A Survey of Approaches to Fault Tolerant Design of VOD Servers: Techniques, Analysis, and Comparison*

Leana Golubchik John C. S. Lui Maria Papadopouli

Abstract

Recent technological advances in digital signal processing, data compression techniques, and high speed computer networking have made on-demand multimedia servers feasible. A challenging task in multimedia systems is to service multiple clients simultaneously, while satisfying the real-time requirement for continuous delivery of objects at specified rates. To accomplish these tasks and realize economies of scale associated with servicing a large user population, a multimedia server can require a large disk subsystem. Although a single disk is fairly reliable, a large disk farm can have an unacceptably high probability of disk failure. Further, due to the real-time constraints, the reliability and availability requirements of multimedia systems can be more stringent than those of traditional information systems. In this paper we focus on the main tradeoffs and issues associated with providing fault tolerance in multidisk VOD systems. We illustrate these tradeoffs through a discussion of several fault tolerance schemes and present a framework for a performability evaluation of possible design choices.

1 Introduction

Recent technological advances in digital signal processing, data compression techniques, and high speed computer networks have made Video-on-Demand (VOD) servers feasible. Challenging tasks in designing such large scale systems include not only satisfying the real-time constraints of continuous delivery of objects, which has been addressed in the following proposals [3, 22, 19, 5, 16, 20, 10] (to name a few) in the form of various data layout and scheduling algorithms, but also providing a high degree of reliability and availability. To exhibit reasonable economies of scale, the VOD server should contain a large number of disks; something on the order of 1000 drives would not be uncommon. For example, a storage subsystem of 1000 (1 gigabyte) disks would provide enough storage

*The research of Leana Golubchik and Maria Papadopouli was supported in part by the NSF CAREER grant CCR-96-25013. The research of John C. S. Lui was supported in part by the RGC and CUHK Direct Grant.

for approximately 220 (100 minute) MPEG-2 movies (at about 6 Mbits/sec) or approximately 880 MPEG-1 movies [9] (at about 1.5 Mbits/sec) or some combination of the two. Similarly, assuming a bandwidth of 4 Mbytes/sec, 1000 disk drives provide enough bandwidth to support approximately 5300 concurrent MPEG-2 users or 21,300 MPEG-1 users. Although a single disk can be fairly reliable, given such a large number of disks, the aggregate rate of disk failures can be high. For instance, if the mean time to failure (MTTF) of a single disk is on the order of 300,000 hours, then the MTTF of *some* disk in a 1000 disk system is on the order of 300 hours (or 12.5 days). Several works on fault tolerance schemes for VOD servers have appeared in the last few years; these include [2, 6, 11, 17, 22, 23]. The issues and tradeoffs involved in providing fault tolerance *in a cost-effective manner* is the focus of this paper.

We first give a general architectural overview of a VOD system. A large-scale VOD server consists of a tertiary storage library, a collection of disks, and a set of processors. The entire database permanently resides on tertiary storage, from which objects are retrieved and placed on disk drives for delivery on demand. The long latency times and high bandwidth cost of tertiary devices usually preclude most objects from being transmitted directly from the tertiary store. And the high cost of buffer space usually precludes most objects from being permanently stored in main memory. Given this architecture, a disk failure does not result in data loss, since a copy of each object is stored on tertiary storage. However, a disk failure can result in interruption of requests in progress. If some of the data for an object currently being displayed is on a disk that fails, then a discontinuity in delivery, termed a *hiccup*, occurs. Since portions of many objects typically reside on a disk, a single disk failure can cause hiccups in the display of multiple objects. These hiccups will occur each time an access to the failed disk is attempted and will continue until a reloaded operational disk replaces the failed disk. Rebuilding a failed disk from tertiary storage can be a slow process. Loading a standby disk with the missing data requires portions of many objects to be loaded from the tertiary storage and many tapes may need to be referenced. Therefore, without some form of fault tolerance, such a system is not likely to be acceptable. That is, providing fault tolerance schemes in VOD servers is essential, since, given the size of the disk subsystem, duration of videos, and the real-time constraints, it is not acceptable to be unable to recover from disk failure *in real time*.

We can divide the existing approaches to fault tolerant design of VOD servers into two categories: redundancy-based techniques (e.g., [2, 22]) and non-redundancy-based techniques (e.g., [23]). That is, redundancy-based techniques, such as parity-based or replication-based schemes, are those that store redundant information which, under failure, is used to *reconstruct* the missing data. Whereas

non-redundancy-based techniques exploit the redundancy that is "inherently" present in video data in order to *approximate* the missing information without the use of redundant information. In this paper, we focus on redundancy-based techniques, partly because there has been significantly more work done in this area and partly because the loss of image quality in non-redundancy-based schemes is somewhat difficult to evaluate. Note that, this loss of quality is a persistent problem, unlike momentary jitters experienced by some redundancy-based schemes.

Redundancy-based techniques can further be subdivided into two categories: (1) those that exploit sequentiality of video data delivery, e.g., [2] and (2) those that disregard sequentiality of video data delivery, e.g., [6]. That is, one can either exploit the fact that video display can be "pre-planned" and layout and retrieve data such that the information retrieved for reconstruction of missing data can be utilized for normal display as well. Or, one can disregard the fact that this information can be utilized for normal display (and not just reconstruction of missing data), and thus discard the data blocks retrieved for reconstruction, i.e., the reading of data blocks for normal display is scheduled independently from the reading of possibly the same data blocks for the purpose of reconstructing (and displaying) the data missing due to failure.

This means that a fundamental tradeoff in choosing one or the other type of an approach is between I/O bandwidth and buffer space. That is, by using a sequential-type scheme, we can utilize the I/O bandwidth better by retrieving each data block only once, and using it for both normal display and reconstruction purposes; however, this efficiency in I/O bandwidth utilization is achieved at the cost of additional buffer space, since the blocks retrieved for reconstruction of missing information must be buffered until they are needed for normal display. In contrast, when using a non-sequential-type scheme, we can discard data as soon as reconstruction of missing information is complete, and hence there is no need for additional buffer space; however, this efficiency in buffer space utilization is achieved at the cost of inefficient bandwidth usage, since, even under normal operation, we must reserve sufficient I/O bandwidth during scheduling of data retrieval so as to be able to access all the information needed for reconstruction of missing data, in real time, when a failure does occur — this includes both parity information *and* the surviving data blocks in the same parity group.

Since it is not immediately clear how to compare savings in I/O bandwidth with savings in buffer space, one approach is to assess this tradeoff through cost considerations. Thus, a meaningful performance measure is $/stream, i.e., one can compare alternative fault tolerance schemes by considering: (a) the overall cost of the system based on system requirements, e.g., in terms of the

cost of disks plus main memory, which are dictated by I/O bandwidth, buffer space, and storage for fault tolerance needs, and (b) the maximum number of streams that can be simultaneously supported by the corresponding VOD architecture. It is worth noting that different schemes have their optimum operating points at different architectural configurations; thus it is not the case that one fault tolerance scheme is absolutely better than another, but rather that one must understand the system requirements and constraints (e.g., minimum number of simultaneously supported streams) and then choose a fault tolerance scheme accordingly. Finally, it must also be noted that the reliability characteristics (such as MTTF) of a particular fault tolerance scheme should not be neglected, since that is the initial motivation for this work; however, it is reasonable to compare schemes which are above an "acceptable" reliability threshold based on the $/stream metric.

In the remainder of this paper, we describe and compare approaches for providing redundancy-based fault tolerance in VOD servers. We use the following metrics in our (performability) comparison of these schemes:

- available disk storage space (for storage of "real" data)
- buffer space requirements
- available disk bandwidth (for delivery of "real" data)
- reliability[1]
- maximum number of simultaneously supported streams
- cost per stream

The organization of the remainder of the paper is as follows. In Section 2 we briefly present background information on design of VOD servers as well as some simple schemes for providing fault tolerance in VOD servers, which we use as "baseline" cases in the remainder of the paper. In Sections 3 and 4 we present various sequential and non-sequential fault tolerance schemes, respectively, as well as discuss the associated tradeoffs. In Section 5 and 6 we present analysis of these schemes, and in Section 6 we discuss their respective merits. Section 7 gives our concluding remarks.

[1] Mean time to failure and mean time to degradation of service (these will be defined later in the paper).

2 Background

We begin this section with a description of a simple disk model and some basic concepts of scheduling of video streams — these will aid in further description of the various fault tolerance schemes as well as the associated performance tradeoffs. We will also give a brief description of a RAID-based technique for providing fault tolerance in traditional systems and then describe two simple schemes for providing fault tolerance in VOD systems — a RAID-based sequential-type scheme and a RAID-based non-sequential-type scheme. These two schemes will serve as "baseline" cases for the remainder of the paper. In the context of these baseline cases, we will illustrate the basic tradeoffs associated with these two types of approaches.

2.1 Video Scheduling

In this section we give a brief description of the notion of scheduling video requests in cycles or groups. We will use the term *stream* to refer to the delivery of a given object at a given time. To achieve efficient use of available disk bandwidth, it is common to organize the scheduling of streams into (time) cycles or groups, e.g., as in [5, 22]. In their simplest form, cycle-based schemes deliver in each cycle the data that is read in the previous cycle. During each time period, data for each active stream is read from the disks into main memory while, concurrently, the data read during the previous cycle is transmitted over the network to display stations. The motivation for this organization is to provide opportunities for seek optimization in order to increase the effective bandwidth of the system; the blocks read from a disk during a cycle can be read in any order (since they are not transmitted until the next cycle), for instance using an elevator type algorithm, and thus seek times can be optimized. The main disadvantage of cycle-based scheduling is the memory buffers required to hold the data read during one cycle until it is transmitted in the next cycle. In the remainder of this paper, for the sake of clearity and brevity, we will consider fault tolerance schemes in the context of cycle-based scheduling only, unless otherwise stated[2].

It is useful to generalize the idea of a cycle as follows (also refer to [2]). Define a unit, B, of disk I/O and let b_0 be the bandwidth requirement of an object. Let k_t be the number of disk storage units that are transmitted per cycle per object. If T_{cyc} is the length of a cycle, then $T_{cyc} = (k_t * B)/b_0$. Then if k_r disk storage units are read in a cycle for a stream, where k_r is an integer multiple of k_t,

[2]Qualitatively similar tradeoffs can be illustrated in the context of non-cycle-based scheduling techniques.

then the data read in one "read cycle" is delivered over the next k_r/k_t cycles. This is illustrated in Figure 1, where Ai refers to block i of object A.

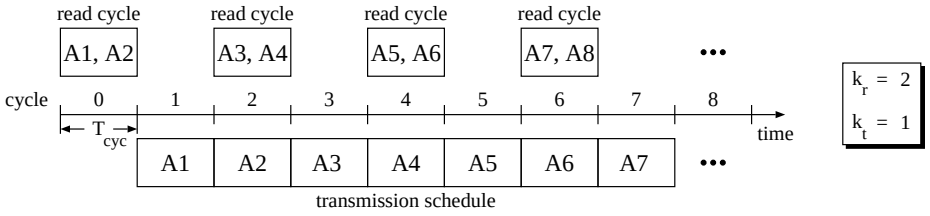

Figure 1: Multiple transmission cycles per read cycle.

Finally, in this paper, we will assume the use of constant bit rate (CBR) data retrieval schemes (e.g., as in [3]), which provide an abstraction of a fixed rate video stream. Furthermore, we will assume provision of guaranteed quality of service (QoS) (e.g., as in [3]). This assumption will be reflected in our models and analysis of the various schemes, as we will have to provide for worst case possibilities and construct upper bounds on system parameters (e.g., cycle time). Note that, an alternative to CBR schemes and guaranteed QoS provisions would be variable bit rate (VBR) schemes (e.g., as in [14]) and/or statisical QoS provisions (e.g., as in [8]). For ease and clarity of exposition we will not consider VBR schemes and/or statistical QoS provisions here; however, many of the issues and tradeoffs discussed in the context of CBR schemes and guaranteed QoS would apply to VBR schemes and/or statistical QoS as well.

2.2 Disk Model

We will assume in the remainder of the paper (unless otherwise stated) that the unit of disk I/O is a track. This is motivated by the reduction in rotational latency[3] achieved as well as by the amortization of the seek overheads over a "large enough" data transfer. We assume for example, that a full track read is started at the next possible sector boundary and therefore there is very little rotational latency.

We use the following simple disk model (as in [2]) for the purpose of discussing basic performance tradeoffs. We will use the following notation for the study and comparison of the different fault tolerance schemes:

[3]Assuming, of course, that the disk is able to start the read at any sector on a track.

τ_{seek}	maximum seek time between the extreme inner and outer cylinders of a disk
τ_{trk}	maximum time attributable to reading a track (or maximum time attributable to rotational delay) *plus* the slowdown and the speedup of the read/write head of the seek time [21]; as in [2], we model this as a cost associated with the reading of a track as opposed to being part of the seek cost.
τ_{read}	maximum time attributable to reading a track (or maximum time attributable to rotational delay) *not including* the slowdown and the speedup of the read/write head of the seek time
B	the number of bytes per track (in megabytes)
D	number of disks in the system
W	size of the video collection of the system (not including the parity information) (in gigabytes)
S	the capacity of a disk (in megabytes)
$T_t(r)$	maximum time to read r tracks[4] $T_t(r) = \tau_{seek} + r * \tau_{trk}$ this equation basically defines the simple disk model
T_{cyc}	the cycle time (sec)
N	(maximum) total number of active streams in a system
N_{eff}^P	maximum number of active streams possible, per disk, under a cycle-based scheme P, without provisions for recovery from failure in real time
N_d^P	maximum number of active streams possible, per disk, under a fault tolerant scheme P, i.e., with provisions for recovery from failure in real time, $N_d^P \leq N_{eff}^P$
k_r	number of tracks read in a "read cycle" per stream
k_t	number of tracks transmitted per stream per cycle; we will assume for simplicity that k_r is an integer multiple of k_t, where $\frac{k_r}{k_t} - 1$ = number of cycles between "read cycles" for the same stream
b_0	object bandwidth (in MBytes/s)
MTTF	mean time to failure of the system
MTTDS	mean time to degradation of service of the system
MTTF(disk)	mean time to failure of a disk
MTTR(disk)	mean time to repair of a disk

[4]This is the time to read tracks for the schemes where the retrieval unit is an integral number of tracks. We will describe a scheme where the retrieval unit is a multiple number of sectors, and in this case the rotational delay will have to be included in the model. In Section 5, we will explore this in more detail. We should also mention that the equation give for $T_t(r)$ is an upper bound — a motivation for this is provisions of guaranteed QoS, as explained in

In terms of the above notation the following expression characterizes T_{cyc} [2]:

$$T_{cyc} = \frac{k_t * B}{b_0}$$

This follows from the definition of a cycle in Section 2.1. For the purpose of computing N, an upper bound on the number of streams that can be supported by the system, we assume that the load is evenly spread over the D disks in the system so that there are $\frac{N*k_r}{D}$ tracks to be read per disk per "read cycle". Then, a constraint, expressing that there must be sufficient time in a cycle to read this number of tracks, is:

$$T\left(\frac{N * k_r}{D}\right) \leq T_{cyc} \quad \Rightarrow \quad \tau_{seek} + \frac{N * k_r}{D} * \tau_{trk} \leq \frac{k_t * B}{b_0}$$

Using this inequality, we can solve for a bound on N or equivalently on N/D, the number of streams per disk:

$$N^P_{eff} = N/D \leq \left[\frac{B * k_t}{b_0} - \tau_{seek}\right] / \left[k_r * \tau_{trk}\right]. \tag{1}$$

where P refers to a particular cycle-based scheme being used.

2.3 Traditional Redundancy-based Schemes

To improve the reliability and availability of a large information system, we can use some fraction of the disk space to store redundant information. Typically, parity-based schemes[18] and mirroring schemes[4] have been used for this purpose[5]. One important goal in designing fault tolerant systems is decreasing the probability of data unavailability while incurring the least cost in disk storage, disk bandwidth, and buffer space. Multiple disks can fail (as long as they are not in the same parity group), and the missing data can still be reconstructed on-the-fly. Only in the unlikely event of two disks in the same parity group failing, would a rebuild from tertiary storage be necessary. This last type of system failure, involving two disks in the same parity group we will refer to as *catastrophic failure*. In a catastrophic failure, portions of objects have to be loaded from tertiary storage devices to reconstruct the contents of the failed disk(s) on spare disk(s). There is another serious type of system failure which we will refer to as *degradation of service*, that can occur when there is insufficient available resources, due to failures, to continue delivering all active requests even though there may still be sufficient information on the disk subsystem to reconstruct the missing

Section 2.1.

[5] Variations on these redundancy schemes, e.g., [15, 6], will be discussed later in the paper.

information. For instance, a single disk failure in a cluster could result in no loss of data, however, the loss of the disk may result in insufficient bandwidth to continue delivery of all requests that were active before the failure occurred. In this case, one or more active requests might have to be terminated and rescheduled for transmission at a later time.

There are three modes of operation for a disk subsystem [15], as originally defined in the context of disk arrays: 1) *normal mode*, where all disks are operational, 2) *degraded mode*, where one (or more) disks have failed, and 3) *rebuild mode*, where the disks are still down, but the process of rebuilding the missing information on spare disks is in progress. Due to lack of space, in this paper we only discuss the system's behavior under normal and degraded modes of operation.

2.4 RAID-based Schemes for VOD Servers

In this section we discuss RAID-based approaches and use them to illustrate the basic tradeoffs and issues in providing fault tolerance in VOD servers. In the remainder of the paper, these schemes serve as "baseline" cases.

RAID-based sequential-type scheme (Streaming RAID)
An example of a sequential-type fault tolerance scheme is the *Streaming RAID* (SR) approach [22][6], where disks are grouped into fixed size clusters of C disks each with one parity disk and $C-1$ data disks. The set of data blocks, one per data disk, and a parity block form a parity group (see Section 2.3). Each object is striped over all the data disks for load balancing. The sequence of parity groups associated with an object are allocated in a round-robin fashion over all the clusters. For each active stream, a parity group is read in each cycle and delivered to the network in the subsequent cycle. Since $C-1$ data tracks are read for a stream in each cycle, $k_r = C-1$ and, since they are all delivered in the next cycle, $k_t = C-1$ (see Section 2.1).

In the event of a disk failure, SR can reconstruct the missing data by a parity computation. Since an entire parity group is read for each active stream in each cycle, if a disk has failed then the missing data that would have been read from that disk can be reconstructed on-the-fly from the other data blocks and the parity block from the same parity group, without hiccups[7]. Note

[6] In what follows we describe a slight generalization of the approach presented in [22].

[7] To account for the computation time to perform the XOR, it may be that the fetch portion of a cycle has to be scheduled to end some number of milliseconds prior to the beginning of data transmission. But this is a minor complication which is not discussed any further in this paper.

that, a RAID5-type organization (where the parity blocks are precessed among all the disks in a cluster) will not alleviate the necessity to reserve (i.e., not utilize under normal operation) $\frac{1}{C}$ of a cluster's bandwidth, since sufficient bandwidth to read parity information must still be reserved, in order to recover from failure in real time.

This scheme can withstand up to one disk failure per cluster before a catastrophic failure occurs (or before there is a degradation of service)[8]. Thus, the main advantage of the Streaming RAID scheme is its high reliability. However, that reliability is gained at the cost of disk storage, disk bandwidth, and buffer space.

Note that, the SR scheme exhibits the same performance characteristics under failure as it does under normal operation. This is partly due to the fact that it exploits the sequential nature of video data delivery and thus, under failure, utilizes the information retrieved for reconstruction of missing data for normal display, i.e., no additional bandwidth is needed under failure. The tradeoff here is the amount of buffer space required to store data, needed for reconstruction of missing information, which is retrieved much earlier than is required by the display rate of the video. Thus, a major disadvantage of the SR approach is the large amount of main memory required per disk. (Of course, a fraction of the disk storage and bandwidth is "wasted" for reliability purposes as well.) Note that, an incentive for a large cluster size is the greater efficiency with respect to disk bandwidth and disk storage (i.e., proportionatly less redundant information will have to be stored and retrieved). However, this must be balanced with the cost of additional main memory which grows linearly with the cluster size[9]. In Section 3 we will discuss schemes which address the problems of both high memory requirements and bandwidth loss and improve on this simple sequential approach. In Sections 5 and 6 we will quantify the above mentioned tradeoffs as well as the improvements discussed in Section 3.

RAID-based non-sequential-type scheme (RAID5)

An alternative to the SR scheme would be to still use a RAID-based system, but disregard the sequentiality of video data delivery. That is, we could use a traditional RAID5 (R5) system, with a (basically) arbitrary construction of parity groups[10]. Data retrieval can proceed one block at a

[8] In this scheme, as will become apparent later, catastrophic failures and degradation of service are "equivalent".

[9] We could achieve a small improvement in buffer space requirements by not reading the parity information during normal operation; however, that could result in hiccups in data delivery during the cycle in which the failure occurs.

[10] Note that, this means that parity groups may be constructed from data blocks belonging to different video objects.

time (i.e., $k_r = 1$ and $k_t = 1$) and thus obviate the need for large amounts of buffer space[11]. Thus, this scheme precludes the possibility that, under failure, information retrieved for reconstruction of missing data can also be utilized for normal display. This means that, to be able to recover from failure in *real time*, we would have to reserve disk bandwidth in the system to insure that when a failure does occur, there will be sufficient bandwidth to retrieve enough information to reconstruct the missing data and to continue with delivery of all requests that were active before the failure. Of course, some additional buffer space would also be needed under failure, for retrieval of data required for reconstruction of missing information.

The R5 scheme is just as reliable as the SR scheme. Furthermore, it exhibits better buffer space utilization characteristics. However, its bandwidth utilization characteristics are quite a bit poorer than those of the SR scheme. In a (basically) read-only system, like a VOD server, a failure of one disk in a cluster results in double the load on all other disks in that cluster. Thus, in the R5 scheme we would have to reserve *half* of the system's bandwidth to insure real-time recovery from disk failure. In other words, we would obtain a large reduction in buffer space requirements at the cost of significant bandwidth loss. In Section 4 we will discuss schemes which address the problem of such high bandwidth loss and improve on this simple non-sequential approach. In Sections 5 and 6 we will quantify the above mentioned tradeoffs as well as the improvements discussed in Section 4.

3 Sequential Schemes

In this section, we discuss sequential-type schemes which improve on various inefficiencies of the basic SR approach (as described in Section 2.4).

3.1 Improving Buffer Space Requirements

As mentioned earlier, one of the inefficiencies of the RAID-based sequential-type approach is the large buffer space requirement (see Section 2.4). In this section we discuss two schemes, the Stag-

[11] There is almost no advantage to reading full parity groups in this case; since most of the blocks in the parity group (except for the one needed for transmission in the next cycle), would be discarded, the only advantage of reading the whole parity group at once would be to prevent hiccups in data delivery during the cycle in which the failure occurs.

gered group scheme and the Non-clustered scheme, which address this problem.

Staggered-group scheme

The Staggered-group scheme (SG), a simple extension to SR, suggested in [22][12] and [2], addresses the main disadvantage of SR, namely the large buffer space requirements, by eliminating the idea that all the data read in one cycle must be delivered in the next cycle. Thus, in this scheme the only difference from the SR scheme is that the data read for an object in one cycle is delivered to the network over the following n cycles. As an example, suppose that we have clusters of size $C = 5$. Then during a cycle that an object A is being read, 4 tracks of real data for A will be read. If we let a cycle length be equal to $\frac{B}{b_0}$, then the four cycles following the read of the data for A will be used to deliver this data. Once every 4 cycles the next parity group is read for A. So the last "transmission" cycle overlaps with the next "read" cycle for A. Similarly, each active stream has the same pattern which repeats every 4 cycles. The SG scheme in effect uses $k_r = C - 1$ and $k_t = 1$ since the cycle length is dictated by the display time of one track of data. Similar "grouping" schemes (although not in the context of fault tolerance or SR) have appeared in other studies [5]. Also, the "prefetching with parity disk" scheme described in [17] is essentially the same as the SG scheme, except that data for each object is read one block at a time, instead of an entire parity group at a time; however, the entire parity group is still stored in memory before its first block is transmitted for display.

The SG scheme has the following characteristics, in comparison with SR: (1) it requires approximately half the memory, (2) the data layout on disk is exactly the same, (3) the only difference in data retrieval is that data read for an object in one cycle can be transmitted over the next several cycles (rather than all in the next cycle), (4) fault tolerance characteristics are the same, and (5) there is a slight cost in disk bandwidth overhead. The savings of approximately half the memory are due to the fact that memory usage is "out of phase" for all streams assigned to different read cycles when one stream is at the point of maximum memory usage (just read its parity group) other streams are at the low ebb of their memory usage. The reason for some loss of disk bandwidth utilization (less streams can be handled) is that the cycles are now shorter and there are fewer requests served per disk per cycle and that in turn means that there are less "opportunities" for seek optimization[13]; this loss in seek optimization is quantified in Section 5.

[12] In [22] it was referred to as "memory sharing with subgrouping and subcycling".

[13] The tradeoffs between improving bandwidth utilization by amortizing seeks over a greater number of streams and increases in buffer space resulting from reading more streams per cycle are investigated in [5].

Non-clustered scheme

In the SR scheme as well as in the SG scheme, an entire parity group is read for each active stream before its first block is transmitted for display (as is the case in the "prefetching with parity disk" scheme in [17], except that it is read one block at a time). This requires a relatively large amount of memory to be allocated to store all the blocks read until they are transmitted. Below we discuss a scheme, introduced in [2], where the constraint of reading an entire parity group before transmitting its first block for display is removed (as we will see later, this could results in a minor degradation in reliability). So, in effect, the cluster size is *decoupled* from the value of k_r, the number of blocks read per object per "read" cycle. In Sections 5 and 6 this Non-clustered(NC) scheme is shown to provide even larger savings in memory (as compared to the SG approach).

All the schemes described thus far are designed to adapt immediately to disk failures without missing the scheduled delivery of any data. It is important to note that most of the memory in the SR and the SG schemes is needed to be able to provide this level of fault tolerance rather than being needed for normal (i.e., fault-free) operation. These observations suggest that much memory could be saved if a (slightly) lower level of fault tolerance were acceptable. Thus, instead of reading an entire parity group in one cycle, we can read only the data which will be delivered during the next cycle. The savings in memory from this scheme as compared to the SR scheme are approximately a factor of C, where C is the number of disks in a cluster. However, this Non-clustered (NC) scheme does not exhibit as good reliability characteristics as the SR or the SG schemes. When a disk failure occurs, in order to recover from it, the affected disk cluster switches to degraded mode of operation, where each stream on this cluster reads an entire parity group at a time, i.e., a cluster operating in degraded mode under the NC scheme behaves just like a cluster operating in normal mode under the SG scheme. There is a short transition phase as the cluster switches to degraded mode, and during this transition phase, some data may be lost and consequently a small number of momentary hiccups will occur. However once the transition to degraded mode is complete, all data will be delivered according to the degraded mode schedule and no additional hiccups will occur. A more accurate description, with smaller data losses and a more detailed discussion of the NC scheme is given in [2].

Note that, if we provide sufficient memory for each cluster in the system to be able to operate under failure, then the buffer space requirement of NC will not be any better than those of the SG scheme. Rather than requiring each cluster to have sufficient memory to run in degraded mode (which is a rare event), an architecture is needed in which (a relatively small amount of) additional buffers space, needed under failure, can be shared, i.e, used by any failed cluster. For instance, as

suggested in [2], there could be one or more extra processors containing a buffer pool to help handle clusters operating in degraded mode. These *buffer servers* could be shared by all the clusters in the system. A cluster in degraded mode sends the data read from the disk to the buffer server and the buffer server takes care of creating the missing data by parity computation and delivering the data on time. (Of course, other architectures are possible.) In a typical system, there might be 100 clusters of 10 disks each, but buffer servers for 5 degraded mode clusters would be sufficient as the probability of more than 5 out of the 100 clusters having a failed disk is extremely low. The mean time to having 5 clusters simultaneously operating in degraded mode is approximately:

$$\frac{MTTF(disk)^5}{1000 * 999 * 998 * 997 * 996 * MTTR(disk)^4}$$

With $MTTF(disk) = 300,000$ hours and $MTTR(disk) = 1$ hour, the mean time until degradation of service (i.e., until utilize all the shared buffer space) would be greater than 250 million years, whereas the mean time to a catastrophic failure (i.e., two failed disks in the same cluster) would be approximately 1141 years. This basically indicates that there is little degradation in system reliability due to providing enough buffer space for only a few clusters to operate under failure, i.e., it is very likely that there will be a catastrophic failure before the system runs out of buffer space for supporting failed clusters.

3.2 Improving Bandwidth Requirements

Thus far we have described sequential-type schemes that have dealt with poor utilization of the buffer space resource. However, one problem which still remains with these schemes is that they can exhibit relatively poor utilization of the bandwidth resource. This can largely be attributed to the fact that in all these schemes the parity disks are not needed during normal operation and therefore their bandwidth, during normal operation, is available but not utilized — it is held in reserve in case of a failure[14]. Below, we describe the Improved-bandwidth (IB) approach, introduced in [2], which addresses this problem; that is, it utilizes the full bandwidth of the system under normal operation but is still able to react to disk failure in real time. For simplicity we discuss this approach in the framework of the SR scheme, *but the IB technique is applicable to other schemes as well.*

Improved-bandwidth scheme

[14]Note that, a RAID5-type organization (where the parity blocks are precessed among all the disks in a cluster) will not alleviate this problem, since sufficient bandwidth for reading parity information must still be reserved, in order to recover from failure in real time.

One way to address the problem of wasted bandwidth resources (for reliability purposes), as described above, is to distribute the parity information associated with data on disk cluster i over the disks of disk cluster $i+1$. For instance, in Figure 2, $X0p$, stored on disk 4 of cluster 1, is the parity information for $(X0, X1, X2, X3)$, which are stored on disks $(0, 1, 2, 3)$ of cluster 0. In this scheme, a failure of a disk in cluster i, will result in an additional load on cluster $i+1$ (refer to Figure 2). Therefore, one important issue in the design of the IB scheme is to be able to accommodate this additional load in real time. When a disk failure occurs in cycle t in cluster i, then cluster i and

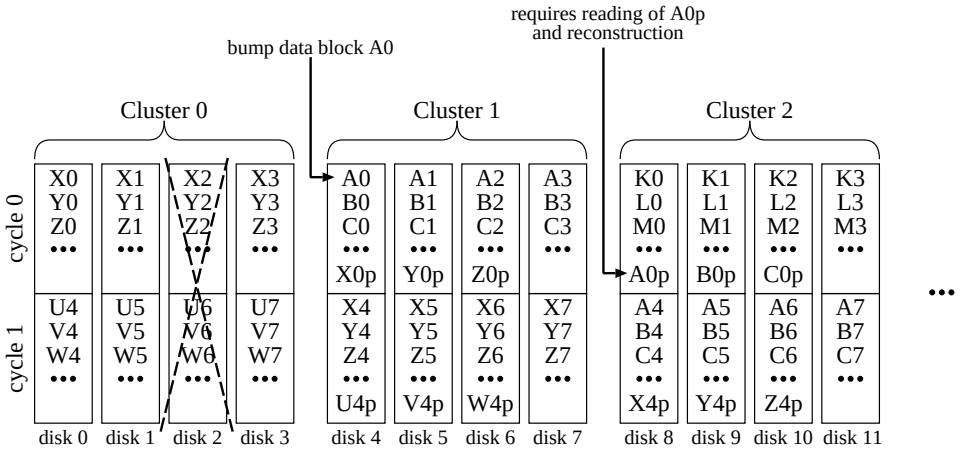

Figure 2: Improved Bandwidth Approach Applied to SR

its right hand neighbors have to perform a "shift to the right" as follows. Cluster i delivers data from all its operational disks plus it reconstructs the missing data by reading the appropriate parity blocks residing on disks of cluster $i+1$. Those disks of cluster $i+1$ that do not have sufficient idle disk bandwidth to serve both the local requests and the parity blocks requests from cluster i drop some of the local requests in favor of reading the parity blocks[15]. The dropped local requests are treated as a partial disk failure of cluster $i+1$ which generates parity block requests on cluster $i+2$. Note that of these dropped blocks, no more than one can be from each parity group. This shift has to propagate to the right until enough idle capacity is found. If there is not enough bandwidth available in the system, then a degradation of service occurs, and one or more requests must be terminated and (possibly) rescheduled at a later time. This situation can arise under two conditions; we elaborate on these conditions below.

The major advantage that this scheme has as compared to the schemes described thus far is

[15]This is similar to how "chained declustering" [13] handles failures; note also, that the Improved-bandwidth approach basically degenerates to the chained declustering approach when the parity group size is equal to 1.

the additional bandwidth available during normal operation. However, it is not as reliable as, for instance, the SR scheme. The number of possible scenarios where the second failure turns out to be catastrophic is greater in this system than with the SR scheme. In general, an SR or an NC system with K clusters, can withstand up to K failures, as long as there is no more than one failure per cluster, before data is lost. In the IB scheme, a failure in each of any two adjacent clusters results in some data loss. Thus an IB system with K clusters can possibly withstand up to $\frac{K}{2}$ failures before some data is lost. The increased sensitivity to a second failure is due to the fact that there are dependencies between parity groups which do not exist in the SR scheme; for instance, disk 4 in Figure 2 belongs to two different "types" of parity groups because it acts as the parity disk for cluster 0 and as a data disk for cluster 1. The mean time to catastrophic failure in this scheme is approximately:

$$\frac{MTTF(disk)^2}{D * (2C-1) * MTTR(disk)}$$

where the $(2C-1)$ factor in the denominator reflects the additional exposure to disk failures. The mean time to catastrophic failure in this case with $MTTF(disk) = 300,000$ hours, $MTTR(disk) =$ 1 hour, $D = 1000$ disks, and $C = 10$ disks per cluster is approximately 540 years rather than 1141 years as in the SR and the NC schemes. However, as long as this is an "acceptable" MTTF of a system, IB can be an attractive scheme for providing fault tolerance in VOD servers (as will be illustrated in Sections 5 and sec-compare).

In addition to catastrophic failure, degradation of service can also occur. If the IB system is running at full capacity, then a disk failure results in degradation of service (however, there is no scheme that can recover from such failure without degradation of service). Furthermore, degradation of service will also occur if during a "shift to the right" the system runs into another failed cluster, operating at full capacity, before it is able to "smooth out" the load created by the failure (in this case, the system would not be able to "shift to the right" past that failed cluster). However, we can improve on both of these situations by reserving a small amount of idle capacity, spread among all the clusters of the system. In that case, the IB system will result in a sufficiently high degree of availability. For instance, if we reserve 5 disks worth of bandwidth available in the system, then the mean time to degradation of service is the same as the mean time to degradation of service in the NC scheme (see Section 3.1) or about 250 million years[16]. (This is based on an optimistic assumption that the parity blocks that have to be read are evenly spread over a cluster.)

[16]Essentially, in both cases we are assuming that failures are rare, and thus we are reserving a *small* amount of available capacity, whether it is disk bandwidth or buffer space, in order to *"share"* it between all the clusters in the system, in case of failure.

On the other hand, the mean time until catastrophic failure is approximately 540 years. Therefore, we can significantly improve system reliability (i.e, mean time until degradation of service) by reserving only a very small fraction of the total disk bandwidth for "smoothing out" the additional load caused by a failure, i.e., it is very likely that there will be a catastrophic failure before the system runs out of bandwidth for performing a "shift to the right" (as described above).

Lastly, if a failure occurs in the middle of a cycle, then it might not be possible to mask the failure for the objects scheduled on the failed disk during that cycle, since parity blocks are not being read concurrently with the data blocks under normal operation. A sophisticated scheduler, during normal operation, might adapt to the system load by reading parity blocks under lightly loaded conditions (and thus avoid isolated hiccups); as the load increases, reading parity blocks can be dropped in favor of supporting more streams.

The "prefetching without parity disk" scheme, described in [17], is essentially the same as the IB scheme, except that the parity information is spread among all other disks in the system, as opposed to just the adjacent cluster. It is not clear whether this approach would balance the additional load due to failure better than the IB scheme, since the IB scheme is able to adjust to changes in workload smoothly, by shifting the load to the right. Furthermore, "prefetching without parity disk" [17] has a serious reliability problem. As stated above, the IB scheme improves the bandwidth efficiency of the system at the cost of reducing the system reliability by approximately a half, since it can handle (simultaneous) single disk failures in at most half of the clusters in the system. The "prefetching without parity disk" [17] gives up much more in system reliability since it will lose some data from the disk subsystem after *any two disk failures*, i.e., after any two disk failures, some data will have to be reconstructed from tertiary storage.

4 Non-sequential Schemes

The main advantage of non-sequential schemes is that they exhibit better buffer space utilization than sequential schemes; however as already mentioned, a major disadvantage is the poor utilization of the disk bandwidth which is due to the necessity to reserve additional bandwidth for real-time recovery from failure. This is basically the same problem that is faced by traditional RAID system, which can suffer from poor performance in degraded modes of operation, as a result of an increase in workload due to failure. Hence, one approach to addressing this problem would be to use a

solution similar to the one used in traditional disk subsystems, namely, achieve a smaller increase in workload under failure (i.e., better degraded mode performance) at the cost of storing more (than "necessary") redundant information (and/or possibly sacrificing some of the reliability, as will become apparent later). One example of a traditional approach is the Clustered RAID scheme, proposed in [15]. We first describe the main idea behind Clustered RAID and then two variations on the adaptation of the Clustered RAID framework to VOD servers, one using the Balanced Incomplete Block Design (BIBD) [12] technique as suggested in [15] and then applied specifically to VOD servers in [17], and the other using the Segmented Information Dispersal (SID) technique [6].

The basic idea behind Clustered RAID is to relax the constraint used in traditional RAID architectures [18], that the parity group size, G, is equal to the cluster size, C — here, "parity group size" refers to the number of data blocks plus the parity block, and "cluster size" refers to the number of disks over which the parity group blocks are distributed. One of the motivations for use of techniques where the parity group size is smaller than the cluster size is a better load balancing characteristics under disk failure. To deliver video data using the Clustered RAID scheme, we can use the same approach as in the case of R5 (see Section 2.4), with the notable exception that we would need to reserve enough bandwidth (in case of failure) on each disk of the cluster to read only $x\frac{G-1}{C-1}$ bytes of data (as compared to x bytes of data in the case of R5) to reconstruct the missing x bytes of data. Of course, by choosing values of G which are small relative to C, we can reduce the amount of bandwidth wasted for reliability purposes (but at the cost of additional disk space). Several methodologies have been proposed for constructing parity groups such that the load is distributed uniformly across the disks in case of a failure — BIBD and SID are two which are described below in the context of VOD servers.

4.1 Improving Bandwidth Requirements through SID

The SID scheme described in [6] uses the Clustered RAID type approach to address the bandwidth inefficiency problem of a traditional RAID system in the context of VOD servers. We briefly describe the SID approach below. As in the case of an R5 scheme (see Section 2.4), both real data and redundant information are stored on each disk. The main feature of the SID scheme is that it maintains the *limited continuous access length* (LCAL) property which states that during the recovery of any k ($1 \leq k \leq b/u$) contiguous interleave units which belong to the data portion of a missing fragment, no more than $\lceil k/q \rceil * u$ bytes must be read from the available fragments, where

q is the dispersal factor and u is the size of the interleave unit (in bytes). Thus q is one parameter which can be used to control the amount of bandwidth wasted for reliability purposes. (This is analogous to the $\frac{G-1}{C-1}$ parameter of the Clustered RAID.)

The LCAL property that characterizes SID guarantees that during the recovery of any k ($1 \leq k \leq \frac{b}{u}$) contiguous interleave units which belong to the data portion of a missing framgement, no more than $\lceil k/q \rceil * u$ bytes must be read from the fragments on the surviving disks of a cluster. Another parameter which will aid in determining the effectiveness of the SID scheme is the redundancy rate c, which is the ratio of the amount of redundant data to the total amount of data[17]. Of course, not all values of q, C (the cluster size), and c will maintain the LCAL property. Hence, part of the difficulty is in choosing proper values and constructing a proper data layout that will correspond to these values.

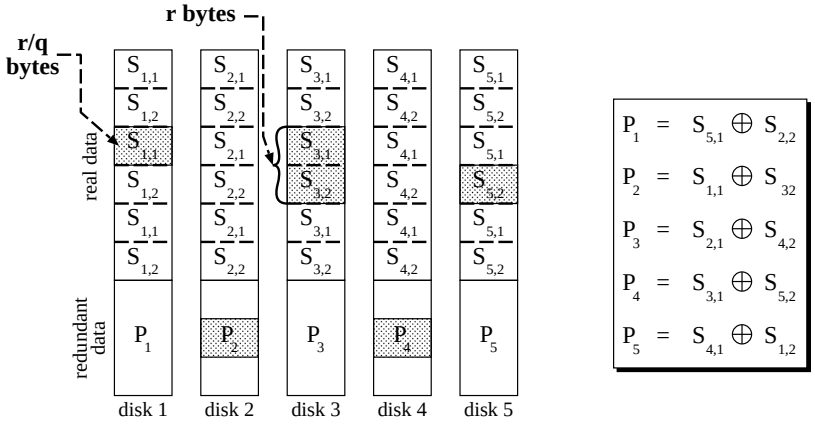

Figure 3: SID organization

We illustrate a special case of the SID organization (namely Constrained SID) through the example of Figure 3. In this example, $C = 5$, $q = 2$, $b = 2$, and $c = \frac{1}{3}$. Each disk i is divided into segments $S_{i,j}$, where $1 \leq j \leq q$, which are $\frac{r}{q}$-interleaved. The interleaving insures that each block of size r contains equal amounts of (real) data from each segment of the disk. The check segment of each fragment contains the parity (exclusive or) of q data segments that belong to q other fragments. If a failure occurs, for instance of disk 3, then, e.g., a shaded region, of size r bytes, of that disk can be reconstructed by reading (contiguous) shaded regions, each of size $\frac{r}{q}$ bytes, of the surviving disks of the cluster. To deliver video data using an SID scheme, we can

[17]There are many more parameters of the SID scheme which are given in [6]; however, for ease of presentation, we will limit our discussion to those that can aid us in presenting a relatively simple version of SID and still allow us to illustrate the advantages and disadvantages of this approach.

use the same approach as in the case of R5, which is reading blocks of r bytes (see Section 2.4), with the notable exception that we would need to reserve enough bandwidth (in case of failure) on each disk of the cluster to read only $\frac{r}{q}$ bytes of data (as compared to r bytes of data in the case of R5) to reconstruct the missing r bytes of data in real time. As already mentioned, the SID scheme depicted in Figure 3 is a Constrained SID organization. In this case, given a cluster of C disks, using the LCAL property it can be shown that $q^2+1 \leq C$ (where a valid q must first be determined — see [6] for details). Of course, in the Constrained SID scheme, larger values of q will result in less storage and bandwidth wasted for reliability purposes. On the other hand, given the above constraints, larger values of q will also *force* larger values of C — this can result in a degradation in system reliability. The performability comparison between SID and R5 is quantified in Sections 5 and 6.

Note that, by carefully choosing the values of the SID parameters, such as q (or C), it is possible to construct SID solutions that are equivalent (i.e., degenerate to) other redundancy schemes, such as R5 [18], interleaved declustering [1, 7] (i.e., a version of full replication), and Clustered RAID [15] (see [6] for details).

In this paper we will be consider several variations of the Constrained SID scheme. The motivation for considering multiple versions of the SID approach is the fact that the size of the retrieval unit and the precise scheduling of data retrieval are not specified in [6], and, as will become apparent in Sections 5 and 6, these choices can significantly affect the performance of the VOD system. Namely, they affect the maximum number of streams that can be active simultaneously, the disk bandwidth utilization, the buffer space requirements, and the cost per stream. The two variations of the SID scheme considered in this paper are SID_s and SID_b. In the SID_s schemes the retrieval unit is a set of q consecutive tracks. During the recovery process, the LCAL property guarantees that to reconstruct these q tracks, each surviving disk of the cluster has to read no more than one track. On the other hand, the SID_b scheme uses one track as its retrieval unit. In this case, under the degraded mode of operation, each surviving disk reads $\frac{track\_size}{q*sector\_size}$ sectors for reconstruction of each missing track. We should also mention, that the total amount of time needed to read these sectors for each missing track includes the maximum rotational latency (plus the slowdown and the speedup of the read/write head of the seek time, as described in Section 2.2) and the time to read the $\frac{track\_size}{q*sector\_size}$ sectors ($\approx \tau_{read}/q$).

Under all SID schemes, bandwidth must be reserved to insure that each disk will have sufficient bandwidth to retrieve the additional data or parity information in order to reconstruct the missing

data of the failed disk in real time. Specifically, for the SID_s variation there will be sufficient bandwidth to retrieve both the data needed to service all active streams and the data needed for reconstruction, if the following constraint is satisfied:

$$\begin{aligned} N_d^{SID_s} &+ \frac{N_d^{SID_s}}{q} \leq N_{eff}^{SID_s} \\ \Rightarrow N_d^{SID_s} &\leq N_{eff}^{SID_s} * \frac{q}{q+1} \end{aligned}$$

where $N_d^{SID_s}$ is the maximum number of streams per disk that can be active under the SID_s scheme. In the SID_b scheme, the surviving disks read data blocks for reconstruction of missing information during the degraded mode of operation, of size less than a single track. The total time that it takes for a disk to read $N_d^{SID_b}$ tracks and $N_d^{SID_b}$ data units (used for reconstruction) is:

$$T_{cyc} = \tau_{seek} + N_d^{SID_b} * \tau_{trk} + N_d^{SID_b} * \left[\tau_{trk} + \frac{\tau_{read}}{q}\right]$$

Given above, in order for the system to guarantee that it can retrieve (in real time) the additional data needed for reconstruction in case of failure, it must also satisfy the following constraint:

$$N_d^{SID_b} \leq \frac{\frac{B}{b_0} - \tau_{seek}}{2\tau_{trk} + \frac{\tau_{read}}{q}}$$

As already mentioned in the R5 scheme reconstruction of r consecutive bytes requires every disk to retrieve r bytes of data or parity information. Therefore, in the case of R5, the following constraint must be satisfied: $N_d^{R5} \leq \frac{N_{eff}^{R5}}{2}$.

4.2 Improving Bandwidth Requirements through BIBD

The main difference between BIBD and SID is in what constraints are placed on the construction of redundant information and on the layout of real and redundant data within a cluster. Of course, these differences can reflect on the performance of the system, both under normal operation and under failure.

The BIBD approach specifies a way of constructing s parity groups of size k (i.e., k disks belong to a parity group), where every pair of disks occurs in exactly λ different parity groups and each disk occurs in exactly r different parity groups. In addition, the total number of disks is equal to u. Thus, a BIBD solution satisfies the following two equations ([12]):

$$\begin{aligned} r * (k-1) &= \lambda * (u-1) \\ s * k &= u * r \end{aligned} \tag{2}$$

A "declustered parity based" scheme, utilizing the BIBD approach in a fault tolerant design of a VOD server is described in [17]. In this scheme, the retrieval unit, per stream, of size b bytes can be viewed as an interleaved form of r data blocks, each of size b/r bytes and each assigned to a different parity group. If D_i is the failed disk, then for each surviving disk D_j in the affected cluster, there are λ parity groups to which the pair (D_i, D_j) belongs. The total number of blocks that a disk has to read for the reconstruction of the retrieval unit is λ. Therefore, if N_d^{BIBD} is the maximum number of active streams on a disk, then (using Eq. (2)) $N_d^{BIBD} * r * \frac{k-1}{u-1}$ additional blocks have to be read by each surviving disk (or $N_d^{BIBD} * \frac{k-1}{u-1} * b$ bytes), where $\frac{k-1}{u-1} \leq 1$. In the case of the R5 scheme, the total number of bytes retrieved by each of the surviving disks is $N_d^{R5} * b$. Therefore, the amount of bandwidth that must be reserved for reconstruction of missing data, in case failure, under the BIBD scheme is potentially smaller than the amount of bandwidth that would have be be reserved under the R5 scheme. Given a BIBD-based scheme, where the retrieval unit consists of r tracks[18], and a scheduling scheme where $k_r = k_t = r$, there will be sufficient bandwidth to retrieve both the data needed to service all active streams and the data needed for reconstruction, if the following constraint is satisfied:

$$\begin{aligned} N_d^{BIBD} &+ N_d^{BIBD} * \frac{k-1}{u-1} \leq N_{eff}^{BIBD} \\ \Rightarrow N_d^{BIBD} &\leq N_{eff}^{BIBD} * \frac{u-1}{k+u-2} \\ \Rightarrow N_d^{BIBD} &\leq \left[\frac{B}{bo * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * r}\right] * \frac{u-1}{k+u-2} \end{aligned} \tag{3}$$

One main difference between the SID and the BIBD schemes is that BIBD cannot guarantee that the data blocks retrieved for reconstruction of missing information during the degraded mode are consecutive. In our model, this difference can affect the bandwidth utilization only if the size of the block read for reconstruction purposes is less than a track. In that case, there will be an additional overhead of the rotational time delay (as described in Section 4.1). Due to the similarities between the SID and the BIBD approaches and to the lack of space, we only analyze the SID scheme in Sections 5 and 6.

[18]This is unrelated to the r parameter given above.

5 Analysis

In this section we analyze the SR, SG, IB, NC, SID_s, SID_b, and R5 schemes based on: (1) reliability considerations, including their susceptibility to catastrophic failure and degradation of service, (2) maximum number of simultaneous display streams they can support, and (3) costs associated with storing redundant information. The penalties reflected in the above given metrics can fall into one of three categories:

disk storage	the amount of disk storage that must be dedicated to redundancy, e.g., parity information, which can not be used to store real data
bandwidth	the amount of bandwidth that must be dedicated to redundancy, e.g., for retrieving parity, which can not be used for retrieving real data
buffer space	the amount of memory needed to support a redundancy scheme, e.g., for storing some portion of a parity group until it can be delivered to display stations

In general, these penalties are not independent of each other. For instance, in certain parity placement schemes (see Section 2.4), a penalty in storage is accompanied by a penalty in bandwidth. Or, as will become evident later in Section 6, it is often possible to tradeoff disk storage cost for buffering cost, and vice versa. (In addition to the quantitative metrics, one should also consider more qualitative factors, such as: (a) complexity of scheduling retrieval and delivery of objects during normal and degraded modes of operation, (b) complexity of data layout, and (c) complexity of the rebuild process. However, due to lack of space we do not discuss these any further.)

Disk space overhead

Let S^p denote the amount of additional disk storage space required by scheme p, where $p = SR$, SG, NC, IB, SID_s, or SID_b. Then:

$$S^{SR} = S^{SG} = S^{NC} = S^{IB} = S^{R5} = \frac{D * S}{C} \quad (4)$$

$$S^{SID_b} = = S^{SID_s} = \frac{D * S}{q} \quad (5)$$

where S is the disk capacity (in megabytes), C is the parity group size, q the dispersal factor of the layout, and D is total number of disks in the system.

Disk bandwidth overhead

The amount of additional disk bandwidth, BW^p, required by scheme p is:

$$BW^{SR} = BW^{SG} = BW^{NC} = \frac{D * d}{C} \quad (6)$$

$$BW^{IB} = K_{IB} * d \quad (7)$$

$$BW^{R5} = \frac{D * d}{2} \quad (8)$$

$$BW^{SID_b} = BW^{SID_s} = \frac{D * d}{q} \quad (9)$$

where d is the bandwidth of a single disk, C is the parity group size, and $K_{IB} * d$ is the disk bandwidth that is reserved in the IB scheme to insure a reasonable mean time to degradation of service (MTTDS) (see Section 3.2).

Reliability

The reliability of the various schemes can be compared using the following equations. The susceptibility to catastrophic failure, i.e., $MTTF^p_{D|C}$ (for each scheme p) as a function of the parity group size, C, and the number of disks in the system, D, is:

$$MTTF^{SR}_{D|C} = MTTF^{SG}_{D|C} = MTTF^{NC}_{D|C} \approx \frac{\text{MTTF(disk)}^2}{D * (C-1) * \text{MTTR(disk)}} \quad (10)$$

$$MTTF^{IB}_{D|C} \approx \frac{\text{MTTF(disk)}^2}{D * (2C-1) * \text{MTTR(disk)}} \quad (11)$$

$$MTTF^{SID_b}_{D|C} = MTTF^{SID_s}_{D|C} \quad (12)$$

$$\approx \frac{\text{MTTF(disk)}^2}{\frac{W}{S} * q * (1+q) * \text{MTTR(disk)}} \quad (13)$$

The mean time to degradation of service for the SR and the SG schemes is the same as their mean time to catastrophic failure. The situation is different for the NC, IB and the SID schemes. The degradation of service in the SID variations[19] or in the NC scheme occurs due to lack of buffer space (i.e., when the $(K_{NC}+1)$st failure results in a need for more buffer space and the buffer pool is empty, where K_{NC} is the number of "buffer nodes" in the system). The degradation of service in the IB scheme occurs when there is a lack of available bandwidth capacity (in the right places) to perform the shift. Hence:

$$MTTDS^{NC} = MTTDS^{IB} = MTTDS^{SID_a}$$

$$\approx \frac{\text{MTTF(disk)}^K}{D*(D-1)*\cdots*(D-K+1)*\text{MTTR(disk)}^{(K-1)}} \quad (14)$$

[19]This is a slight generalization of the SID approach described in [6]; sharing of a buffer pool between clusters was not considered in [6].

Schemes	k_r	k_t
SR	$C-1$	$C-1$
NC	1	1
IB	$C-1$	$C-1$
$R5$	1	1
SID_s	q	q
SID_b	1	1

Table 1: Characterization of schemes.

where $a = s, b$ and the assumptions are that in the case of the NC scheme and in all the SID variations, there is sufficient buffer space to mask $K = K_{NC}$ failures, whereas in the case of the IB scheme, there is $K = K_{IB}$ disks worth of bandwidth capacity reserved in the system[20] (see Sections 3.1 and 3.2 for details). (In the remainder of the paper, we will use the notation K_P for the number of disk failures that scheme P will be able to mask.) Note that, given the same amount of redundant information and the same total number of disks in the NC, IB, and SID schemes, the SID scheme exhibits poorer reliability characteristics. This is due to the dependence between the cluster size and the amount of redundant information in a SID-based scheme (see constraints given in Section 4.1), which forces the system into using larger (and fewer) clusters, and thus results in a lower MTTF.

Number of Simultaneously Supported Streams

The number of simultaneously supported streams, N, was given in Eq. (1). Thus, the *maximum* number of simultaneously supported streams, N^p, for each scheme p is (refer to Table 1 for details):

$$N^{SR} = \left\lfloor \frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)} \right\rfloor * D\frac{C-1}{C} \tag{15}$$

$$N^{SG} = \left\lfloor \frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}} \right\rfloor * D\frac{C-1}{C} \tag{16}$$

$$N^{NC} = \left\lfloor \frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}} \right\rfloor * D\frac{C-1}{C} \tag{17}$$

$$N^{IB} = \left\lfloor \frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)} \right\rfloor * (D - K_{IB}) \tag{18}$$

[20]There is an additional constraint, in the case of the IB scheme (which is not considered in this computation); namely, the reserved capacity has to be in the "right" places, since more than two failures in a single stretch of clusters with no available capacity results in degradation of service.

$$N^{SID_s} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * q}\right] * D * \left[\frac{q}{q+1}\right] \quad (19)$$

$$N^{R5} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] \left[\frac{D}{2}\right] \quad (20)$$

$$N^{SID_b} = \frac{\frac{B}{b_0} - \tau_{seek}}{2\tau_{trk} + \frac{\tau_{read}}{q}} * D \quad (21)$$

Additional Buffer Space

At this point we can compute the buffer space requirements[21], BF^p, based on the maximum number of streams supported by each scheme, p. The number of buffers necessary per stream per cycle in the SR scheme is $2C$; hence:

$$BF^{SR} = 2C * B * \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)}\right] * D\frac{C-1}{C} \quad (22)$$

The number of buffers necessary per each set of $C-1$ streams in each cycle of the SG scheme is $(C+1)+(C-1)+(C-2)+\cdots+3+2 = \frac{C(C+1)}{2}$, hence;

$$BF^{SG} = \frac{C(C+1)}{2} * B * \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] * D\frac{C-1}{C} * \frac{1}{C-1} \quad (23)$$

The number of buffers necessary per stream per cycle in an NC scheme under normal operation is simply 2; the number of buffers necessary per stream per cycle in an NC scheme under failure, is the same as in the SG scheme, except that the NC scheme only provides enough buffer space for K_{NC} (one per cluster) failures; hence:

$$BF^{NC} = 2 * B * \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] * D\frac{C-1}{C} + \left[\frac{BF^{SG}}{(D\frac{C-1}{C})/C} * K_{NC}\right] \quad (24)$$

The number of buffers necessary per stream per cycle in the IB scheme is the same as in the case of the SR scheme, except that no buffer space needs to be reserved for storing parity data. Therefore the number of buffers necessary per each steam is $2(C-1)$, and hence:

$$BF^{IB} = 2(C-1) * B * \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)}\right] * (D - K_{IB}) \quad (25)$$

The buffer space requirements for the R5 scheme are (where we provide sufficient buffer space to mask no more than K_{R5}, in order to make a fair comparison with other schemes):

$$BF^{R5} = B * N_d^{R5} * [2 * D + (C-1) * K_{R5}] \quad (26)$$

[21] For ease of exposition, in this section we assume double buffering in most cases; of course, this can be improved upon in an actual implementaion.

Finally, the buffer space requirements for the SID-based schemes can be computed as follows:

$$BF^{SID_s} = B * q^2 * (\frac{2 * D + K_{SID_s} * (q+1)}{q+1}) * (\frac{B}{b_o * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * q}) \quad (27)$$

$$BF^{SID_b} = B * \frac{\frac{B}{b_0} - \tau_{seek}}{2\tau_{trk} + \frac{\tau_{read}}{q}} * (2 * D + K_{SID_b} * (q+1)) \quad (28)$$

Cost Per Stream

Since it is not immediately clear how to compare savings in I/O bandwidth (when sequential-type schemes are used) with savings in buffer space (when using non-sequential-type schemes are used), one approach is to assess this tradeoff through cost considerations. Thus, as already mentioned, a meaningful performance measure is $/stream, i.e., one can compare alternative fault tolerance schemes by considering: (a) the overall cost of the system based on system requirements, e.g., in terms of the cost of disks plus main memory, which are dictated by I/O bandwidth, buffer space, and storage for fault tolerance needs, and (b) the maximum number of streams that can be simultaneously supported by the corresponding VOD architecture.

Therefore, given that cost is a constraint, we would like to study how the different schemes affect the cost per stream metric. As we improve the disk storage efficiency (by increasing the parity group size) and hence the cost of the *disk subsystem*, we increase the *buffering* cost. However, additional memory space and larger clusters also allow us to increase the maximum number of streams that we can support simultaneously. As an example use of our analysis, consider the problem of sizing and selecting data layout and scheduling schemes for a system with a fixed working set size, W, which is the amount of real data that we would like to store on the disk-subsystem. In the following equations, the cost per stream is derived for the SR, SG, NC, IB, R5, SID_b, and SID_s schemes, where $Cost^p$, for each scheme p is defined as the cost of the buffer space plus the cost of the disk storage subsystem which need to be provided in order to support a fixed working set W and the fault tolerance scheme p.

$$\frac{Cost^{SR}}{N^{SR}} = C_b * 2 * C * B + C_d * \frac{C}{C-1} * \frac{S}{\left[\frac{B}{bo * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)}\right]} \quad (29)$$

$$\frac{Cost^{SG}}{N^{SG}} = C_b * \frac{C * (C+1)}{2 * (C-1)} * B + C_d * \frac{C}{C-1} * \frac{S}{\frac{B}{bo * \tau_{trk} - \frac{\tau_{seek}}{\tau_{trk}}}} \quad (30)$$

$$\frac{Cost^{NC}}{N^{NC}} = C_b * \left[2B + \frac{B * K_{NC} * C * (C+1)}{2 * (C-1) * \frac{D * (C-1)}{C^2}}\right] + C_d * \frac{S}{\left[\frac{B}{bo * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] * \frac{C-1}{C}} \quad (31)$$

$$\frac{Cost^{IB}}{N^{IB}} = C_b * 2 * (C-1) * B + C_d * \frac{\frac{D*S}{D-K_{IB}}}{\left[\frac{B}{bo*\tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}*(C-1)}\right]} \quad (32)$$

$$\frac{Cost^{R5}}{N^{R5}} = C_b * B * q * (2 + \frac{K_{R5} * q}{D}) + C_d * S * \frac{2}{\left[\frac{B}{bo*\tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right]} \quad (33)$$

$$\frac{Cost^{SID_s}}{N^{SID_s}} = C_b * B * q * (2 + \frac{K_{SID_s} * (q+1)}{D}) + C_d * S * \frac{1+\frac{1}{q}}{\left[\frac{B}{bo*\tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}*q}\right]} \quad (34)$$

$$\frac{Cost^{SID_b}}{N^{SID_b}} = C_b * B * (2 + \frac{K_{SID_b} * (q+1)}{D}) + C_d * S * \frac{1}{\left[\frac{\frac{B}{bo}-\tau_{seek}}{2\tau_{trk}+\frac{\tau_{read}}{q}}\right]} \quad (35)$$

6 Comparison of Schemes

In this section we compare the different schemes using two metrics: the buffering requirements and the cost per stream. The motivation for this comparison is to illustrate the issues and tradeoffs associated with fault tolerant design of VOD servers, rather than to determine the "best" approach. It is worth noting (as will become apparent in the figures), that different schemes have their optimum operating points at different architectural configurations; thus it is not the case that one fault tolerance scheme is absolutely better than another, but rather that one must understand the system requirements and constraints (e.g., minimum number of simultaneously supported streams) and then choose a fault tolerance scheme accordingly.

Given that the probability of K_P failures is sufficiently low (i.e., we achieve an "acceptable" level of system reliability), in the remainder of this section we will fix this parameter and concentrate mostly on the cost/performance comparison. We will consider a video database with a fixed amount of "real" data, W[22]. The system parameters used in the comparisons are described in Table 2. The total number of disks is maintained at the minimum required to hold the video collection and the parity information. Before discussing the results presented in this section, we would like to note the difference between a parity group size and a cluster group size, as used in this section. Here, the parity group size refers to the number of real data blocks in a parity group (i.e., not including the parity block). Note that, for the SID variations discussed here, the parity group size refers to the dispersal factor q where the constraint on the cluster size must be maintained, i.e., $C_{SID_a} = q^2 + 1$, for $a = s, b$. Clearly, schemes with the same parity group size have the same amount of redundant

[22]Since the amount of "real" data is kept constant, the number of disks in the storage system depends on the particular fault tolerance scheme being used, i.e., D is a function of W, C, and q.

Parameter	Value
b_o	3.5 Mb/sec
B	0.05 MB
τ_{seek}	24 ms
τ_{trk}	12ms
τ_{read}	8.3ms
K_P	5 failures
1 sector	100 blocks
Size of Real Data W	1000GB
Disk Capacity S	1000MB
Disk Cost C_d	\$0.3 per MB
Memory Cost C_b	\$30 per MB
MTTF(disk)	300,000 hours
MTTR(disk)	1 hour

Table 2: System parameters.

information.

Figure 4 illustrates the buffering requirements for the SID-based schemes as a function of the parity group size. Clearly, the SID_s scheme needs more buffer space, since in that scheme q tracks are retrieved per stream per cycle, as opposed to only a single track per stream per cycle as in the case of the SID_b scheme[23]. Furthermore, the buffer space needed for the SID_s scheme increases as the parity group size increases. This is due to an increase in the size of the retrieval unit of each stream. On the other hand, as can be seen from Figure 4, the memory requirements of the SID_b scheme change at a slower rate, as a function of the parity group size.

Figure 5 depicts the cost per stream of the SID-based schemes as a function of the parity group size. The cost per stream, of course, is a function of the the total number of streams that can be supported simultaneously. Note that, SID_s has relatively large buffer space requirements (see Figure 4). However, its performance, as measured in terms of cost per stream, is better than that of SID_b, which exhibited lower buffer space requirements. This is due to its ability to support a larger number of streams simultaneously, since it reads whole tracks at a time and is thus more

[23]As already mentioned, for the experiments with SID, we tried different retrieval unit sizes, since this was not specified in [6]. For example, in SID_b, under the degraded mode of operation, each disk reads $\frac{track\_size}{q*sector\_size}$ sectors for reconstruction of each missing track.

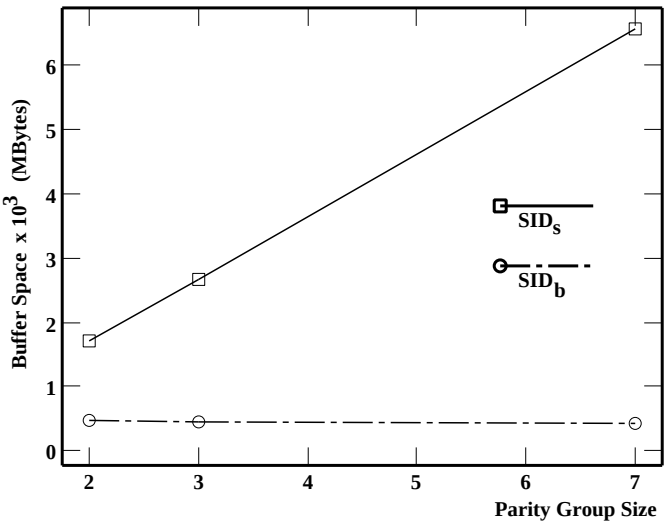

Figure 4: Memory requirements of SID based-schemes.

efficient at utilizing disk bandwidth than SID_b. This illustrates the merits of considering a "proper" performance metric, namely, in this case, \$/stream.

Next, we discuss results for the two sequential schemes, namely NC and IB, using the same metrics. Figure 6 illustrates the buffering requirements of NC as a function of the parity group size. As the parity group size increases, the buffering requirements of NC increase very slowly (this increase is due to the increase in the size of clusters which consequently results in an increase in the amount of buffer space that must be allocated to support K_{NC} clusters operating under failure). From Figure 6, we observe that the buffering requirements of the IB scheme are greater than those of the SID_s scheme. This is due to the fact that IB can support a larger number of streams simultaneously than SID_s.

Figure 7 illustrates the cost per stream curves as a function of the parity group size for sequential schemes (NC and IB) as well as non-sequential schemes (SID_s, SID_b, and RAID5). For a small parity group size, the IB scheme has the best performance as measured in terms of cost per stream (e.g., \$43 at the parity group size of 1)[24]. SID_s reaches its minimum cost per stream at a parity group size of 3 which then increases slowly as the parity group size continues to grow. Note also, that NC and SID_s perform similarly at a parity group size of 3. However, as was emphasized

[24]Essentially this is the mirroring case, where the second copy can be used to serve more requests. Furthermore, since disk capacity is growing more rapidly than bandwidth capacity, full replication schemes are becoming more cost effective.

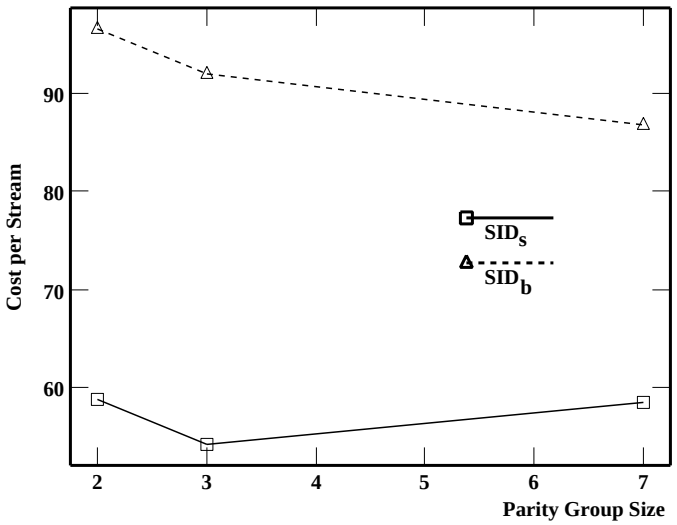

Figure 5: Cost per stream for SID-based schemes.

earlier, in the case of SID-based schemes, larger parity group sizes result in larger cluster sizes (and fewer clusters) and consequently, this results in a lower MTTF. Therefore, when choosing among schemes with approximately the same cost per stream, the scheme that *achieves that cost at a higher reliability* (e.g., this can be "approximated" here by the parity group size)[25] is more desirable.

In what follows, we evaluate the various schemes under different changes in technology as reflected in memory and disk costs. First, we consider a reduction in the disk cost from \$0.3 per MB to \$0.1 per MB while keeping the rest of the parameters fixed as listed in Table 2. Figure 9 displays the cost per stream curves using the reduced disk cost.

The cost per sream for the NC decreases at a low rate, and when the parity group size reaches 7, the cost per stream of NC reaches the value of \$18.5. Hence a 66% reduction in the disk cost results in a $\approx$ 62% reduction in the cost per stream. Note that, in Figure 9 for parity group size larger than 3, the NC scheme has a lower cost per stream than IB. The decrease in the difference between cost per stream of IB and NC, as compared to Figure 7, is due to the fact that

[25] Basically, here we are using the parity group size as an "approximate" measure of reliability. One thing missing from this evaluation is the consideration of the rebuild process, i.e., it is possible that a scheme using a larger parity group size is able to rebuild the failed disk quicker, if it is also using a significantly larger cluster size (as may be the case for the SID schemes) — in this case, it is not immediately obvious (without performing reliability analysis) which of the schemes would have a longer MTTF. However, the system's behavior in the rebuild mode is outside the scope of this paper.

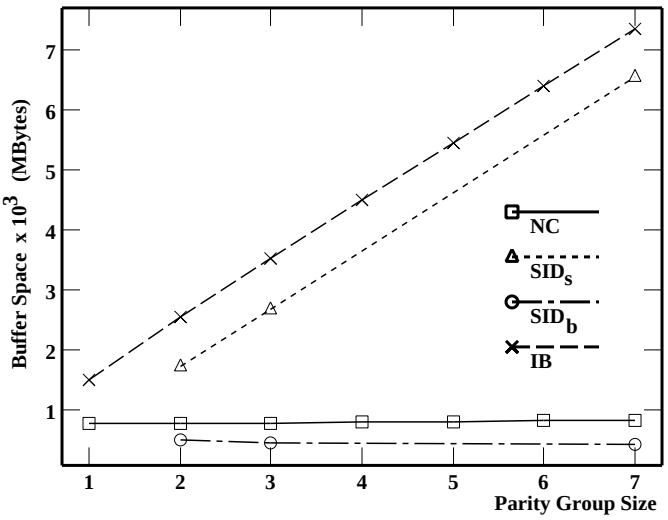

Figure 6: Memory requirements.

the main advantage of IB is its efficient bandwidth utilization — as the disk cost decreases, efficient bandwidth utilization becomes less important to the overall cost per stream. Furthermore, it can be shown, using Eqs. (32) and (33), that the reduction in disk cost has greater effect on the NC and the SID-based schemes, than on the IB scheme. Also note that (in Figure 10), with the reduced disk cost, NC performs better than SID_s for small parity group sizes (e.g., 2), in constrast to the results in Figure 8. Thus, it is not the case that one scheme is absolutely better than another, but rather that one must understand the tradeoffs as well as the system's requirements and constraints and then choose a fault tolerance scheme accordingly.

Next we consider a reduction in memory cost from \$30 per MB to \$10 per MB while keeping the rest of the parameters fixed as listed in Table 2. Figure 11 displays the cost per stream curves using the reduced memory cost.

This reduction in memory cost does not have as great of an effect on the cost per stream of the SID_b and the NC schemes (up to 5%), as it does on the cost per stream of the IB and the SID_s schemes (up to 25%). Note that, with the new reduced memory cost system (refer to Figure 11), the rate at which the cost per stream of the IB scheme changes, as a function of the parity group size, is small, as compared to the original system and the reduced disk cost system (see Figures 8 and 10, respectively). This is due to the fact that the inefficiencies of the IB approach (*as applied to the SR scheme*) are mainly due to large buffer space requirements (as opposed to inefficient use of disk bandwidth, as is the case for other schemes).

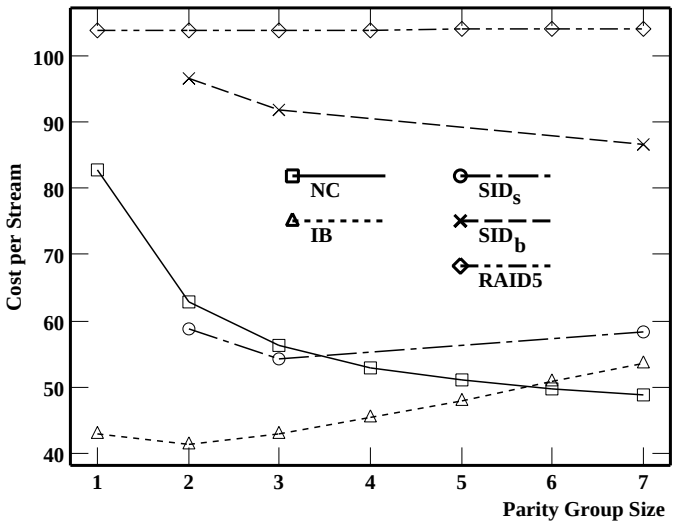

Figure 7: Cost per stream.

7 Conclusions

In summary, fault tolerance issues in multimedia storage systems are complicated by the necessity to recover from failure in real time. In this paper, we discussed several parity schemes for providing reliability and availability in a Video-on-Demand server, and we evaluated them using several performance metrics which included: storage overhead, bandwidth utilization, buffer space requirements, MTTF, MTTDS, and cost per stream. We studied the cost per stream characteristics of several sequential and non-sequential schemes as a function of the parity group size, which affects the reliability characteristics of the system. We have shown that improvements in reliability, which depend on the amount of redundant information stored and on how this information is placed on disks and retrieved, must be balanced against degradation in performance, disk storage overhead, loss of bandwidth, and increase in cost. In conclusion, we would like to impress upon the reader that the main point of this paper is the exposition of tradeoffs and issues associated with designing fault tolerant VOD servers rather than a description of specific schemes. Hence, it is not the case that one fault tolerance scheme is absolutely better than another fault tolerance scheme, but rather that one must understand the tradeoffs presented in this work as well as one's system constraints and requirements and then choose a fault tolerance scheme accordingly.

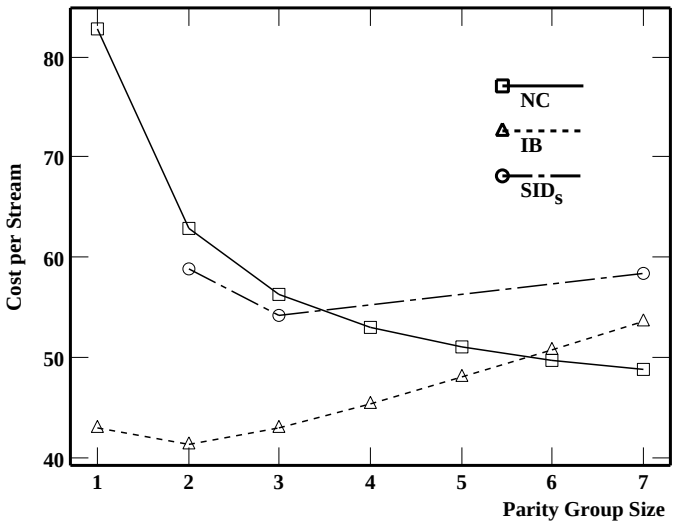

Figure 8: Cost per stream.

References

[1] DBC/1012 Database Computer System Manual Release 2.0. Technical Report Document No. C10-0001-02, Teradata Corporation, Nov 1985.

[2] S. Berson, L. Golubchik, and R. R. Muntz. Fault Tolerant Design of Multimedia Servers. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 364–375, San Jose, CA, May 1995.

[3] Steven Berson, Shahram Ghandeharizadeh, Richard R. Muntz, and Xiangyu Ju. Staggered Striping in Multimedia Information Systems. *SIGMOD*, 1994.

[4] D. Bitton and J. Gray. Disk Shadowing. *VLDB*, pages 331–338, 1988.

[5] M. Chen, D. Kandlur, and P. Yu. Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams. *ACM Multimedia '93*, pages 235–242, 1993.

[6] A. Cohen and W. A. Burkhard. Storage architectures for continuous digital video retrieval. In *Proceedings of SPIE, Photonics East*, 1995.

[7] G. Copeland and T. Keller. A Comparison of High-Availability Media Recovery Techniques. *ACM SIGMOD Conference*, pages 98–109, 1989.

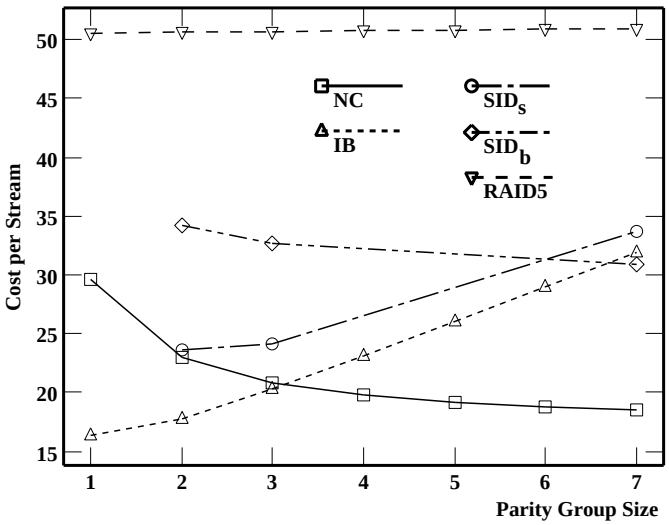

Figure 9: Cost per stream using the reduced disk cost.

[8] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley. Channel Allocation under Batching and VCR Control in Movie-On-Demand Servers. Technical Report RC19588, IBM Research Report, Yorktown Height, NY, 1994.

[9] D. Le Gall. MPEG: a Video Compression Standard for Multimedia Applications. *Communications of the ACM*, April 1991.

[10] S. Ghandeharizadeh and C. Shahabi. On Multimedia Repositories, Personal Computers, and Hierarchical Storage. In *Proceedings of the Second ACM International Conference on Multimedia*, October 1994.

[11] L. Golubchik and R. R. Muntz. Fault tolerance issues in multidisk video-on-demand storage servers. In *Proceedings of SPIE, Photonics East*, 1995.

[12] Jr. M Hall. *Combinatorial Theory.* John Wiley & Sons, 1986.

[13] H. Hsiao and D. J. DeWitt. Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines. *Proc. of Data Engineering*, pages 456–465, 1990.

[14] S. W. Lau and J. C. S. Lui. A Novel Video-On-Demand Storage Architecture for Supporting Constant Frame Rate with Variable Bit Rate Retrieval. In *Proc. of the 5th Intl. Conf. on Network and Operating System Support for Digital Audio and Video*, April 1995.

[15] Richard R. Muntz and John C.S. Lui. Performance Analysis of Disk Arrays Under Failure. *VLDB Conference*, pages 162–173, 1990.

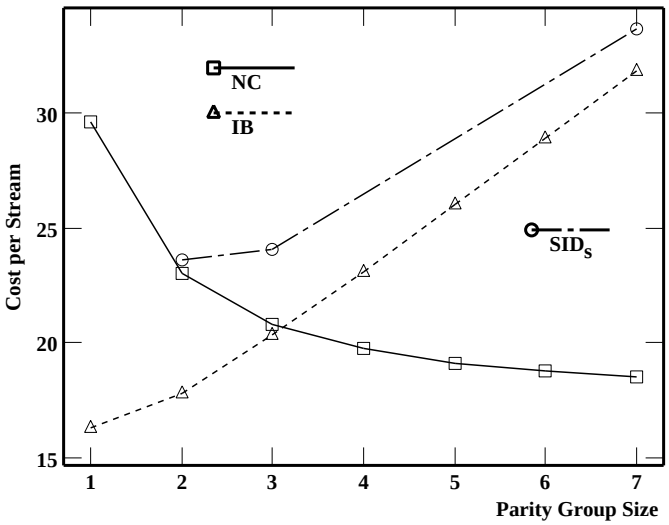

Figure 10: Cost per stream using the reduced disk cost.

[16] B. Ozden, A. Biliris, R. Rastogi, and A. Silberschatz. A Low-Cost Storage Server for Movie on Demand Databases. *Proc. of the 20th Intl. Conf. on Very Large Data Bases*, Sept. 1994.

[17] B. Ozden, R. Rastogi, P. Shenoy, and A. Silberschatz. Fault-tolerant Architectures for Continuous Media Servers. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 79–90, Montreal, Canada, June 1996.

[18] David A. Patterson, Garth Gibson, and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). *ACM SIGMOD Conference*, pages 109–116, 1988.

[19] P. V. Rangan and H. M. Vin. Efficient Storage Techniques for Digital Continuous Multimedia. *IEEE Transactions on Knowledge and Data Engineering*, August 1993.

[20] P. V. Rangan and H. M. Vin. Designing File Systems for Digital Video and Aduio. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, October 1991.

[21] Chris Ruemmler and John Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer Magazine*, pages 17–28, March 1994.

[22] F. A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID - A Disk Array Management System For Video Files. *ACM Multimedia Conference*, pages 393–399, 1993.

[23] H. M. Vin, P. J. Shenoy, and S. S. Rao. Efficient failure recovery in multidisk mulltimedia servers. In *Proceedings of SPIE, Photonics East*, 1995.

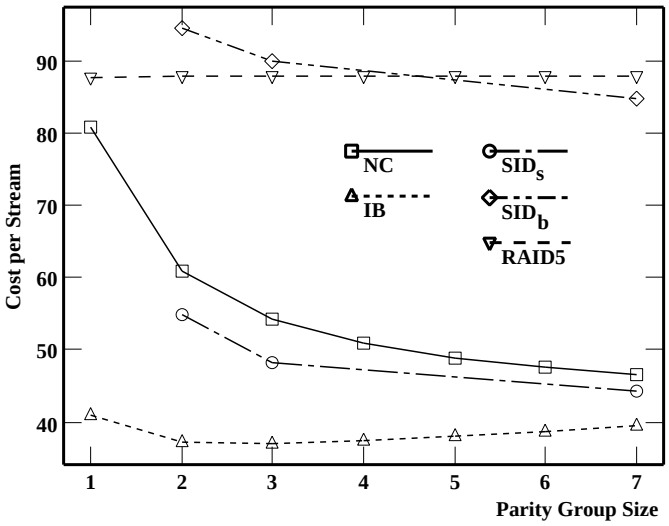

Figure 11: Cost per stream using the reduced memory cost.

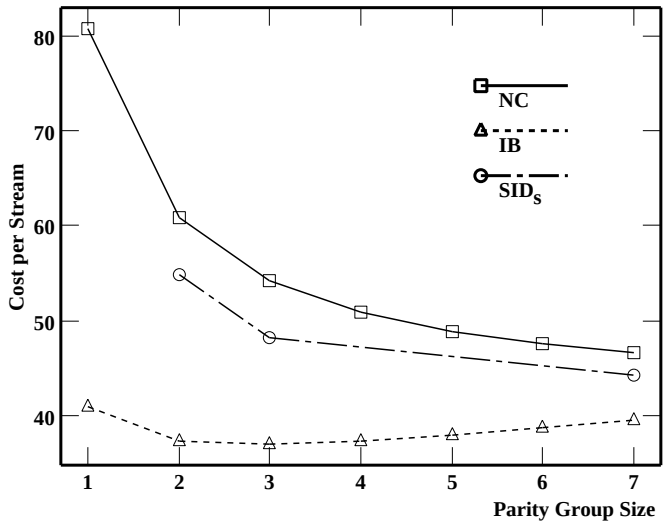

Figure 12: Cost per stream using the reduced memory cost.