Performance and Reliability Study for Distributed Video Servers: Mirroring or Parity?

Jamel Gafsi and Ernst W. Biersack

Institut EURECOM
B.P. 193, 06904 Sophia Antipolis, FRANCE
{gafsi,erbi}@eurecom.fr

***Abstract*— We study video server performance and reliability. We classify several reliability schemes based on the redundancy technique used (mirroring vs. parity) and on the distribution granularity of redundant data. Then, we propose for each scheme its adequate data layout. To calculate the server reliability, we apply discrete modeling based on Markov chains. Further, we focus on the trade-off between achieving high reliability and low per stream cost. Our results show that, in contrast to intuition, for the same degree of reliability, mirroring-based schemes always outperform parity-based schemes in terms of per stream cost and also restart latency after disk failure. Our results also show that a mirroring scheme that copies original data of a single disk onto a subset of all disks significantly improves the server reliability and slightly increases the per stream cost as compared to the classical interleaved mirroring scheme.*

Keywords— **Distributed Multimedia, Video Servers, Reliability Modeling.**

I. Introduction

Video servers typically store large video files. As the number of files to be stored increases, the number of storage components required, typically SCSI hard disks, increases as well. However, the larger the number of disks, the more vulnerable to disk failures the video server becomes. In order to ensure uninterrupted service even in the presence of disk failures, a server must be able to reconstruct lost information. This can be achieved by using redundant information. Two reliability techniques are applied: **mirroring** [1, 2, 3, 4] and **parity** [5, 6, 7, 8]. Unlike parity, which adds a small storage overhead for parity data, mirroring requires twice as much storage volume as in the non-fault tolerant case. However, mirroring significantly simplifies the design and the implementation of video servers, since it does not require any synchronization of reads or additional processing time to decode lost information, which is the case for parity. In order to achieve higher reliability, added resources (storage volume, main memory space, and I/O bandwidth) are required. The amount of additional resources mainly depends on the **distribution granularity** of the reliability technique used. The distribution granularity is defined as the parity group size for parity and the number of disks across which original data of a single disk is replicated for mirroring. We explore in this paper the trade-off between reliability and per stream cost depending on both, the reliability technique used (parity/mirroring) and its distribution granularity. The video server is assumed to use *round-based scheduling*; the service time is divided into equal-size time intervals. Each admitted client is served once every time interval: the **service round**. Further, in order to optimize seek overhead, the server uses the SCAN algorithm for data retrieval from disks. Finally, the server contains N server nodes, to which D disk drives are connected. The server nodes are identical and each node contains D_n disks. Let us assume that D is a multiple of N as: $D = N \cdot D_n$. A very important design issue of a video server concerns the way data is distributed (striped) over its disks. We assume that each video object is partitioned into video blocks that are distributed over *all* disks of the server following a round robin fashion. Further, we adopt the so called Coarse Grained (CGS) striping algorithm that retrieves for one stream *one large* video block from a single disk during a service round. During the next service round, the *next* video block is retrieved from possibly a different disk [7, 9, 10, 11].

II. Reliability Schemes

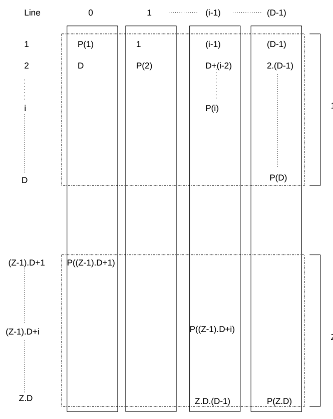
We classify reliability schemes depending on the technique used (parity/mirroring) and on the distribution granularity of redundant data.

II.-A Parity-Based Reliability

Parity-based techniques store *parity* data in addition to *original* data (RAID2-6). When a disk failure occurs, parity information is used to reconstruct the failed original data. RAID5 [5] requires a small amount of additional storage volume for each video object to protect against failure, since one parity block is needed for the $(D-1)$ original blocks. The $(D-1)$ original blocks and the parity block build a **parity group**. Figure 1 proposes a disk layout for RAID5. Figure 1(a) shows for a video server with D disks how one video object is stored using RAID5. The data placement within the server is represented by placing the numbers inside a matrix that contains D columns (*the disks*) and $(Z \cdot D)$ **retrieval lines** (*the parity groups*). Figure 1(b) shows the storage layout of original and parity blocks on the disk $(i-1)$ $(i \in [1, ..., D])$. $P(i)$ denotes the parity block of line i. We introduce for RAID5-based schemes a class called the **One-to-All** scheme.

In order to increase server reliability, clustering schemes are proposed in the literature. We consider only clustering schemes that divide the server into, normally homogeneous, C independent parity groups. The parity group size D_c is typically much smaller than the total number of disks D. Let us assume that $D = C \cdot D_c$. Since a parity group covers some and not all disks, we call this organization the **One-to-Some** scheme. We depict in Figure 2 a storage layout of original and parity data of the portion of a video object that is stored on parity group j (($j \in [1..C]$).

The main disadvantage of parity-based techniques is the need for additional buffer storage or I/O bandwidth resources when working in disk failure mode, since in the worst case the whole parity group must be retrieved and temporarily kept in the buffer to reconstruct lost data. We have distinguished in [11] between the second read strategy and the buffering strategy. The former doubles the I/O bandwidth re-


(a) Storage matrix of a video object.

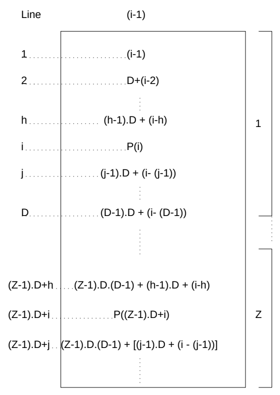

(b) Storage of a video object on disk i.

Fig. 1: One-to-All Parity data layout for one video object stored on a video server with D disks.

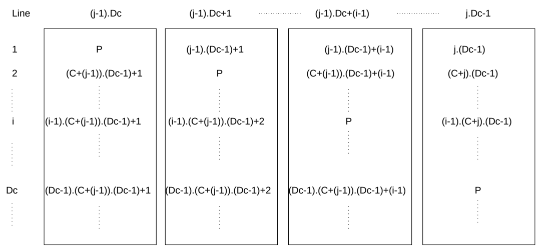


Fig. 2: One-to-Some Parity data layout of a portion of a video object on parity group j ($j \in [1..C]$).

quirement [12], whereas the latter requires D (D_c) times as much buffer space compared to when working with no disk failure. We will restrict our discussion to the buffering strategy, since it achieves higher throughput than the second read strategy.

II.-B Mirroring-Based Reliability

Based on the chained declustering [2], the so called **One-to-One** scheme replicates original blocks stored on disk i onto disk ($(i+1)$ mod D) ($i \in [0..D-1]$). This scheme can tolerate up-to $\frac{D}{2}$ disk failures (best case). However, it does not survive more than one disk failure if the two failed disks are *consecutive* (worst case). Additionally, the load of a failed disk is entirely shifted to a single disk, which results on load-imbalances during disk failure mode.

In [4], a mirroring scheme, based on the interleaved declustering, was proposed that uniformly distributes the load of a failed disk over all the remaining disks in the server (the *One-to-All* scheme). Figure 3 shows how original blocks of disk $(i-1)$ are replicated over the remaining $(D-1)$ server disks using the One-to-All scheme.

We depict in Figure 3 the storage layout of a One-to-All mirroring scheme.

Clusters can be also built for mirroring (**clustered mirroring**). Original data of one disk are only replicated over a sub-set and not all server disks (the *One-to-Some* scheme). We assume C independent groups, each containing D_c disks. Figure 4 proposes the data layout of the portion of one video object that is stored on cluster j using the One-to-Some scheme. The Figure also shows how original data of disk $(j-1) \cdot D_c + (i-1)$ are replicated over the remaining disks inside group j.

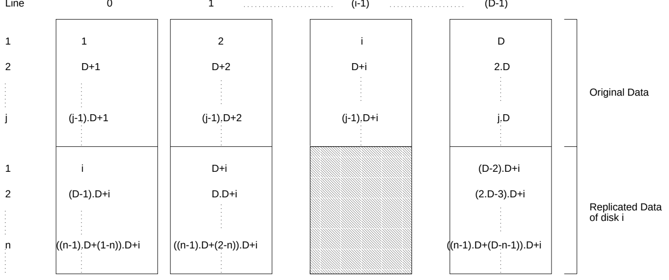


Fig. 3: One-to-All mirroring data layout for one video object stored on a video server with D disks.

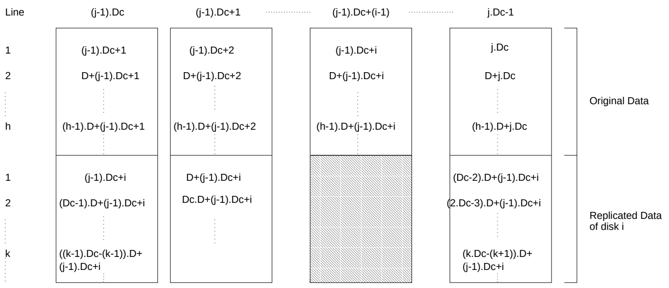


Fig. 4: One-to-Some mirroring data layout of a portion of a video object on group j

II.-C Classification of Reliability Schemes

Table 1 gives an overview of the different schemes described:

Distribution Granularity	Reliability Scheme	
	Mirroring-based	Parity-based
One-to-One	Chained declustering	XXX
One-to-All	Interleaved declustering	RAID5-based
One-to-Some	Clustered mirroring	Clustered parity

Table 1: Classification of Reliability Schemes and Distributions

Based on the classification in Table 1, we will compare the different schemes in the remainder of the paper. We focus in section III. on server performance that includes server throughput, resource (storage, memory, and I/O bandwidth) requirements, and per stream cost. Section IV. investigates quality of service including restart latency after disk failure and server reliability. Server reliability is evaluated based on discrete Markov models. We will distinguish between independent disk failures and the more complicated modeling of dependent component failures. The results of section III. and IV. lead to the conclusions of this paper, which we present in section V..

III. Server Performance

The main performance metric is the maximum number of streams Q_s that the server can simultaneously admit, which is called server throughput. Adding fault tolerance within the server requires additional resources in terms of storage volume, buffer space, and I/O bandwidth capacity and therefore increases server cost.

III.-A Throughput

The maximum number of streams Q_d that one disk can admit without assuming fault-tolerance in a video server is $Q_d = \frac{\frac{b_{dr}}{r_p} - 2 \cdot t_{seek}}{\frac{b_{dr}}{r_d} + t_{rot} + t_{stl}}$ [11]. Thereby, $r_p (= 1.5 \ \mathrm{Mbps})$ denotes the video playback rate, $r_d (= 40 \ \mathrm{Mbps})$ is the disk's transfer rate, $t_{stl} (= 1.5 \ \mathrm{ms})$ is the settle time, $t_{seek} (= 20 \ \mathrm{ms})$ is the maximum seek time, $t_{rot} (= 11.11 \ \mathrm{ms})$ is the worst case rotational latency, and $b_{dr} (= 1 \ \mathrm{Mbit})$ denotes the block size. The parameter values are those of the Seagate Swift SCSI disks [13]. Adding fault-tolerance, each disk reserves a portion of its available bandwidth to be used during disk failures. For each disk, the amount of bandwidth reserved for failure mode does not depend on the reliability technique used, but depends only on the *distribution granularity* of redundant data. Let us take the One-to-All scheme to better illustrate this statement. For parity, after one disk has failed, at most one parity block is needed for each parity group (1 parity block for the $(D-2)$ remaining original blocks). For mirroring, at most one replicated block is needed during disk failure for each *line* (see Figure 3). Consequently, for each parity group (line) and each stream, one additional block must be retrieved in the presence of a disk failure assuming the One-to-All scheme (parity or mirroring). Analogously, the number of additional blocks needed is the same for parity and mirroring assuming the One-to-Some scheme. Therefore, given a certain distribution granularity the amount of I/O bandwidth that must be reserved for failure mode and thus the throughput are the same for parity and mirroring. We show in Table 2 the server throughput Q_s for different distribution granularities (One-to-One, One-to-Some, and One-to-All) [11]. The throughput values in table 2 consider the average case and not the worst case.

Distribution Scheme	Throughput Q_s
One-to-One (Mirroring)	$\frac{D \cdot Q_d}{2}$
One-to-All (Mirroring/Parity)	$D \cdot \left\lfloor \frac{D-1}{D} \cdot Q_d \right\rfloor$
One-to-Some (Mirroring/Parity)	$D \cdot \left\lfloor \frac{D_c-1}{D_c} \cdot Q_d \right\rfloor$

Table 2: Server throughput for different distribution granularities

Since the half of the available disk bandwidth must be reserved for failure mode for the One-to-One scheme, the throughput is cut into half as compared with the non-fault tolerant case. Thus, we will limit our discussion in the remainder of this paper to the One-to-All and the One-to-Some schemes.

III.-B Buffer Requirement

Considering the SCAN scheduling algorithm, the worst case buffer requirement for one served stream is twice the block size b_{dr}. Thus, the buffer requirement B_s for a server with Q_s concurrent streams and working without disk failure is $B_s = 2 \cdot b_{dr} \cdot Q_s$. Mirroring replicates original blocks belonging to a single disk over all or a sub-set of disks. During disk failure mode, blocks that would have been retrieved from the failed disk are retrieved from the corresponding disks. Thus, original blocks are replaced by mirrored blocks. Accordingly, mirroring requires the same amount of buffer B_s during normal operation mode and during disk failure mode. Further, the same buffer is required for all mirroring distribution granularities. On the other hand, parity needs to perform a X-OR operation over a set of blocks in order to reconstruct a lost block. In fact, during normal operation mode, the buffer is immediately liberated after consumption. When a single disk fails, original as well as parity blocks that belong to the same parity group are sequentially retrieved (during consecutive service rounds) from consecutive disks and must be *temporarily stored* on the buffer (for many service rounds) until the lost original block is regenerated. Since buffer overflow must be avoided, the buffer requirement is calculated for the worst case situation, where the whole parity group should be contained in the buffer to reconstruct the lost block. Further, a buffer size of one disk retrieval block should be additionally reserved to store the first retrieved block of the next parity group. Thus, the total buffer requirement during disk failure is $D \cdot b_{dr} \cdot Q_s = \frac{D \cdot B_s}{2}$ for the One-to-All parity scheme and is $D_c \cdot b_{dr} \cdot Q_s = \frac{D_c \cdot B_s}{2}$ for the One-to-Some parity scheme.

III.-C Per Stream Cost

To get the stream cost, we proceed as follows. We calculate the total server costs $\$_{server}$ as the cost of the storage volume V (hard disks) and the main memory volume B (buffer requirements) when operating in disk failure mode as: $\$_{server} = P_{mem} \cdot B + P_{disk} \cdot V$, where P_{mem} denotes the price of $1 \ \mathrm{Mbyte}$ of main memory and P_{disk} represents the price of $1 \ \mathrm{Mbyte}$ of hard disk. Typical price figures are $P_{mem} = 13\$$ and $P_{disk} = 0.5\$$. Since these prices change very fast, we will consider the relative costs by introducing the cost ratio α between P_{mem} and P_{disk}: $P_{mem} = \alpha \cdot P_{disk}$. The server cost becomes therefore: $\$_{server} = P_{mem} \cdot B + \frac{P_{mem}}{\alpha} \cdot V = P_{mem} \cdot \left(B + \frac{V}{\alpha}\right)$. The per stream cost is obtained by dividing the total server cost $\$_{server}$ by the overall server throughput Q_s: $\$_{stream} = \frac{\$_{server}}{Q_s} = \frac{P_{mem} \cdot \left(B + \frac{V}{\alpha}\right)}{Q_s}$

III.-D Performance Comparison

We compare mirroring and parity techniques assuming different distribution granularities in terms of their per stream costs. Given a value of D, we calculate for each scheme the throughput that can be reached as well as the amount of buffer needed. Then we derive the per stream cost for each scheme. For the One-to-Some schemes (parity/mirroring), we take $D_c = 10$ as a good trade-off between server throughput and disk/buffer requirement.

Figure 5 shows the per stream cost results for the One-to-All parity scheme (Par_{O2A}), the One-to-Some parity scheme (Par_{O2S}), the One-to-Some mirroring scheme (Mir_{O2S}), and the One-to-All mirroring scheme (Mir_{O2A}). The parameter α takes the value $\alpha = \frac{13}{0.5} = 26$, which is the actual memory/disk ratio. We observe that the two mirroring schemes (One-to-All mirroring and One-to-Some mirroring) have *lower per stream cost* than the two parity schemes (One-to-All parity and One-to-Some parity. Further, the One-to-Some parity scheme has a much lower per stream cost than the One-to-All parity scheme (lower buffer requirement). On the other hand, the One-to-Some mirroring scheme provides a slightly higher per stream cost than the One-to-All mirroring scheme (the same resource requirements, but a smaller throughput).

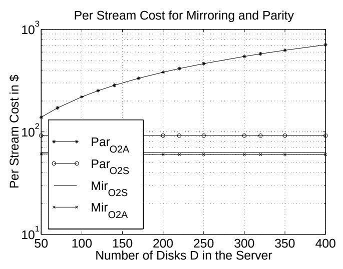


Fig. 5: Per stream cost.

IV. Quality of Service

IV.-A *Restart Latency*

The restart latency denotes the time that elapses between the service interruption due to disk failure and the time data is reconstructed. Since mirroring (One-to-All and One-to-Some) retrieves the replication of a lost block after a disk failure, the worst case restart latency T_r equals the service round duration ($T_r = \tau$). However, parity needs in the worst case to retrieve all surviving blocks and the parity block of a parity group to reconstruct the lost block. Since the retrieval of each block occurs in a separate service round (CGS-based retrieval), the worst case restart latency is $T_r = D \cdot \tau$ for the One-to-All parity scheme and is $T_r = D_c \cdot \tau$ for the One-to-Some parity scheme.

IV.-B *Reliability Modeling*

We define the server reliability at time t as the probability that the server survives until time t assuming that all server components are initially operational. The server survives as long as its working components deliver the functionality expected. Server reliability depends on the distribution granularity applied, since the latter determines the number of components that are allowed to fail without making the server fail. However, *server reliability does not depend on whether mirroring is used or parity.*

We use a **Markov-chain** to model the server reliability [14, 15]. The disk mean time to failure $MTTF_d$ takes a relatively pessimistic value (100000 hours) and the disk mean time to repair $MTTR_d$ takes the value of 72 hours.

We use a *discrete Markov model*: We divide the time in steps and we examine the behavior of the model only at the beginning of each step. The use of a discrete Markov model is justified since we deal with very high values of $MTTF_d$ (about 11.5 years) and the time between two time steps is relatively small (1 day). Furthermore, we use the following Taylor expansion of the exponential function for $t = 1$: P$[\text{ a disk will fail during the next day }] = 1 - e^{-\lambda_d} = \lambda_d$. To build the *transition matrix* of the discrete Markov model, we calculate the probability $p_{i,j}(t_n)$ that the server changes from state i to state j between time steps t_{n-1} and t_n. Note that $p_{i,i}(t_n)$ is the probability that the server remains in state i between time steps t_{n-1} and t_n and is defined as: $p_{i,i}(t_n) = 1 - \sum_{j=0, j\neq i}^{s-1} p_{i,j}(t_n)$. Finally, $p_i(t_n)$ denotes the probability that the server is in state i at time t_n. Let $v^{(t_n)}$ be the state vector of the server: $v^{(t_n)} = \big(p_0(t_n), p_1(t_n), \cdots, p_{(s-1)}(t_n)\big)$. The video server works at time $t_0 = 0$ and all its components work. Thus, $v^{(0)} = (1, 0, \cdots, 0)$. From the Markov model, we derive the transition matrix $M^{(0)}$ at the time step t_0. Given the initial vector $v^{(0)}$ and the initial transition matrix $M^{(0)}$, we can calculate the value of the reliability function $R^s(t_n)$ at every time step t_n: We calculate the state-probability vector $v^{(t_n)}$ as $v^{(t_n)} = v^{(0)} \cdot M^{(t_{n-1})}$. Having the values of the vector $v^{(t_n)}$, we then calculate $R^s(t_n)$ at time step t_n as $R^s(t_n) = \sum_{i=0}^{s-2} p_i(t_n) = 1 - p_{(s-1)}(t_n)$. Given $R^s(t_n)$, the mean server lifetime $MTTF_s$ is approximated as $MTTF_s = \sum_{n=0}^{\infty} R^s(t_n)$.

IV.-B.1 Independent Disk Failures

If we consider independent disk failures, the One-to-All scheme is not able to recover lost data after two consecutive disk failures. The transition matrix $M^{(0)}$ at time step t_0 is:

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

The corresponding Markov model is shown in Figure 6, where: $\lambda_1 = D \cdot \lambda_d$, $\lambda_2 = (D-1) \cdot \lambda_d$, and $\mu = \mu_d$

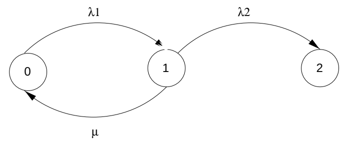
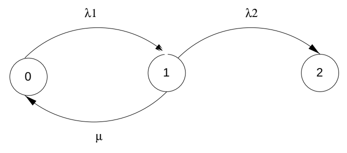


Fig. 6: Markov Model for the One-to-All-Assignment Scheme.

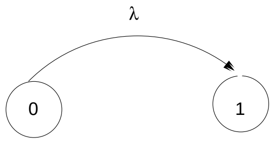
The server reliability function is then: $R^s(t_n) = \sum_{i=0}^{1} p_i(t_n) = p_{(0)}(t_n) + p_{(1)}(t_n) = 1 - p_{(2)}(t_n)$

The One-to-Some scheme behaves differently. In fact, the server fails if one of the groups (the first one) fails. Figure 7(a) shows the model for one group and Figure 7(b) shows the server level model.

The corresponding values of the parameters used in figure 7(a) are: $\lambda_1 = D_c \cdot \lambda_d$, $\lambda_2 = (Dc - 1) \cdot \lambda_d$, and $\mu = \mu_d$



(a) Markov Model for a Group of the One-to-Some-Assignment Scheme.



(b) Server Markov Model of the One-to-Some-Assignment Scheme.

Fig. 7: Markov Model for the One-to-Some-Assignment Scheme.

The transition matrix $M_c^{(0)}$ for a single group at time step t_0 is:

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

As $v^{(0)} = (1,0,0)$, the reliability function $R(t_n)^c$ of a single group is: $R^c(t_n) = \sum_{i=0}^{1} p_i(t_n) = p_{(0)}(t_n) + p_{(1)}(t_n) = 1 - p_{(2)}(t_n)$.

The group mean lifetime $MTTF_c$ is therefore: $MTTF_c = \sum_{n=0}^{\infty} R^c(t_n) = \sum_{n=0}^{\infty}(1-p_{(2)}(t_n))$

Let us now calculate the server reliability function. We assume that the group failure distribution is also exponential. The parameter λ in Figure 7(b) is then: $\lambda = C \cdot \lambda_c$, where λ_c denotes the already calculated group failure rate: $\lambda_c = \frac{1}{MTTF_c}$.

We derive the server transition matrix $M_s^{(0)}$ at time step t_0 as follows:

$$M_s^{(0)} = \begin{pmatrix} 1 - C\cdot\lambda_c & C\cdot\lambda_c \\ 0 & 1 \end{pmatrix}$$

Hence the server reliability function $R^s(t_n)$: $R(t_n)^s = 1 - p_{(1)}(t_n)$

IV.-B.2 Dependent Component Failures

As we described in section I., our video server consists of a set N of nodes, each containing D_n disks. Disks belonging to the same server node share common hardware. When a node fails due to interface, hardware, or controller failure, all disks that are connected to it can not deliver data any more and are considered as failed disks. We show in this section the impact of complete node failures on the server reliability for both, the One-to-All and the One-to-Some scheme.

Like a disk failure, a node failure is assumed to be repairable and the repair rate μ_n is exponentially distributed. The node failure rate λ_n is also an exponential function. Let $\lambda_n = \frac{1}{MTTF_n}$ be the failure rate of a server node, where $MTTF_n = 100000$ hours is the node mean life time. Further, let $\mu_n = \frac{1}{MTTR_n}$ be the repair rate of a failed server node, where $MTTR_n = 72$ hours is the node mean repair time.

Since the copies of original data stored on one disk are spread over *all* remaining disks of the server for the one-2-all scheme, a node failure is not tolerable. Figure 8 shows the Markov model for this scheme with dependent component failure. The server fails as far as two consecutive disk failures or a node failure occur.

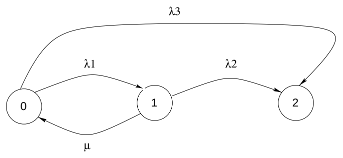


Fig. 8: Markov Model for the One-to-All-Assignment Scheme with Dependent component Failures.

The states 0, 1, and 2 denote respectively the initial state, the one disk failure state, and the server failure state.

The values of the parameters in figure 8 are:

- $\lambda_1 = D \cdot \lambda_d$: The probability that the first disk out of the D disks fails.
- $\lambda_2 = (D-1)\cdot\lambda_d + N\cdot\lambda_n$: The probability that the second disk out of the remaining $(D-1)$ disks or the first node fails.
- $\lambda_3 = N\cdot\lambda_n$: The probability that the first node fails.
- $\mu = \mu_d$.

The transition matrix $M^{(0)}$ at time step t_0 is

$$M^{(0)} = \begin{pmatrix} 1-\lambda_1-\lambda_3 & \lambda_1 & \lambda_3 \\ \mu & 1-\mu-\lambda_2 & \lambda_2 \\ 0 & 0 & 1 \end{pmatrix}$$

For the One-to-Some scheme, we assume that each group contains exactly *one* disk from each server node. This method was introduced in [16] and also used in [17] to build a cluster of disks. The authors called this scheme the **orthogonal RAID** (only for parity). The advantage of the orthogonal RAID is that it allows complete nodes to fail without loosing any data. In [16, 17], orthogonal RAID was used for a parity-based server. In this paper, we apply this orthogonality principle for both, parity and mirroring and call it the *orthogonal One-to-Some* scheme.

Based on the assumption above, we first model the reliability for a single group (Figure 9) and then the server reliability using again Figure 7(b)).

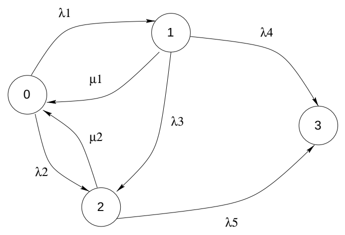


Fig. 9: Markov Model for one group with dependent component failures.

The states 0, 1, 2, and 3 of Figure 9 denote respectively the initial state, the one disk failure state, the one node failure state, and the group failure state. The parameter values used are:

- $\lambda_1 = D_c \cdot \lambda_d$: The probability that the first disk failure occurs inside the group (One out of the D_c disks of the group).

- $\lambda_2 = N \cdot \lambda_n$: The probability that the first node failure occurs. A complete node failure means for one group the failure of a single disk.
- $\lambda_3 = \lambda_n$: The probability that the node fails, which the already failed disk belongs to.
- $\lambda_4 = (D_c - 1) \cdot \lambda_d + (N - 1) \cdot \lambda_n$: The probability that the second disk of the group fails or the first node fails that does not contain the already failed disk.
- $\lambda_5 = (D_c - 1) \cdot \lambda_d + (N - 1) \cdot \lambda_n$: The probability that the second node fails or another disk of the group fails.
- $\mu_1 = \mu_d$, and $\mu_2 = \mu_n$.

The group transition matrix $M_c^{(0)}$ at time step t_0 is:

$$M_c^{(0)} = \begin{pmatrix} 1-\lambda_1-\lambda_2 & \lambda_1 & \lambda_2 & 0 \\ \mu_1 & 1-\mu_1-\lambda_3-\lambda_4 & \lambda_3 & \lambda_4 \\ \mu_2 & 0 & 1-\mu_2-\lambda_5 & \lambda_5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Having the corresponding Markov models (the initial transition matrices) for the dependent component failure case, we derive the server reliability for the One-to-All and the One-to-Some schemes as previously shown in section IV.-B.1.

IV.-C Reliability Comparison

We have shown in section III.-D that mirroring outperforms parity in terms of per stream cost. Now, we consider one reliability technique (mirroring or parity) and focus on different distribution granularities. Based on the models presented in section IV.-B, we compare the One-to-All and the One-to-Some schemes in terms server reliability. We present the reliability results for both cases, independent disk failures and dependent component failures. Assuming the latter case, we will evaluate the benefits of the orthogonal One-to-Some scheme as compared to the One-to-All scheme.

IV.-C.1 Independent Disk Failures

Figure 10 shows the server reliability for the One-to-All and the One-to-Some schemes assuming independent disk failures. We observe that the One-to-Some scheme has a higher server reliability than the One-to-All scheme. For instance, the server reliability after 2 years of operation is 0.81 for the One-to-Some scheme and is only 0.33 for the One-to-All scheme.

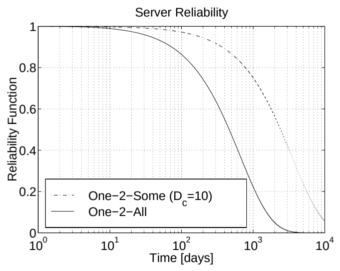
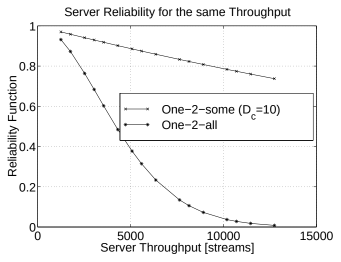
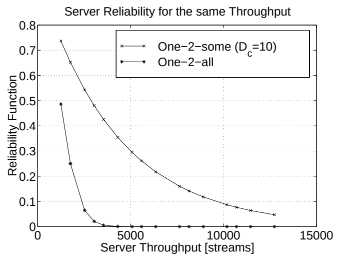


Fig. 10: Server reliability for One-to-All and One-to-Some for the same number of disks $D = 100$ assuming independent disk failures.

In Figure 11 we calculate the server reliability for both schemes after t_n days of non-interrupted operation while assuming the same throughput. We remind that in order to achieve the same throughput, the One-to-Some and the One-to-All schemes require approximately the same amount of disks (see Table 2). The parameter t_n takes the values 182 (Figure 11(a)) and 1825 (Figure 11(b)). We observe that the server reliability decreases for both schemes as the server throughput increases (number of disks required grows). The One-to-Some scheme has for both values of t_n a higher server reliability than the One-to-All scheme. Further, the decrease of server reliability is more dramatic for the One-to-All scheme than for the One-to-Some scheme.



(a) Server reliability after 6 months of operation ($t_n = 182$).



(b) Server reliability after 5 years of operation ($t_n = 1825$).

Fig. 11: Server reliability of One-to-All and One-to-Some for the same throughput assuming independent disk failures.

IV.-C.2 Dependent Component failures

Based on the models described in Figures 8 and 9, we plot in Figure 12 the server reliability for One-to-All and the orthogonal One-to-Some schemes. It is shown that the latter performs best regarding server reliability. As an example, the server reliability for the orthogonal One-to-Some scheme after 2 years of server operation is about 0.61. The One-to-All scheme, however, achieves a very low server reliability (0.05).

Figures 13(a) ($t_n = 6$ months) and 13(b) ($t_n = 5$ years) show that the orthogonal One-to-Some scheme performs best in terms of server reliability when assuming the same throughput. We observe that for the dependent case, already for small video servers (small number of disks), the difference in terms of server reliability between the orthogonal One-to-Some scheme and the One-to-All scheme is significant. In fact, even for small video severs, the server reliability for the One-to-All scheme is very low and decreases fast as throughput increases and also as t_n grows. If we compare the results in Figures 11 and 13 for the same value of

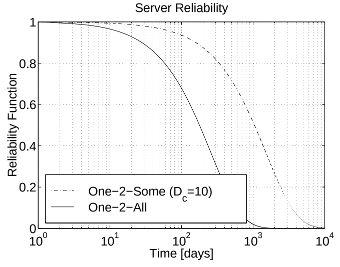
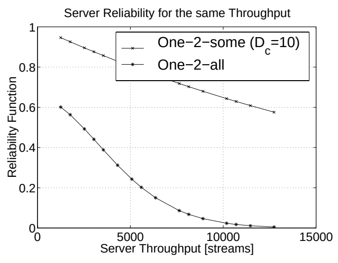
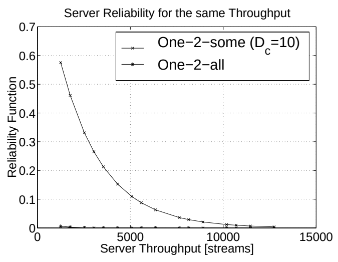


Fig. 12: Server reliability for One-to-All and One-to-Some for the same number of disks $D = 100$ assuming dependent component failures.



(a) Server reliability after 6 months of operation ($t_n = 182$).



(b) Server reliability after 5 years of operation ($t_n = 1825$).

Fig. 13: Server reliability of One-to-All and One-to-Some for the same throughput (dependent component failures).

t_n, we realize that the server reliability is lower for the case of dependent component failures. We also notice the dramatic degradation of the server reliability for the One-to-All scheme in the dependent case compared the the independent case, which is not observed for the orthogonal One-to-Some scheme, which allows a complete node to fail.

V. Conclusion

We have discussed several reliability and distribution techniques and proposed their adequate data layouts. Further, we have compared the performance and the reliability of the schemes considered. We have seen that for the same distribution granularity, the mirroring-based technique outperforms the parity-based technique in terms of the cost per stream and the worst case restart latency. We have also shown using discrete Markov modeling that the One-to-Some scheme *considerably* increases the server reliability as compared to the One-to-All scheme. For the case of dependent component failures, our results show that the orthogonal One-to-Some scheme, which survives even a complete node failure, has a much higher server reliability than the One-to-All scheme. In summary, we believe that the orthogonal One-to-Some mirroring scheme is the best scheme since it has the lowest cost per stream and the highest server reliability.

Acknowledgement

Eurecom's research is partially supported by its industrial partners: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

References

[1] E. K. Lee, *Performance Modeling and Analysis of Disk Arrays*. PhD thesis, University of California at Berkley, 1993.
[2] 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.
[3] 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.
[4] A. Mourad, "Doubly-striped disk mirroring: Reliable storage for video servers," *Multimedia, Tools and Applications*, vol. 2, pp. 253–272, May 1996.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] J. Gafsi and E. W. Biersack, "Data striping and reliablity aspects in distributed video servers," *In Cluster Computing: Networks, Software Tools, and Applications*, February 1999.
[12] 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.
[13] Seagate Disc Home, http://www.seagate.com/disc/disctop.shtml.
[14] A. Hoyland and M. Rausand, *System Reliability Theory: Models and Statistical Methods*, vol. 518. John Wiley and Sons, 1994.
[15] 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.
[16] G. A. Gibson, *Redundant Disk Arrays: Reliable, Parallel Secondary Storage*. PhD thesis, University of California at Berkley, December 1990.
[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.