Designing Efficient Fault Tolerant VOD Storage 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 Video-on-Demand (VOD) servers feasible. A challenging task in such systems is servicing multiple clients simultaneously while satisfying real-time requirements of continuous delivery of objects at specified bandwidths. To accomplish these tasks and realize economies of scale associated with servicing a large user population, a VOD server requires a large disk subsystem. Although a single disk is fairly reliable, a large disk farm can have an unacceptably high probability of disk failure. Furthermore, due to real-time constraints, the reliability and availability requirements of VOD systems are even more stringent than those of traditional information systems. In this paper we present the main tradeoffs associated with providing fault tolerance in multidisk VOD systems. We illustrate these tradeoffs through a discussion of several fault tolerance schemes as well as by evaluating their performance under normal and degraded modes of operation.

1 Introduction

Recent technological advances in digital signal processing, data compression techniques, and high speed computer networking 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, 20, 17, 5, 14, 18, 9] (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) disk system would provide enough storage for approximately 220 (100 minute) MPEG-2 movies (about 6 Mbits/sec) or approximately 880 MPEG-1 movies [8] (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 too high. The mean time to failure (MTTF) of a single disk is on the order of $300,000$ hours; this means that 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, 10, 15, 20, 21]. The issues 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 preclude objects from being transmitted directly from the tertiary store. 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 objects are typically striped over the disks, a single disk failure can cause multiple hiccups in the display of many objects. These hiccups will occur on each access to the failed disk 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 tertiary store; many tapes may need to be referenced and that is very time consuming. 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 not to have schemes for recovery 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, 20]) and non-redundancy-based techniques (e.g. [21]). 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 storing 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 image quality of non-redundancy-based schemes is somewhat subjective. This loss of quality in non-redundancy-based schemes is a persistent problem, unlike some momentary jitters experienced by 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, a fault tolerance scheme can either exploit the fact that video display can be "pre-planned" and utilize the information retrieved for reconstruction of missing data for normal display. Or it 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 separately from the reading of the same data blocks for the purpose of reconstructing (and displaying) the data missing due to failure. In the remainder of this paper, we describe and compare sequential-type and non-sequential-type 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 (mean time to failure and mean time to degradation of service)
- number of simultaneously supported streams
- cost of the storage subsystem (disks plus buffer space) per stream

The organization 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 Sections 5 and 6 we present analysis of these schemes and a discussion on their respective merits. Section 7 gives our conclusions.

2 Background

We begin this section with a description of a simple disk model and some basic concepts on 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 disk 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, e.g., as in [20] (or similarly into groups, as in [5]). 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) and thus seek times can be optimized, for instance using an elevator type algorithm. 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[1].

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 of an object. Let k_t be the number of disk storage units that are transmitted per cycle. If T_{cyc} is the length of a cycle, then $T_{cyc} = (k_t * B)/b_0$. Then if k_r disk

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

storage units are read in a cycle for a stream, where k_r is an integer multiple of k_t, then the data read in one "read cycle" is delivered in the next k_r/k_t cycles. This is illustrated in Figure 1, where Ai refers to block i of object A.

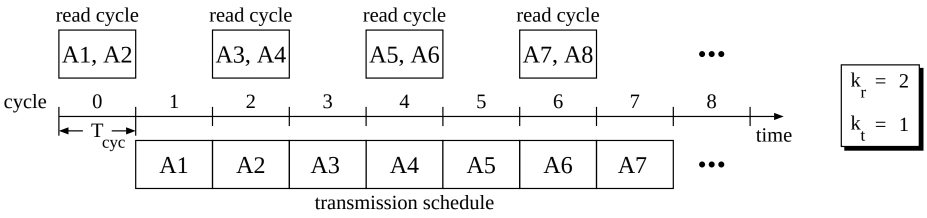

Figure 1: Multiple transmission cycles per read cycle.

2.2 Disk Model

We use the following simple disk model (as in [2]) for the purpose of discussing basic performance tradeoffs. We will use the following notation in the remainder of this paper:

τ_{seek}	maximum seek time between the extreme inner and outer cylinders of a disk
τ_{rot}	average rotational latency time including the slowdown and speedup fraction of the seek time
τ_{trk}	maximum time attributable to reading a track *as well as* the slowdown and the speedup of the READ/WRITE head of the seek time [19]; as in [2], we take the point of view that this cost is associated with the reading of the track as opposed to part of the seek cost
τ_{blk}	maximum time attributable to reading a block (or sector)
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[2]
D'	number of disks in the system from which "real" data is read; specifically, D' does not include disks which are devoted to parity as are some disks in a RAID3 type architecture
S	the capacity of a disk (in megabytes)
$T_t(r)$	maximum time to read r tracks[3] $T_t(r) = \tau_{seek} + r * \tau_{trk}$

	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}	effective number of active streams on a disk (maximum number of stream that a disk can support)
N_d	maximum number of active streams on a disk, $N_d \leq N_{eff}$
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 k_r/k_t = number of cycles between "read cycles" for the same stream
b_0	object bandwidth in MB/s (megabytes per second)
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

We will assume in the remainder of the paper, unless otherwise stated, that the unit of disk I/O is a track. This is largely motivated by the reduction in rotational latency achieved — we assume for instance, that a full track read is started at the next sector boundary and therefore, there is very little rotational latency. 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 equation, we can solve for a bound on N or equivalently on N/D', the number of streams per disk:

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

where $\frac{k_t * B}{b_0} - \tau_{seek}$ is the minimum useful read time per disk per cycle and $k_t * \tau_{trk}$ is the time to read k_t tracks.

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 schemes[16] and mirroring schemes[4] have been used for this purpose[4]. 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 is called a *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 call a *degradation of service*, that can occur when there is insufficient available resources, due to failures, to continue delivering all active requests. 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, an active request might have to be terminated and rescheduled for transmission at a later time.

There are three modes of operation for a disk subsystem [13], 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

RAID-based sequential-type scheme (Streaming RAID)

An example of a sequential-type fault tolerance scheme is the *Streaming RAID* (SR) approach [20][5], where disks are grouped into fixed sized 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 on the parity disk form

[4]Variations on these redundancy schemes, e.g., [13, 6], will be discussed later in the paper.

[5]In what follows we describe a slight generalization of the approach presented in [20].

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, $k_r = C - 1$ in the model of Section 2.2 and, if they are all delivered in the next cycle, then $k_t = C - 1$.

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 partity block from the same parity group, without hiccups[6]. This scheme can withstand up to one disk failure per cluster before a catastrophic failure occurs (or before there is a degradation of service). Thus, the main advantage of the Streaming RAID scheme is its high reliability. However, that reliability is gained at the cost of both disk storage and disk bandwidth.

Note that, the SR scheme exhibits the same performance characteristics under failure as it does under normal operation. This is 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, which grows linearly with the cluster size[7]. Note that, the incentive for a large cluster size is the efficiency with respect to disk bandwidth and disk storage; however, this must be balanced against the cost of additional main memory.

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. Data retrieval can proceed one block at a

[6]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 transmission of the data. But this is a minor complication which is not discussed any further in this paper.

[7]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.

time (i.e., $k_r = 1$ and $k_t = 1$) and thus obviate the need for large amounts of buffer space[8]. 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 under disk failure. In other words, we would obtain a large reduction in buffer space requirements at the cost of significant bandwidth loss. In Sections 4 we will discuss schemes which address the problem of such a 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 suggested in Section 4.

3 Sequential Schemes

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 Staggered group scheme and the non-clustered scheme, which address this problem.

[8]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.

3.1 Improving Buffer Space Requirements

Staggered-group scheme

The Staggered-group scheme (SG), a simple extension to SR, suggested in [20][9] and [2], addresses the main disadvantage of SR, namely the large buffer space requirements, by eliminating the idea that 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. 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 $k_t = 1$ since the basic 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 [15] 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.

Hence, 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 in the face of a disk failure is exactly the same, and (5) there is a slight cost in disk bandwidth overhead. The savings of approximately half of main memory derives from 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) then 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[10]; this loss in seek optimization is quantified in Section 5.

[9] In [20] it was referred to as "memory sharing with subgrouping and subcycling".

[10] 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 in each cycle for each active stream. 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 in one cycle is removed (as we will see later, this could results in 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 suggested 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 compared to the SR scheme is approximately a factor of C, where C is the number of disks in a cluster. However, the NC scheme does not exhibit as good reliability characteristics as the SR or the SG schemes. When a disk failure occurs, the affected disk cluster will have to switch to degraded mode of operation, where each stream in 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 will be a short transition phase as the cluster switches to degraded mode, and during this transition phase, some data may not be delivered and consequently a small number of 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 lesser 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 requirements will not be any better than those of the SG scheme. Rather than each cluster have all the memory it needs to run in degraded mode (which is a rare event), an architecture is needed in which the additional buffers space, needed under failure, can be 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 a degradation of service 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 schemes that have dealt with the poor utilization of the buffer space resource. However, one problem with these schemes which remains is that they still 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. 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

Instead of having dedicated parity disks, which are only used for reading in case of failure, we can intermix data and parity information on the same disk, which suggests the possibility of using the bandwidth of all disks during normal operation. One way of accomplishing this 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. The system design then only has to show how to accommodate a failure of a disk in cluster i (in real time), since this will result in an additional load on cluster $i+1$ (refer to Figure 2). When a disk failure occurs in cycle

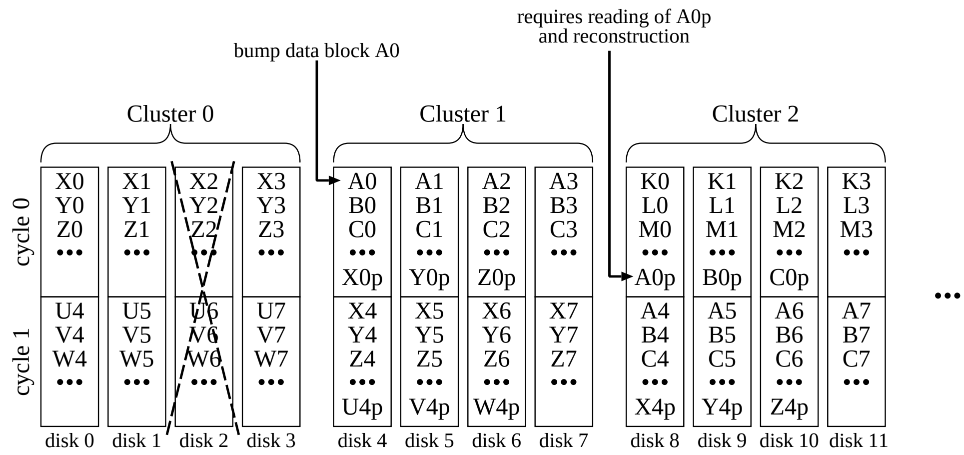

Figure 2: Improved Bandwidth Approach Applied to SR

t in cluster i, then cluster i and 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[11]. 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. 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 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 or causes streams to be dropped is greater in this system than with the SR scheme. In general, a SR or a NC system with K clusters, can withstand up to K failures, as long as there

[11] This is similar to how "chained declustering" [12] 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.

is no more than one failure per cluster, before data is lost. In the IB scheme, a failure in each of the two adjacent clusters results in data loss. Thus an improved bandwidth system with K clusters can possibly withstand up to $\frac{K}{2}$ failures before 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, i.e., certain disks belong to two parity groups; for instance, disk 4 in Figure 2 belongs to two different 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.

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 should result in a 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[12]. (This is based on an optimistic assumption that the parity blocks that have to be read are evenly spread over a cluster which should be an accurate enough estimate for system sizing.) On the other hand, the mean time until catastrophic failure is approximately 540 years. This basically indicates that we can significantly improve system reliability by reserving only a small amount of 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).

[12]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.

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 avoiding 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 [15], 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" [15] 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 half, since it can handle single disk failures in at most half of the clusters in the system. The "prefetching without parity disk" [15] gives up much more in system reliability since it will lose data from the disk subsystem after *any two disk failures*, i.e., after any two disk failures, data will have to be reconstructed from tertiary storage.

4 Non-sequential Schemes

The main advantage of non-sequential schemes is the better buffer space utilization, 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 sacrificing some of the reliability. One example of a (more traditional) approach is the Clustered RAID scheme, proposed in [13]. 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) [11] technique [15] and the other using a

Segmented Information Dispersal (SID) technique [6].

The basic idea behind Clustered RAID is to relax the assumption, made in traditional RAID architectures [16], that the parity group size, G, should be equal to the cluster size, C, where "parity group size" refers to the number of data blocks plus the parity block, and the "cluster" size refers to the number of disks over which the parity group blocks are distributed. Thus, 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 relatively small to C, we can reduce the amount of bandwidth wasted for reliability purposes. Several methodologies have been proposed for constructing the parity groups and distributing the data uniformly across the disks in case of a disk failure — BIBD and SID are two which are described below.

4.1 Improving Buffering Requirements through SID

The SID scheme described in [6] uses the Clustered RAID framework to address the bandwidth inefficiency problem of a traditional RAID system. We briefly describe the SID approach below. As in the case of a 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 the reconstruction of a block of r bytes will require reading of at most $\frac{r}{q}$ *contiguous* bytes from any disk in the cluster, where q is the dispersal factor. Thus q is one parameter which can control the amount of bandwidth wasted for reliability purposes. (This is analogous to the $\frac{G-1}{C-1}$ parameter of the Clustered RAID.) 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[13]. 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 the proper values and constructing a proper data layout that will correspond to these values.

In this section we describe a special case of SID, termed Constrained SID, where redundant

[13]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.

information is constructed using q blocks of data, each of size $\frac{r}{q}$. Consequently, $c = \frac{1}{q+1}$ and $C \geq q^2 + 1$ (see [6] for details). We illustrate this approach through the example of Figure 3. In this example, $C = 5$, $q = 2$, and $c = \frac{1}{3}$. Each disk i is divided into segments $S_{i,j}$, where

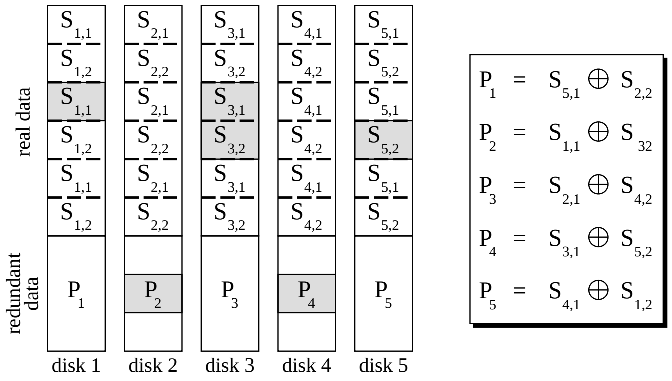

Figure 3: SID Organization

$1 \leq j \leq q$, 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. If a failure occurs, for instance of disk 3, then, for example, the shaded region, of size r bytes, of that disk can be reconstructed by reading the (contiguous) shaded regions, each of size $\frac{r}{q}$ bytes, of the surviving disks of the cluster. To deliver video data using the SID scheme, we can 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. Of course, in the case of 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, C, it is possible to construct SID solutions that are equivalent (i.e., degenerate to) other redundancy schemes, such as R5 [16], interleaved declustering [1, 7] (i.e., a version of full replication), and Clustered RAID [13] (see [6] for details).

In this paper we will be consider several variations on the SID scheme. They mainly differ in choice of the size of the video slice (i.e retrieval unit). This choice affects the maximum number of streams that can be active simultaneously, the disk bandwidth utilization, the buffer space requirements, and the cost per stream. The four variations of the SID scheme considered in this paper are SID_1, SID_p, SID_s, and SID_b. In the SID_1, SID_p and 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 disk of the cluster has to read no more than one track. The SID_p scheme retrieves an entire parity group in each cycle, and discards all but one track in the next cycle. (Recall that the SID is a non-sequential scheme). 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 disk reads $\frac{track\_size}{q*sector\_size}$ sectors for reconstruction of each missing track.

Under all SID schemes, bandwidth must be reserved to insure that each disk will have enough bandwidth to retrieve the additional data or parity information in order to reconstruct the missing data of the failed disk. Specifically, for the SID_1, SID_b, and SID_s variations 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:

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

where $N_d^{SID_a}$ is the maximum number of streams per disk that can be active under the SID_a scheme, and $a = s, p, 1$. 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. In this case, we must consider an additional rotational latency time, t_{avg}^{rot}, which is attributed to reading some number of sectors (less than a track) as well as to the slowdown and the speedup fraction of the seek time. The total time that it takes for a disk to read N_d tracks and N_d data units (used for reconstruction) is:

$$T_{cyc} = t_{max}^{seek} + N_d * t_{trk} + N_d * \left[t_{avg}^{rot} + \frac{t_{blk} * B}{sector\_size * q} \right]$$

where the size of one unit is $\frac{B}{sector\_size*q}$. Since the system has to guarantee that it can retrieve the additional data needed for reconstruction in case of failure, it must also satisfy the following constraint:

$$N_d \leq \frac{\frac{B}{b_0} - t_{max}^{seek}}{t_{trk} + t_{avg}^{rot} + \frac{t_{blk}*B}{sector\_size*q}}$$

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, $N_d = \frac{N_{eff}}{2}$

4.2 Improving Buffering Requirements through BIBD

The main difference between BIBD and SID is in which constraints are placed on the construction of redundant information and on the layout of real and redundant data within the 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:

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

A "declustered parity based" scheme, utilizing the BIBD approach in a fault tolerant design of a VOD server is described in [15]. 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 is the maximum number of active streams on a disk, then (using Eq. (2)) $N_d * r * \frac{k-1}{u-1}$ additional blocks have to be read by each surviving disk (or $N_d * \frac{k-1}{u-1} * b$ bytes). Note that, $\frac{k-1}{u-1} \leq 1$, and in the case of the R5 scheme the total number of bytes retrieved by each of the surviving disks is $N_d * b$. Therefore, the amount of bandwidth that must be reserved for reconstruction of missing data, in case failure, under the BIBD scheme is 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, 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:

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

The 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. 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.

5 Analysis

In this section we analyze the SR, SG, IB, NC, SID_p, SID_s, SID_1, SID_b, and R5 schemes based on: 1) reliability considerations, including their susceptibility to catastrophic failure and degradation of service, 2) number of simultaneous display streams they can support, and 3) penalties or costs associated with storing redundant information. These penalties 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 actual data

bandwidth the amount of bandwidth that must be dedicated to redundancy, e.g., for retrieving parity, which can not be used for retrieving actual data

buffer space the amount of memory needed to support of 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 the 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, SID_1, SID_b or SID_p. Then:

$$S^{SR} = S^{SG} = S^{NC} = S^{IB} = S^{R5} = \frac{D * S}{C} \tag{3}$$

$$S^{SID_b} = S^{SID_p} = S^{SID_s} = S^{SID_1} = \frac{D * S}{q} \tag{4}$$

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. (Recall that D' is the number of disks from which data is read which, for an IB scheme, is equal to D, but for all other schemes is equal to $\frac{C-1}{C} * D$.)

Disk bandwidth overhead

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

$$
\begin{aligned}
BW^{SR} &= BW^{SG} = BW^{NC} = \frac{D * d}{C} && (5)\\
BW^{IB} &= K_{IB} * d && (6)\\
BW^{R5} &= \frac{D * d}{2} && (7)\\
BW^{SID_b} &= BW^{SID_p} = BW^{SID_s} = BW^{SID_1} = \frac{D * d}{q} && (8)
\end{aligned}
$$

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 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:

$$
\begin{aligned}
MTTF^{SR}_{D|C} &= MTTF^{SG}_{D|C} = MTTF^{NC}_{D|C} \approx \frac{\text{MTTF(disk)}^2}{D * (C-1) * \text{MTTR(disk)}} && (9)\\
MTTF^{IB}_{D|C} &\approx \frac{\text{MTTF(disk)}^2}{D * (2C-1) * \text{MTTR(disk)}} && (10)\\
MTTF^{SID_b}_{D|C} &= MTTF^{SID_s}_{D|C} = MTTF^{SID_1}_{D|C} = MTTF^{SID_p}_{D|C} && (11)\\
&\approx \frac{\text{MTTF(disk)}^2}{\frac{W}{S} * q * (1+q) * \text{MTTR(disk)}} && (12)
\end{aligned}
$$

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 and the IB schemes. The degradation of service in the SID variations[14] 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

[14]This is a slight generalization of the SID approach described in [6], where sharing of a buffer pool between clusters is not considered.

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_p	$q*(q+1)$	q
SID_1	1	1
SID_b	1	1

Table 1: Characterization of schemes.

in the IB scheme occurs when there is a lack of available bandwidth capacity (in the right places) to perform the shift. Hence:

$$\begin{aligned} MTTDS^{NC} &= MTTDS^{IB} = MTTDS^{SID} \\ &\approx \frac{\mathrm{MTTF(disk)}^K}{D*(D-1)*\cdots*(D-K+1)*\mathrm{MTTR(disk)}^{(K-1)}} \end{aligned} \tag{13}$$

where 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[15]. (See Sections 3.1 and 3.2 for details.) The SID-based schemes share the same mean time to having 5 clusters simultaneously operating in degraded mode with the NC (for the same redundancy rate). 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 (see constraints given in Section 4.1), which forces SID into having larger (and fewer) clusters, and thus a lower MTTF.

Number of Simultaneously Supported Streams

The number of simultaneously supported streams, N, which 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):

[15]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.

$$N^{SR} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)}\right] * D\frac{C-1}{C} \tag{14}$$

$$N^{SG} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] * D\frac{C-1}{C} \tag{15}$$

$$N^{NC} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] * D\frac{C-1}{C} \tag{16}$$

$$N^{IB} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)}\right] * (D - K_{IB}) \tag{17}$$

$$N^{SID_s} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * q}\right] * D * \left[\frac{q}{q+1}\right] \tag{18}$$

$$N^{R5} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right]\left[\frac{D}{2}\right] \tag{19}$$

$$N^{SID_p} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * q}\right] * D * \left[\frac{q}{(q+1)^2}\right] \tag{20}$$

$$N^{SID_1} = \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] * D * \left[\frac{q}{q+1}\right] \tag{21}$$

$$N^{SID_b} = \frac{\frac{B}{b_0} - \tau_{max}^{seek}}{\tau_{trk} + \tau_{avg}^{rot} + \frac{\tau_{blk} * B}{sector\_size * q}} * D \tag{22}$$

Additional Buffer Space

At this point we can compute the buffer space requirements, BF^p (including parity data), 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:

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

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;

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

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

$$BF^{NC} = 2 * B * \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] * D\frac{C-1}{C} + \left[\frac{BF^{SG}}{(D\frac{C-1}{C})/C} * K_{NC}\right] \tag{25}$$

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 parity. Therefore the number of buffers necessary per each steam is $2(C-1)$, and hence:

$$BF^{IB} = 2(C-1) * B * \left[\frac{B}{b_0 * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)}\right] * (D - K_{IB}) \tag{26}$$

The buffer space requirements for the R5 scheme are:

$$BF^{R5} \quad = \quad B * N_d^{R5} * [2 * D + (C-1) * K] \tag{27}$$

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

$$BF^{SID_s} \quad = \quad B * q^2 * \left(\frac{2 * D + K * q}{q+1}\right) * \left(\frac{B}{b_o * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * q}\right) \tag{28}$$

$$BF^{SID_p} \quad = \quad B * \left(\frac{(2 * D + K * q) * q^2}{(q+1)^2}\right) * \left(\frac{B}{b_o * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * q}\right) \tag{29}$$

$$BF^{SID_b} \quad = \quad B * \frac{\frac{B}{b_0} - \tau_{max}^{seek}}{\tau_{trk} + \tau_{avg}^{rot} + \frac{t_{blk} * B}{sector\_size * q}} * (2 * D + K * q) \tag{30}$$

Cost Per Stream

Assuming that cost is a constraint, we would like to study how the different schemes affect the cost per stream. 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 results, consider the problem of sizing and selecting data layout and scheduling for a system with a fixed working set size, W, the amount of real data that we would like to store on the disk-subsystem, and required number of streams to be supported. In the following equations, the cost per stream is derived for the SR, SG, NC, IB, R5, SID_b, SID_1, SID_p, and SID_s schemes.

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

$$\frac{Cost^{SG}}{D * N_d} = C_b * \frac{C * (C+1)}{2 * (C-1)} * B + C_d * \frac{C}{C-1} * \frac{S}{\frac{B}{bo * \tau_{trk} - \frac{\tau_{seek}}{\tau_{trk}}}} \tag{32}$$

$$\frac{Cost^{NC}}{D * N_d} = C_b * \left[2B + \frac{B * K * C * (C+1)}{2 * (C-1) * \frac{D * (C-1)}{C^2}}\right] + C_d * \frac{S}{\left[\frac{B}{bo * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right] * \frac{C-1}{C}} \tag{33}$$

$$\frac{Cost^{IB}}{D * N_d} = C_b * 2 * (C-1) * B + C_d * \frac{\frac{D * S}{D - K}}{\left[\frac{B}{bo * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk} * (C-1)}\right]} \tag{34}$$

$$\frac{Cost^{R5}}{D * N_d} = C_b * B * q * \left(2 + \frac{Kq}{D}\right) + C_d * S * \frac{2}{\left[\frac{B}{bo * \tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right]} \tag{35}$$

$$\frac{Cost^{SID_1}}{D*N_d} = C_b * B * (2 + \frac{Kq}{D}) + C_d * S * \frac{1+\frac{1}{q}}{\left[\frac{B}{bo*\tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}}\right]} \tag{36}$$

$$\frac{Cost^{SID_s}}{D*N_d} = C_b * B * q * (2 + \frac{Kq}{D}) + C_d * S * \frac{1+\frac{1}{q}}{\left[\frac{B}{bo*\tau_{trk}} - \frac{\tau_{seek}}{\tau_{trk}*q}\right]} \tag{37}$$

$$\frac{Cost^{SID_b}}{D*N_d} = C_b * B * (2 + \frac{Kq}{D}) + C_d * S * \frac{1+\frac{1}{q}}{\left[\frac{\frac{B}{b_o} - \tau_{seek}}{\tau_{trk} + T_{rot} + (\frac{\tau_{blk}*B}{sector\_size*q})}\right]} \tag{38}$$

6 Comparison of Schemes

In this section we compare the different schemes using two metrics: the buffering requirements and the cost per stream. We assume sufficient resources to mask $K = 5$ failures and a video database with a fixed amount of "real" data, W[16]. As stated earlier in Section 3.1, the probability of 5 failures is extremely low. The system parameters used in the experiments 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. Whereas parity group size and cluster size for the R5, NC, and IB schemes are identical, for the SID variations discused here, the parity group size refers to the dispersal factor q and the cluster size, $C_{SID} = q^2 + 1$. Clearly, schemes with the same parity group size have the same amount of redundant information.

Figure 4 illustrates the buffering requirements for each SID-based scheme as a function of the parity group size. Clearly, the SID_s scheme needs more buffer space than the others, since in that scheme q tracks are retrieved per stream per cycle, as opposed to only one a single track per stream per cycle as in the case of the SID_1 and SID_b schemes. Furthermore, the buffer space needed for the SID_s increases as the parity group increases. On the other hand, as can be seen from Figure 4, the memory requirements of the remaining schemes are relatively insensitive to the parity group size.

Figure 5 displays the cost per stream of the SID-based schemes as a function of the parity group size. As was expected, the cost per stream is a function of the the total number of streams that can be supported simultaneously. For example, SID_p has relatively small buffer space requirements

[16] 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}	20 ms
τ_{rot}	8 ms
$\tau_{speedup\_slowdown}$	5 ms
τ_{trk}	20ms
τ_{blk}	0.15 ms
K	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_m	$30 per MB
MTTF(disk)	300,000 hours
MTTR(disk)	1 hour

Table 2: System parameters.

(see Figure 4). However, its performance, as measured in terms of cost per stream, is still poor. This is due to its inability to support a large number of streams simultaneously. Similarly, SID_b, looses some bandwidth, and consequently supports less streams simultaneously. This is due to the additional rotational delay incurred as a result of reading data blocks that are smaller than a single track. Thus, SID_b also has a higher cost per stream than the SID_1 and the SID_s schemes. In addition, in Figure 5, SID_s and SID_1 have approximately the same cost per stream, although the buffering requirements of SID_s are significantly greater than those of SID_1, as illustrated in Figure 4.

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 (or cluster size). Note that, given the same amount of redundant information, both NC and SID_1 can support the same number of streams simultaneously. As the parity group size increases, the buffering requirements of NC increase slowly. 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. From Eqs. (26)

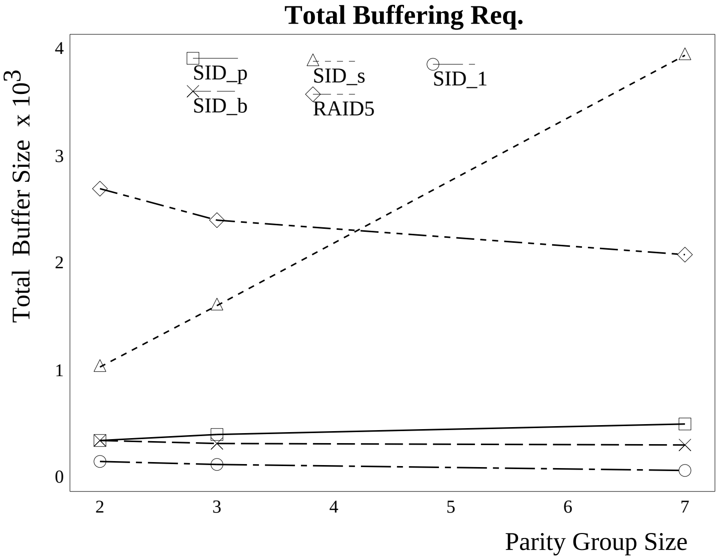

Figure 4: Memory requirements of SID based-schemes.

and (29), we can derive the sizes of a database, $W \leq \frac{(q^2+2q+1)*q}{2*(q+1)} * \frac{S*K}{2}$, for which the buffering requirements of IB are smaller than those of SID_s. In the graphs presented in this section, the database is of size 1000GB. Hence, in these figures IB has larger buffer space requirements than SID_s, since the database size is above the threshold of $\frac{(q^2+2q+1)*q}{2*(q+1)} * \frac{SK}{2}$.

Figure 7 illustrates the cost per stream curves as a function of the parity group size for sequential schemes (e.g., NC and IB) as well as non-sequential schemes (e.g., SID_s, SID_1, and SID_b). For a small parity group size, the IB scheme has the best performance as measured in terms of cost per stream (e.g., \$65 at the parity group size of 2). The cost per stream decreases as the parity group size increases for the SID_1 scheme. And, at a parity group size of 7, IB and SID_1 have approximately the same cost per stream. 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, SID_1, and SID_s peform similarly at a parity group size of 5. However, as was emphasized 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 approximate the same cost per stream, the scheme that *achieves that cost at a smaller parity group size* is more desirable.

In what follows, we evaluate the different schemes under different forecasts for changes in memory and disk costs. First, we consider a reduction in the disk cost from \$0.3 per MB to \$0.1 per

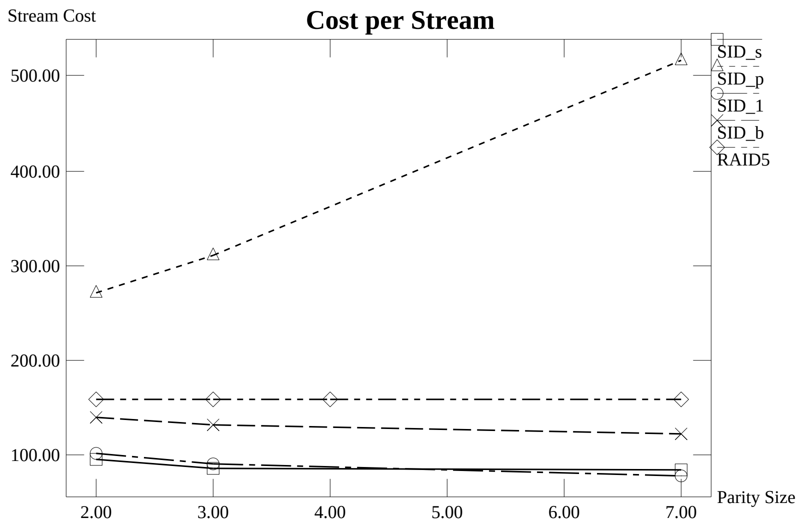

Figure 5: Cost per stream for SID-based schemes.

MB while keeping the rest of the parameters fixed as listed in Table 2. Figure 8 displays the cost per stream curves using the reduced disk cost. When the parity group size reaches 7, the cost per stream of SID_1 and SID_b reach their minimum value of \$26. Hence a 66% reduction in the disk cost results in a 74% reduction in the cost per stream. Furthermore, NC decreases at a similar rate. Note that, in Figure 8 for a parity group size of 4, the NC and SID_1 schemes have a lower cost per stream than IB. In addition, IB maintains its optimal cost per stream at a parity group size of 2. The decrease in the difference between cost per stream of IB and NC (as well as SID_1), as compared to Figure 7, is due to the fact that the main advantage of IB is its efficient bandwidth utilization — as the disk cost decreases, effective bandwidth utilization becomes less important to the overall cost per stream. Furthermore, it can be shown, using Eqs.(34), (35), and (37), that the reduction in disk cost has a greater affects on the NC and the SID-based schemes, than on the IB scheme.

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 9 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_1, the SID_b, and the NC schemes (less than 5%), as it does on the cost per stream of the IB and the SID_s schemes. Note that, with a reduced memory cost, the rate at which the cost per stream of the IB scheme changes is very small. For example a 66% reduction in the memory cost results in only a 7.6% reduction in the cost per stream for the IB

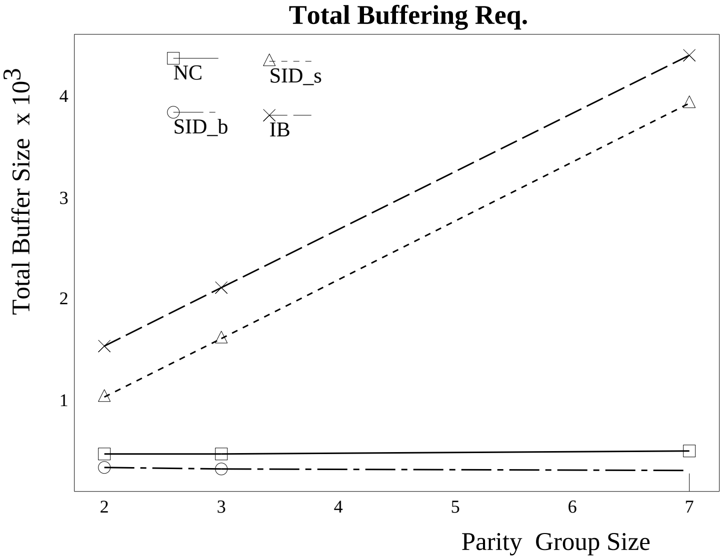

Figure 6: Memory requirements.

scheme.

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 include: 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 also determines 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, must be balanced against degradation in performance, due 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 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 then choose

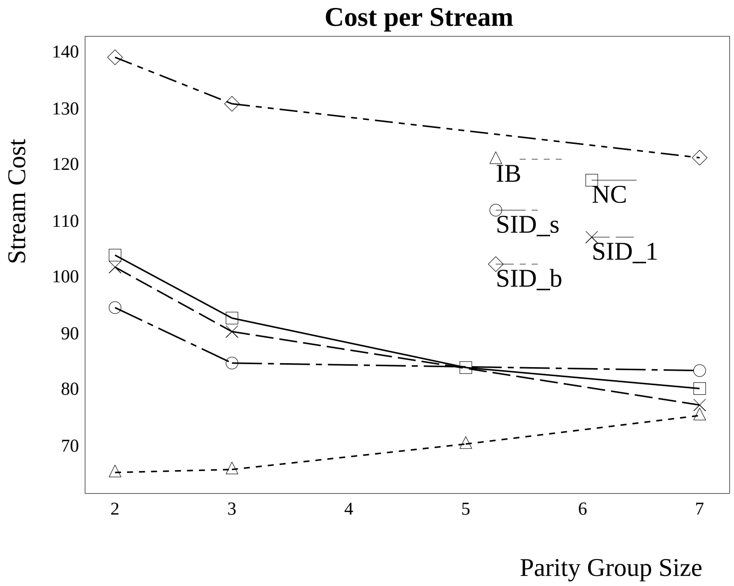

Figure 7: Cost per stream.

a fault tolerance scheme accordingly.

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.

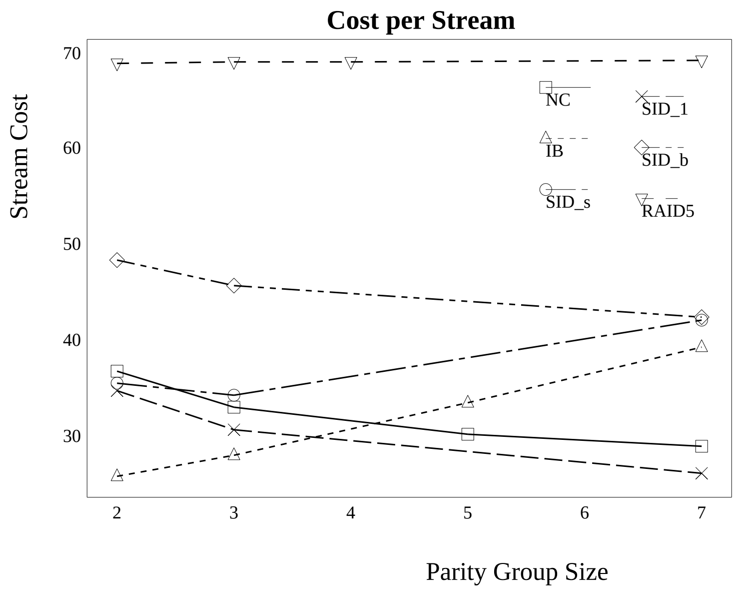

Figure 8: Cost per stream using the reduced disk cost.

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

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

[9] 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.

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

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

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

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

[14] 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.

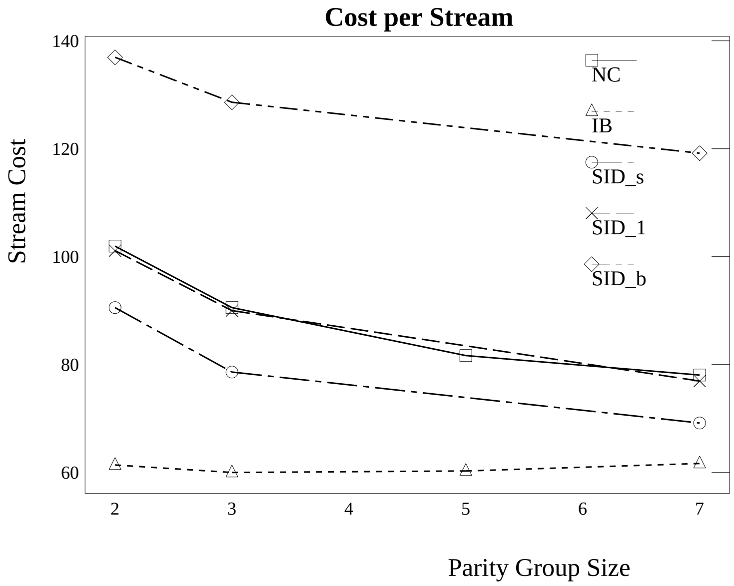

Figure 9: Cost per stream using the reduced memory cost.

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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.

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