Modeling and Performance Comparison of Reliability Strategies for Distributed Video Servers

Jamel Gafsi, Ernst W. Biersack
email: (gafsi,erbi)@eurecom.fr, Fax: 0033.4.93.00.26.27
Institut EURECOM, 2229 Route des Crêtes
06904 Sophia Antipolis Cedex, FRANCE

October 18, 1999

Abstract

Large scale video servers are typically based on disk arrays that comprise multiple nodes and many hard disks. Due to the large number of components, disk arrays are susceptible to disk and node failures that can affect the server reliability. Therefore, fault-tolerance must be already addressed in the design of the video server. For fault-tolerance, we consider parity-based as well as mirroring-based techniques with various distribution granularities of the redundant data. We identify several reliability schemes and compare them in terms of the server reliability and per stream cost. To compute the server reliability, we use continuous time Markov chains that are evaluated using the SHARPE software package. Our study covers independent disk failures and dependent component failures. We propose a new mirroring scheme called Grouped One-to-One scheme that achieves the highest reliability among all schemes considered. The results of this paper indicate that dividing the server into independent groups achieves the best compromise between the server reliability and the cost per stream. We further find that the smaller the group size, the better the trade-off between a high server reliability and a low per stream cost.

Keywords: Distributed Video Servers, Reliability Modeling, Markov Chains, SHARPE, Performance and Cost Analysis

1 Introduction

1.1 Server Design Issues

Many multimedia applications such as online news, interactive television, and video-on-demand require large video servers that are capable of transmitting video data to thousands of users. In contrary to traditional file systems, video servers are subject to real-time constraints that impact the storage, retrieval, and delivery of video data. Furthermore, video servers must support very high disk bandwidth for data retrieval in order to serve a large number of video streams simultaneously. The most attractive approach for implementing a video server relies on *disk arrays* that (i) achieve high I/O performance and high storage capacity, (ii) can gradually grow in size, and (iii) are very cost efficient. Unfortunately, large disk arrays are vulnerable to disk failures, which results in poor reliability for the video server. A challenging task is therefore to design video servers that provide not only high performance but also high reliability.

The video server considered in this paper is composed of many disk arrays, which are also referred to as **server nodes**. Each server node comprises a set of magnetic disk drives as illustrated in Figure 1 and is directly attached to the network. A video to store is divided into many blocks and the blocks are distributed among *all* disks and

nodes of the video server in a round robin fashion. All server nodes are identical, each node containing the same number of disks D_n. The total number of server disks D is then $D = N \cdot D_n$, where N denotes the number of server nodes.

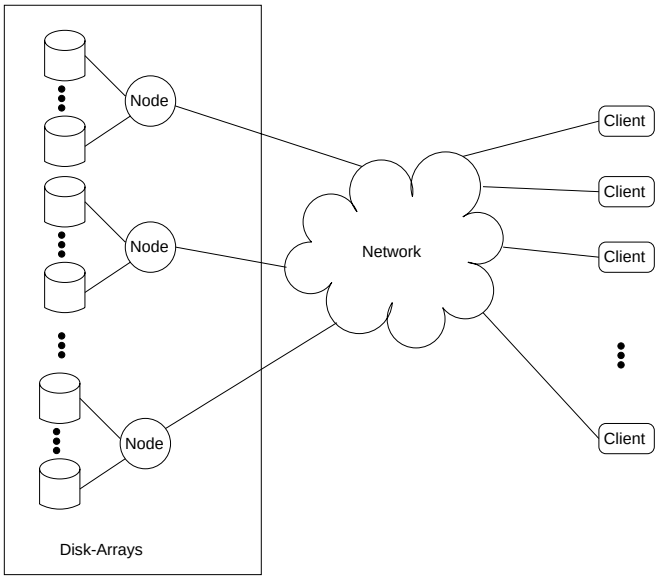


Figure 1: Video Server Architecture

A client that consumes a video from the server is connected to *all* server nodes and is served once every time interval called the **service round**. A video block that is received at the client during service round i is consumed during service round $i+1$. Further, since the transfer rate of a single disk is much higher than the stream playback rate, each disk can serve many streams during one service round. In order to efficiently retrieve multiple blocks from a disk during one service round, the video server applies the well known SCAN algorithm that optimizes the seek overhead by reordering the service of the requests.

A very important decision in a video server design concerns the way data is distributed (striped) over its disks. To avoid hot spots, each video is partitioned into video blocks that are distributed over *all* disks of the server as already mentioned. Based on the way data are retrieved from disks, the literature distinguishes between the Fine-Grained (FGS) and the Coarse-Grained (CGS) Striping algorithms. FGS retrieves for one stream (client) multiple, typically small, blocks from many disks during a *single* service round. A typical example of FGS is RAID3 as defined by Katz et al. [1]. Derivations of FGS include the streaming RAID of Tobagi et al. [2], the staggered-group scheme of Muntz et al. [3], and the configuration planner scheme of Ghandeharizadeh et al. [4]. The main drawback of FGS is that it suffers from large buffer requirements that are proportional to the number of disks in the server [5, 6]. CGS retrieves for one stream *one large* video block from a *single* disk during a single service round. During the next service round, the *next* video block is retrieved from possibly a different disk. RAID5 is the classical example of CGS. Oezden et al. [7, 6] showed that CGS results in higher throughput than FGS for the same amount of resources (see also Vin et al. [8], Beadle et al. [9], and our contribution [5]). Accordingly, we will adopt CGS to store original video data on the video server.

The paper is organized as follows. Section 2 classifies reliability schemes based on (i) whether mirroring or parity is used and (ii) the distribution granularity of *redundant* data. Related work is discussed at the end of the section. Section 3 studies reliability modeling for the reliability schemes considered. The reliability modeling is based on Continuous Time Markov Chains (**CTMC**) and concerns the case of independent disk failures as well as the case of dependent component failures. We focus in section 4 on the server performance, where we compare the per stream cost for the different reliability schemes. Section 5 emphasizes the trade-off between the server reliability

and the per stream cost and studies the effect of varying the group size on the server reliability and the per stream cost. The results of section 5 lead to the conclusions of this paper, which are presented in section 6.

1.2 Our Contribution

In the context of video servers, reliability has been addressed previously either by applying parity-based techniques (RAID2–6), e.g. [10, 8, 6, 11, 5], or by applying mirroring-based techniques (RAID1), e.g. [12, 13, 14]. However, all of the following aspects have not been considered together:

- Comparison of several parity-based and mirroring-based techniques under consideration of both, the video server performance and cost issues. Our cost analysis concerns the storage and the buffering costs to achieve a given server throughput.
- Reliability modeling based on the distribution granularity of redundant data in order to evaluate the server reliability for each scheme considered. We will perform a detailed reliability modeling that incorporates the case of independent disk failures and the case of dependent component failures.
- Performance, cost, and reliability trade-offs of different parity-based as well as mirroring-based techniques. We will study the effect of varying the group size on the server reliability and the per stream cost and determine the best value of the group size for each technique.

2 Video Server Reliability

We use CGS to store/retrieve *original* video blocks, since it outperforms FGS in terms of the server throughput. Adding fault-tolerance within a video server implies the storage of *redundant (replicated/parity)* blocks. What remains to be decided is how *redundant* data is going to be stored/retrieved on/from the server. For mirroring, we limit ourselves to *interleaved declustering* schemes [15], where original data and replicated data are spread over all disks of the server. For parity, we limit ourselves to RAID5-like schemes, where parity blocks are evenly distributed over all server disks. We retain these schemes since they distribute the load uniformly among all server components. Additionally, we consider the case where only original blocks of a video are used during normal operation mode. During disk failure mode, replicated/parity blocks are needed to reconstruct lost original blocks that are stored on the failed component.

Mirroring (also called *RAID 1* [16, 17]) consists in storing *copies* of the original data on the server disks. The main disadvantage of mirroring schemes is the 100% storage overhead due to storing a complete copy of the original data. Reliability based on parity consists in storing *parity* data in addition to original data (RAID2-6). When a disk failure occurs, parity blocks together with the remaining original data are used to reconstruct failed original blocks. The RAID5 [18] parity scheme requires one parity block for each $(D-1)$ original blocks, where D is the total number of server disks. The $(D-1)$ original blocks and the one parity block constitute a **parity group**. Although the additional storage volume is small for parity-based reliability, the server needs additional resources in terms of I/O bandwidth or main memory when working in disk failure mode. In fact, in the worst case, the whole *parity group* must be retrieved and temporarily kept in the buffer to reconstruct a lost block. In [5], we have distinguished between the *second read strategy* and the *buffering strategy*. The second read strategy doubles the I/O bandwidth requirement [19], whereas the buffering strategy increases the buffer requirement as compared to the failure free mode. We will restrict our discussion to the buffering strategy, since it achieves about twice as much throughput as the second read strategy [5, 20]. We will see that the buffering strategy becomes more attractive in terms of server performance (lower buffer requirements) and also regarding the server reliability when the size of the parity group decreases.

2.1 Classification of Reliability Schemes

Reliability schemes differ in the technique (parity/mirroring) and in the *distribution granularity* of redundant data. We define below the distribution granularity of redundant data.

- For the parity technique, the distribution granularity of redundant data is determined by whether the parity group comprises *All* (D) or *Some* (D_c) disks of the server. For the latter case, we assume that the server is partitioned into independent **groups** and that all groups are the same size, each of them containing D_c disks. Let C denote the number of groups in the server ($C = \frac{D}{D_c}$).
- For the mirroring technique, the distribution granularity of redundant data has two different aspects:
  - The first aspect concerns whether the original blocks of *one* disk are replicated on *One*, *Some* (D_c), or *All remaining* ($D - 1$) disks of the server.
  - The second aspect concerns how a *single original* block is replicated. Two ways are distinguished. The first way replicates *one* original block *entirely* into *one* replicated block [13], which we call **entire block replication**. The second way partitions *one* original block into *many sub-blocks* and stores each sub-block on a *different* disk [14], which we call **sub-block replication**. We will show later on that the distinction between entire block and sub-block replication is decisive in terms of server performance (throughput and per stream cost).

Table 1 classifies mirroring and parity schemes based on their distribution granularity. We use the terms **One-to-One**, **One-to-All**, and **One-to-Some** to describe whether the distribution granularity of redundant data concerns *one* disk (mirroring), *all* disks (mirroring/parity), or *some* disks (mirroring/parity) disks. For the One-to-One scheme, only mirroring is possible, since One-to-One for parity would mean that the size of each parity group equals 2, which consists in replicating each original block (mirroring). Hence the symbol "XXX" in the table.

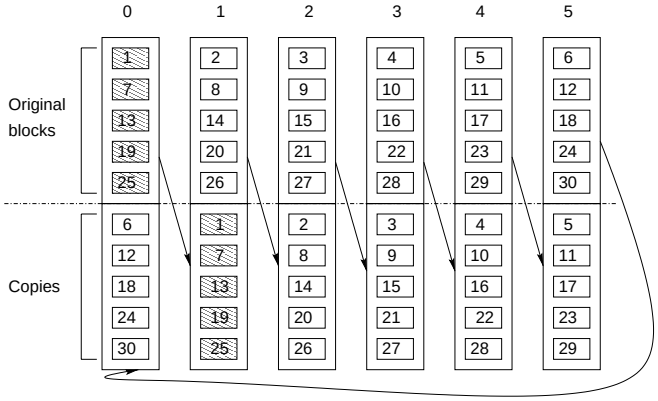
	Mirroring	**Parity**
One-to-One	Chained declustering [21, 22]	XXX
One-to-All	Entire block replication (doubly striped) [13, 23] Sub-block replication [15]	RAID5 with one group [18]
One-to-Some	Entire block replication Sub-block replication [14]	RAID5 with many groups [3, 8, 7]

Table 1: Classification of the different reliability schemes

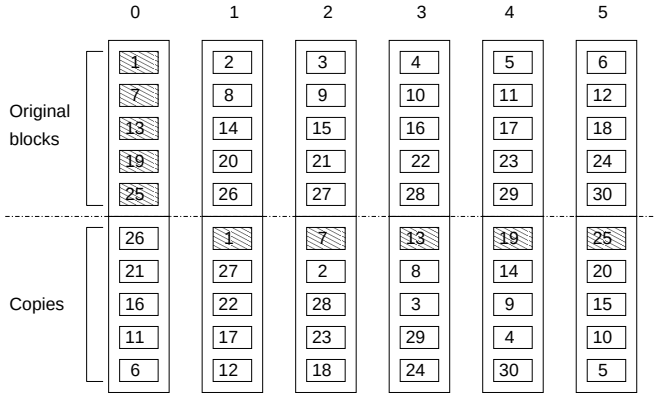
Table 1 distinguishes seven schemes. We will give for each of these schemes an example of the data layout . Thereby, we assume that the video server contains 6 disks and stores a single video. The stored video is assumed to be divided into exactly 30 *original* blocks. All schemes store original blocks in the same order (round robin) starting with disk 0 (Figures 2 and 3). What remains to describe is the storage of redundant data for each of the schemes.

Figure 2 presents examples of the mirroring-based schemes. These schemes have in common that each disk is partitioned into two separate parts, the first part storing original blocks and the second part storing replicated blocks.

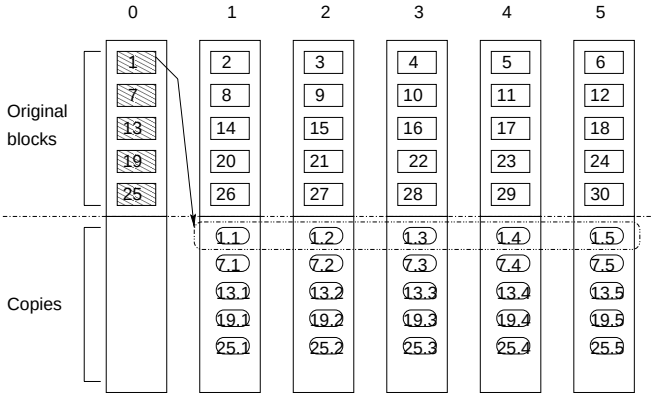
As illustrated in Figure 2(a), the One-to-One mirroring scheme ($Mirr_{one}$) simply replicates original blocks of one disk onto another disk. If one disk fails, its load is *entirely* shifted to its *replicated* disk, which creates load-imbalances within the server (the main drawback of the One-to-One scheme).



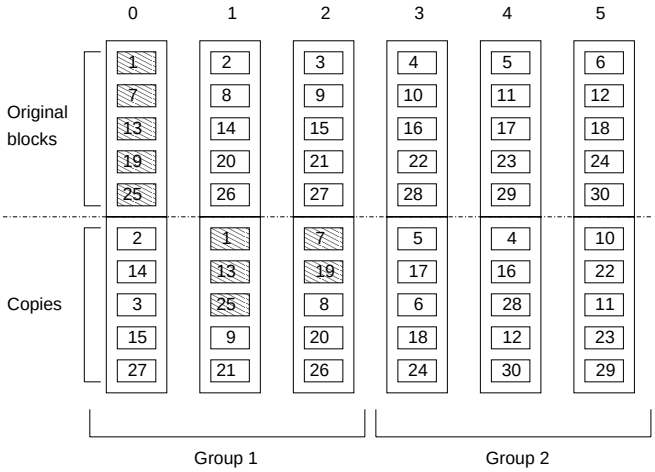
(a) One-to-One Organization $Mirr_{One}$.



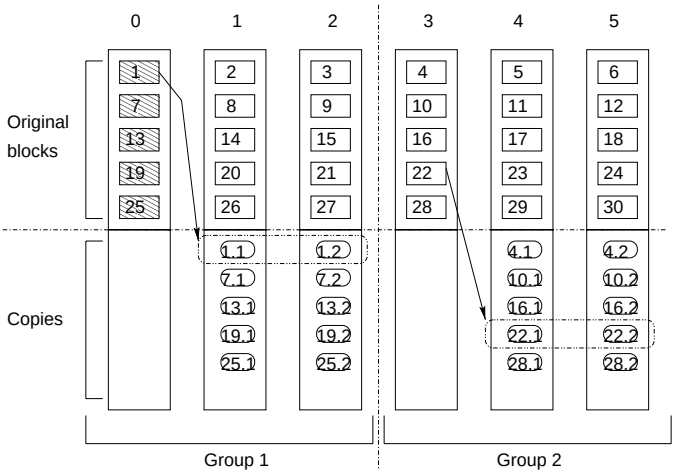
(b) One-to-All Organization with Entire block replication $Mirr_{all-entire}$.



(c) One-to-All Organization with Sub-blocks replication $Mirr_{all-sub}$.



(d) One-to-Some Organization with Entire block replication $Mirr_{some-entire}$.



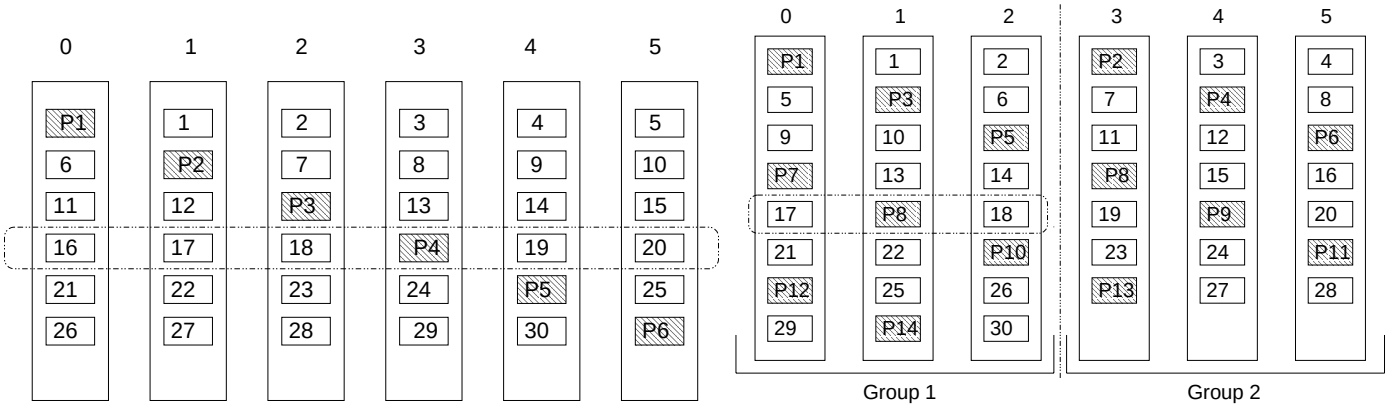
(e) One-to-Some Organization with Sub-blocks replication $Mirr_{some-sub}$.

Figure 2: Mirroring-based schemes.

In order to distribute the load of a failed disk evenly among the remaining disks of the server, the One-to-All mirroring scheme is applied as shown in Figures 2(b) and 2(c). Figure 2(b) depicts entire block replication ($Mirr_{all-entire}$) and Figure 2(c) depicts sub-block replication ($Mirr_{all-sub}$). In Figure 2(c), we only show how original blocks of disk 0 are replicated over disks 1, 2, 3, 4, and 5. If we look at Figures 2(b) and 2(c), we realize that only a single disk failure is allowed. When two disk failures occur, the server cannot ensure the delivery of all video data.

The One-to-Some mirroring scheme trades-off load-imbalances of the One-to-One mirroring scheme and the low reliability of the One-to-All mirroring scheme. In fact, as shown in Figures 2(d) (entire block replication, $Mir_{some-entire}$) and 2(e) (sub-block replication, $Mir_{some-sub}$), the server is divided into multiple (2) independent groups. Each group locally employs the One-to-All mirroring scheme. Thus, original blocks of one disk are replicated on the remaining disks of the group and therefore the load of a failed disk is distributed over all remaining disks of the group. Further, since each group tolerates a single disk failure, the server may survive multiple disk failures.

Figure 3 presents two layout examples of RAID5 that correspond to the One-to-All parity scheme, Par_{all} (Figure 3(a)) and the One-to-Some parity scheme, Par_{some} (Figure 3(b)). In Figure 3(a) the parity group size is 6, e.g. the 5 original blocks 16, 17, 18, 19, and 20 and the parity block P_4 build a parity group. In Figure 3(b) the parity groups size is 3, e.g. the 2 original blocks 17 and 18 and the parity block P_8 build a parity group.



(a) One-to-All Organization Par_{all}.

(b) One-to-Some Organization Par_{some}.

Figure 3: Parity-based Schemes.

Looking at Figures 2 and 3, we observe that all One-to-All schemes (mirroring with entire block replication ($Mirr_{all-entire}$), mirroring with sub-block replication ($Mirr_{all-sub}$), and RAID5 with one group (Par_{all})) only tolerate one disk failure. All these schemes therefore have the same server reliability. The same property holds for all One-to-Some schemes (mirroring with entire block replication, mirroring with sub-block replication, and RAID5 with C groups), since they all tolerate at most a single disk failure on each group. Consequently, it is enough for our reliability study to consider the three schemes (classes): One-to-One, One-to-All, and One-to-Some. However, for our performance study we will consider in section 4 the different schemes of Table 1.

2.2 Related Work

Based on RAID [1], reliability has been addressed previously in the literature either in a general context of file storage, or for video server architectures. Mechanisms to ensure fault-tolerance by adding redundant information

to original content can be classified into **parity-based** schemes and **mirroring-based** schemes.

An extensive amount of work has been carried out in the context of parity-based reliability, see e.g. [24, 2, 17, 10, 25, 26, 7, 27, 28]. These papers ensure a reliable real-time data delivery, even when one or some components fail. These papers differ in the way (i) they stripe data such as RAID3 (also called FGS) or RAID5 (also called CGS), and (ii) allocate parity information within the server (dedicated, shared, declustered, randomly, sequentially, SID, etc.), and (iii) the optimization goals (throughput, cost, buffer requirement, load-balancing, start-up latency for new client requests, disk bandwidth utilization, etc.).

Video servers using mirroring have been proposed previously, see e.g. [12, 15, 23, 14, 29, 21, 22]. However no *reliability modeling* has been carried out. Many mirroring schemes were compared by Merchant et al. [15], where some striping strategies for replicated disk arrays were analyzed. Depending on the striping granularity of the original and the replicated data, they distinguish between the uniform striping (CGS for original and replicated data in dedicated or in chained form) and the dual striping (original data are striped using CGS and replicated data are striped using FGS). However, their work is different from our study in many regards. First, the authors assume that both copies are used during normal and failure operation mode. Second, the comparison of different mirroring schemes is based on the *mean response times* and on *the throughput* achieved without taking into account server reliability. Finally, the authors do not analyze the impact of varying the distribution granularity of redundant data on server reliability and server performance.

In a general context of Redundant Arrays of Inexpensive Disks, Trivedi et al. [30] analyzed the reliability of RAID1-5 and focused on the relationship between disk's $MTTF$ and system reliability. Their study is based on the assumption that a RAID system is partitioned into *cold* and *hot* disks, where only hot disks are active during normal operation mode. In our case, we study reliability strategies for video servers that do not store redundant data separately on *dedicated* disks, but distribute original and redundant data evenly among *all* server disks. Gibson [31] uses continuous Markov models in his dissertation to evaluate the performance and reliability of parity-based redundant disk arrays.

In the context of video servers, reliability modeling for *parity-based schemes* (RAID3, RAID5) has been performed in [32] and RAID3 and RAID5 were compared using Markov reward models to calculate server availability. The results show that RAID5 is better than RAID3 in terms of the so-called performability (availability combined with performance).

To the best of our knowledge, there is no previous work in the context of video servers that has compared several mirroring and parity schemes with various distribution granularities in terms of the server reliability and the server performance and costs.

3 Reliability Modeling

3.1 Motivation

We define the server reliability at time t as the probability that the video server is able to access *all* videos stored on it provided that all server components are initially operational. The server survives as long as its working components deliver any video requested to the clients. As we have already mentioned, the server reliability depends on the distribution granularity of redundant data and is independent of whether mirroring or parity is used. In fact, what counts for the server reliability is the number of disks/nodes that are allowed to fail without causing the server to fail. As an example, the One-to-All mirroring scheme with entire block replication ($Mirr_{all-entire}$) only tolerates a single disk failure. This is also the case for $Mirr_{all-sub}$ and for Par_{all}. These three schemes have therefore the same server reliability. In light of this fact, we use the term One-to-All to denote all of the three schemes for the purpose of our reliability study. Analogous to One-to-All, the term One-to-

Some will represent the three schemes $Mirr_{some-entire}$, $Mirr_{some-sub}$, and Par_{some} and the term One-to-One denotes the One-to-One mirroring scheme $Mirr_{one}$.

We use Continuous Time Markov Chains (CTMC) for the server reliability modeling [33] . CTMC has discrete state spaces and continuous time and is also referred to as *Markov process* in [34, 35]. We assume that the mean time to failure ($MTTF$) of every component is **exponentially distributed**. The server $MTTF_s$ and the server reliability $R^s(t)$ have the following relationship (assuming that $MTTF_s < \infty$): $MTTF_s = \int_0^\infty R^s(t)dt$ [34]. The mean time to disk failure $MTTF_d$ equals $\frac{1}{\lambda_d}$ and the mean time to disk repair $MTTR_d$ equals $\frac{1}{\mu_d}$.

To build the *state-space diagram* [34] of the corresponding CTMC, we introduce the following parameters: s denotes the total number of states that the server can have; i denotes a state in the Markov chain with $i \in [0..(s-1)]$; $p_i(t)$ is the probability that the server is in state i at time t. We assume that the server is fully operational at time $t_0 = 0$ and state 0 is the initial state. Additionally, state $(s-1)$ denotes the system failure state and is assumed to be an *absorbing state* (unlike previous work [32], where a Markov model was used to compare the performance of RAID3 and RAID5 and allowed the repair of an overall server failure). When the video server attains state $(s-1)$, it is assumed to stay there for an infinite time. This previous assumption allows to concentrate our reliability study on the interval between the initial start up time and the time, at which the *first* server failure occurs. Thus: $p_0(0) = 1,\ p_i(0) = 0\ \forall i \in [1..(s-1)]$, and $p_{(s-1)}(\infty) = 1$. The server reliability function $R^s(t)$ can then be computed as [34]: $R^s(t) = \sum_{i=0}^{s-2} p_i(t) = 1 - p_{(s-1)}(t)$

We present in the remainder of this section the Markov models for the three schemes One-to-All, One-to-One, and One-to-Some, assuming both, (i) independent disk failures (section 3.2) and (ii) dependent component failures (section 3.3).

3.2 Reliability Modeling for Independent Disk Failures

3.2.1 The One-to-All Scheme

With the One-to-All scheme ($Mirr_{all-entire}$, $Mirr_{all-sub}$, Par_{all}), data are lost if at least two disks have failed. The corresponding state-space diagram is shown in Figure 4, where states 0, 1, and F denote respectively the initial state, the one disk failure state, and the server failure state.

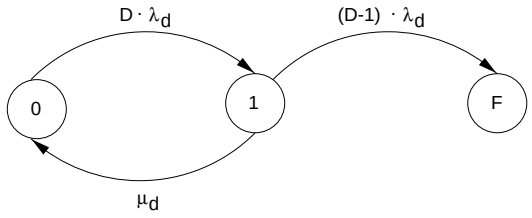


Figure 4: State-space diagram for the One-to-All Scheme.

The generator matrix Q_s of this CTMC is then:

$$Q_s = \begin{pmatrix} -D \cdot \lambda_d & D \cdot \lambda_d & 0 \\ \mu_d & -\mu_d - (D-1) \cdot \lambda_d & (D-1) \cdot \lambda_d \\ 0 & 0 & 0 \end{pmatrix}$$

3.2.2 The One-to-One Scheme

The One-to-One scheme ($Mirr_{one}$) is only relevant for mirroring. As the One-to-One schemes stores the original data of disk i on disk $((i+1) \bmod D)$, the server fails if *two consecutive* disks fail. Depending on the location of the failed disks, the server therefore tolerates a number of disk failures that can take values between 1 and $\frac{D}{2}$. Thus, the number of disks that are allowed to fail without making the server fail can *not* be known in advance, which makes the modeling of the one-to-one scheme complicated: Let the D server disks be numbered from 0 to $D-1$. Assume that the server continues to operate after $(k-1)$ disk failures. Let $P^{(k)}$ be the probability that the server does not fail after the k^{th} disk failure. $P^{(k)}$ is also the probability that no disks that have failed are consecutive (adjacent). We calculate in Appendix A the probability $P^{(k)}$ for all $k \in [2..\frac{D}{2}]$. It is obvious that $P^{(1)} = 1$ and $P^{(\frac{D}{2}+1)} = 0$.

Figure 5 shows the state-space diagram for $Mirr_{one}$. If the server is in state i ($i \geq 1$) and failure $i+1$ happens, then the probability that the server fails (state F) equals $(1 - P^{(i+1)})$ and the probability that the server continuous operating (state $i+1$) equals $P^{(i+1)}$.

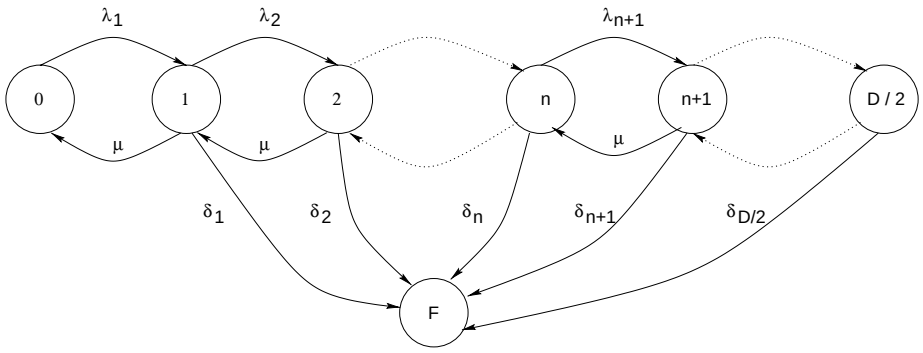


Figure 5: State-space diagram for the One-to-One scheme.

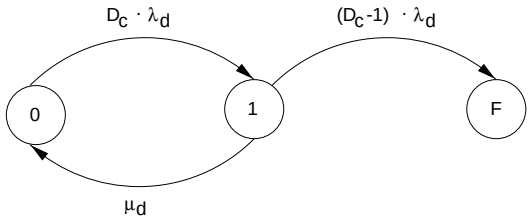
The parameters of Figure 5 have the following values: $\lambda_1 = D \cdot \lambda_d$, $\lambda_2 = (D-1) \cdot \lambda_d \cdot P^{(2)}$, $\lambda_3 = (D-2) \cdot \lambda_d \cdot P^{(3)}$, $\lambda_{n+1} = (D-n) \cdot \lambda_d \cdot P^{(n+1)}$, $\delta_1 = (D-1) \cdot \lambda_d \cdot (1 - P^{(2)})$, $\delta_2 = (D-2) \cdot \lambda_d \cdot (1 - P^{(3)})$, $\delta_n = (D-n) \cdot \lambda_d \cdot (1 - P^{(n+1)})$, $\delta_{n+1} = (D-(n+1)) \cdot \lambda_d \cdot (1 - P^{(n+2)})$, $\delta_{\frac{D}{2}} = \frac{D}{2} \cdot \lambda_d$, and $\mu = \mu_d$. The corresponding generator matrix Q_s of the CTMC above is:

$$Q_s = \begin{pmatrix} -\lambda_1 & \lambda_1 & 0 & 0 & \cdots & 0 & 0 \\ \mu & -\mu-\lambda_2-\delta_1 & \lambda_2 & 0 & \cdots & 0 & \delta_1 \\ 0 & \mu & -\mu-\lambda_3-\delta_2 & \lambda_3 & \cdots & 0 & \delta_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \mu & -\mu-\delta_{\frac{D}{2}} & \delta_{\frac{D}{2}} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \quad (1)$$

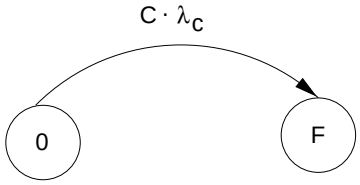
3.2.3 One-to-Some Scheme

The One-to-Some scheme (mirroring/parity) builds independent groups. The server fails if one of its C groups fails and the group failure distribution is assumed to be exponential. We first model the group reliability and then derive the server reliability. Figures 6(a) and 6(b) show the state-space diagrams of one group and of the server respectively. In Figure 6(b), the parameter C denotes the number of groups in the server and λ_c denotes the group failure rate.

The generator matrix Q_c for the CTMC of a single group is:



(a) State-space diagram of one group.



(b) State-space diagram of the server.

Figure 6: State-space diagrams for the One-to-Some Scheme.

$$Q_c = \begin{pmatrix} -D_c \cdot \lambda_d & D_c \cdot \lambda_d & 0 \\ \mu_d & -\mu_d - (D_c - 1) \cdot \lambda_d & (D_c - 1) \cdot \lambda_d \\ 0 & 0 & 0 \end{pmatrix}$$

The group reliability function $R^c(t)$ at time t is $R^c(t) = \sum_{i=0}^{1} p_i(t) = p_{(0)}(t) + p_{(1)}(t) = 1 - p_{(2)}(t)$. The group mean lifetime $MTTF_c$ is then derived from $R^c(t)$. To calculate the overall server reliability function, we assume that the group failure distribution is exponential. The server failure rate is thus $C \cdot \lambda_c$ with $\lambda_c = \frac{1}{MTTF_c}$. The server generator matrix Q_s of the CTMC of Figure 6(b) is therefore:

$$Q_s = \begin{pmatrix} -C \cdot \lambda_c & C \cdot \lambda_c \\ 0 & 0 \end{pmatrix} \tag{2}$$

3.3 Reliability Modeling for Dependent Component Failures

Dependent component failures mean that the failure of a single component of the server can affect other server components. We recall that our server consists of a set N of server nodes, where each node contains a set of D_n disks. Disks that belong to the same node have common components such as the node's CPU, the disk controller, the network interface, etc.. When any of these components fails, all disks contained in the affected node are unable to deliver video data and are therefore considered as failed. Consequently, a single disk does not deliver video data anymore if itself fails or if one of the components of the node fails to which this disk belongs. We present below the models for the different schemes for the case of dependent component failures. Similarly to a disk failure, a node failure is assumed to be repairable. The failure rate λ_n of the node is exponentially distributed with $\lambda_n = \frac{1}{MTTF_n}$, where $MTTF_n$ is the mean life time of the node. The repair rate μ_n of a failed node is exponentially distributed as $\mu_n = \frac{1}{MTTR_n}$, where $MTTR_n$ is the mean repair time of the node.

For mirroring and parity schemes, we apply the so called **Orthogonal RAID** mechanism whenever groups must be built. Orthogonal RAID was discussed in [31] and [17]. It is based on the following idea. Disks that belong to the same group must belong to different nodes. Thus, the disks of a single group do not share any (common) node hardware components. Orthogonal RAID has the property that the video server survives a complete node failure: When one node fails, all its disks are considered as failed. Since these disks belong to different groups, each group will experience at most one disk failure. Knowing that one group tolerates a single disk failure, all groups will survive and therefore the server will continue operating. Until now, Orthogonal RAID was only applied in the context of *parity* groups. We generalize the usage of Orthogonal RAID for both, mirroring and parity.

In order to distinguish between disk and node failure when building the models of the schemes considered, we will present each state as a tuple $[i, j]$, where i gives the number of disks failed and j gives the number of nodes failed. The failure (absorbing) state is represented with the letter F as before.

3.3.1 The One-to-All Scheme

For the One-to-All schemes ($Mirr_{all-entire}$, $Mirr_{all-sub}$, and Par_{all}), double disk failures are not allowed and therefore a complete node failure causes the server to fail. Figure 7 shows the state-space diagram for the One-to-All scheme for the case of dependent component failures. The states of the model denote respectively the initial state ($[0, 0]$), the one disk failure state ($[1, 0]$), and the server failure state (F).

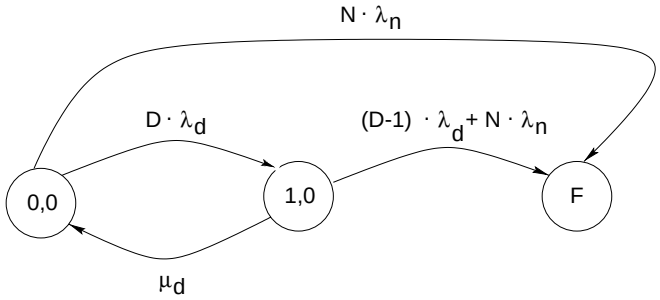


Figure 7: State-space diagram for the One-to-All scheme with dependent component failures.

The generator matrix Q_s is then:

$$Q_s = \begin{pmatrix} -D \cdot \lambda_d - N \cdot \lambda_n & D \cdot \lambda_d & N \cdot \lambda_n \\ \mu_d & -\mu_d - (D-1) \cdot \lambda_d - N \cdot \lambda_n & (D-1) \cdot \lambda_d + N \cdot \lambda_n \\ 0 & 0 & 0 \end{pmatrix}$$

3.3.2 The Grouped One-to-One Scheme

Considering dependent component failures, the One-to-One scheme as presented in Figure 2(a) would achieve a very low server reliability since the server immediately fails if a single node fails. We propose in the following an organization of the One-to-One scheme that tolerates a complete node failure and even $\frac{N}{2}$ node failures in the best case. We call the new organization the **Grouped One-to-One** scheme. The Grouped One-to-One organization keeps the initial property of the One-to-One scheme, which consists in *completely* replicating the original content of one disk onto another disk. Further, the Grouped One-to-One organization divides the server into independent groups, where disks belonging to the same group have their replica inside this group. The groups are built based on the Orthogonal RAID principle and thus disks of the same group belong to different nodes as Figure 8 shows. Figure 8 assumes one video containing 40 original blocks and is stored on a server that is made of four nodes ($N_1, \cdots, N_4$), each containing two disks. Inside one group, up-to $\frac{D_c}{2}$ disk failures can be tolerated, where $D_c = 4$ is the number of disks inside each group. The Grouped One-to-One scheme can therefore survive $\frac{N}{2} = 2$ node failures in the best case (the server in Figure 8 continues operating even after nodes N_1 and N_2 fail). In order to distribute the load of a failed node among possibly *many* and not only *one* of the surviving nodes, the Grouped One-to-One scheme ensures that disks belonging to the same node have their replica on disks that do not belong to the *same* node [1].

[1] Assume that node N_1 has failed, then its load is shifted to node N_3 (replica of disk 0 are stored on disk 4) and to node N_4 (replica of disk 1 are stored on disk 7)

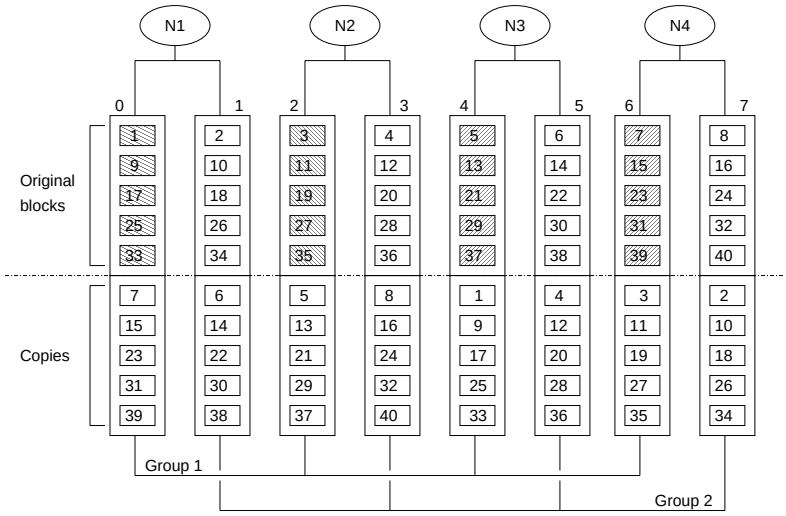


Figure 8: Grouped One-to-One scheme for a server with 4 nodes, each with 2 disks ($D_c = N = 4$).

In order to model the reliability of our Grouped One-to-One scheme, we first study the behavior of a single group and then derive the overall server reliability. One group fails when two consecutive disks inside the group fail. We remind that two disks are consecutive if the original data of one of these disks are replicated on the other disk (for group 1 the disks 0 and 4 are consecutive, whereas the disks 0 and 2 are not). Note that the failure of one disk inside the group can be due to (i) the failure of the disk itself or (ii) the failure of the whole node, to which the disk belongs. After the first disk failure, the group continues operating. If the second disk failure occurs inside the group, the group may fail or not depending on whether the two failed disks are consecutive. Let $P^{(2)}$ denotes the probability that the two failed disks of the group are not consecutive. Generally, $P^{(k)}$ denotes the probability that the group does not fail after the k^{th} disk failure inside the group. $P^{(k)}$ is calculated in Appendix A.

The state-space diagram of one group for the example in Figure 8 is presented in Figure 9. The number of disks inside the group is $D_c = 4$. State $[i, j]$ denotes that $i + j$ disks have failed inside the group, where i disks, themselves, have failed and j nodes have failed. Obviously, all of the i disks that have failed belong to different nodes than all of the j nodes that have failed. We describe in the following how we have built the state-space diagram of Figure 9. At time t_0, the group is in state $[0, 0]$. The first disk failure within the group can be due to a single disk failure (state $[1, 0]$) or due to a whole node failure (state $[0, 1]$). Assume that the group is in state $[1, 0]$ and one more disk of the group fails. Four transitions are possible: (i) the group goes to state $[2, 0]$ after the second disk of the group has failed itself and the two failed disks are not consecutive, (ii) the group goes to state $[1, 1]$ after the node has failed, on which the second failed disk of the group is contained and the two failed disks are not consecutive, (iii) the group goes to state F after the second disk of the group has failed (disk failure or node failure) and the two failed disks are consecutive, and finally (iv) the group goes to state $[0, 1]$ after the node has failed, to which the first failed disk of the group belongs and thus the number of failed disks in the group does not increase (only one disk failed). The remaining of the state-space diagram is to derive in an analogous way.

The parameters of Figure 9 are the following:

$\lambda_1 = D_c \cdot \lambda_d$, $\lambda_2 = N \cdot \lambda_n$, $\lambda_3 = (D_c - 1) \cdot \lambda_d \cdot P^{(2)}$, $\lambda_4 = (N - 1) \cdot \lambda_n \cdot P^{(2)}$,

$F_1 = ((D_c - 1) \cdot \lambda_d + (N - 1) \cdot \lambda_n) \cdot (1 - P^{(2)})$, $\lambda_5 = (D_c - 2) \cdot \lambda_d \cdot P^{(3)}$, $\lambda_6 = (N - 2) \cdot \lambda_n \cdot P^{(3)}$, $F_2 = ((D_c - 1) \cdot \lambda_d + (N - 1) \cdot \lambda_n) \cdot (1 - P^{(2)})$, $\lambda_7 = (\frac{D_c}{2} \cdot \lambda_d) + (\frac{N}{2} \cdot \lambda_N)$, $\mu_1 = \mu_d$, and $\mu_2 = \mu_n$.

The generator matrix Q_c for a group is:

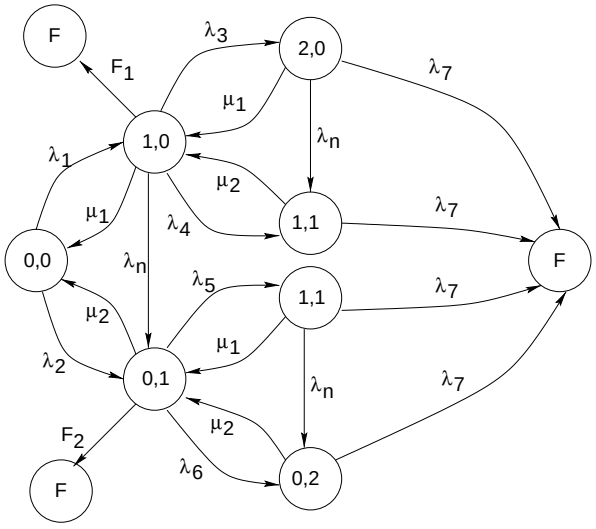


Figure 9: State-space diagram of one group for the Grouped One-to-One scheme with dependent component failures ($D_c = N = 4$).

$$Q_c = \begin{pmatrix} -\lambda_1 - \lambda_2 & \lambda_1 & \lambda_2 & 0 & 0 & 0 & 0 & 0 \\ \mu_1 & -A & \lambda_n & \lambda_3 & \lambda_4 & 0 & 0 & F_1 \\ \mu_2 & 0 & -B & 0 & \lambda_5 & \lambda_6 & F_2 & \\ 0 & \mu_1 & 0 & -\mu_1 - \lambda_n - \lambda_7 & \lambda_n & 0 & 0 & \lambda_7 \\ 0 & \mu_2 & 0 & 0 & -\mu_2 - \lambda_7 & 0 & 0 & \lambda_7 \\ 0 & 0 & \mu_1 & 0 & 0 & -\mu_1 - \lambda_n - \lambda_7 & \lambda_n & \lambda_7 \\ 0 & 0 & \mu_2 & 0 & 0 & 0 & -\mu_2 - \lambda_7 & \lambda_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

where $A = \mu_1 + \lambda_n + \lambda_3 + \lambda_4 + F_1$ and $B = \mu_2 + \lambda_5 + \lambda_6 + F_2$.

From the matrix Q_c we get the group mean life time $MTTF_c$, which is used to calculate the overall server reliability. The state-space diagram for the server is the one of Figure 6(b), where the parameter λ takes the value $C \cdot \lambda_c$ and λ_c denotes the failure rate of one group with $\lambda_c = \frac{1}{MTTF_c}$. The server reliability is then calculated analogously to Eq. 2.

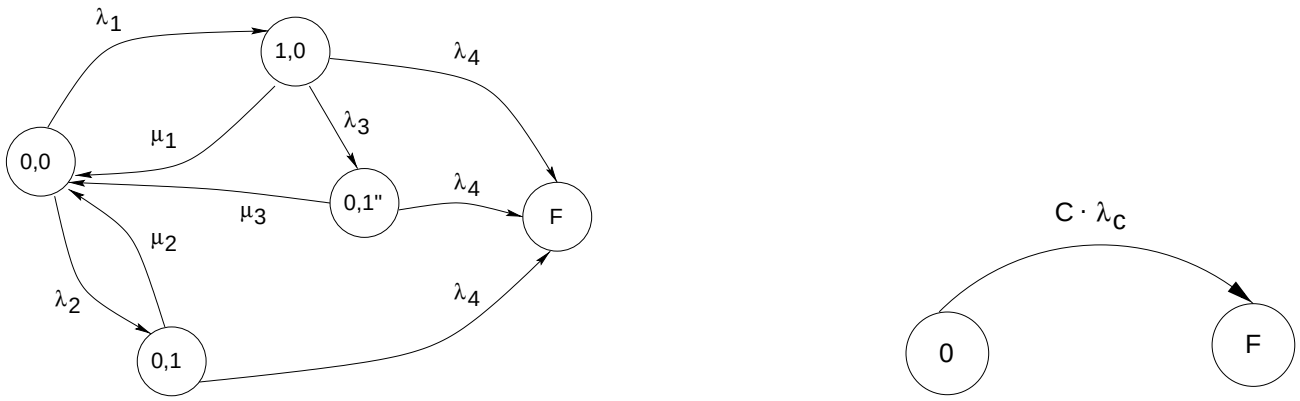
Note that the example described in Figure 9 considers a small group size ($D_c = 4$). Increasing D_c increases the number of states contained in the state-space diagram of the group. In general, the number of states for a given D_c is: $1 + \sum_{i=0}^{\frac{D_c}{2}} 2^i = 2^{\frac{D_c}{2}+1}$. We present in Appendix B a general method for building the state-space diagram of one group containing D_c disks.

3.3.3 The One-to-Some Scheme

We use Orthogonal RAID for all One-to-Some schemes ($Mirr_{some-entire}$, $Mirr_{some-sub}$, and Par_{some}). If we consider again the data layouts of Figures 2(d), 2(e), and 3(b), Orthogonal RAID is then ensured if the following holds: Node 1 contains disks 0 and 3; node 2 contains disks 1 and 4; and node 3 contains disks 2 and 5.

For the reliability modeling of the One-to-Some scheme, we first build the state-space diagram for a single group (Figure 10(a)) and then compute the overall server reliability (Figure 10(b)). The states in Figure 10(a) denote the following: the initial state ($[0, 0]$), the state where one disk fails ($[1, 0]$), the state where one node fails resulting

in a single disk failure within the group ($[0,1]$), the state where one disk and one node have failed and the failed disk belongs to the failed node ($[0,1"]$), and the one group failure state (F). The parameter values used in Figure 10 are: $\lambda_1 = D_c \cdot \lambda_d$, $\lambda_2 = N \cdot \lambda_n$, $\lambda_3 = \lambda_n$, $\lambda_4 = (D_c - 1) \cdot \lambda_d + (N - 1) \cdot \lambda_n$, $\mu_1 = \mu_d$, $\mu_2 = \mu_n$, and $\mu_3 = min(\mu_d, \mu_n)$,



(a) State-space diagram for one group.

(b) State-space diagram for the server.

Figure 10: State-space diagrams for the One-to-Some Scheme for the case of dependent component failures.

The generator matrix Q_c for a group is:

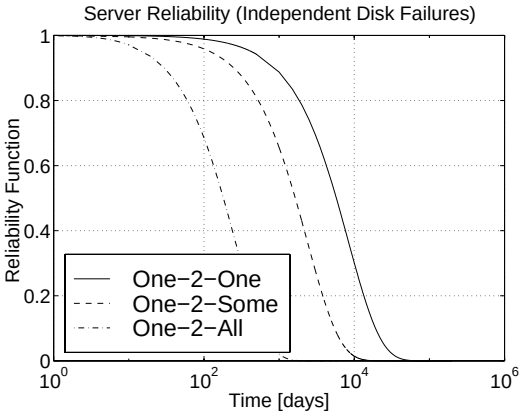
$$Q_c = \begin{pmatrix} -\lambda_1 - \lambda_2 & \lambda_1 & \lambda_2 & 0 & 0 \\ \mu_1 & -\mu_1 - \lambda_3 - \lambda_4 & 0 & \lambda_3 & \lambda_4 \\ \mu_2 & 0 & -\mu_2 - \lambda_4 & 0 & \lambda_4 \\ \mu_3 & 0 & 0 & -\mu_3 - \lambda_4 & \lambda_4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

3.4 Reliability Results

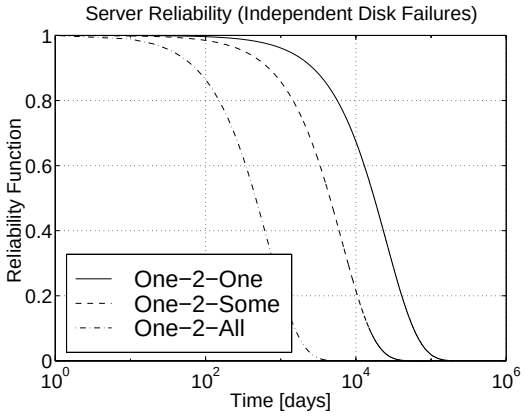
We resolve our continuous time Markov chains using the SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) [33] tool for specifying and evaluating dependability and performance models. Sharpe takes as input the generator matrix and computes the server reliability at a certain time t. The results for the server reliability are shown in Figures 11 and 12. The total number of server disks considered is $D = 100$ and the number of server nodes is $N = 10$, each node containing 10 disks. We examine server reliability for two failure rates, $\frac{1}{60000 \text{ hours}}$ and $\frac{1}{100000 \text{ hours}}$, which are pessimistic values.

Figure 11 plots the server reliability for the One-to-One, One-to-All, and One-to-Some schemes for the case of independent disk failures. As expected, the server reliability for the One-to-One scheme is the highest. The One-to-Some scheme exhibits higher server reliability than the One-to-All scheme. Figures 11(a) and 11(b) also show how much the server reliability is improved when mean time to disk failure increases (λ_d decreases). For example, for the One-to-One scheme and after 10^4 days of operation, the server reliability is about 0.3 for $\lambda_d = \frac{1}{60000 \text{ hours}}$ and is about 0.66 for $\lambda_d = \frac{1}{100000 \text{ hours}}$.

Figure 12 depicts the server reliability for the Grouped One-to-One, the One-to-All, and the One-to-Some schemes for the case of dependent component failures. We observe that the Grouped One-to-One scheme provides a better reliability than the One-to-Some scheme. The One-to-All scheme has the lowest server reliability,



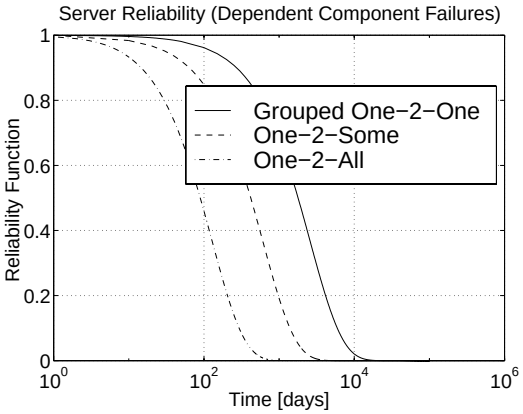
(a) Server reliability for $\lambda_d = \frac{1}{60000 \text{ hours}}$ and $\mu_d = \frac{1}{72 \text{ hours}}$.



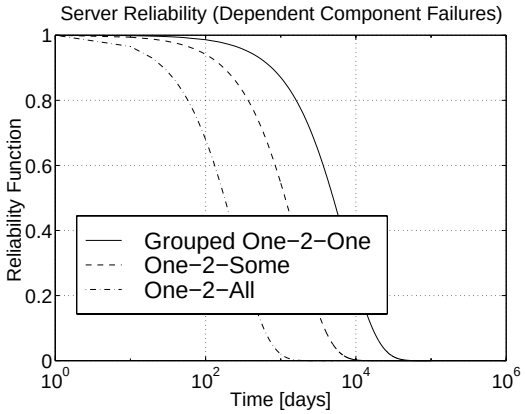
(b) Server reliability for $\lambda_d = \frac{1}{100000 \text{ hours}}$ and $\mu_d = \frac{1}{72 \text{ hours}}$.

Figure 11: Server reliability for the three schemes assuming independent disk failures with $D = 100$ $D_c = N = 10$.

e.g. for $\lambda_d = \lambda_n = \frac{1}{100000 \text{ hours}}$ and after three years, the server reliability is 0 for the One-to-All scheme, 0.51 for the One-to-Some scheme, and 0.85 for the Grouped One-to-One scheme. Figures 12(a) and 12(b) show that the server reliability increases when λ_d (λ_n) decreases.



(a) Server reliability for $\lambda_d = \lambda_n = \frac{1}{60000 \text{ hours}}$ and $\mu_d = \mu_n = \frac{1}{72 \text{ hours}}$.



(b) Server reliability for $\lambda_d = \lambda_n = \frac{1}{100000 \text{ hours}}$ and $\mu_d = \mu_n = \frac{1}{72 \text{ hours}}$.

Figure 12: Server reliability for the three schemes assuming dependent component failures with $D = 100$ and $D_c = N = 10$.

Comparing Figures 11 and 12, we see, as expected, that the server reliability is higher for the independent disk failure case than for the dependent component failure case. We restrict our further discussion to the case of dependent component failures since it is more realistic than the case of independent disk failures.

4 Server Performance

An important performance metric for the designer and the operator of a video server is the maximum number of streams Q_s that the server can simultaneously admit, which is referred to as the **server throughput**. Adding fault-tolerance within a server requires additional resources in terms of storage volume, main memory and I/O bandwidth capacity. As we will see, the reliability schemes discussed differ not only in the throughput they achieve, but also in the amount of additional resources they need to guarantee uninterrupted service during disk failure mode. Throughput is therefore not enough to compare server performance of these schemes. Instead, we use the **cost per stream**. We calculate in section 4.1 the server throughput for each of the schemes. Section 4.2 focuses on buffer requirements. Section 4.3 compares then the different schemes with respect to the cost per stream.

4.1 Server Throughput

The admission control policy decides, based on the remaining available resources, whether a new incoming stream is admitted. The CGS striping algorithm serves a list of streams from a single disk during one service round. During the next service round, this list of streams is shifted to the next disk. If Q_d denotes the maximum number of streams that a single disk can serve simultaneously (*disk throughput*) in a non fault-tolerant server, then the overall server throughput Q_s is simply $Q_s = D \cdot Q_d$. Accordingly, we will restrict our discussion to disk throughput. Disk throughput Q_d is given by Eq. 3 [5], where meaning and value of the different parameters are listed in Table 2. The disk parameter values are those of Seagate and HP for the SCSI II disk drives [36].

$$Q_d \cdot \left(\frac{b}{r_d} + t_{rot} + t_{stl} \right) + 2 \cdot t_{seek} \leq \tau$$
$$Q_d = \frac{\tau - 2 \cdot t_{seek}}{\frac{b}{r_d} + t_{rot} + t_{stl}} \tag{3}$$

Parameter	Meaning of Parameter	Value
r_p	Video playback rate	1.5 Mbps
r_d	Inner track transfer rate	40 Mbps
t_{stl}	Settle time	1.5 ms
t_{seek}	Seek time	10.39 ms
t_{rot}	Worst case rotational latency	9.33 ms
b	Block size	1 Mbit
τ	Service round duration	$\frac{b_{dr}}{r_p}$ sec

Table 2: Performance Parameters

To allow for fault-tolerance, each disk reserves a portion of its available I/O bandwidth to be used during disk failure mode. Since the amount of reserved disk I/O bandwidth is not the same for all schemes, the disk throughput will also be different.

Let us start with the Grouped One-to-One scheme $Mirr_{one}$. Since the original content of a single disk is entirely replicated onto another disk, half of each disk's I/O bandwidth must be kept unused during normal operation mode to be available during disk failure mode. Consequently the disk throughput Q_{One}^{mirr} is simply the half of Q_d: $Q_{One}^{mirr} = \frac{Q_d}{2}$.

For the One-to-All mirroring scheme $Mirr_{all-entire}$ with entire block replication, the original blocks of one disk are spread among the other server disks. However, it may happen that the original blocks that would have been required from a failed disk during a particular service round are *all* replicated on the *same* disk (worst case situation). In order to guarantee deterministic service for this worst case, half of the disk I/O bandwidth must be reserved to disk failure mode. Therefore, the corresponding disk throughput $Q^{mirr}_{All-Entire}$ is: $Q^{mirr}_{All-Entire} = \frac{Q_d}{2}$.

The worst case retrieval pattern for the One-to-Some mirroring scheme $Mirr_{some-entire}$ with entire block replication is the same as for the previous scheme and we get: $Q^{mirr}_{Some-Entire} = \frac{Q_d}{2}$. Since the three schemes $Mirr_{one}$, $Mirr_{all-entire}$, and $Mirr_{some-entire}$ achieve the same throughput, we will use the term $Mirr_{Entire}$ to denote all of them and Q^{mirr}_{Entire} [2] to denote their disk throughput:

$$Q^{mirr}_{Entire} = \frac{Q_d}{2} \tag{4}$$

For the One-to-All mirroring scheme $Mirr_{all-sub}$ with sub-block replication, the situation changes. In fact, during disk failure mode, each disk retrieves at most $Q^{mirr}_{All-Sub}$ original blocks and $Q^{mirr}_{All-Sub}$ replicated *sub-blocks* during one service round. Let us assume that sub-blocks have the same size b^{all}_{sub}, i.e. $b = b^{all}_{sub} \cdot (D-1)$. The admission control formula becomes:

$$\begin{aligned} Q^{mirr}_{All-Sub} \cdot \left((\frac{b}{r_d} + t_{rot} + t_{stl}) + (\frac{b^{all}_{sub}}{r_d} + t_{rot} + t_{stl}) \right) + 2 \cdot t_{seek} \leq \tau \\ \Rightarrow \qquad Q^{mirr}_{All-Sub} = \frac{\tau - 2 \cdot t_{seek}}{\frac{b + b^{all}_{sub}}{r_d} + 2 \cdot (t_{rot} + t_{stl})} \end{aligned} \tag{5}$$

Similarly, the disk throughput $Q^{mirr}_{Some-Sub}$ for One-to-Some mirroring with sub-block replication $Mirr_{some-sub}$ is:

$$Q^{mirr}_{Some-Sub} = \frac{\tau - 2 \cdot t_{seek}}{\frac{b + b^{some}_{sub}}{r_d} + 2 \cdot (t_{rot} + t_{stl})} \tag{6}$$

where b^{some}_{sub} denotes the size of a sub-block as $b = b^{some}_{sub} \cdot (D_c - 1)$ and D_c is the number of disks contained on each group.

We now consider the disk throughput for the parity schemes. Recall that we study the buffering strategy and not the second read strategy for lost block reconstruction. For the One-to-All parity scheme Par_{all}, one parity block is needed for every $(D-1)$ original blocks. The additional load of each disk consisting in retrieving parity blocks when needed can be seen from Figure 3(a). In fact, for one stream in the worst case all requirements for parity blocks concern the same disk, which means that at most one parity block is retrieved from each disk every D service rounds. Consequently, each disk must reserve $\frac{1}{D}$ of its I/O bandwidth for disk failure mode. The disk throughput Q^{par}_{All} is then calculated as:

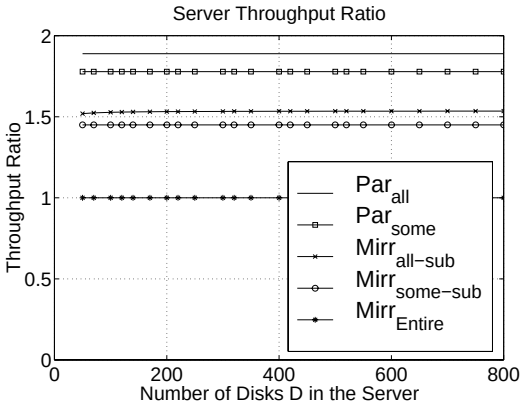
$$Q^{par}_{All} = Q_d - \lceil \frac{Q_d}{D} \rceil \tag{7}$$

Analogous to the One-to-All parity scheme, the One-to-Some parity scheme Par_{some} has the following disk throughput Q^{par}_{Some}:

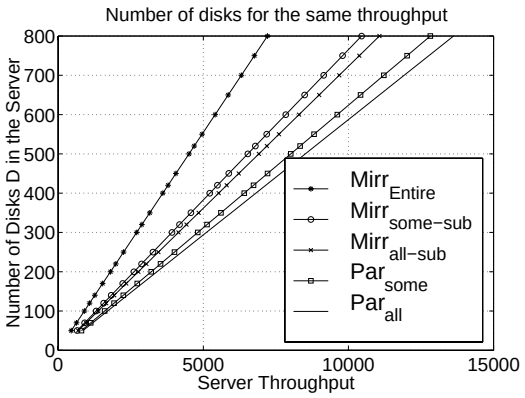
$$Q^{par}_{Some} = Q_d - \lceil \frac{Q_d}{D_c} \rceil \tag{8}$$

[2] These three schemes share the common property that each original block is entirely replicated into one block.

In Figure 13(a), we take the throughput value Q^{mirr}_{Entire} of $Mirr_{Entire}$ as base line for comparison and plot the ratios of the server throughput as a function of the total number of disks in the server.



(a) Throughput Ratios.



(b) Number of disks required for the same Server throughput.

Figure 13: Throughput results for the reliability schemes with $D_c = 10$.

Mirroring schemes that use entire block replication ($Mirr_{Entire}$) provide lowest throughput. The two mirroring schemes $Mirr_{all-sub}$ and $Mirr_{some-sub}$ that use sub-block replication have throughput ratios of about 1.5. The performance for $Mirr_{all-sub}$ is slightly higher than the one for $Mirr_{some-sub}$ since the sub-block size $b^{all}_{sub} < b^{some}_{sub}$. Parity schemes achieve higher throughput ratios than mirroring schemes and the One-to-All parity scheme Par_{all} results in the highest throughput. The throughput for the One-to-Some parity scheme Par_{some} is slightly smaller than the throughput for Par_{all}. In fact, the parity group size of $(D-1)$ for Par_{all} is larger than D_c. As a consequence, the amount of disk I/O bandwidth that must be reserved for disk failure is smaller for Par_{all} than for Par_{some}. In order to get a quantitative view regarding the I/O bandwidth requirements, we reverse the axes of Figure 13(a) to obtain in Figure 13(b) for each scheme the number of disks needed to achieve a given server throughput.

4.2 Buffer Requirement

Another resource that affects the cost of the video server and therefore the cost per stream is main memory. Due to the speed mismatch between data retrieval from disk (transfer rate) and data consumption (playback rate), main memory is needed at the server to temporarily store the blocks retrieved. For the SCAN retrieval algorithm, the worst case buffer requirement for one served stream is twice the block size b. Assuming the normal operation mode (no component failures), the buffer requirement B of the server is therefore $B = 2 \cdot b \cdot Q_s$, where Q_s denotes the server throughput.

Mirroring-based schemes replicate original blocks that belong to a single disk over one, all, or a set of disks. During disk failure mode, blocks that would have been retrieved from the failed disk are retrieved from the disks that store the replica. Thus, mirroring requires the same amount of buffer during normal operation mode and during component failure mode independently of the distribution granularity of replicated data. Therefore, for all mirroring schemes considered (Grouped One-to-One $Mirr_{one}$, One-to-All with entire block replication $Mirr_{all-entire}$, One-to-All with sub-block replication $Mirr_{all-sub}$, One-to-Some with entire block replication $Mirr_{some-entire}$, and One-to-Some with sub-block replication $Mirr_{some-sub}$) the buffer requirement during

component failure is $B^{mirr} = B$.

Unlike mirroring-based schemes, parity-based schemes need to perform a X-OR operation over a set of blocks to reconstruct a lost block. In fact, during normal operation mode the buffer is immediately liberated after consumption. When a disk fails, original blocks as well as the parity block that belong to the same parity group are sequentially retrieved (during consecutive service rounds) from consecutive disks and must be *temporarily stored* in the buffer for as many service rounds that elapse until the lost original block will be reconstructed. Since buffer overflow must be avoided, the buffer requirement is calculated for the worst case situation where the whole parity group must be contained in the buffer before the lost block gets reconstructed. An additional buffer size of one block must be also reserved to store the first block of the next parity group. Consequently, during component failure, the buffer requirement B_{all}^{par} for Par_{all} is $B_{all}^{par} = D \cdot b \cdot Q_s = \frac{D}{2} \cdot B$, and the buffer requirement B_{some}^{par} for Par_{some} is $B_{some}^{par} = D_c \cdot b \cdot Q_s = \frac{D_c}{2} \cdot B$. Note that the buffer requirement for Par_{all} depends on D and therefore increases linearly with the number of disks in the server. For Par_{some}, however, the group size D_c can be kept constant while the total number of disks D varies. As a result, the buffer requirement B_{some}^{par} for Par_{some} remains unchanged when D increases.

4.3 Cost Comparison

The performance metric we use is the per stream cost. We first compute the total server cost $\$_{server}$ and then derive the cost per stream $\$_{stream}$ as: $\$_{stream} = \frac{\$_{server}}{Q_s}$. We define the server cost as the cost of the hard disks and the main memory dimensioned for the component failure mode: $\$_{server} = P_{mem} \cdot B + P_d \cdot V_{disk} \cdot D$, where P_{mem} is the price of $1\ \mathrm{Mbyte}$ of main memory, B the buffer requirement in Mbyte, P_d is the price of $1\ \mathrm{Mbyte}$ of hard disk, V_{disk} is the storage volume of a single disk in MByte, and finally D is the total number of disks in the server. Current price figures – as of 1998 – are $P_{mem} = \$13$ and $P_d = \$0.5$. Since these prices change frequently, we will consider the relative costs by introducing the *cost ratio* α between P_{mem} and P_d: $\alpha = \frac{P_{mem}}{P_d}$. Thus, the server cost function becomes: $\$_{server} = P_{mem} \cdot B + \frac{P_{mem}}{\alpha} \cdot V_{disk} \cdot D = P_{mem} \cdot \left(B + \frac{V_{disk} \cdot D}{\alpha}\right)$ and the per stream cost is:

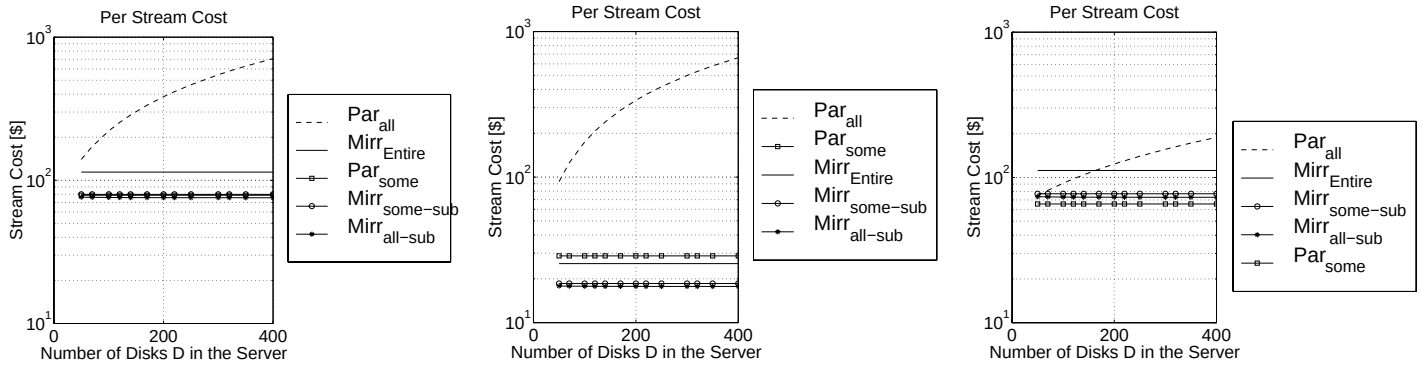
$$\$_{stream} = \frac{\$_{server}}{Q_s} = \frac{P_{mem} \cdot \left(B + \frac{V_{disk} \cdot D}{\alpha}\right)}{Q_s} \tag{9}$$

To evaluate the cost of the five different schemes, we compute for each scheme and for a given value of D the throughput Q_s achieved and the amount of buffer B required to support this throughput. Note that we take $D_c = 10$ for the schemes $Mirr_{some-entire}$, $Mirr_{some-sub}$, and Par_{some}.

Figure 14 plots the per stream cost for the schemes Par_{all}, Par_{some}, $Mirr_{Entire}$, $Mirr_{some-sub}$, and $Mirr_{all-sub}$ for different values of the cost ratio α. We recall that the notation $Mirr_{Entire}$ includes the three mirroring schemes $Mirr_{all-entire}$, $Mirr_{some-entire}$, and the Grouped One-to-One $Mirr_{one}$ that experience the same throughput and require the same amount of resources. In Figure 14(a), we consider $\alpha = \frac{13}{0.5} = 26$ that presents the current memory/hard disk cost ratio. Increasing the value of $\alpha = \frac{P_{mem}}{P_d}$ means that the price for the disk storage drops faster than the price for main memory: In Figure 14(b), we multiply the current cost ratio by five to get $\alpha = 26 \cdot 5 = 130$ [3]. On the other hand, decreasing the value of α means that the price for main memory drops faster than the price for hard disk: In Figure 14(c) we divide the current cost ratio by five to get $\alpha = \frac{26}{5} = 5.2$ [4].

[3] To illustrate the faster decrease of the price for hard disk as compared to the one for main memory, we consider the current price for main memory ($P_{mem} = \$13$) and calculate the new *reduced* price for hard disk ($P_d = \frac{13}{130} = \$0.1$).

[4] Analogously, to illustrate the faster decrease of the price for memory as compared to the price for hard disk, we take the current price for hard disk ($P_d = \$0.5$) and calculate the new *reduced* price for memory ($P_{mem} = 0.5 \cdot 5.2 = \$2.6$).



(a) $\alpha = \frac{13}{0.5} = 26$. (b) $\alpha = \frac{13}{0.1} = 130$. (c) $\alpha = \frac{2.6}{0.5} = 5.2$.

Figure 14: Per stream cost for different values of the cost ratio α with $D_c = 10$.

The results of Figure 14 indicate the following:

- The increase or the decrease in the value of α as defined above means a decrease in either the price for hard disk or the price for main memory respectively. Hence the overall decrease in the per stream cost in Figures 14(b) and 14(c) as compared to Figure 14(a).

- Figure 14(a) ($\alpha = 26$) shows that the One-to-All parity scheme (Par_{all}) results in the *highest* per stream cost that increases when D grows. In fact, the buffer requirement for Par_{all} is highest and also increases linearly with the number of disks D and thus resulting in the highest per stream cost. Mirroring schemes with entire block replication ($Mirr_{Entire}$) have the second worst per stream cost. The per stream cost for the remaining three schemes ($Mirr_{all-sub}$, $Mirr_{some-sub}$, and Par_{some}) is roughly equal and is *lowest*. The best scheme is the One-to-All mirroring scheme with sub-block replication ($Mirr_{all-sub}$). It has a slightly smaller per stream cost than the One-to-Some mirroring scheme with sub-block replication ($Mirr_{some-sub}$) due to the difference in size between b_{sub}^{all} and b_{sub}^{some} (see the explanation in section 4.1).

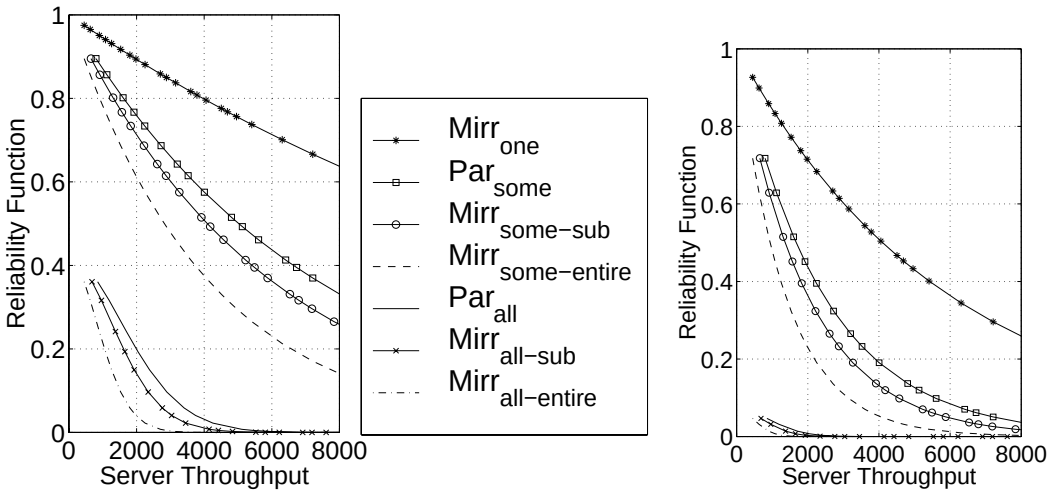
- The increase in the cost ratio α by a factor of five ($\alpha = 130$ in Figure 14(b)) slightly decreases the per stream cost of Par_{all} and results in a *dramatic* decrease in the per stream cost of all three mirroring schemes and also the parity scheme Par_{some}. For instance the per stream cost for Par_{some} decreases from $\$78.64$ down-to $\$28.72$ and the per stream cost of $Mirr_{some-sub}$ decreases from $\$79.79$ down-to $\$18.55$. All three mirroring schemes become more cost efficient than the two parity schemes.

- The decrease in the cost ratio α by a factor of five ($\alpha = 5.2$ in Figure 14(c)) affects the cost of the three mirroring schemes very little. As an example, the per stream cost for $Mirr_{some-sub}$ is $\$79.79$ in Figure 14(a) and is $\$77.19$ in Figure 14(c). On the other hand, decreasing α, i.e. the price for main memory decreases faster than the price for hard disk, clearly affects the cost of the two parity schemes. In fact, Par_{some} becomes the most cost efficient scheme with a cost per stream of $\$65.64$. Although the per stream cost of Par_{all} decreases significantly with $\alpha = 5.2$, it still remains the most expensive for high values of D. Since Par_{all} has the highest per stream cost that linearly increases with D, we will not consider this scheme in further cost discussion.

5 Server Reliability and Performance

5.1 Server Reliability vs. Per Stream Cost

Figure 15 and Table 3 depict the server reliability and the per stream cost for the different reliability schemes discussed herein. The server reliability is computed after 1 year (Figure 15(a)) and after 3 years (Figure 15(b)) of server operation. The results in Figure 15 are obtained as follows. For a given server throughput, we calculate for each reliability scheme the number of disks required to achieve that throughput. We then compute the server reliability for each scheme and its respective number of disks required. Table 3 shows the normalized per stream cost for different values of α. We take the per stream cost of $Mirr_{one}$ as base line for comparison and divide the cost values for the other schemes by the cost for $Mirr_{one}$. We recall again that the three schemes $Mirr_{one}$, $Mirr_{all-entire}$ and $Mirr_{some-entire}$ have the same per stream cost since they achieve the same throughput given the same amount of resources (see section 4).



(a) Server reliability after 1 year of server operation with $\lambda_d = \lambda_n = \frac{1}{100000 \text{ hours}}$.

(b) Server reliability after 3 years of server operation with $\lambda_d = \lambda_n = \frac{1}{100000 \text{ hours}}$.

Figure 15: Server reliability for the same server throughput with $D_c = 10$.

	$\alpha = 26$	$\alpha = 130$	$\alpha = 5.2$
$Mirr_{one}$ $Mirr_{some-entire}$ $Mirr_{all-entire}$	1	1	1
Par_{some}	0.688	1.129	0.588
$Mirr_{some-sub}$	0.698	0.729	0.691
$Mirr_{all-sub}$	0.661	0.696	0.653

Table 3: Normalized stream cost (by $Mirr_{one}$) for different values of α with $D_c = 10$

The three One-to-All schemes Par_{all}, $Mirr_{all-sub}$, and $Mirr_{all-entire}$ have poor server reliability even for a low values of server throughput, since they only survive a single disk failure. The difference in reliability between these schemes is due to the fact that Par_{all} requires, for the same throughput, fewer disks than

$Mirr_{all-sub}$ that in turn needs fewer disks than $Mirr_{all-entire}$ (see Figure 13(b)). The server reliability of these three schemes decreases dramatically after three years of server operation as illustrated in Figure 15(b)). Accordingly, these schemes are not attractive to ensure fault tolerance in video servers and hence we are not going to discuss them more in the remainder of this paper. We further discuss the three One-to-Some schemes Par_{some}, $Mirr_{some-sub}$, and $Mirr_{some-entire}$ and the Grouped One-to-One scheme $Mirr_{one}$. Based on Figures 15(a) and 15(b), $Mirr_{one}$ has a higher server reliability than the three One-to-Some schemes Par_{some}, $Mirr_{some-sub}$, and $Mirr_{some-entire}$.

From Table 3, we see that $Mirr_{one}$, which has the same per stream cost as $Mirr_{some-entire}$, has a per stream cost about 1.5 higher than $Mirr_{some-sub}$. Par_{some} has the highest per stream cost for a high value of α ($\alpha = 130$) and is the most cost effective for a small value of α ($\alpha = 5.2$).

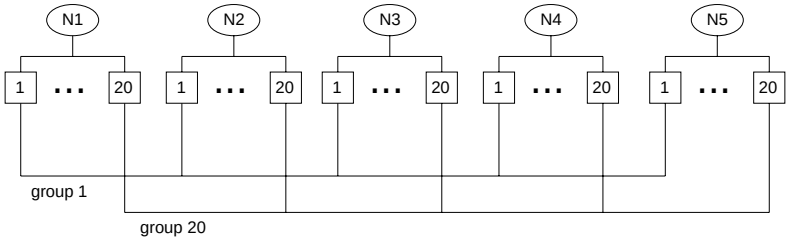
In summary, we see that the best scheme among the One-to-Some schemes is Par_{some} since it has a low per stream cost and requires fewer disks than $Mirr_{some-sub}$ and thus provides a higher server reliability than both, $Mirr_{some-sub}$ and $Mirr_{some-entire}$. Since $Mirr_{some-entire}$ achieves much lower server reliability than $Mirr_{one}$ for the same per stream cost, we conclude that $Mirr_{some-entire}$ is not a good scheme for achieving fault tolerance in a video server.

Based on the results of Figure 15 and Table 3, we conclude that the three schemes: $Mirr_{one}$, Par_{some}, and $Mirr_{some-sub}$ are the good candidates to ensure fault tolerance in a video server. Note that we have assumed in Figure 15 for all these three schemes the same value $D_c = 10$. $Mirr_{one}$ has the highest server reliability but a higher per stream cost as compared to the per stream cost of Par_{some} and $Mirr_{some-sub}$. For the value $D_c = 10$, the two schemes Par_{some} and $Mirr_{some-sub}$ have a lower per stream cost but also a lower server reliability than $Mirr_{one}$. This difference in server reliability becomes more pronounced as the number of disks in the video server increases. We will see in the next section how to determine the parameter D_c for the schemes Par_{some} and $Mirr_{some-sub}$ in order to improve the trade-off between the server reliability and the cost per stream.

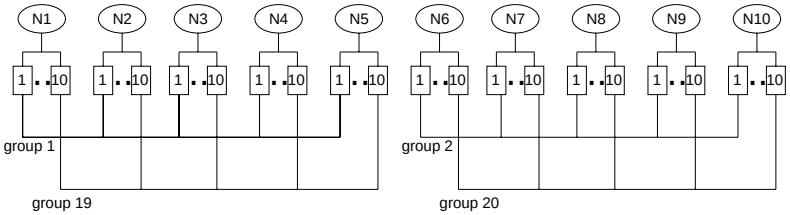
5.2 Determining the Group Size D_c

This section evaluates the impact of the group size D_c on the server reliability and the per stream cost. We limit our discussion to the three schemes: our $Mirr_{one}$, Par_{some}, and $Mirr_{some-sub}$. Remember that we use the Orthogonal RAID principle to build the independent groups (see section 3.3). Accordingly, disks that belong to the same group are attached to different nodes. Until now, we have assumed that the group size D_c and the number of nodes N are constant ($D_c = N = 10$). In other terms, increasing D leads to an increase in the number of disks D_n per node. However, the maximum number of disks D_n is limited by the node's I/O capacity. Assume a video server with $D = 100$ disks and $D_c = 5$. We plot in Figure 16 two different ways to configure the video server. In Figure 16(a) the server contains five nodes ($N = 5$), where each node consists of $D_n = 20$ disks. One group contains $D_c = 5$ disks, each belonging to a different node. On the other hand, Figure 16(b) configures a video server with $N = 10$ nodes, each containing only $D_n = 10$ disks. The group size is again $D_c = 5$, i.e. a single group does not stretch across all nodes. Note that the number of groups C is the same for both configurations ($C = 20$). When the video server grows, the second alternative suggests to add new nodes (containing new disks) to the server, whereas the first alternative suggests to add new disks to the existing nodes. Since D_n must be kept under a certain limit given by the node's I/O capacity, we believe that the second alternative is more appropriate to configure a video server.

We consider two values the group size: $D_c = 5$ and $D_c = 20$ for the remaining three schemes $Mirr_{one}$, Par_{some}, and $Mirr_{some-sub}$. Figures 17(a) and 17(b) depict the server reliability for $Mirr_{one}$, Par_{some}, and $Mirr_{some-sub}$ after one year and after three years of server operation, respectively. Table 4 shows for these schemes the normalized per stream cost with different values of α and with $D_c = 5$ and $D_c = 20$. We take again the per stream cost of $Mirr_{one}$ as base line for comparison and divide the cost values for the other schemes by



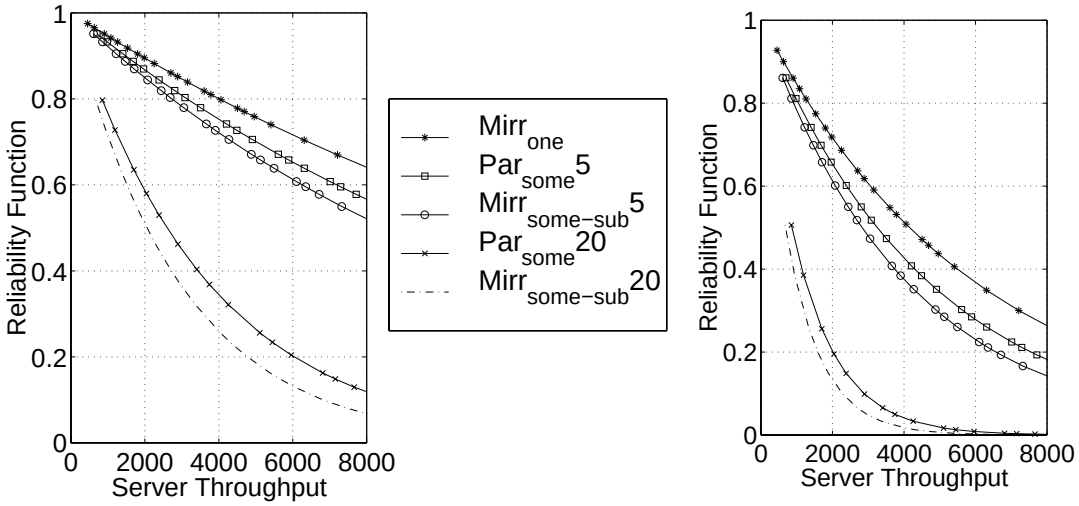
(a) $D_c = N = 5$ and $D_n = 20$.



(b) $D_c = 5$, $N = 10$, and $D_n = 10$.

Figure 16: Configuring a video server with $D = 100$ and $D_c = 5$.

the cost for $Mirr_{one}$.



(a) Server reliability after 1 year of server operation with $\lambda_d = \lambda_n = \frac{1}{100000 \text{ hours}}$ and $D_c = 5, 20$.

(b) Server reliability after 3 years of server operation with $\lambda_d = \lambda_n = \frac{1}{100000 \text{ hours}}$ and $D_c = 5, 20$.

Figure 17: Server reliability with $D_c = 5, 20$.

The results of Figure 17 and Table 4 are summarized as follows:

- The server reliability of $Mirr_{one}$ is higher than for the other two schemes. As expected, the server re-

	D_c	$\alpha = 26$	$\alpha = 130$	$\alpha = 5.2$
$Mirr_{one}$		1	1	1
Par_{some}	20	0.798	1.739	0.584
Par_{some}	5	0.695	0.879	0.653
$Mirr_{some-sub}$	20	0.678	0.717	0.671
$Mirr_{some-sub}$	5	0.745	0.771	0.739

Table 4: Normalized stream cost (by $Mirr_{one}$) for different values of α and D_c.

liability increases for both, Par_{some} and $Mirr_{some-sub}$ with decreasing D_c. In fact, as D_c decreases, the number of groups grows and thus the number of disk failures (one disk failure per group) that can be tolerated increases as well.

- Depending on the value of α, the impact of varying the group size D_c on the per stream cost differs for Par_{some} and $Mirr_{some-sub}$. For $Mirr_{some-sub}$, the cost per stream decreases as the group size D_c grows for all three values of α considered. Indeed, the sub-block size to be read during disk failure is inversely proportional to the value of D_c. Consequently, the server throughput becomes smaller for decreasing D_c and the per stream cost increases. For Par_{some} with $\alpha = 26$ and $\alpha = 130$, the per stream cost decreases as D_c decreases. However, this result is reversed with $\alpha = 5.2$, where the per stream cost of Par_{some} is higher with $D_c = 5$ than with $D_c = 20$. The following explains the last observation:

  1. A small value of α (e.g. $\alpha = 5.2$) signifies that the price for main memory decreases faster than the one for hard disk and therefore main memory does not *significantly* affect the per stream cost for Par_{some} independently of the group size D_c.
  2. As the group size D_c decreases, the amount of I/O bandwidth that must be reserved on each disk for the disk failure mode increases. Consequently, the throughput is smaller with $D_c = 5$ than with $D_c = 20$. As a result, the per stream cost for Par_{some} increases when the group size D_c decreases.
  3. Since the memory cost affects *only little* the per stream cost of Par_{some} for a small value of α, the weight of the amount of I/O bandwidth to be reserved on the per stream cost becomes more *visible* and therefore the per stream cost of Par_{some} increases as D_c decreases.

  Note that the per stream cost of Par_{some} ($D_c = 20$) is *lowest* for $\alpha = 5.2$, whereas it is *highest* for $\alpha = 130$.

- Par_{some} has always a higher server reliability than $Mirr_{some-sub}$. Further, for the values $\alpha = 26$ and $\alpha = 130$, Par_{some} has a higher cost per stream than $Mirr_{some-sub}$ given the same value of D_c. However, for the value $\alpha = 5.2$, Par_{some} becomes more cost effective than $Mirr_{some-sub}$.

- Based on the reliability results and for the high values of α, e.g. $\alpha = 26, 130$, we observe that a small group size ($D_c = 5$) considerably increases the server reliability and decreases the per stream cost for Par_{some}. For $Mirr_{some-sub}$, the server reliability increases as D_c decreases, but also the per stream cost slightly increases whatever the value of α is.

In summary, we have shown that the three schemes $Mirr_{one}$, Par_{some}, and $Mirr_{some-sub}$ are good candidates to ensure fault-tolerance in a video server. The Grouped One-to-One scheme $Mirr_{one}$ achieves a higher reliability than the other two schemes at the expense of the per stream cost that is about 1.5 times as high. For Par_{some}, the value of D_c must be small to achieve a high server reliability and a low per stream cost. For $Mirr_{some-sub}$, the value of D_c must be small to achieve a high server reliability at the detriment of a *slight* increase in the per stream cost.

6 Conclusions

In the first part we have presented an overview of several reliability schemes for distributed video servers. The schemes differ by the type of redundancy used (mirroring or parity) and by the distribution granularity of the redundant data. We have identified seven reliability schemes and compared them in terms of the server reliability and the cost per stream. We have modeled server reliability using Continuous Time Markov Chains that were evaluated using the SHARPE software package. We have considered both cases: independent disk failures and dependent component failures.

The performance study of the different reliability schemes led us to introduce a novel reliability scheme, called the Grouped One-to-One mirroring scheme, which is derived by the classical One-to-One mirroring scheme. The Grouped One-to-One mirroring scheme $Mirr_{one}$ outperforms *all* other schemes in terms of server reliability. Out of the seven reliability schemes discussed the $Mirr_{one}$, Par_{some}, and $Mirr_{some-sub}$ schemes achieve both, high server reliability and low per stream cost. We have compared these three schemes in terms of server reliability and per stream cost for several memory and hard disk prices and various group sizes. We found that the smaller the group size, the better the trade-off between high server reliability and low per stream cost.

A Calculation of $P^{(k)}$ for the One-to-One Scheme

We calculate in the following the probability $P^{(k)}$ that the video server that uses the One-to-One mirroring scheme does not fail after k disk/node failures.

Let us consider the suite of n units. Note that the term unit can denote a disk (used for the independent disk failure case) or a node (used for the dependent component failure mode). Since we want to calculate the probability that the server does not fail having k units down, we want those units not to be *adjacent*. Therefore we are looking for the sub-suites $i_1, i_2, \cdots, i_k \in [1, \cdots, n]$ such that:

$$\forall l \in [1..k-1], i_{l+1} - i_l > 1 \tag{10}$$

Let us call S the set of these suites. We introduce a bijection of this set of suites to the set $j_1, j_2, \cdots, j_k \in [1, \cdots, n-k+1]$ such that $j_1 = i_1, j_2 = i_2 - 1, j_3 = i_3 - 2, \cdots, j_k = i_k - (k-1)$.

Introducing the second suite j allows to suppress condition (10) on the suite i_l, since the suite j is strictly growing, whose number of elements are thus easy to count: it is the number of strictly growing functions from $[1..k]$ in $[1..n-(k-1)]$, that is $C(n-(k-1), k) = \frac{(n-k+1)!}{k!\cdot(n-2\cdot k+1)!}$

This result though doesn't take into account the fact that the units number 1 and n are adjacent. In fact, two scenarii are possible

- The first scenario is when unit 1 has already failed. In this case, units 2 and the n are not allowed to fail, otherwise the server will fail. We have then a set of $n-3$ units among which we are allowed to pick $k-1$ non-adjacent units. Referring to the case that we just solved, we obtain: $C(n-3-(k-2), k-1) = \frac{(n-k-1)!}{(k-1)!\cdot(n-2\cdot k-2)!}$
- The second scenario is when the first unit still works. In this case, k units are chosen among the $n-1$ remaining ones. This leads us to the value $C(n-1-(k-1), k) = C(n-k, k) = \frac{(n-k)!}{k!\cdot(n-2\cdot k)!}$

The number of possibilities N_k that we are looking for is given by : $N_k = C(n-1-k, k-1) + C(n-k, k)$

Consequently, for a given number k of failed units (disks/nodes), the probability $P^{(k)}$ that the server does not fail after k unit failures is calculated as:

$$P^{(k)} = \frac{C(n-1-k, k-1) + C(n-k, k)}{C(n,k)}$$

B Building the group state-space diagram for the Grouped One-to-One scheme

We show in the following how to build the state-space diagram of one group containing D_c disks for the Grouped One-to-One scheme. We focus on the transitions from state $[i,j]$ to higher states and back. We know that state $[i,j]$ represents the total number $(i+j)$ of disks that have failed inside the group. These failures are due to i disk failures and j node failures. We also know that the number of states in the state-space diagram is $2^{\frac{D_c}{2}+1}$. The parameters i and j must respect the condition: $i+j \leq \frac{D_c}{2}$, since at most $\frac{D_c}{2}$ disk failures are tolerated inside one group. We distinguish between the two cases: (i) $i+j < \frac{D_c}{2}$ and (ii) $i+j = \frac{D_c}{2}$. Figure 18 shows the possible transitions and the corresponding rates for the case (i), whereas Figure 19 shows the transition for the case (ii).

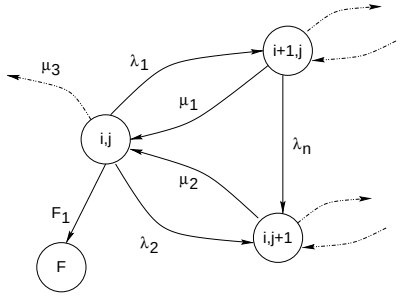


Figure 18: Transitions from state $[i,j]$ to higher states and back, for $(i+j) < \frac{D_c}{2}$.

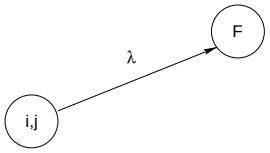


Figure 19: Transition to the failure state for $(i+j) = \frac{D_c}{2}$.

The parameters in the two figures have the following values: $\lambda_1 = (D_c - (i+j)) \cdot \lambda_d \cdot P^{(i+j+1)}$, $\lambda_2 = (N-(i+j)) \cdot \lambda_n \cdot P^{(i+j+1)}$, $F_1 = ((D_c-(i+j)) \cdot \lambda_d + (N-(i+j)) \cdot \lambda_n) \cdot (1 - P^{(i+j+1)})$, $\mu_1 = \mu_d$, $\mu_2 = \mu_n$, $\mu_3 = (\mu_d \ \text{OR} \ \mu_n)$ and $\lambda = (Dc - \frac{D_c}{2}) \cdot \lambda_d + (N - \frac{D_c}{2}) \cdot \lambda_n$.

References

[1] D. A. Patterson, G. Gibson, and R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," in *Proceedings of the 1988 ACM Conference on Management of Data (SIGMOD)*, (Chicago, IL), pp. 109–116, June 1988.

[2] F. A. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming raid(tm) – a disk array management system for video files," in *Proceedings of the 1st ACM International Conference on Multimedia*, (Anaheim, CA), August 1993.

[3] S. Berson, L. Golubchik, and R. R. Muntz, "Fault tolerant design of multimedia servers," in *Proceedings of SIGMOD'95*, (San Jose, CA), pp. 364–375, May 1995.

[4] S. Ghandeharizadeh and H. K. Seon, "Striping in multi-disk video servers," in *Proceedings in the SPIE International Symposium on Photonics Technologies and Systems for Voice, Video, and Data Communications*, 1995.

[5] J. Gafsi and E. W. Biersack, "Data striping and reliablity aspects in distributed video servers," *In Cluster Computing: Networks, Software Tools, and Applications*, vol. 2 (1), pp. 75–91, February 1999.

[6] B. Ozden *et al.*, "Disk striping in video server environments," in *Proc. of the IEEE Conf. on Multimedia Systems*, (Hiroshima, Japan), pp. 580–589, jun 1996.

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

[8] 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)*, (Hiroshima), pp. 144–150, June 1996.

[9] S. A. Barnett, G. J. Anido, and P. Beadle, "Predictive call admission control for a disk array based video server," in *Proceedings in Multimedia Computing and Networking*, (San Jose, California, USA), pp. 240, 251, February 1997.

[10] 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*, vol. 2, July 1994.

[11] L. Golubchik, J. C.-S. Lui, and M. Papadopouli, "A survey of approaches to fault tolerant design of vod servers: Techniques, analysis, and comparison," *Parallel Computing Journal*, vol. 24, no. 1, pp. 123–155, 1998.

[12] D. Bitton and J. Gray, "Disk shadowing," in *Proc. of the 14th int. conference on VLDB, L. A., Aug. 1988*, pp. 331–338, 1988.

[13] A. Mourad, "Doubly-striped disk mirroring: Reliable storage for video servers," *Multimedia, Tools and Applications*, vol. 2, pp. 253–272, May 1996.

[14] W. Bolosky *et al.*, "The tiger video fileserver," in *6th Workshop on Network and Operating System Support for Digital Audio and Video*, (Zushi, Japan), Apr. 1996.

[15] A. Merchant and P.-S. Yu, "Analytic modeling and comparisons of striping strategies for replicated disk arrays," *IEEE Transactions on Computers*, vol. 44, pp. 419–433, Mar. 1995.

[16] E. K. Lee, *Performance Modeling and Analysis of Disk Arrays*. PhD thesis, University of California at Berkley, 1993.

[17] 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*, 1994.

[18] 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*, vol. 26, pp. 145–185, June 1994.

[19] 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*, vol. 2, July 1994.

[20] J. Gafsi and E. W. Biersack, "Performance and cost comparison of mirroring- and parity-based reliability schemes for video servers," in *Proceedings of KiVS'99*, (Darmstadt, Germany), March 1999.

[21] H. I. Hsiao and D. J. DeWitt, "Chained declustering: A new availability strategy for multiprocessor database machines.," in *In Proceedings of the Int. Conference of Data Engeneering (ICDE), 1990*, pp. 456–465, 1990.

[22] L. Golubchik, J. C. Lui, and R. R. Muntz, "Chained declustering: Load balancing and robustness to skew and failures," in *In Proceedings of the Second International Workshop on Research Issues in Data Engineering: Transaction and Query Processing, Tempe, Arizona*, (Tempe, Arizona), pp. 88–95, February 1992.

[23] A. Mourad, "Issues in the design of a storage server for video-on-demand," *Multimedia Systems*, vol. 4, no. 2, pp. 70–86, 1996.

[24] E. K. Lee *et al.*, "Raid-ii: A scalabale storage architecture for high-bandwidth network file service," Tech. Rep. UCB/CSD 92/672, University of California, Berkeley, Feb. 1992.

[25] S. Ghandeharizadeh and S. H. Kim, "Striping in multi-disk video servers," in *Proc. High-Density Data Recording and Retrieval Technologies Conference*, SPIE, Oct. 1995.

[26] A. Cohen and W. Burkhard, "Segmented information dispersal (SID) for efficient reconstruction in fault-tolerant video servers," in *Proc. ACM Multimedia 1996*, (Boston, MA), pp. 277–286, Nov. 1996.

[27] 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.

[28] Y. Birk, "Random raids with selective exploitation of redundancy for high performance video servers," in *NOSSDAV97*, LNCS, Springer, May 1997.

[29] M.-S. Chen *et al.*, "Using rotational mirrored declustering for replica placement in a disk-array-based video server," *Multimedia Systems*, vol. 5, pp. 371–379, Dec. 1997.

[30] M. Malhotra and K. S. Trivedi, "Reliability analysis of redundant arrays of inexpensive disks," *Journal of Parallel and Distributed Computing*, vol. 17, pp. 146–151, 1993.

[31] G. A. Gibson, *Redundant Disk Arrays: Reliable, Parallel Secondary Storage*. PhD thesis, University of California at Berkley, December 1990.

[32] S. A. Barnett and G. J. Anido, "Performability of disk-array-based video servers," *Multimedia Systems*, vol. 6, pp. 60–74, 1998.

[33] R. A. Sahner, K. S. Trivedi, and A. Puliafito, *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic Publishers, 1996.

[34] A. Hoyland and M. Rausand, *System Reliability Theory: Models and Statistical Methods*, vol. 518. John Wiley and Sons, 1994.

[35] N. Randolph, *Probability, Stochastic Processes, and Queeing Theory*. Springer-Verlag, 1995.

[36] S. Ghandeharizadeh, S. H. Kim, C. Shahabi, and R. Zimmermann, *Placement of Continuous Media in Multi-Zone Disks*, ch. 2, pp. 23–59. Multimedia Information Storage and Management, Kluwer Academic Publishers, soon m. chung ed., 1996.