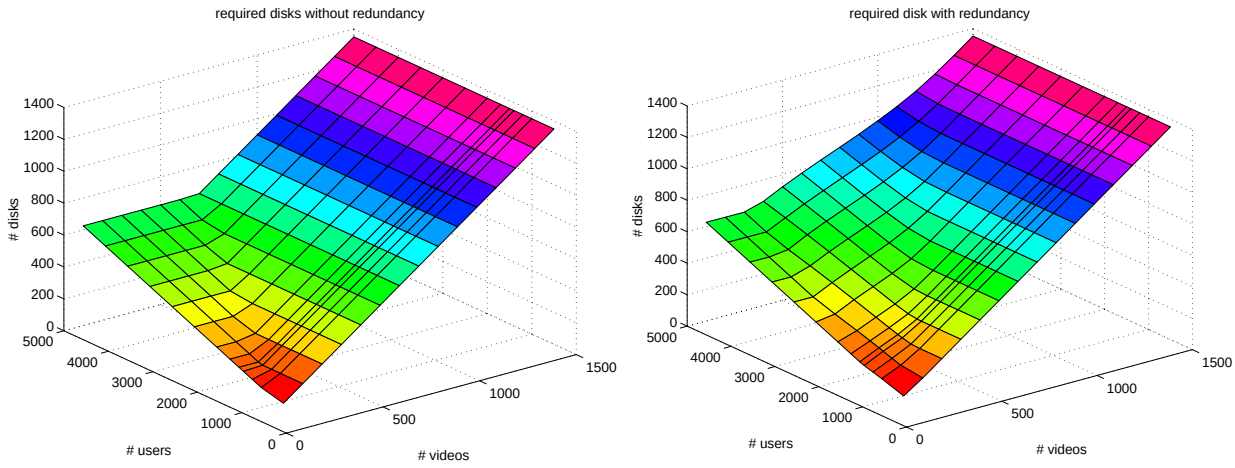

Figure 3. Total number of disk required for a standard (a) server and a fault tolerant (b) server as function of the number of clients and the number of users ($\phi = 0.271$)

where the server changes from being bandwidth limited to being storage limited. On the bend itself, the server is balanced and (EQ 8) describes how the bend extends:

$$\frac{n_{clients} r_{video}}{r_{disk}} - \frac{n_{videos} r_{video} \tau_{video} 3600}{V_{disk}} = 0 \quad . \qquad \text{(EQ 8)}$$

Fig. 4 shows the ratio of videos that are replicated and the resulting additional costs in terms of additional disks that must

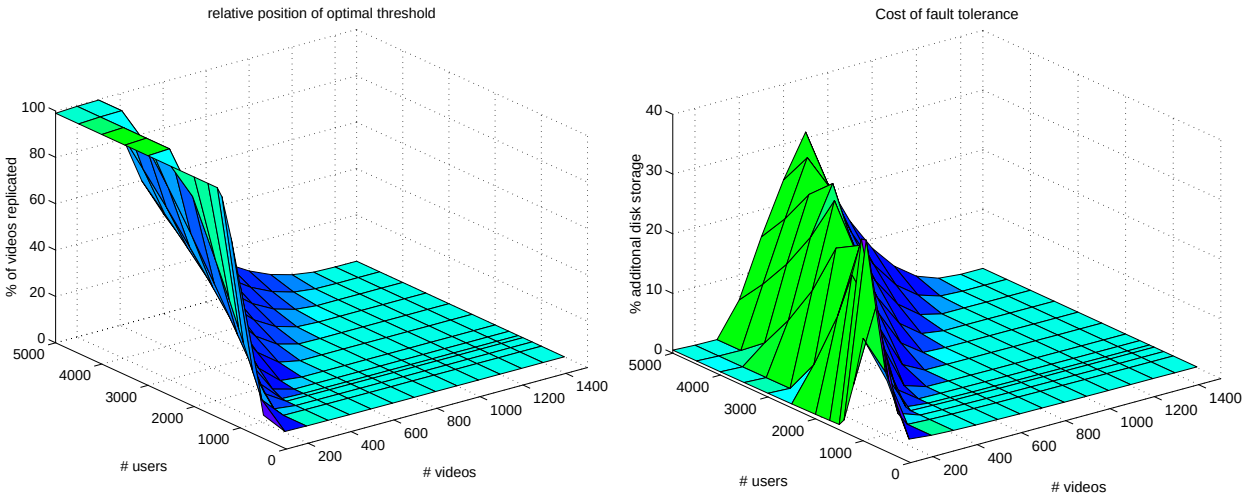

Figure 4. a) optimal threshold as h_{opt}/n_{video} ; b) cost of fault tolerance in terms of add. disks

be added. Fig. 4a shows that replication is done mainly for configurations that are originally bandwidth limited. For highly unbalanced, bandwidth limited configurations all videos are replicated. This is possible without incurring the high costs of additional disks, because a highly bandwidth limited server has plenty of spare storage space. The replication ratio decreases gradually for configurations that are more balanced. For configurations that are extremely storage limited the ratio reaches 0% since here the server can provide enough spare I/O bandwidth to have all clients use RAID 5 in the failure case.

The additional cost that our R5M incurs can be expressed as the additional number of disks that are required to make the server fault tolerant. Fig. 4b shows the percentage of disks (compared to the non-fault tolerant server) that must be added. It is obvious that the highest costs are incurred for server configurations that were originally balanced (the bend in Fig. 3a) as there is neither spare I/O bandwidth for RAID 5 clients, nor spare storage capacity to replicate videos readily available. Instead, additional disk drives must be added immediately if the server is made fault tolerant. For extremely bandwidth or storage limited servers no additional disks need to be added to make them fault tolerant, as there is either enough spare storage to replicate all videos or enough spare bandwidth to have all users use RAID 5 in the case of a disk failure.

3.4 Double Disk Failures

For large disk configurations with hundreds of disks, the probability of double disk failures increases significantly. One way to address this problem is to introduce a striping group size that is smaller than the overall number of disks resulting in several smaller striping groups within the disk array. Videos are still stored over all disks of the array [10], so the advantages of wide striping are preserved. On the other hand, the array can now sustain double disk failures as long as they occur in different striping groups.

The downside of a smaller striping group size is that the amount of storage needed to store parity information is increased as there is an additional parity disk for **each** striping group. For administrative purposes, the total number of disks must be a multiple of the striping group size resulting in some additional disks needed to fill up the last striping group of the disk array.

The resulting total amount of disk drives is shown in Fig. 5 for two different scenarios. In Fig. 5a, a video server configuration for 500 users and 100 videos is shown (c.f. Fig. 1) and in Fig. 5b the configuration is for 5000 users and 1000 videos for parity group sizes of 10, 20, 50, and 100. As expected, the number of disks at h_{opt} is larger than for a single striping group. The decision whether to use smaller striping groups instead of a single group must be made based on trading-off avail-

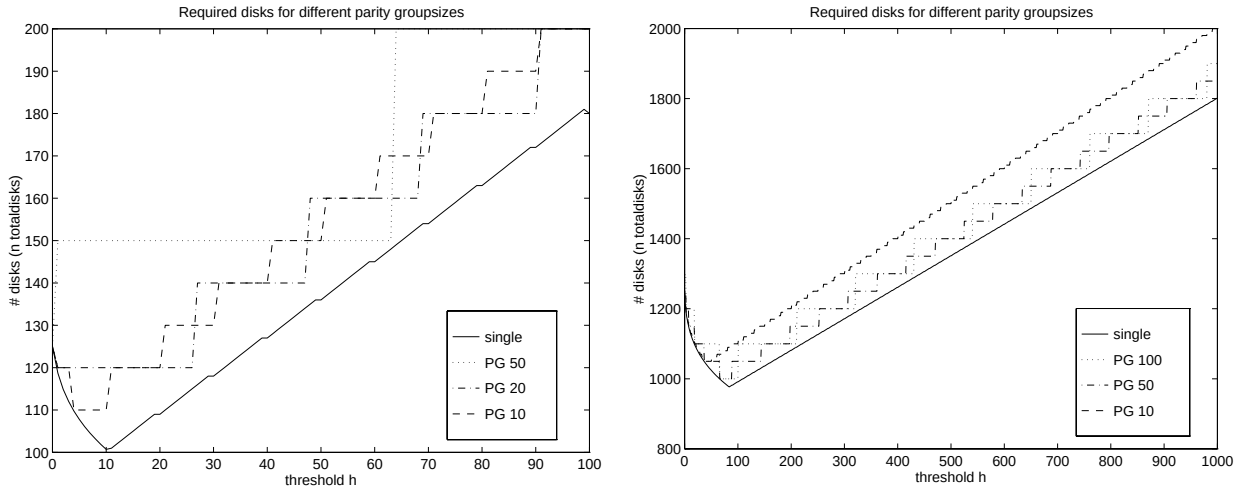

Figure 5. Required Disks for different parity group size ($\phi = 0.271$) a) parity group sizes (10,20,50); b) parity group sizes (100,50,20,10)

ability requirements versus the cost of multiple groups. One can also see that the size of a striping group must be reasonably related to the total number of disks to avoid the staircase effect if the striping group size is too large. In Fig. 5a, a striping group size of 50 requires a comparably large number of disks just to fill up the last striping group.

Table 3: Storage overhead and reliability for different parity group sizes

		parity group size				
Configuration		**10**	**20**	**50**	**100**	**single PG**
500 users 100 videos	**#disks/%**	110/110%	120/120%	150/150%	-	100/100%
	MTTL in h	$7.58 \cdot 10^6$	$3.29 \cdot 10^6$	$1.02 \cdot 10^6$		$7.58 \cdot 10^5$
5000 users 1000 videos	**#disks/%**	1050/107%	1020/104%	1000/102%	1000/102%	977/100%
	MTTL in h	$7.94 \cdot 10^5$	$3.87 \cdot 10^5$	$1.53 \cdot 10^5$	$7.58 \cdot 10^4$	$7.87 \cdot 10^3$

Table 3 summarizes the effect of different sized striping groups on our scheme. It also gives values for the *Mean Time To Data Loss* MTTL, i.e. that a double disk failure occurs, of the resulting disk array. Smaller parity group sizes increase the MTTL by one or more orders of magnitude for less than 10% additional disks.

4. RELATED WORKS

R5M offers deterministic service guarantees even when operating in failure mode. R5M keeps the additional number of disks that must be added low by combining RAID 5 and replication and by considering the client request behavior. In the context of video servers, we are the first to propose the combined use of RAID 5 and mirroring to assure the timely *retrieval* of video data under disk failure and to compute the *cost optimal combination of mirroring and parity* for a given popularity distribution.

Combined replication and parity in the context of *file* servers has been recently proposed by Wilkes [11]: For active data, two copies are held in the upper level of the hierarchy, for inactive data a single copy with RAID 5 parity protection is held in the lower level of the hierarchy. The migration between the two levels is automatic and is based whether or not a file is *write*-active.

In the context of video servers, reliability under failure has been addressed before: Mourad [6] assures 100% service continuation under disk failure by using full mirroring to achieve fault tolerance. Therefore, the storage volume required doubles. To reduce the costs of mirroring, he proposes an optimized disk layout taking into account Zone Recording used in modern disk drives. He does not exploit the request distribution for the stored material and he does not investigate how the costs for mirroring depend on the configuration of a video server (storage vs. bandwidth limited).

Tewari et al. [10] describe a so called *software* RAID that is comparable to our wide stripe configuration. Apart from disk failures, they also consider node failures in a clustered video server consisting of several server nodes. The authors' work is based on earlier work investigating a clustered video server [9] where they use queuing models to evaluate the performance of their server. The same modelling seems to be used for [10]. The service model is therefore inherently probabilistic as opposed to our deterministic model. The authors develop an analytical model that allows to investigate the effects of different parameters for their software RAID. They optimize with respect to loss probability versus achievable disk utilization. The resulting performance numbers indicate that they do not achieve a superior disk utilization compared to our scheme. Moreover, additional memory buffer must be provided to achieve acceptable values for the utilization increasing the total cost of the server. The authors do not exploit the expected request distribution for stored material.

Chen et al. [1] propose a replication scheme for narrow stripe video servers. They define a replication threshold that determines whether a given movie will be replicated or not. Whenever a video needs a share of the I/O capacity of a single disk array of the video server larger than the threshold, it is replicated. This choice of the threshold addresses mainly load balancing problems in narrow stripe video servers. The authors do not consider other forms of redundancy for non-replicated videos. Thus, users watching such videos will experience loss of service. The number of lost users is between 5% and 20%. The disk bandwidth utilization that can be achieved is about 80% which is comparable to our scheme. On the other hand, disk space is wasted due to the narrow stripe organization of the server. Especially, for high replication thresholds this under-utilization is substantial (70%). For lower thresholds the storage utilization improves. However, due to the higher number of replications more disk storage is required.

Tetzlaff and Flynn [8] describe a replication scheme that improves load balancing and availability in *narrow stripe* video servers. Their main focus is again on load balancing effects of replication. Their service model is inherently probabilistic as opposed to our deterministic model. They disqualify wide striping due to the effect a single disk failure has in such a scenario. They don't consider other schemes, such as RAID, to make wide striping fault tolerant.

Ozden [7] presents two approaches using RAID to reconstruct the missing data under disk failure: (i) r*eservation of a contingency I/O bandwidth* to reconstruct the lost striping block by reading all the remaining blocks in the parity group and (ii) *prefetching* all the striping blocks in a parity group to reconstruct the lost striping block. While (i) requires additional I/O bandwidth (ii) requires additional RAM buffer.

In Table 4, we summarize the features of the different reliability schemes. R5M is the only scheme to exploit both replication and RAID methods to establish fault tolerance without any client loss. The only two schemes that guarantee uninterrupted service for all clients are the ones proposed by Mourad and by Ozden.

Table 4: Comparison of different reliability schemes

Scheme	Striping	Utilization	RAID 5	Replication	100% service
R5M	wide	~80%	yes	yes	yes
Mourad	wide		no	yes	yes
Tewari	wide	~80%	yes	no	no
Chen	narrow	~80%	no	yes	no
Tetzlaff	narrow	~90%	no	yes	no
Ozden	wide		yes	no	yes

5. CONCLUSION

Video servers based on large disk arrays suffer from the same reliability problems as disk arrays in legacy applications like file or database servers. Even though individual disks might well have MTTF of several 100000 hours, a disk array with a large number of disks has a MTTF that is much lower. RAID organizations try to overcome the problem of disk failures by storing redundant information in form of parity along with the data. When a disk fails the parity information together with the information on the surviving disks can be used to reconstruct the missing data.

The disadvantage of RAID organization, like RAID 5, is that the available I/O bandwidth is reduced by 50% during operation with a failed disk. In a video server application, this is unacceptable since the available I/O bandwidth directly influences the number of concurrent clients that a video server can service. Therefore, new array organizations must be found that improve the bandwidth available from a disk array operating with a failed disk.

We proposed a scheme, called R5M, that combines standard RAID 5 parity organization with replication of very popular video material to yield a cost-effective video server that can sustain its full load of clients even when it is operating with a failed disk. We showed how our scheme adapts to different configurations concerning the number of clients and the number of videos that a server can support. To determine how many and which videos are to be replicated, we were relying on the fact that the popularity distribution of the individual videos stored on the server follows a Zipf distribution. Should this assumption give rise to concern, our scheme can be made conservative by choosing a conservative parameter for the Zipf distribution, effectively underestimating the bias of the distribution.

6. REFERENCES

[1] M.-S. Chen et al. Using rotational mirrored declustering for replica placement in a disk-array-based video server. In *Proc. 3rd ACM Conference on Multimedia*, pages 121–130, San Francisco, CA, Nov. 1995.

[2] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. Raid: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.

[3] M. Holland, G. Gibson, and D. Siewiorek. Architectures and algorithms for on-line failure recovery in redundant disk arrays. *Journal of Distributed and Parallel Databases*, 2(3), July 1994.

[4] D. E. Knuth. *The Art of Computer Programming - Sorting and Searching*, volume 3, chapter 6, pages 397–399. Addision-Wesley, Menlo Park, CA, 1973.

[5] T. D. C. Little et al. A digital on-demand video service supporting conten-based queries. In *Proc. 1st ACM International Conference on Multimedia*, Anaheim, CA, August 1993.

[6] A. Mourad. Reliable disk striping in video-on-demand servers. In *Proceedings of the 2nd IASTED/ISMM International Conference Distributed Multimedia Systems and Applications*, pages 113–118, Stanford, CA, August 1995.

[7] B. Ozden et al. Fault-tolerant architectures for continuous media servers. In *SIGMOD International Conference on Management of Data 96*, pages 79–90, June 1996.

[8] W. Tetzlaff and R. Flynn. Block allocation in video servers for availability and throughput. In *Proceedings of IS&T/SPIE Symposium on Multimedia Computing and Networking*, volume 2667, San Jose, CA, Jan. 1996.

[9] R. Tewari, D. M. Dias, W. Kish, and H. Vin. Design and performance tradeoffs in clustered video servers. In *Proceedings IEEE International Conference on Multimedia Computing and Systems (ICMCS'96)*, pages 144–150, Hiroshima, June 1996.

[10] R. Tewari, D. M. Dias, W. Kish, and H. Vin. High availability for clustered multimedia servers. In *Proceedings of International Conference on Data Engineering*, New Orleans, LA, February 1996.

[11] J. Wilkes et al. The HP Autoraid hierarchical storage system. *ACM Trans. Comput. Syst.*, 14(1), Feb. 1996.