

CHAPTER 9

MODELING A MULTIMEDIA SYSTEM FOR VOD SERVICES

Antonio Puliafito, Salvatore Riccobene,
Giancarlo Iannizzotto, Lorenzo Vita

9.1. INTRODUCTION

The constant increase in the performance and economic accessibility of modern calculation and communication systems has directed the interest of both researchers and industry towards the implementation of new forms of highly interactive multimedia communication such as Video on Demand. These systems belong to the Continuous Media (CM) category¹, which feature a continuous flow of data from a service manager (mass storage system) to a multimedia user interface (client).

The implementation of CM systems, however, shows up certain important limits in current mass memory technology, which is essentially electromechanical. For some time now, in fact, disks have constituted serious bottlenecks for performance in certain applications. The recent spread of parallel processing systems and multimedia applications has increased these limits even further.

Besides the inherent technological limitations, one of the main limits of mass storage systems derives from the fact that up to now time constraints have not been considered as high-priority requirements. Designers have mainly concentrated on forms of organization that are capable of ensuring a good trade-off between reliability and average

performance. The insertion of a new user (new connection) thus leads to a general reduction in the QoS already connected users are offered.

VOD systems², on the other hand, require the transfer of large (and often fixed) amounts of data with strong time constraints (deadlines): a management policy which, in case of insertion of a new user, degrades the QoS of previously connected users is definitely unacceptable. The system has to guarantee each user the bandwidth he requires for the application he is using, from the moment he is inserted to the end of his worksession, with no kind of degradation, unless it is due to accidental events.

A possible solution to the problem of mass memories is given by the parallelization of I/O subsystems by using Disk Array techniques³. Disk Arrays are a set of hard disks organized in such a way that the various I/O requests can be served either by different units (parallelization of requests) or by several units at the same time (parallelization of service). The current trend is to organize them in such a way that both kinds of parallelization are possible: in any case, the main goal consists of the uniform distribution of the workload over all the units of the system.

The aim of this chapter is to define and analyze a disk array system for the storage of specific data for CM applications. Focusing on VOD service systems in particular, a redundancy scheme based on an Information Dispersal Algorithm (IDA)⁴ is used. A Petri Net model is developed to analyze performance, assessing the overhead introduced and the distribution of response times for a generic read request.

The chapter is organized as follows: Section 9.2 introduces the problem of storing data for VOD applications, describes the solution proposed and outlines the main features of the IDA coding used. Sections 9.3 and 9.4 present the Petri Net model of the system and the results obtained. The authors' conclusions and suggestions for future research are given in Section 9.5.

9.2. DISK ARRAY FOR MULTIMEDIA SERVERS

As compared with traditional applications, VOD services enjoy a series of specific features, mainly connected with typical properties of the data treated:

- the transfer rate required is particularly high;
- the data flow has to be kept more or less continuous and is subject to very strong time constraints, unlike common applications;
- as the data transferred is mainly audio/video sequences, it is highly likely that a request for data concerning a certain set of video frames will be followed by a request for the frames which come immediately afterwards. It is therefore reasonable to schedule these requests in anticipation by means of simple prefetching algorithms, thus optimizing and reducing service times;
- the system writing phase takes place off-line or at times when the load is low, so requests subject to time constraints are almost exclusively read requests.

A system designed for the management of VOD services therefore has to be based on an architecture which is able to exploit these particular features.

One possible architecture², divided into three logical levels, is composed as follows: the lowest logical level comprises high-capacity mass storage devices with a low access speed and a low cost/storage ratio. This level deals with long-term storage of film sequences.

The central level is the real server of the system and comprises devices which have to offer not only a high storage capacity, but also considerable access speed. These devices also have to be able to manage a large number of connections at the same time. It is thus reasonable to envisage the use of disk array technology.

The third level refers to the end user or user group with particular features in common and provides an access interface with extremely high-speed but low-capacity memory devices (something similar to cache memories) for the temporary storage of data.

In this chapter we will concentrate exclusively on the middle level, describing an architecture based on disk-array technology designed specifically for VOD services.

The main feature of the organization proposed is the adoption of IDA coding⁴ in the data storage phase (a more detailed description of IDA is given in Appendix A).

Up to now the main advantage of IDA coding was its potential for enhancing system reliability. The possibility of inserting an arbitrary

quantity of redundancy in data treatment, in terms of both storage and transmission, provides high MTBF values for any system. However, with particular reference to mass memory storage systems it should be pointed out that the use of this technique has not been very successful so far. This is surely due to the fact that it is not possible to reconcile the features of IDA with the needs of a general-purpose disk array.

One major drawback lies in the fact that the probability of faults occurring simultaneously in more than one unit is not really very high. With RAID5³ technology, as spare disks are available, a faulty disk can be replaced very quickly, thus reducing to a minimum the time window during which a fault in a second unit would necessarily cause failure of the whole system. In most applications using a single additional unit to record is sufficient and economically feasible.

Another drawback of using IDA is that there is a certain degradation in performance in write operations, as each operation requires access to all the units in the group. Operations concerning a limited amount of data are therefore greatly penalized.

In this chapter we want to show that IDA coding can be used efficiently with disk array systems and performance can even be enhanced if the workload applied has certain characteristics. The use of a particular management policy related to the typical features of VOD systems, in fact, makes it possible to eliminate the drawbacks outlined above, to the advantage of performance.

The load generated by systems for the management of VOD services consists almost exclusively of read requests, each of which refers to large amounts of data. Write operations, on the other hand, are sporadic and generally take place off-line. In addition, as for read requests, they refer to large amounts of data. This means that the performance drawbacks described above - due to the size of the blocks IDA operates on - are no longer a problem.

IDA coding provides a large number of units from which the same data can be read; a read operation can thus be performed on the fastest units and the system's response times are considerably reduced. The choice does not necessarily have to be made a priori, but can be run-time.

Let us assume, for instance, a system with N disks. They do not all need to be used to store the same file. If, in fact, the value of N is high, it is more convenient to store a file on a subset, called a group, within which IDA coding is used. If G is its size, we get $G = R + B$,

where R is the redundancy introduced and B is the base to which this redundancy is applied. If we set $G = 6$ units, for example, and we want a 50% redundancy we will have $B = 4$ and $R = 2$. The files will therefore be subdivided in series of four blocks. Coding is applied to each block, thus obtaining the 6 blocks to be stored on the six disks in the group. Thanks to the IDA technique any subset of B disks can be chosen from the G making up the group to reconstruct the original information. It is clear that as the value of R increases, the number of sets data can be read from increases considerably. The idea is therefore to send a read request to all the disks in the group and wait for the B fastest ones, thus preventing overloaded disks from slowing the system down. For the remaining $G - B$ disks (which are R), there are two possibilities:

1. the corresponding subrequest is being served; in this case, even though an abort operation is still possible, there will be a functioning overhead (the disk will be working unnecessarily);
2. the subrequest is still queued. In this case it can be removed without uselessly loading the relative disk.

The aim of the following section is therefore to determine how much useless work is done by the system disks if such a management policy is used.

9.3. MODEL OF THE SYSTEM

In this paragraph we develop a model of the disk array proposed in order to evaluate the overhead introduced by the management policy used.

Of the various available methods we chose modeling by Stochastic Reward Nets (SRNs)⁵, which associate the various markings with reward indexes, so as to obtain effective evaluation indexes. The system examined comprises a set of 12 disks in groups of three units each. The division of the disks into groups was made on the basis of the Partial Dynamic Declustering (PDD) policy described in [6]. The division is not a static one: groups are formed dynamically, according to the current workload conditions on the various units. The main advantage of

dynamic management is efficient distribution of the workload over the mass storage subsystem.

The use of IDA coding does not entail any significant change in the PDD technique; indeed, it is quite transparent to it. Fig. 9.1 shows the model of a generic group comprising three units. For the IDA coding we fixed $N = 3$ and $R = 1$, so $B = 2$. A generic read request addressed to the group is split into three subrequests, each addressed to all the disks. The request can be considered to have been served when two units have terminated their part. The model will also have to show the behaviour of the third (and last) subrequest, once the other two have been served. In other words, it is necessary to assess whether it is still queueing or already being served. As the disks also belong to other groups, in fact, as provided for by PPD, the corresponding subrequests proceed asynchronously up the various queues, so both situations are possible. To model this behaviour, the queue for a generic disk has been described using a set of places. Each disk has two types of traffic - one relating to the group being examined and the other to all the other groups it belongs to. Below we will indicate the former as internal and the latter as external.

External requests follow the flow $T_{ext}, Pa, ta, Pb, T_{Serv.a}$. The queue for these requests is subdivided into two places, Pa and Pb . If there are no internal requests queued, ta is enabled and the queued external requests are accumulated in Pb , to be served by the transition $T_{Serv.a}$. The arrival of an internal request (firing of TG) places a mark in Pc . If there are other requests already queued (in Pb) tb is blocked, thus preventing the mark from passing to Pd . A mark in Pc disables the transition ta , so all the external requests following the arrival of the internal request will remain in Pa . With this technique it is therefore possible to discriminate between external requests which have arrived before an internal one and those which arrived afterwards. FIFO management of the queue is thus guaranteed, in accordance with the protocol used. Once all the previous external requests have been served, the internal request can be served as well (enabling of tb). We point out that there are three exit routes for the marks representing internal requests: the transitions tc, T_{over} and $T_{Serv.b}$. Transition tc is enabled when there are two marks in $Pout$. This situation indicates that the minimum number of responses on the part of the disks has been reached. If the third subrequest, the delayed one, is still in Pc , i.e. still queueing (as there are other requests ahead of it), tc eliminates it,

thus without it representing a workload for the disk, by using a reward function which returns 1 on the occurrence of the following condition:

$$(\#Pout + \#Pc1 + \#Pc2 + \#Pc3) = 3$$

It can easily be seen that when this condition occurs the system is doing useless work.

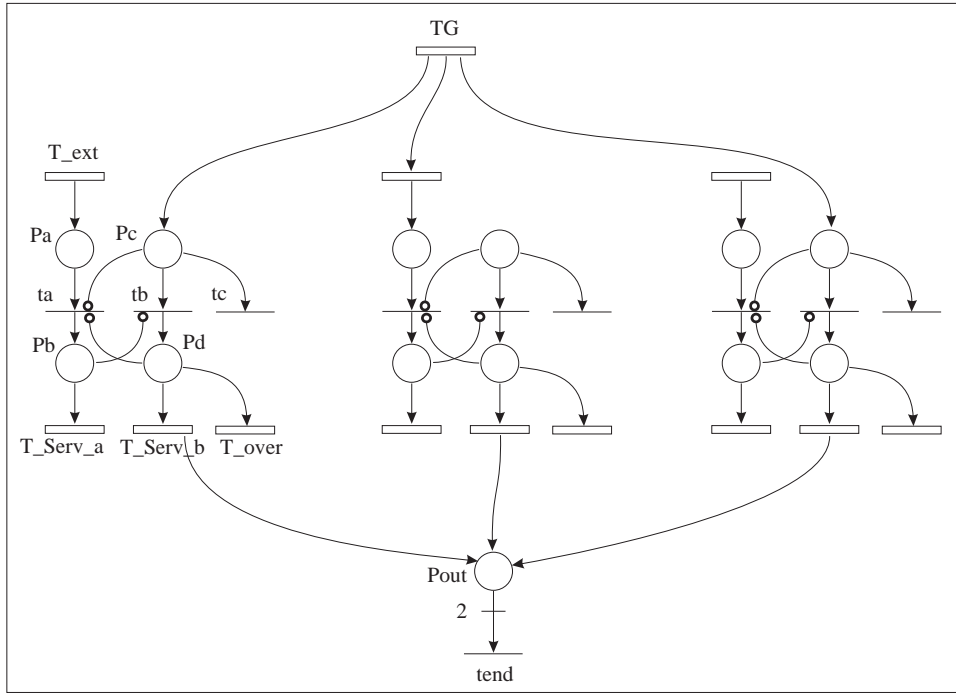


Figure 9.1: Petri Net model.

As regards the characteristics of the disks, the following values are assumed (for each disk):

Substained Transfer rate	5 MBytes/s;
Mean Seek Time	9 ms;
Mean Latency Rotational Time	5 ms.

Table 9.1.: Disk parameters

The workload applied to the system was hypothesized as being made up of multimedia transactions for VOD services, with films coded using the MPEG1 standard. The average throughput per connection is therefore 1.5 Mb/s. Assuming that each request addressed to the system refers to 25KB of data, each connection will involve a workload of 7.5 req/s. This value determines the firing rate of the transition *TG*.

The rate of T_{ext} was made to vary from a minimum of 7.5 to a maximum of 37.5, which correspond respectively to a total system workload of 180-540 req/s and therefore 4.5-13.5 MB/s (see table 9.2).

Fig. 9.2 shows the results obtained from an analytical solution of the model. As can be seen, the overhead introduced tends to decrease as the workload increases. This can be accounted for by the fact that, due to the dynamic management of the groups, when the traffic increases the time the corresponding subrequests spend queueing becomes mutually uncorrelated, and so the probability that, once the minimum number of requests needed to reconstruct data has been reached, the other subrequests (only one in our model) will still be queueing and not in service phase. It can also be seen from the fig. 9.2 that, although the overhead is quite high when workload values are low, it is not much higher than 5% for high values. If low values of workload are assumed the system can support the overhead introduced by this management policy quite well.

On the other hand, the advantages obtained in terms of service times are considerable, as will be pointed out in the following section.

	rate TG (req/s)	rate T _{ext} (req/s)	Total rate to the system
1	7.5	7.5	180
2	7.5	15	270
3	7.5	22.5	360
4	7.5	30	450
5	7.5	37.5	540

Table 9.2.: Analytical evaluation parameters

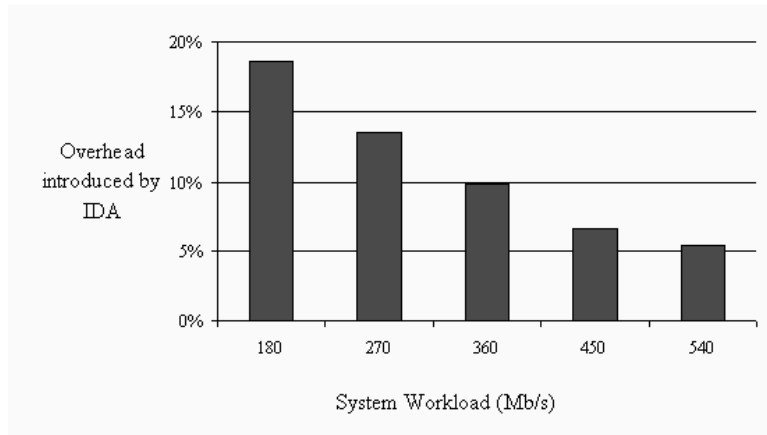


Figure 9.2: Overhead introduced by IDA.

9.4. EVALUATION OF SYSTEM RESPONSE TIME

The aim of this section is to evaluate the architecture described above in strictly performance terms. The parameter we used to do so was the system response time to a generic I/O request.

Response time is identified in literature as the time between generation of a request and completion of the operation requested. In the specific case of mass memory subsystems, it can be divided into six parts:

- Tos = time taken by the O.S. to process the request and translate it into low-level orders;
- Tw = time spent in queueing (in the controller or in the O.S. memory);
- RPS = rotational positioning sensing delay: time to make a full revolution when, in the absence of disk cache, the channel is not free when the disk is ready;
- Ts = seek time: time required for the heads to find the right cylinder;
- Trl = rotational latency time: time necessary to find the required sector;
- Trw = reading/writing time: time required to transfer data to or from the main memory.

In our assessment we will not take the first term into account, as it depends almost exclusively on the implementation of the specific operating system. We will not take the RPS into account either, as it is possible to use techniques to reduce this delay to zero.

The last three terms are generally indicated as the service time, as they mainly depend on the characteristics of the disk. The probabilistic distribution of these times was studied in [7].

According to [7] we model the disk access delay, the sum of the seek and latency times, with a normal distribution and the transfer time with a deterministic time depending on the requests. To deal with non

exponentially distributed events, we use phase type approximation⁸. The generic transition T_Serv_x , which models the service time in Fig. 9.1 is thus replaced by two cascades of Erlang distributions, the characteristics of which are given in Table 9.3.

Distribution	Order	Mixing Probability	Mean [ms]	Single Stage Rate [ms ⁻¹]
1	10	0.2926	1.455	6.869
2	10	0.7073	27.106	0.369

Table 9.3.: Parameters for the two mixed Erlang distributions

For a more detailed description of the procedure see [9].

To calculate the response time *pdf*, we used the tagged customer method. We calculated the steady-state probability of the states of the equivalent Markov chain as seen by a generic customer. Then the *MTTA* was calculated for each state, considering the absorbing state to be the one in which the tagged customer has been served. This method is described in detail in [5].

Obviously in our specific case the tagged customer coincides with the arrival of an internal request. The network in Fig. 9.1 was modified as shown in Fig. 9.3. To calculate the steady-state probabilities, the part relating to each disk has been simplified by combining places Pa and Pb in a single place $Pext$. The network in Fig. 9.3b, on the other hand, represents the model to be solved to obtain the *MTTA* values. In this case the place $Pout$ becomes the absorbing one.

The results obtained using this method are given in Fig. 9.4, which shows the probability distribution functions for applied load values of 9, 11.25 and 13.5 MB/s.

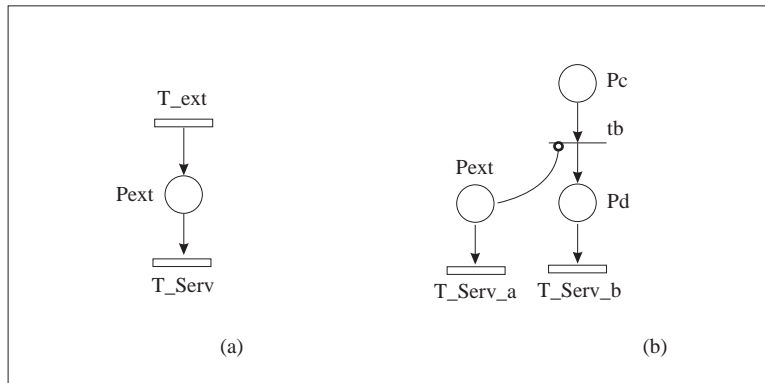


Figure 9.3: Petri Nets for the Tagged Customer Method.

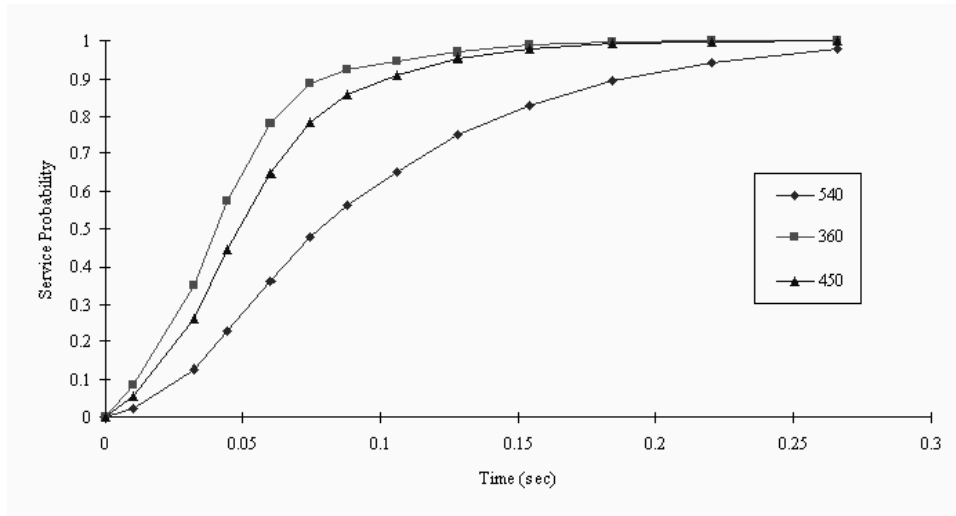


Figure 9.4: Response time distributions.

As can be seen, even with very low deadlines (100, 150 ms) the system has a very high probability of serving requests. This means that in a VOD service the probability of loss of videoframes is very low. Even when the workload is high, the system manages to serve the requests in a short time (200 ms). Knowledge of these response time probability distribution curves is indispensable for the design of a VOD system as it allows an appropriate size to be chosen for the Disk Array and a maximum threshold to be determined for the acceptable number of connections.

9.5. CONCLUSIONS

With the organization presented in the chapter it is possible to implement an efficient server for VOD services. The Petri net model described allows the performance obtainable from the system to be evaluated. The work also shows the great advantages that can be had, in terms of performance, from combined use of Partial Dynamic Declustering and IDA coding in the presence of typical CM workloads. Further improvements could be obtained by managing the service queues with prefetching algorithms, as it is possible to predict future requests on the basis of those currently being served. This allows high-quality

multimedia services to be provided.

9.A. APPENDIX - THE IDA TECHNIQUE

The *IDA* technique functions as follows. In an initial phase the information is subdivided in such a way as to form fixed-size macroblocks which contain the minimum unit processed by *IDA* in each operation. Appropriate choice of the size of these macroblocks therefore has to be made according to the features of the system and the data to be processed.

Each macroblock is then divided further into M blocks of a constant size, D (see Fig. 9.5).

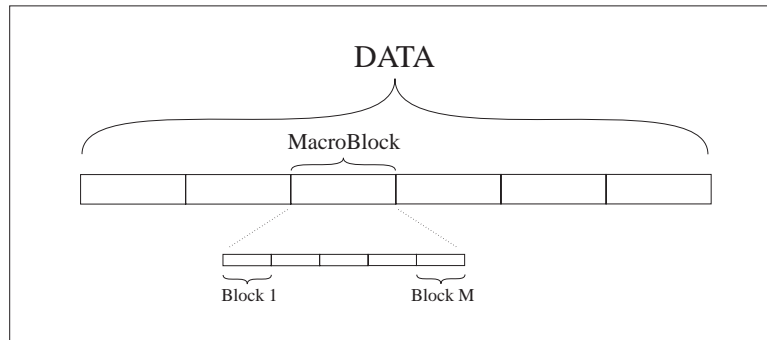


Figure 9.5: Ida coding.

If R is the desired redundancy value, i.e. R represents the number of units to be added, then the total number of units needed for storage will be $N = M + R$. Starting from the M initial blocks, the algorithm will build N new blocks, also of size D (see Fig. 9.6). Each of the N output blocks will be obtained by a linear combination of the M input blocks. To perform this operation it is necessary to use an $M \times N$ matrix. Choice of the matrix (i.e. the coding) is such that any minimum subset of M elements chosen from the N output blocks will be made up of linearly independent elements. In other terms, starting from any subset of M elements from the N outputs, it is possible to reconstruct the M input blocks. The only condition required for decoding is identification of the blocks from which to decode the original information, i.e. the index (from 1 to N) associated to each element.

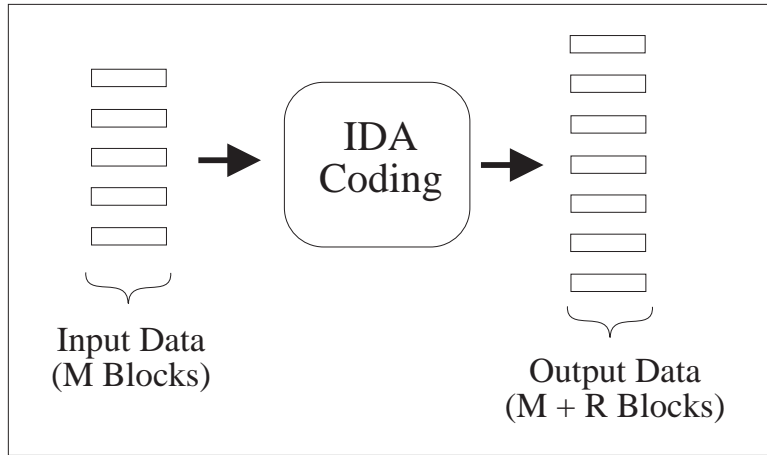


Figure 9.6: *IDA* coding phase.

Parity encoding can be seen as a particular case of *IDA* in which the first M blocks of the N to be recorded are exactly the same as the initial data and the $M + 1$ -th block is calculated by an *XOR* operation on the input data. With parity encoding we always have $R = 1$ and $N = M + 1$. Normally *IDA* is used to obtain an increase in the *MTBF* of a system. According to the system's reliability features, it is possible to choose the value, R , of redundancy in such a way as to obtain the total *MTBF* value desired. Even if the system has a very low *MTBF* value, suitable choice of the value of R will bring it back to satisfactory values. Table 9.4 gives the *MTTF* values for a disk array system with a space of 10 disks. It compares solutions based on *RAID1*, *RAID5* and *IDA*. the disks are all the same and have an *MTBF* of 1000h. The basic system (without redundancy) as a whole will have an *MTBF* of 1000h (0.11 years).

Scheme	Disks no.	MTBF
None	11	0.11
RAID 5	11	23
IDA	11	23
IDA	12	5284
IDA	13	1393837
IDA	20	3.7×10^{15}
RAID 1	20	114

(values expressed in years)

Table 9.4: *MTTF* values for a disk Array with B=10 disk units

These values were calculated according to [3] and do not take into account situations of correlated disk failures, which would considerably reduce the values given above. As regards the MTBF for RAID1, only the case corresponding to 20 disks is reported, because in this case total duplication of the disks is required (every disk has a mirror disk). Similarly, for the RAID5 case only the value corresponding to 11 disks is given, because only one disk is required to store parity information. The table also highlights the great flexibility of the IDA solution for redundancy purposes.

REFERENCES

1. D.J. Gemmel, H.M. Vin, D.D. Kandlur, PV. Rangan, L.A. Rowe, 'Multimedia storage servers: a tutorial', Computer, Vol. 28, No 5, May 1995.
2. T.D.C. Little, D. Venkatesh, "Prospects for interactive Video-on-Demand", IEEE Multimedia, Vol. 1, No 3, Fall 1994.
This paper is a survey on the technological considerations for designing a large-scale, distributed, interactive multimedia system. It examines the basic problems and proposes some implementation solutions.
3. P.M. Chen, E.K. Lee, G.A.Gibson ,R.H. Katz, D.A. Patterson , "RAID: High-Performance, Reliable Secondary Storage", ACM Computing Surveys, Vol. 26, No 2, June 1994.
This work is a detailed survey dealing with the problematics of Disk Arrays and standardized architectures. In particular, a detailed description of RAID systems is presented.
4. M.O. Rabin, "Efficient Dispersal of Information for Security Load Balancing and Fault Tolerance", ACM Journal of the Association for Computing Machinery, Vol.36 No 2, April 1989, pp 335-348.
This paper describes a new information coding system which can be used for storing and for transmitting data on a network not totally reliable.
5. J.K. Muppala. K.S. Trivedi, V. Mainkar, V.G. Kulkarni, "Numerical computation of response time distributions using stochastic reward

nets”, Annals of Op. Research, 48 (1994), pp.155-184.

6. V. Catania, A. Puliafito, S. Riccobene, L. Vita, “Design and Performance Analysis of a Disk Array System”, IEEE Transaction on Computer, Vol. 44, No 10, October 1995, pp. 1236-1247.

This work presents a new technique, named Partial Dynamic Declustering, for dynamic management of disk groups in a Disk Array. Comparisons are made with other existing architectures.

7. M.Y. Kim, A.N. Tantawi, “Asynchronized disk interleaving: Approximating access delays”, IEEE Transaction on Computer, Vol.40, No. 7, July 1991, pp. 801-810.

8. M.A. Johnson, “Selecting Parameters of phase distributions: combining nonlinear programming, heuristics and Erlang distributions”, ORSA J. on Computing, Vol.5, No. 1, 1993, pp. 69-83.

9. M. Malhotra, A.L. Reibman, “Selecting and implementing phase approximations for semi-Markov models”, Stochastic Models, 9 (4), pp. 473-506, 1993.