Storage Replication and Layout in Video-on-Demand Servers

Scott D. Stoller[1]* and John D. DeTreville[2]

[1] Dept. of Computer Science, Cornell University, Ithaca, NY 14853-7501, USA.
stoller@cs.cornell.edu
[2] Digital Equipment Corporation, Systems Research Center
130 Lytton Avenue, Palo Alto CA 94301-1044, USA.
jdd@pa.dec.com

Abstract. We propose and analyze an architecture for storage servers in large Video on Demand (VoD) systems. We describe a method for distributing the collection of titles among the levels of the storage hierarchy, based on estimates of the mean demand for each title. The resulting distribution minimizes cost for a given level of performance. Since high availability is desirable in VoD systems, we consider the use of mirroring or parity-based redundancy (*à la* RAID) and estimate the effect on the system's cost and availability. In the very-large-scale storage systems needed for VoD, the placement of disk arrays on the pool of computers must be chosen carefully to provide high availability for the least cost. We propose a strategy for arranging disk arrays on a pool of PCs; our strategy is inspired by Holland and Gibson's work on parity declustering for RAID.

1 Introduction

A Video on Demand (VoD) system provides access to a library of video (*e.g.*, digitized movies) by multiple independent subscribers in homes or offices. Large VoD systems would have been impossible to deploy cost-effectively with the technology of two decades ago; two decades from now they should seem easy. VoD systems should be a common application on the coming "information superhighway."

To better anticipate future design issues, we create analytical models of VoD systems and their cost and performance. We concern ourselves with large systems and with very high performance, the better to illustrate the challenges in engineering VoD systems.

* This work was performed in part at the Digital Equipment Corporation Systems Research Center. This material is based on work supported in part by the Office of Naval Research under contract N00014-91-J-1219, the National Science Foundation under Grant No. CCR-9003440, DARPA/NSF Grant No. CCR-9014363, NASA/DARPA grant NAG-2-893, and AFOSR grant F49620-94-1-0198. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not reflect the views of these agencies.

A VoD system in a large metropolitan area might serve 10^4 to 10^5 simultaneous viewers (N_V) at its busiest. The library of movies, *etc.*, might easily contain 10^3 to 10^4 titles (N_T). Although home viewers might tolerate significant delays in starting or restarting a movie, we require a latency of at most a few seconds for such operations; this allows stored video also to be used in interactive applications. We assume that the popularities of the titles follow a Zipf-like distribution, *i.e.*, that the popularity of the i^{th} title is proportional to $1/i$.

1.1 General Architectural Assumptions

Each title is striped across an array of G storage devices (*i.e.*, G is the group size). During playback, the data stream must enter the distribution network at the same rate r it is to be viewed: a few Mb/s. As this is an order of magnitude slower than disks can read, each disk array can serve roughly $10G$ streams. We choose to concentrate equipment at a central site, where it can be shared, such that each viewer needs only a network connection and a simple set-top unit (STU) to decode the stream and provide a user interface.

We assume a number of server computers at the central site, each with its own disks, interconnected via a high-speed network, and attached to the distribution network. Since the limiting factor at the servers is the bandwidth of the memory-I/O bus, we choose the PC architecture for the servers, as commodity hardware minimizes cost per bandwidth.

In a large VoD system, component failures will be common. We say that the servers can *tolerate* a failure if, despite that failure, the servers can continue to provide nearly uninterrupted service to all N_V viewers. (We allow a small hiccup in scheduling that could be masked by a buffer in the STU.)

1.2 Focus: High-level Server Design

We restrict our attention to the physical and logical architecture of the servers. We seek the optimal distribution of titles between disk and RAM, and the optimal layouts of titles among disks and PCs, in order to minimize cost and maximize availability at a given level of performance.

In a VoD system, unlike a general-purpose file system, the data are large and static, real-time response is required, bandwidth can be pre-allocated, and the usage patterns of the viewers are partly known. We base our design on these features.

2 Placing Titles in the Storage Hierarchy

The function of the servers is playback of stored video data. Various storage media are available, each with different cost and performance. The available storage media are called the *storage hierarchy*. Descending the hierarchy corresponds to *decreasing* cost per unit capacity and *increasing* cost per unit throughput.

Server design includes deciding where in the hierarchy to store each title. Multiple copies of a title may be needed to serve all of the requested streams for that title; this is *replication for throughput.* We arrange titles in the storage hierarchy with more popular titles in higher levels; storing a popular title at a lower level would require more copies of that title, counterbalancing the lower cost per unit capacity at the lower level. For concreteness, we assume that the storage hierarchy comprises RAM, (magnetic) disk, and (magnetic) tape, in that order. Generalizing our analysis to accommodate additional storage media is straightforward.

We quantify the above placement argument for the particular case of allocating titles between RAM and disk. The number of streams that can be served from each copy of a title depends on the layout of the data—in particular, it can be increased by striping a title across an array of storage devices. We assume a title in RAM is striped across G PCs; a title on disk is striped across G disks. (Relaxing the assumption that the same group size is used for both media does not affect our conclusions.) Let t_{D} and t_{PC} denote the mean throughput of a disk and (the I/O-bus of) a PC, respectively. Let p_{D} and p_{PC} denote the cost of a disk and PC, respectively. If s streams are needed, then the number of copies needed for bandwidth if the title is stored in RAM or on disk is approximately $i_{\mathrm{RAM}} = \frac{sr}{Gt_{\mathrm{PC}}}$ or $i_{\mathrm{D}} = \frac{sr}{Gt_{\mathrm{D}}}$, respectively.

Assume, for now, that the disks are *throughput-limited* (*i.e.*, the total required disk throughput equals the total available disk throughput). Then the total costs of serving this title from RAM or disk are approximately $P_{\mathrm{RAM}} = \lceil i_{\mathrm{RAM}} \rceil drp_{\mathrm{RAM}} + i_{\mathrm{RAM}} Gp_{\mathrm{PC}}$ or $P^t_{\mathrm{D}} = i_{\mathrm{D}} Gp_{\mathrm{D}} + \frac{t_{\mathrm{D}}}{t_{\mathrm{PC}}} i_{\mathrm{D}} 2Gp_{\mathrm{PC}}$, respectively, where d is the duration of the title.[3]The second summands are the cost of the needed PC throughput; the factor of 2 in P^t_{D} reflects data on disk crossing the PC's I/O-bus twice before transmission. Viewing these costs as functions of s, we see that they intersect at some value of s, which we denote s^t_{RAM}. The system cost is minimized by storing titles with $s \geq s^t_{\mathrm{RAM}}$ in RAM, and titles with $s < s^t_{\mathrm{RAM}}$ at a lower level.

The above calculation assumes the disk system is throughput-limited; note that no such assumption was made for RAM. If the disk system is sufficiently *capacity-limited* (*i.e.*, the total required disk storage space equals the total available disk storage capacity), then the cost of serving the streams for this title from disk is just the cost of the storage space; roughly, $P^c_{\mathrm{D}} = i_{\mathrm{D}} \frac{dr}{c_{\mathrm{D}}} p_{\mathrm{D}}$.

Of course, the intersection of P_{RAM} and P^c_{D} defines a different cut-off between RAM and disk. Since we don't know *a priori* which cut-off is correct, we compute the cut-off s_{RAM} as follows. We calculate an initial estimate for s_{RAM} by assuming the disk system is throughput-limited—this estimate is s^t_{RAM}. We then check whether the resulting disk system is actually throughout-limited. If so, we are done; if not, we increment s_{RAM} repeatedly until the disk system *is* throughput-limited. This method often leads to a final configuration where the disk system is at the balance-point of being capacity-limited *and* throughput-limited. The examples in Section 5 illustrate this.

[3] Slight modifications are required if redundancy is used. For example, if titles in RAM are mirrored, then the storage cost per title in RAM doubles.

Distributing Titles Between Disk and Tape. It is tempting to store the least popular titles only on tape when they are not being viewed and copy them to disk as needed. However, calculations similar to those above show that, except in VoD systems with low ratios of viewers to titles, the tape throughput needed for on-demand copying costs (almost) as much as the disk capacity that is saved. We conclude that using tape in this way is generally not worthwhile, especially if the additional complexity of supporting on-demand loading of titles from tape is taken into account.

3 Redundancy in RAM

Another major design decision for VoD servers is the form of redundancy to use. Different redundancy schemes may be more attractive at different levels of the storage hierarchy, so we discuss the choices separately for RAM and disk. In this section, we discuss briefly the resources needed for titles in RAM for no redundancy, parity, and mirroring. Availability is analyzed in Section 6.

The use of parity for fault-tolerance is well-known from RAID [1]. The resource requirements depend on where missing data is reconstructed. Reconstruction in the STU instead of the servers has some benefits (*e.g.*, reducing the net cost of the servers and STUs) and some drawbacks (*e.g.*, increasing the performance requirements for the STUs).

If reconstruction is done in the STU, then the STU must be fast enough to reconstruct the missing data in real-time. Since the STU is processing a single video stream, this requirement is modest.

If reconstruction is done in the server, extra PC throughput is required for the PCs to exchange data to reconstruct the missing data. The task of reconstruction can be distributed among many PCs (not just those in the affected array) to reduce the additional PC throughput needed.

The cost of mirroring is dominated by the cost of the extra RAM needed for storage.

4 Disk Organization and Redundancy

After deciding which titles to store on disk, we must still choose the layout of the titles on the disks and the layout of the disks on the PCs. For the former, we continue to assume that each copy of a title on disk is striped across exactly G disks. The arrangement of these disk arrays on the PCs affects the availability. The best choice depends on the form of redundancy.

In this section, we determine the resource requirements for titles on disk with each redundancy option; availability is analyzed in Section 6. The requirements depend on both the form of redundancy and the *tolerated failures*, *i.e.*, the types of failures the system tolerates. For example, using mirroring when the system must tolerate any single disk failure requires doubling the disk capacity.

Two principles help determine the best organization. For economy, the organization should minimize the resources needed to tolerate tolerated failures. For availability, it should minimize the number of viewers affected by non-tolerated failures.

4.1 Organizations with No Redundancy

A system with no redundancy tolerates no failures. Availability is maximized by distributing each disk array onto as few PCs as possible, since this minimizes the number of viewers affected by each PC failure.

4.2 Organizations with Parity: Declustered Disk Arrays

We use parity to tolerate failure of a single disk or PC.[4] One simple organization is to arrange the PCs into arrays of size G. Each array of disks is spread across an array of PCs; each PC has at most one disk from each disk array. This makes it possible for the system to tolerate a PC failure.

The resource requirements for the servers depend on where reconstruction is done. Reconstruction in the STU requires few additional resources. The resources needed for reconstruction in the servers depend on the transmission schedule in the event of failure. One simple schedule has each PC in the affected array synchronously read data from disk, exchange the data with the other PCs, reconstructs its part of the missing data, then transmits its data to the STUs according to the normal transmission schedule. In the event of a failure, PCs read data earlier than normal but transmit it at the usual time, so additional buffer RAM is needed.[5] The main costs of parity-based redundancy are this buffer RAM and the extra PC throughput needed for reconstruction.

With the organization sketched above, in the event of the failure of a PC with $n^{\mathrm{D}}_{\mathrm{PC}}$ disks, the extra load on the surviving PCs in the affected array of PCs is $n^{\mathrm{D}}_{\mathrm{PC}}$ times the extra load resulting from a single disk failure, as is the extra RAM and throughput needed. The resource requirements can be significantly reduced by better distributing the extra load caused by a PC failure. Intuitively, we think of the PCs as a large homogeneous pool and arrange the disk arrays on the PCs to minimize the average number of disk arrays "shared" by each pair of PCs.

More precisely, we say that two PCs *share* a disk array iff both of the PCs have a disk in that disk array. Let $\sigma_i = \max(\{\mathit{shared}(i,j) \mid j \neq i\})$, where i and j name PCs and $\mathit{shared}(i,j)$ is the number of disk arrays shared by i and j. A *declustered disk array* organization is one that (nearly) minimizes $\bar{\sigma} = N_{\mathrm{PC}}^{-1} \sum_i \sigma_i$, where N_{PC} is the number of PCs in the system. These organizations are closely analogous to those proposed by Holland and Gibson for arranging data stripes among a pool of PCs [5]. Using the formulas in their Section 4.2, we

[4] Our availability analysis takes into account that some double disk failures can also be tolerated.

[5] This buffering can be eliminated by reading the data from disk twice, but our calculations show that the extra disk throughput generally costs more than the buffer RAM.

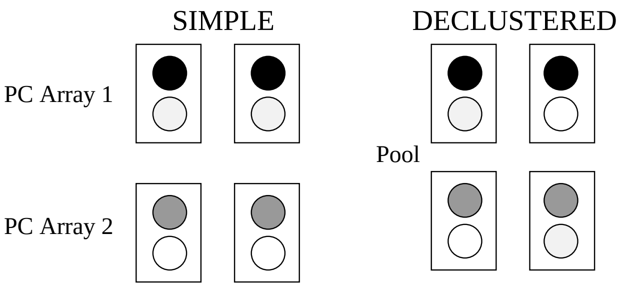

Fig. 1. Example of declustering. The rectangles are PCs; the circles, disks. Shading represents disk arrays. Note that $\bar{\sigma} = 2$ for the simple organization and $\bar{\sigma} = 1$ when declustered.

expect $\bar{\sigma} \approx (n^{\mathrm{D}}_{\mathrm{PC}})^2(G-1)/(N_{\mathrm{D}} - n^{\mathrm{D}}_{\mathrm{PC}})$, where N_{D} is the number of disks in the system. It suffices to use an organization with approximately this value of $\bar{\sigma}$.

Figure 1 contains a simple example of declustering. Section 5 illustrates the cost savings from declustering disk arrays. Declustering reduces availability, but only slightly.

4.3 Organizations with Mirroring

As with parity, the system tolerates the failure of a single disk or PC. As in the first organization with parity, the PCs are divided into arrays of size G, and disk arrays are striped across arrays of PCs. To minimize the PC throughput needed to handle a PC failure, we again use a form of declustering: the backups of different disk arrays on a PC array are placed on different PC arrays. The resource analysis for mirroring is similar to that for parity.

5 Hardware Cost

We estimate hardware costs for various server designs, based on the analysis sketched above. We adopt many simplifications of reality, so our figures are rough estimates of actual cost. For example, although the video streams may be variable-bit-rate, we use just the mean bit-rate in our calculations. We adopt only simplifications that are roughly "orthogonal" to the design issues under consideration, so our analysis should reflect the *relative* costs for different designs.

We assume that the network can multicast data; for example, an ATM network based on the AN2 switch [9] can multicast. This is useful for the most popular titles: multiple viewers can receive the same stream, so the servers need never to supply more than (say) one stream per second per title.

Our calculations are based on the expected hardware cost and performance figures for late 1995 given in Table 1. We take the mean bit-rate of a stream

Table 1. Expected hardware cost and performance figures for late 1995.

RAM Cost: $p_{\mathrm{RAM}} = \$14/\mathrm{MB}$	Disk Controller Throughput: $t_{\mathrm{ctrlr}} = 20$ MB/s Cost: $p_{\mathrm{ctrlr}} = \$250$
Disk Striping unit: $B = 200$ KB Capacity: $c_{\mathrm{D}} = 4.3$ GB Mean Throughput: $t_{\mathrm{D}} = 4$ MB/s Cost: $p_{\mathrm{D}} = \$1600$	PC I/O Throughput (PCI bus): $t_{\mathrm{PC}} = 90$ MB/s Cost: $p_{\mathrm{PC}} = \$4000$

Table 2. Cost and availability of a medium VoD system ($N_T = 10^4, N_V = 10^5$)

Redundancy Organization	None	Parity Simple	Parity Declus.	Mirror	Mir/Par Declus.	ParSTU Simple	ParSTU Declus.
N_T^{RAM}	18	18	18	0	18	18	18
RAM (GB)	50	390	73	5	119	395	73
Disk-cap. (10^3)	6.2	6.3	6.3	12.4	6.3	6.3	6.3
Disk-tput.(10^3)	6.2	6.3	6.3	9.5	6.3	6.3	6.3
Disk Ctrlr(10^3)	1.6	1.6	1.6	3.1	1.6	1.6	1.6
PC	700	1800	1000	860	810	720	710
Network (10^3\$)	0	110	110	0	70	0	0
Cost (10^6\$)	13.8	23.3	15.7	24.1	15.5	18.9	14.4
ir (year^{-1})	.14	.027	.021	.019	.015	.015	.015

to be $r = 3$ Mb/s; this corresponds to VHS-quality video encoded with MPEG-2. Increasing the group size G improves load-balancing and reduces replication of titles but reduces availability. We take $G = 50$; this is just large enough to require negligible replication of titles for throughput. Thus, by storing titles in RAM, striping widely, and taking advantage of network multicast, we nearly eliminate replication for throughput; we use redundancy primarily for enhancing availability.

Tables 2 and 3 describe a "medium" VoD system ($N_T = 10^4, N_V = 10^5$) and a "large" VoD system ($N_T = 10^4, N_V = 4 \times 10^5$), respectively. The three sections of each table give the design, its hardware requirements, and the resulting availability. The "Redundancy" row gives the forms of redundancy used for RAM and for disk (in that order, if two forms are given), or for both (if only one form is given). "Par(ity)" denotes parity with reconstruction in the servers; "ParSTU" denotes reconstruction in the STU. N_T^{RAM} is the number of titles stored in RAM.

The "Disk-cap." and "Disk-tput." rows count the disks required for capacity and for throughput; the number of disks needed is the maximum of these. The "PC" row counts the PCs needed. The "Network" row estimates the cost of the internal network used by the servers to interchange reconstruction data. The cost of the external network is not included, since it is constant among these

Table 3. Cost and availability of a large VoD system ($N_T = 10^4, N_V = 4 \times 10^5$)

Redundancy Organization	None	Parity Simple	Parity Declus.	Mirror	Par/Mir	ParSTU Simple	ParSTU Declus.	ParSTU/Mir
N_T^{RAM}	193	193	193	96	193	193	193	193
RAM (GB)	517	1388	546	516	528	1388	546	528
Disk-cap. (10^3)	6.2	6.3	6.3	12.3	12.3	6.3	6.3	12.3
Disk-tput.(10^3)	15.6	15.9	15.9	18.7	15.9	15.9	15.9	15.9
Disk Ctrlr(10^3)	3.9	4.0	4.0	4.7	4.0	4.0	4.0	4.0
PC	2190	5550	3630	2360	3390	2220	2190	2220
Network (10^3\$)	0	442	442	0	0	0	0	0
Cost (10^6\$)	41.9	68.5	49.0	47.7	47.6	54.7	42.8	42.7
ir (year^{-1})	.15	.062	.057	.037	.052	.029	.033	.035

designs. The row labeled "ir" is discussed in Section 6.

In the medium system, the optimum value of N_T^{RAM} results in the disk system being both throughput-limited and capacity-limited. In the large system, the disk system is throughput-limited but not capacity-limited.

Comparing "Parity" and "Mir/Par" for the medium system shows that the reduced storage cost from parity in RAM is offset by the cost of the PC throughput needed for reconstruction. The same effect can be seen in the large system by comparing the columns "Mirror" and "Par/Mir."

For titles on disk, we see that parity is more attractive in the medium system, while mirroring is more attractive in the large system. For titles in RAM, parity is more attractive than mirroring if we reconstruct in the STUs; otherwise, parity and mirroring are about equally expensive and mirroring has slightly better availability. For both systems, the cheapest redundancy option increases the system cost by about 3% for reconstruction in the STUs, and by about 13% otherwise.

6 Availability

Redundancy can increase availability. We quantify availability as ir, the "mean rate of observed interruptions due to server failures, per STU." "Observed interruptions" do not include failures when the STU is not in use. To quantify this, we introduce two parameters: the mean fractional load on the system, denoted $meanFL$; and the maximum fraction of STUs active at once, denoted $maxFA$. Note that the total number of STUs is $N_{\mathrm{V}}/maxFA$, and that the mean number of active STUs is $N_{\mathrm{V}} meanFL$. We take $meanFL = 1/4$ and $maxFA = 1/3$. As a sanity check on these values, note that each STU is in use $(meanFL)(maxFA)$ of the time, *i.e.*, 14 hours/week. These values represent a future time when VoD has largely supplanted broadcast TV and videotape rental in the U.S. It is easy to see the effect of choosing other values for these parameters, since ir is directly proportional to each.

Our availability calculations are based on a classification of failures by cause. Following Gray and Reuter [4], we take the set *Causes* of possible causes of failures to be: hardware (subdivided into RAM, disk, and PC), operations, maintenance, environment, and software. In equations, we abbreviate the last four causes by their first letter. Let $r(c)$ denote the rate of failures with cause c.

To estimate the effect of redundancy on availability, one must consider the tolerance to failures of each cause. We propose the following model. Each failure is assumed to have the effect of rendering inoperative one physical or *logical* component of the system. In this model, the set *CType* of types of components that a failure may affect are: RAM (one word of RAM), D (one disk), DA (one disk array), PC (one PC), PCA (one PC array), or SYS (the entire system). For a given system design, for each component type t, let $\tau(t)$ denote the number of viewers whose streams are interrupted by a given failure of type t.

Our model postulates that, of all the failures with a given cause c, some fraction λ_t^c render inoperative a component of type t. For example, a software error in the OS will probably crash a single PC, while a software error in a scheduling module is likely to affect an entire array of PCs, so λ_{PC}^S and λ_{PCA}^S are both non-zero. The mean number of viewers interrupted by a failure with cause c is the weighted sum

$$\kappa(c) = \sum_{t \in CType} \lambda_t^c \tau(t) \ . \tag{1}$$

Multiple Failures. To model multiple failures, we generalize the above definitions slightly, by taking the domains of κ and r to be non-empty subsets of *Causes*. For example, $r(\{\mathrm{D}, \mathrm{D}\})$ is the rate of double disk failures, and $\kappa(\{\mathrm{D}, \mathrm{D}\})$ is the mean number of viewers interrupted by a double disk failure. Each of these rates $r(C)$ is a parameter of the model; for example, we are free to choose $r(\{\mathrm{D}, \mathrm{D}\})$ to reflect correlations between disk failures. The earlier discussion is still valid, when occurrences of $r(c)$ and $\kappa(c)$ are interpreted as abbreviations for $r(\{c\})$ and $\kappa(\{c\})$, respectively. Thus, we continue to use equation (1) to define κ on singleton sets. In contrast, we do not assume any particular form for the equations defining κ on sets of cardinality greater than one; these equations are derived using probabilistic arguments and contain only the parameters introduced above. Finally, the mean rate of observed interruptions due to server failures, per STU, is

$$ir = \frac{(meanFL)(maxFA)}{N_{\mathrm{V}}} \sum_{\emptyset \subset C \subseteq Causes} r(C)\kappa(C) \ . \tag{2}$$

6.1 Availability Calculations

The availabilities in Tables 2 and 3 were obtained using the model sketched above. Table 4 contains the failure rates used in our calculations. They are based primarily on data from surveys of Tandem customers [3, 4] and on product information (*e.g.*, [7]). The failure rates reported in the Tandem surveys are *per*

Table 4. Failure rates by cause.

$r(\{\mathrm{RAM}\})$	$= (3 \times 10^7 \text{ hour})^{-1}$	per 2MB bank (selected)
	$(3 \times 10^8 \text{ hour})^{-1}$	per 2MB bank (unselected)
$r(\{\mathrm{D}\})$	$= (3 \times 10^5 \text{ hour})^{-1}$	per disk
$r(\{\mathrm{PC}\})$	$= (50 \text{ year})^{-1}$	per PC
$r(\{O\})$	$= (75 \text{ year})^{-1}$	per 100 PCs
$r(\{M\})$	$= (420 \text{ year})^{-1}$	per 100 PCs
$r(\{E\})$	$= (170 \text{ year})^{-1}$	per system
$r(\{S\})$	$= (30 \text{ year})^{-1}$	per 3 PCs
$r(\{\mathrm{D}, \mathrm{D}\})$	$= 10(N_{\mathrm{D}} r(\{\mathrm{D}\}))^2 MTTR_{\mathrm{D}}$	per system
$r(\{\mathrm{D}, \mathrm{PC}\})$	$= (N_{\mathrm{D}} r(\{\mathrm{D}\}))(N_{\mathrm{PC}} r(\{PC\}))$ $(MTTR_{\mathrm{D}} + MTTR_{\mathrm{PC}})$	per system
$r(\{\mathrm{PC}, \mathrm{PC}\})$	$= 10(N_{\mathrm{PC}} r(\{PC\}))^2 MTTR_{\mathrm{PC}}$	per system

system, for systems with an average of three processors. Therefore, we scale their software failure rate by $N_{\mathrm{PC}}/3$ and (somewhat arbitrarily) their operations and maintenance failure rates by $N_{\mathrm{PC}}/100$. Following Gray's comments [3], we compensate for underreporting of failures by increasing the reported rates for operations, environment, and software failures by 100%, 100%, and 5%, respectively. The factors of 10 in $r(\{\mathrm{D}, \mathrm{D}\})$ and $r(\{\mathrm{PC}, \mathrm{PC}\})$ account for correlated failures. Assuming a pool of spare disks is available, $MTTR_{\mathrm{D}}$ is dominated by the time needed to fill one of the spare disks with the appropriate data. With mirroring, this is just the time needed to fill the spare disk with data, so $MTTR_{\mathrm{D}} \approx c_{\mathrm{D}}/t_{\mathrm{D}}$. Reconstructing data from parity requires more work, so with parity, we take $MTTR_{\mathrm{D}} \approx 1$ hour. Replacing a PC may require human intervention, so we take $MTTR_{\mathrm{PC}} = 1$ day.

Unfortunately, we are not aware of any empirical studies on which to base the values of the parameters λ_t^c, so for our present calculations, we merely chose values that seem plausible, namely

	RAM	D	DA	PC	PCA	SYS
λ^{O}	0	0	0.3	0.6	0.09	0.01
λ^{M}	0	0	0.05	0.89	0.05	0.01
λ^{E}	0	0	0.05	0.05	0.05	0.85
λ^{S}	0	0	0.3	0.6	0.09	0.01

.

The values of λ^t for hardware failures (*i.e.*, $t \in \{\mathrm{RAM}, \mathrm{D}, \mathrm{PC}\}$) are obvious; for example, $\lambda_{\mathrm{D}}^{\mathrm{D}} = 1$ and $\lambda_{\mathrm{PC}}^{\mathrm{D}} = 0$.

The formulas for $\tau(t)$ and $\kappa(C)$ depend on the form of redundancy and the disk organization. For illustration, we give the formulas for parity with the simple (not declustered) organization. Let N_V^{D} denote the number of viewers served from disk. With parity, the system can tolerate the failure of a word of RAM, a disk, or a PC, but not the failure of an entire disk array or PC array, so

$$\begin{array}{llll} \tau(\mathrm{RAM}) &= 0 & \tau(\mathrm{DA}) &= N_{\mathrm{V}}^{\mathrm{D}}/N_{\mathrm{DA}} \\ \tau(\mathrm{D}) &= 0 & \tau(\mathrm{PCA}) &= N_{\mathrm{V}}/N_{\mathrm{PCA}} \\ \tau(\mathrm{PC}) &= 0 & \tau(\mathrm{SYS}) &= N_{\mathrm{V}} \ , \end{array} \tag{3}$$

where $N_{\mathrm{DA}} = N_{\mathrm{D}}/G$ and $N_{\mathrm{PCA}} = N_{\mathrm{PC}}/G$ are the numbers of disk arrays and PC arrays, respectively. These formulas, together with equation (1), determine the contributions of single failures to ir.

Multiple failures involving more than two causes are so infrequent that their contribution to ir is negligible. Compared to single failures, double failures make a modest but non-negligible contribution to ir. Since the total contribution of double failures to ir is modest, we keep only the largest double-failure terms. The largest rates of single failures are for software, disk, and PC, so we consider only the six combinations of these.

Since the redundancy schemes discussed here do not provide (complete) tolerance to software failures, the contribution to ir from single software failures dominates contributions from double-failure terms involving software failures.[6]Thus, it suffices to include contributions from the three double-failure terms involving combinations of disk and PC. Simple probabilistic arguments yield the following estimates:

$$\kappa(\{\mathrm{D},\mathrm{D}\}) = ((G-1)/(N_{\mathrm{D}}-1))\tau(\mathrm{DA}) \tag{4}$$

$$\kappa(\{\mathrm{D},\mathrm{PC}\}) = (n_{\mathrm{PC}}^{\mathrm{D}} G/N_{\mathrm{D}})\tau(\mathrm{DA}) \tag{5}$$

$$\kappa(\{\mathrm{PC},\mathrm{PC}\}) = ((G-1)/(N_{\mathrm{PC}}-1))\tau(\mathrm{PCA}) \ . \tag{6}$$

Similar reasoning is used to obtain formulas for other organizations and other forms of redundancy. These calculations yield the figures for ir in Tables 2 and 3. We conclude that parity or mirroring provide comparable increases in availability: adding either form of redundancy reduces the interruption rate by 75–90%.

7 Conclusions

We have examined high-level design of VoD servers. Our designs and analysis contain novel features. We have described a method for distributing the collection of titles among the levels of the storage hierarchy. Our method is specific to VoD only in that it requires estimates of the mean demand for each file. This problem has also been studied by Tetzlaff *et al.* [8] and by Doğanata and Tantawi [2].

In a very-large-scale storage system, the placement of entire disk arrays is an important issue. If parity-based redundancy is used, we propose arranging disk arrays on a pool of PCs using techniques similar to those used to arrange data stripes on a pool of disks [5]. This idea is not specific to VoD. Choosing the placement of disk arrays is complementary to allocating titles to disk arrays [6].

[6] Since our redundancy schemes provide *partial* tolerance to software failures, this claim is not obvious, but it is easily checked by estimating a few of these terms.

Our availability analysis reflects the increased tolerance to failures *of all kinds* that redundancy provides. For example, mirroring provides tolerance to some software failures, as well as tolerance to certain hardware failures. We estimate tolerance to non-hardware failures by modeling the effects of non-hardware failures as failures of *logical components*.

Numerical studies of our model show that intelligent use of redundancy in a VoD system increases cost moderately (by about 13%) and improves availability significantly (reducing interruptions by 75–90%). In some systems, it is attractive to use different redundancy schemes for titles in different levels of the storage hierarchy.

More work is needed to study the interaction between the storage architectures proposed here and other crucial elements of VoD systems, such as admission control, scheduling, and full VCR-like functionality.

Acknowledgments

The authors thank Ed Lee for helpful discussions.

References

1. P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. Raid: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.
2. Y. N. Doğanata and A. N. Tantawi. Making a cost-effective video server. *IEEE Multimedia*, 1(4):22–30, Winter 1994.
3. Jim Gray. Why do computers stop and what can be done about it? In *Proc. Fifth Symposium on Reliability in Distributed Software and Database Systems*, pages 3–12, 1986.
4. J. Gray and A. Reuter. *Transaction processing: concepts and techniques.* Morgan Kaufmann, 1993.
5. M. Holland and G. Gibson. Parity declustering for continuous operation in redundant disk arrays. In *Proc. Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-V)*, pp. 23–35. ACM Press, 1992.
6. T. D. C. Little and D. Venkatesh. Probabilistic assignment of movies to storage devices in a video-on-demand system. In *Proc. Fourth International Conference on Network and Operating System Support for Digital Audio and video (NOSSDAV '93)*, pp. 213–224, 1993.
7. Micron Semiconductor, Inc. *DRAM Data Book*, 1993.
8. W. Tetzlaff, M. Kienzle, and D. Sitaram. A methodology for evaluating storage systems in distributed and hierarchical video servers. In *COMPCON 94*, pp. 430-439. IEEE Computer Society Press, Spring 1994.
9. C. P. Thacker. *AN2 Switch Overview.* Digital Equipment Corporation, Systems Research Center, August 1994. In preparation.

This article was processed using the LaTeX macro package with LLNCS style