Segmented Information Dispersal (SID) Data Layouts for Digital Video Servers [1]

Ariel Cohen Walter A. Burkhard

Gemini Storage Systems Laboratory
Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093-0114

Abstract

We present a novel data organization for disk arrays—*Segmented Information Dispersal* (SID). SID provides protection against disk failures while ensuring that the reconstruction of the missing data requires only relatively small contiguous accesses to the available disks. SID has a number of properties that make it an attractive solution for fault-tolerant video servers. Under fault-free conditions, SID performs as well as RAID 5 and organizations based on balanced incomplete block designs (BIBD). Under failure, SID performs much better than RAID 5 since it significantly reduces the size of the disk accesses performed by the reconstruction process. SID also performs much better than BIBD by ensuring the contiguity of the reconstruction accesses. Contiguity is a very significant factor for video retrieval workloads, as we demonstrate in this paper. We present SID data organizations with a concise representation which enables the reconstruction process to locate efficiently the needed video and check data.

Keywords

algorithms, data structures, declustering, disk arrays, separated difference sets, video servers.

1 Introduction

Digital video server systems must provide timely delivery of video stream data to an ensemble of users even in degraded modes in which one or more system disks are not operational. One design problem is the following: for a given set of disks, which video data layout can provide this service while avoiding buffer starvation and overflow as well as providing fault-tolerance? We present a novel data layout scheme to solve this problem. These layouts can greatly reduce the added workload served by the operational disks when the disk array experiences failures.

The most widely considered approaches for this problem are based on RAID 5 or RAID 3 data layouts [21]. The staggered striping data organization of Berson, Ghandeharizadeh, Muntz and Ju provides effective disk bandwidth utilization for both small and large workloads [4]. The video servers studied by Berson, Golubchik and Muntz [3] employ a RAID 5 data layout and utilize a modified and expanded read schedule in degraded mode requiring additional buffering. There can be transition difficulties in which data is not delivered on time. A method proposed by Vin, Shenoy and Rao [28] does not rely on RAID 5 parity redundancy but on redundancy properties of the video data itself. Streaming RAID of Tobagi, Pang, Baird, and Gang is one of the first commercial video servers [26]. There has been considerable work on disk array declustering as well; much of this centers on studies and application of balanced incomplete block designs BIBD [11] to transaction processing with workload characteristics vastly different from those of video streams. This includes the work of Holland and Gibson [14], Ng and Mattson [18], Reddy, Chandy and Banerjee [22], Alvarez, Burkhard and Cristian [1], Alvarez, Burkhard, Stockmeyer, and Cristian [2] as well as Schwarz, Steinberg, and Burkhard [25]. Disk array declustering for video servers based on BIBD has been considered by Cohen [9] and independently by Ozden, Rastogi, Shenoy and Silberschatz [20]. The very recent overlay striping data layout of Triantafillou and Faloutsos [27] maintains high throughput across a wide range of workloads by "dynamically" selecting an appropriate stripe size.

[1] Work supported in part by grants from Symbios Logic, Wichita, Kansas, and the University of California MICRO program.

A considerable advantage of our data layout scheme is that it achieves a large number of concurrent streams while requiring only small buffering per stream. At the same time, the layouts are simple and obtain load balancing across the system in spite of vastly differing preferences for various "movies." Moreover, we obtain reasonable performance in degraded operation with one or more disk failures without adverse effects on the fault-free performance. The cost of this degraded performance improvement is additional storage space.

The paper is an elaboration of the abstract presented within the ACM Multimedia Conference, November 1996 [7]. The paper is organized as follows: Section 2 contains an example of various candidate data layout schemes for video servers, Section 3 presents our segmented information dispersal data layout, Section 4 contains our analysis of the performance results where we determine the buffer space required per video stream for RAID 3, RAID 5, and SID, the effect of varying the dispersal factor, and the impact of discontiguous data layouts. We present several video server designs in Section 5. Our conclusions are given within the final section.

2 Design Alternatives Overview

To familiarize the reader with our approach and demonstrate its simplicity, we describe the architecture of a simple dedicated video server consisting of five disks. The disk array contains only video and check data; system files and metadata are stored elsewhere. Movies are divided into equal sized portions referred to as *slices*. We use this terminology to emphasize the difference between our approach, in which the size of each redundant data fragment is typically smaller than the size of a slice, and the RAID 5 approach in which the sizes of slices and redundant data fragments are exactly the same. Each slice is stored contiguously (except for the RAID 3 organization.) Our approach is directly applicable to constant bit rate and variable bit rate compression schemes such as MPEG-1 [16] and MPEG-2 [13, 15]. For MPEG encoded movies, a slice size is typically a few hundred kilobytes.

The videos are stored using disk striping, a technique in which consecutive slices are allocated to the disks in a round robin fashion except for RAID 3. Two immediate benefits are that this approach allows presentation of multiple concurrent streams and that the approach provides load balancing across the disks without knowledge of access frequencies for the stored videos. The RAID 3 data layout provides fine-grain striping while the other data layouts provide coarse-grain striping. Fine-grain striping works well for very lightly loaded systems but coarse-grain striping is much better for heavily loaded systems [6, 10, 27]

Our scheme provides for the inevitable disk failure. We present the single failure resilient version here but multiple failures can be accommodated [9]. For all the schemes presented here, the redundancy calculations utilize the ubiquitous and efficient exclusive-or operation.

We present four distinct data layouts; the first is an array of data disks often referred to as "just a bunch of disks" (JBOD), the second is RAID 3, the third is RAID 5, and finally an example of our scheme *segmented information dispersal* (SID). Each layout has some advantages and disadvantages which we mention as well. For "larger" video servers, one approach utilizes several instances of these data organizations; we will consider such examples in the design section of the paper. We assume that all video materials will be stored within the server. Typically, our servers will be able to service more streams than the number of full length videos stored within them. We assume the server contains the most frequently requested video materials and many clients will be independently requesting access to the same video. We return to these points in the final section of the paper.

The JBOD data organization is shown in Figure 1. The notation $Sx.y$ designates slice y of movie x. The round-robin assignment of slices to disks achieves load balancing. The scheme utilizes 100 percent of the storage capacity of the five disks for video data. However, a disk failure results in a non-delivered slice every fifth delivery for all movies; it essentially results in complete loss of service. The shaded slice indicates a typical fault-free access for a single video stream.

The RAID 3 data organization [21] is shown in Figure 2. Each slice is decomposed into four fragments each residing on a distinct disk. The notation $Sx.y.z$ designates fragment z of slice y of movie x, and $Px.y$ denotes the check data of the fragments of slice y of movie x (e.g. $P1.0 = S1.0.0 \oplus S1.0.1 \oplus S1.0.2 \oplus S1.0.3$.) The data within the shaded rectangles is read in parallel by all disks. The use of slice fragments also achieves load balancing. The scheme utilizes 80 percent of its storage capacity for video data. The array can accommodate a single disk failure. The shaded slice fragments indicate the typical fault-free access for a single video stream. No degradation of performance occurs since the parity can be computed fast enough.

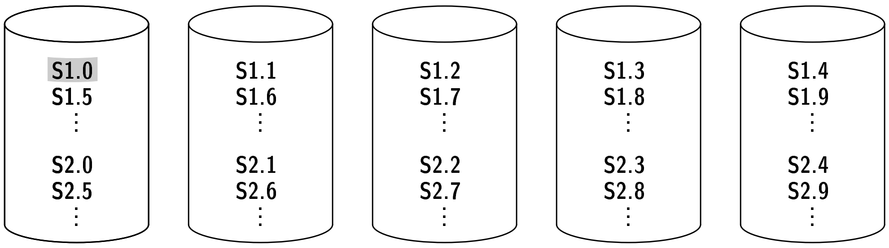

Figure 1: JBOD with slice striping

Note that we will always access the parity data (even though it is not shaded) in parallel with the video slice data. However, this organization suffers from poor fault-free performance due to the dedicated parity disk which is not utilized under the fault-free mode and the cumulative effects of fine-grained striping.

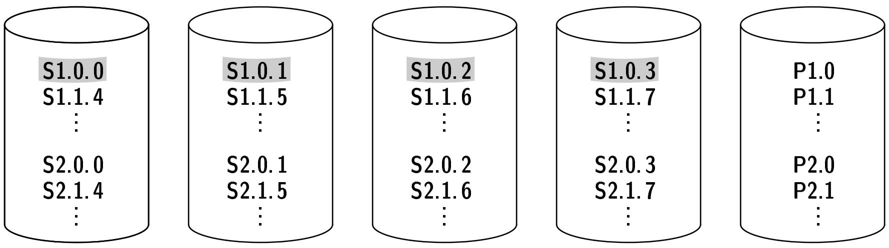

Figure 2: RAID 3 Data Layout

The RAID 5 data organization [21] is shown is Figure 3. Individual slices are stored contiguously on single disks; consecutive slices are stored in round robin fashion. $Sx.y$ denotes slice y of movie x. $Px.z$ denotes the parity of the slices y such that $z = \lfloor y/4 \rfloor$ for movie x; that is, for the slices that appear in the same row. The shaded slice indicates the typical fault-free access for a single video stream. The RAID 5 data layout obtains load balancing and the scheme utilizes 80 percent of its storage capacity for video data. The array can accommodate a single disk failure. This organization has good fault-free performance, but it suffers from poor performance under failure since the workload on the surviving disks doubles. The doubling arises by assuming only that the read load is evenly divided among the disks. Each surviving disk will have to read one additional slice for each video stream that was to be serviced by the failed disk. Here $Px.z = Sx.4z \oplus Sx.(4z+1) \oplus Sx.(4z+2) \oplus Sx.(4z+3)$; $Px.z$ resides on disk $(4-z) \bmod 5$.

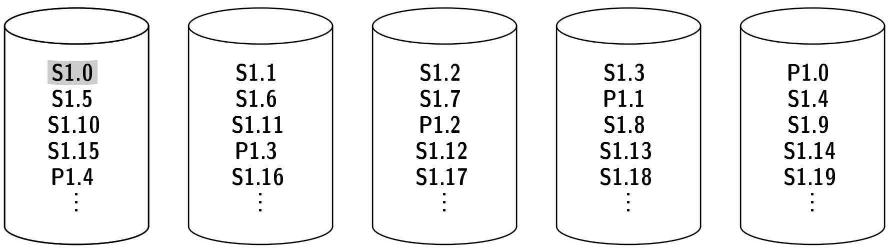

Figure 3: RAID 5 Data Layout

The SID data organization is shown in Figure 4. Individual slices are stored contiguously on single disks; consecutive slices are stored in a round robin fashion. Each slice has an associated redundant/check data fragment. In this configuration, the redundant data is one-half the size of the slice. (In this paragraph, any attribute of SID that depends on the configuration will by associated with the phrase "in this configuration.") Each slice is logically partitioned into two equal sized data fragments in this configuration. The notation $Fx.y.z$ designates data fragment y of slice z of movie x and $Px.z$ denotes the check fragment stored on the disk with data fragments $Fx.0.z$ and $Fx.1.z$. The slice $Sx.z$ from Figure 3 is exactly the data fragments $Fx.0.z$ and $Fx.1.z$ juxtaposed. In this configuration, if $d = z \bmod 5$ (the disk on which the slice is stored) and $r = \lfloor z/5 \rfloor$ (the slice "row number" within the data layout), then $Px.z = Fx.0.((d+1) \bmod 5 + 5r) \oplus$

$Fx.1.((d+4) \bmod 5 + 5r)$; thus, for example,

$$P1.0 = F1.0.1 \oplus F1.1.4 \qquad P1.1 = F1.0.2 \oplus F1.1.0 \qquad P1.2 = F1.0.3 \oplus F1.1.1$$
$$P1.3 = F1.0.4 \oplus F1.1.2 \qquad P1.4 = F1.0.0 \oplus F1.1.3.$$

The shaded fragments are contiguous and represent a typical slice access under fault-free operation. In this configuration, the layout utilizes 66.66 percent of its storage capacity for video data; other SID layouts can have larger or smaller storage utilization. Now for example, if disk 3 fails, we can reconstruct F1.0.3 and F1.1.3 by accessing fragments P1.2, F1.1.1, P1.4, and F1.0.0. Since each redundant fragment is only one-half the size of a slice, the incremental workload will be considerably smaller than for a RAID 5 layout.

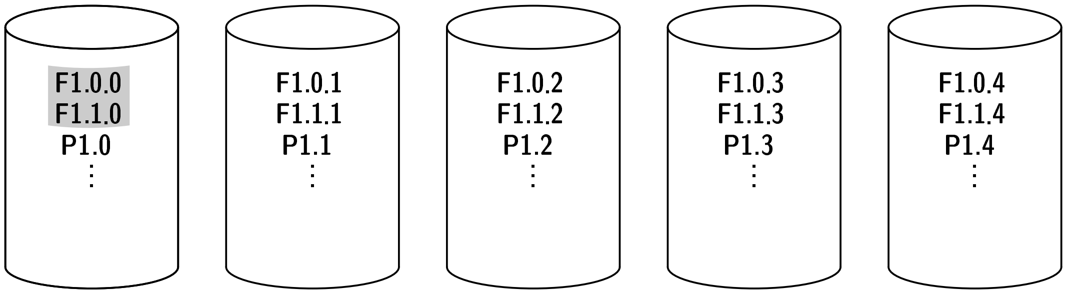

Figure 4: SID

One possible read scheduling is now described. Time during video presentation is divided into equal-duration *reading cycles* when exactly one slice is read for each stream from a disk[2] In the case of RAID 3 organizations, for each slice, the data is obtained by reading from all disks but one. Consequently, the remainder of this discussion does not apply to RAID 3. The streams being serviced by the video server are partitioned into groups which we call *cohorts*. Each cohort has an associated service list of slices (and fragments under degraded mode of operation) to be read during the reading cycle. Since video slice data is allocated in a round robin fashion, cohorts logically cycle through the disks moving from one disk to the next at the start of each reading cycle.

Each cohort has a maximum number of streams it can contain; this number will be limited by the buffer space and by the capability of the disk drives composing our video server. When the number of streams in a cohort is lower than this maximum number, the cohort contains one or more *free slots*. When the display of a new stream needs to be initiated, the system waits until a cohort with a free slot is about to be served by the disk where the first slice of the requested movie resides; the new stream is then incorporated into this cohort service list. When a stream ends, it is dropped from its cohort; this results in a free slot which can be used to initiate another new stream. Figure 5 shows an example of reading cycles along with their cohort service lists in a RAID 5 or SID video server with five disks. In this example, a cohort may contain up to four streams; there are currently three free slots.

reading cycle *t*	reading cycle *t*+*1*
disk0 service list: S1.0, S2.10, S5.5, S1.25	disk0 service list: S3.10, S4.5, S8.15
disk1 service list: S3.6, S4.11, S2.1	disk1 service list: S1.1, S2.11, S5.6, S1.26
disk2 service list: S2.7, S6.12, S1.2	disk2 service list: S3.7, S4.12, S2.2
disk3 service list: S7.3, S3.13, S5.8, S4.3	disk3 service list: S2.8, S6.13, S1.3
disk4 service list: S3.9, S4.4, S8.14	disk4 service list: S7.4, S3.14, S5.9, S4.4

Figure 5: SID and RAID 5 fault-free reading cycles

We conclude our discussion by noting again the load reduction exemplified by our SID example above. The load reduction occurs by reducing the size of the reconstruction reads; this reduces the transfer time

[2]This applied to constant bit rate compression. For variable bit rate compression, *at most* one slice is read for each stream during a reading cycle (see Section 4).

which is a dominant component of the buffer replenishment latency. Suppose that disk 3 is inoperative during reading cycle $t+1$. The SID reconstruction equations for disk 3 slice fragments are

$$Fx.0.(3+5j) = Px.(2+5j) \oplus Fx.1.(1+5j) \qquad Fx.1.(3+5j) = Px.(4+5j) \oplus Fx.0.5j$$

where x designates the movie and $j \geq 0$ the slice row number within the data layout. The revised SID service lists are illustrated within Figure 6; the fragments to the right of the semicolon in reading cycle $t+1$ are one-half the size of the slices. The RAID reconstruction invariants are expressed as

$$Px.j \oplus Sx.4j \oplus Sx.(4j+1) \oplus Sx.(4j+2) \oplus Sx.(4j+3) = 0$$

where x designates the movie and $j \geq 0$ the data layout row number. The revised RAID service lists are illustrated within Figure 7. Under failure, SID and RAID 5 behave quite differently; within RAID 5 the reading cycle would be longer since all objects accessed are the same size while SID's additional accesses are all much smaller.

reading cycle *t*	reading cycle *t+1*
disk0 service list: S1.0, S2.10, S5.5, S1.25	disk0 service list: S3.10, S4.5, S8.15; F2.0.5, F6.0.10, F1.0.0
disk1 service list: S3.6, S4.11, S2.1	disk1 service list: S1.1, S2.11, S5.6, S1.26; F2.1.6, F6.1.11, F1.1.1
disk2 service list: S2.7, S6.12, S1.2	disk2 service list: S3.7, S4.12, S2.2; P2.7, P6.12, P1.2
disk3 service list: S7.3, S3.13, S5.8, S4.3	disk3 service list:
disk4 service list: S3.9, S4.4, S8.14	disk4 service list: S7.4, S3.14, S5.9, S4.4; P2.9, P6.14, P1.4

Figure 6: SID Reading cycles with disk 3 inoperative

reading cycle *t*	reading cycle *t+1*
disk0 service list: S1.0, S2.10, S5.5, S1.25	disk0 service list: S3.10, S4.5, S8.15; S2.10, S6.15, S1.0
disk1 service list: S3.6, S4.11, S2.1	disk1 service list: S1.1, S2.11, S5.6, S1.26; S2.11, P6.3, S1.1
disk2 service list: S2.7, S6.12, S1.2	disk2 service list: S3.7, S4.12, S2.2; P2.2, S6.12, S1.2
disk3 service list: S7.3, S3.13, S5.8, S4.3	disk3 service list:
disk4 service list: S3.9, S4.4, S8.14	disk4 service list: S7.4, S3.14, S5.9, S4.4; S2.9, S6.14, P1.0

Figure 7: RAID 5 reading cycles with disk 3 inoperative

Within SID, the slices $S2.8$, $S6.13$, and $S1.3$ are calculated by noting that $Sx.(3+5j)$ consists of $Fx.0.(3+5j)$ and $Fx.1.(3+5j)$ juxtaposed. These calculations are performed at the conclusion of reading cycle $t+1$.

SID calculations:

$$\begin{aligned} F2.0.8 &= P2.7 \oplus F2.1.6 & F2.1.8 &= P2.9 \oplus F2.0.5 \\ F6.0.13 &= P6.12 \oplus F6.1.11 & F6.1.13 &= P6.14 \oplus F6.0.10 \\ F1.0.3 &= P1.2 \oplus F1.1.1 & F1.1.3 &= P1.4 \oplus F1.0.0. \end{aligned}$$

RAID 5 calculations:

$$\begin{aligned} S2.8 &= P2.2 \oplus S2.10 \oplus S2.11 \oplus S2.9 \\ S6.13 &= P6.3 \oplus S6.15 \oplus S6.12 \oplus S6.14 \\ S1.3 &= P1.0 \oplus S1.0 \oplus S1.1 \oplus S1.2. \end{aligned}$$

3 Segmented Information Dispersal (SID)

The SID data organization provides a middle ground between the two well-known extremes RAID level 1 and RAID level 5 as well as capturing the goal outlined at the end of the previous section. The (n,q)-SID design has parameters n and q designating the number of disks within the array and the *dispersal factor* respectively. We give a functional description of the scheme here; a formal description is given in [8]. Every slice contains $k \cdot q$ data bytes, composed of q juxtaposed fragments each of size k bytes; a slice of data is stored contiguously on a single disk. The dispersal factor q designates the number of juxtaposed fragments constituting a slice. The parameter k is not central to our discussion other than it determines the slice size.

Every slice of data has an associated *check* fragment consisting of k bytes that is stored on the same disk as the slice. The check fragment value is the exclusive-or of q data fragments each residing on one of q other disks. The fragments composing the check fragments provide single disk failure tolerance. That is, any slice of data can be reconstructed by accessing the surviving disks and obtaining at most one fragment (k bytes) from each. SID requires the check fragments as well as data fragments to have size k bytes. However, the typical data access, in fault-free operation, transfers a slice ($k \cdot q$ bytes of data) residing on a single disk. The redundancy ratio, measuring the ratio of the size of the check data to the size of the check plus user data, is $1/(q+1)$ for the SID schemes; in RAID 1, it is $1/2$ and in RAID 5, it is $1/n$. In our overview examples, the (5,2)-SID of Figure 4 has a redundancy ratio of $1/3$ and the 5 disk RAID 5 in Figure 3 has a redundancy ratio of $1/5$.

An approach to obtaining suitable SID data layouts is presented immediately after discussing the relationship between n and q. Suppose a slice of $k \cdot q$ bytes must be recovered. Each of its q fragments resides within one of q check fragments. Furthermore each of these check fragments requires $q-1$ other data fragments so we can obtain the desired fragment. Thus, we must access $q + q(q-1) = q^2$ fragments each of size k. If each of these fragments resides on a distinct disk and we include the failed disk, we obtain

$$n \geq q^2 + 1. \tag{1}$$

Thus, we see that q cannot be any larger than $\sqrt{n-1}$ within the SID scheme. We will refer to this inequality as the *necessary condition.* As a technical aside, we note that there are at most four designs in which equality holds; i.e. n equals q^2+1. These are SID designs for q equal to $2, 3, 7,$ and possibly 57. Our previous SID example is the q equals 2 case. A significant consequence of q being less than $\sqrt{n-1}$ is that during degraded operation the reconstruction work is evenly distributed over only q^2 disks rather than the surviving $n-1$. Consequently, we must forgo our desire to evenly distribute the reconstruction workload over all surviving disks other than in the three (or possibly four) exceptional cases. The non-existence of (q^2+1, q)-SID designs is discussed in detail within [8]; this follows from known graph theoretic results regarding the non-existence of q-regular graphs with q^2+1 nodes and girth 5, referred to as strongly-regular graphs, except for q equal to 2,3,7, and possibly 57.

One approach to obtaining SID designs is exhaustive search together with backtracking; this will obtain any and all such designs but it is an enormous computational task. The technique we present now, referred to as *separated difference sets,* provides (n,q)-SID designs in which q is reasonably large but not necessarily maximal.

Definition: A *separated difference set*, denoted (n,q)-SDS, is a set $C = \{c_0, c_1, \ldots, c_{q-1}\}$ of q positive integers less than n satisfying the following condition. Let D be the set of all differences between ordered pairs of distinct elements of C,

$$D = \{\ (c_i - c_j) \bmod n \mid 0 \leq i,j < q \text{ and } i \neq j\ \}$$

then $|C \cup D| = q^2$.

The name designates the fact that each of the $q \times (q-1)$ differences must be distinct and they must also differ from the q elements of C. Separated difference sets are very straightforward to construct even by exhaustive search. As an example, we consider a (5,2)-SDS in which C is $\{1,4\}$; then the difference set D is $\{2,3\}$ and we see that $C \cup D$ contains the necessary four elements. As a slightly larger example, we consider an $(n,3)$-SDS design; here C could be $\{1, 4, n-1\}$. Then the difference set D is $\{2, 3, 5, n-5, n-3, n-2\}$ and the union of the two contains the required nine elements provided n is at least 11.

An (n,q)-SDS immediately provides an (n,q)-SID design. The elements of the C set designate *offsets* for the parity calculations. There are n disks, labeled $0, 1, \ldots, n-1$, each containing slice and check data. Each slice S_z consists of q juxtaposed fragments denoted $F_{0,z}$, $F_{1,z}$, $\cdots$, $F_{q-1,z}$; the associated parity fragment P_z contains the exclusive-or of the following data fragments:

$$F_{0,(d+c_0) \bmod n + nr},\ F_{1,(d+c_1) \bmod n + nr}, \ldots,\ F_{q-1,(d+c_{q-1}) \bmod n + nr}, \tag{2}$$

where $d = z \bmod n$ (the disk on which slice S_z resides) and $r = \lfloor z/n \rfloor$ (the "row number" of slice S_z in the layout). This notation is the same as in section 2 except the movie number is omitted. The elements of the C set determine only the offsets for parity. The selection of fragments is arbitrary provided all are covered within the design. The explicit choices above cover all the data fragment in the SID design. Our example in

Figure 4 is based on the (5,2)-SDS in which C is $\{1,4\}$. A slightly larger example is presented at the end of this section.

We now verify that the data layouts constructed above are viable SID designs. We show that the reconstruction of a slice S_z will entail fetching q^2 fragments on q^2 distinct disks within our SDS based SID design. We have noted, in determining inequality 1 above, that q^2 fragments must be accessed. The following set of parity fragments contain stripe data for S_z:

$$P_{(d-c_0) \bmod n+nr},\ P_{(d-c_1) \bmod n+nr}, \ldots,\ P_{(d-c_{q-1}) \bmod n+nr}, \tag{3}$$

where, as before, $d = z \bmod n$ and $r = \lfloor z/n \rfloor$. Parity fragment $P_{(d-c_i) \bmod n+nr}$ was computed using the following data fragments:

$$F_{0,(d-c_i+c_0) \bmod n+nr},\ F_{1,(d-c_i+c_1) \bmod n+nr}, \ldots,\ F_{q-1,(d-c_i+c_{q-1}) \bmod n+nr}. \tag{4}$$

First we show that the set of disks containing the data fragments within $P_{(d-c_i) \bmod n+nr}$ which reside on disks $(d - c_i + c_j) \bmod n$ for $0 \le j < q$, are distinct from the disks containing stripe data for S_z. Otherwise, we have $d - c_i + c_j \equiv d - c_k \bmod n$ for some j and k. The expression $(d - c_k) \bmod n$ denotes a parity disk listed in 3 and the expression $(d - c_i + c_j) \bmod n$ denotes a disk, listed in 4, containing the fragment data needed by the parity disk $(d - c_k) \bmod n$. However, this cannot happen within an SDS construction since we would have a difference $c_i - c_j$ equivalent to an offset c_k and this would diminish the cardinality of $C \cup D$.

The other possibility is that the set of disks containing the fragments within $P_{(d-c_i) \bmod n+nr}$ and $P_{(d-c_j) \bmod n+nr}$ includes disk d as well as at least one other disk in common. Then we would have $d - c_i + c_k \equiv d - c_j + c_h \bmod n$ with $i \neq j$, $k \neq i$, and $h \neq j$. Each expression denotes a disk that contains a data fragment for the i^{th} or j^{th} parity disk. However, this cannot happen within an SDS construction since we would have a pair of equivalent differences which would diminish the cardinality of $C \cup D$. Thus, our SDS based SID data layouts operate as claimed.

Table 1 presents SDS based SID designs for $5 \le n \le 100$. For each value of n, the table shows the obtained q and the set of SDS offsets. These SDS configurations were obtained via computer search. For certain values of n, a higher dispersal factor appears in parentheses. For these cases, a solution with a higher dispersal factor is known [8], but it cannot be specified using the SDS representation.

We conclude the section with a slightly larger example design: a (11,3)-SDS based SID design. In this data layout, 25% of the disk space is devoted to redundant data. However, the performance, under single disk failure, is much better than that of a 4 disk RAID level 5 data layout also with 25% of the disk space devoted to redundant data; we consider server performance in detail in the next section. Figure 8 presents the parity calculations for the first slice "row" of the data layout. The next rows follow the same pattern obtained by using the SDS {1,4,10} (e.g. $P_{11} = F_{0,12} \oplus F_{1,15} \oplus F_{2,21}$, $P_{12} = F_{0,13} \oplus F_{1,16} \oplus F_{2,11}$, etc.)

$P_0 = F_{0,1} \oplus F_{1,4} \oplus F_{2,10}$	$P_1 = F_{0,2} \oplus F_{1,5} \oplus F_{2,0}$	$P_2 = F_{0,3} \oplus F_{1,6} \oplus F_{2,1}$
$P_3 = F_{0,4} \oplus F_{1,7} \oplus F_{2,2}$	$P_4 = F_{0,5} \oplus F_{1,8} \oplus F_{2,3}$	$P_5 = F_{0,6} \oplus F_{1,9} \oplus F_{2,4}$
$P_6 = F_{0,7} \oplus F_{1,10} \oplus F_{2,5}$	$P_7 = F_{0,8} \oplus F_{1,0} \oplus F_{2,6}$	$P_8 = F_{0,9} \oplus F_{1,1} \oplus F_{2,7}$
$P_9 = F_{0,10} \oplus F_{1,2} \oplus F_{2,8}$	$P_{10} = F_{0,0} \oplus F_{1,3} \oplus F_{2,9}$	

Figure 8: (11,3)-SDS based SID data parity calculation scheme.

The reconstruction calculations for a failed disk access fragments on nine of the ten surviving disks. Figure 9 presents the calculation scheme for failed disk 3; one fragment is obtained from disks 0,1,2,4,5,6,8,9, and 10.

$F_{0,3} = P_2 \oplus F_{1,6} \oplus F_{2,1}$	$F_{1,3} = P_{10} \oplus F_{0,0} \oplus F_{2,9}$	$F_{2,3} = P_4 \oplus F_{0,5} \oplus F_{1,8}$

Figure 9: (11,3)-SDS based SID reconstruction calculation scheme for disk 3.

Table 1: SID data layout designs for $5 \leq n \leq 100$

n	q	*SDS offsets*	n	q	*SDS offsets*	n	q	*SDS offsets*
5	2	1 4	37	5	1 3 7 12 30	69	7	1 3 7 12 22 30 56
6	2	1 5	38	5	1 3 7 17 30	70	7	1 3 7 12 22 45 53
7	2	1 6	39	5	1 3 7 12 22	71	7	1 3 7 12 22 30 46
8	2	1 7	40	5(6)	1 3 7 12 20	72	7	1 3 7 12 20 34 49
9	2	1 8	41	5	1 3 7 12 20	73	8	1 3 7 15 31 36 54 63
10	2(3)	1 9	42	5	1 3 7 12 20	74	7	1 3 7 12 20 36 54
11	3	1 4 10	43	6	1 3 8 25 31 35	75	7	1 3 7 12 20 34 50
12	3	1 4 11	44	6	1 3 20 29 34 41	76	7	1 3 7 12 20 30 46
13	3	1 4 12	45	6	1 3 9 14 29 36	77	7	1 3 7 12 20 30 44
14	3	1 4 13	46	6	1 3 8 28 32 38	78	8	1 6 16 25 29 36 62 76
15	3	1 4 14	47	6	1 3 7 17 22 30	79	8	1 3 9 22 46 50 57 62
16	3	1 4 15	48	6	1 3 8 14 18 30	80	8	1 3 9 19 30 44 61 76
17	3	1 4 16	49	6	1 3 7 12 29 37	81	8	1 3 7 18 34 39 62 74
18	3(4)	1 4 17	50	6(7)	1 3 7 15 24 35	82	8	1 3 9 16 35 40 62 73
19	4	1 3 7 12	51	6	1 3 7 12 20 30	83	8	1 3 7 22 35 40 60 76
20	4	1 3 8 17	52	6	1 3 7 12 22 35	84	8	1 3 7 17 25 36 48 57
21	4	1 3 8 18	53	6	1 3 7 12 22 40	85	8	1 3 7 20 25 36 48 78
22	4	1 3 11 16	54	6	1 3 7 15 26 39	86	8	1 3 7 12 33 51 64 79
23	4	1 3 7 16	55	6	1 3 7 12 20 30	87	8	1 3 7 12 22 39 57 65
24	4	1 3 7 12	56	6	1 3 7 12 20 36	88	8	1 3 7 12 22 30 58 75
25	4	1 3 7 12	57	7	1 3 13 32 36 43 52	89	8	1 3 7 12 20 35 49 65
26	4	1 3 7 12	58	6	1 3 7 12 20 38	90	8	1 3 7 15 20 43 64 75
27	4	1 3 7 12	59	6	1 3 7 12 20 34	91	9	1 3 9 27 49 56 61 77 81
28	4	1 3 7 12	60	7	1 3 14 20 24 29 36	92	8	1 3 7 12 20 43 67 77
29	4	1 3 7 12	61	7	1 3 7 15 25 36 45	93	8	1 3 7 12 20 34 49 70
30	5	1 4 13 23 29	62	7	1 3 10 31 39 45 50	94	8	1 3 7 12 20 36 46 74
31	5	1 3 8 12 18	63	7	1 3 7 15 20 31 41	95	8	1 3 7 12 20 34 44 60
32	5	1 3 7 12 25	64	7	1 3 7 26 38 43 56	96	8	1 3 7 12 20 30 46 61
33	5	1 3 8 20 30	65	7	1 3 7 16 27 37 49	97	9	1 4 9 20 35 45 47 77 84
34	5	1 3 7 12 27	66	7	1 3 7 12 20 41 51	98	8(9)	1 3 7 12 20 30 46 68
35	5	1 3 7 12 20	67	7	1 3 7 12 20 30 46	99	8	1 3 7 12 20 30 46 78
36	5	1 3 7 12 29	68	7	1 3 7 12 20 43 53	100	8	1 3 7 12 20 30 64 79

4 Performance Results

We conducted a study which compares the performance of several possible data organizations. Our results concern the buffer size per stream requirements for various data layout and cohort size configurations. Larger reading cycle times imply larger buffer sizes. We are trying to maximize the number of users for a given buffer size. The models are now presented along with the results.

Our reading cycle is flexible and we utilize the following seek optimization implementation referred to as the "nearest rule" [5] to minimize the expected actuator movement:

1. Sort the slices according to the cylinder location.
2. Determine the distance between the current position of the actuator and first and last slices of the sorted list. Move actuator to the cylinder of the slice at the closest end of the service list.
3. Access the slices according to the sorted list.
4. Seek the closest extreme (inner-most or outer-most) cylinder of the disk.

We employ the parameters provided by disk manufacturers to determine disk seek times. Seek time models have two distinct domains [12, 23]: one is the square root portion and the other is the affine portion. The boundary between the two portions is designated by b. Let the maximum possible seek distance be denoted by d_{max} (i.e. the number of cylinders minus 1). We estimate the time required to seek d cylinders as follows:

$$s(d) = \begin{cases} u_1 + v_1 d & \text{if } d \geq b \\ u_2 + v_2\sqrt{d} & \text{if } d \leq b \end{cases} \tag{5}$$

Since we know the the track-to-track, maximum and average seek times, denoted t_{min}, t_{max}, and t_{ave} respectively, let X be a random variable which measures the lengths of seeks in a random seek workload for the disk, and let $C = d_{max} + 1$ be the number of cylinders. The probability of a length d seek is determined as follows:

$$Prob[X = d] \;=\; 2\frac{C-d}{C(C-1)}$$

which is calculated assuming all pairs of cylinders are equally-likely. We can now find u_1, v_1, u_2, and v_2 by solving the following system of four linear equations:

$$\begin{aligned} t_{min} &= u_2 + v_2 \\ t_{max} &= u_1 + v_1 d_{max} \\ t_{ave} &= \sum_{d=1}^{b} Prob[X = d]\,(u_2 + v_2\sqrt{d}) \;+\; \sum_{d=b+1}^{d_{max}} Prob[X = d]\,(u_1 + v_1 d) \\ u_1 + v_1 b &= u_2 + v_2\sqrt{b} \end{aligned}$$

The last equation arises at the boundary between the two regions within the seek time model. Of course, if the disk manufacturer provides these model parameters, these calculations are unnecessary. Typically, manufacturers provide only t_{max}, t_{min} and t_{ave} values. Table 2 summarizes our disk parameter nomenclature as well as providing typical values.

Since we will be reading several slices in one sweep across the cylinders, we must determine the maximum seek latency per disk. Assuming that we read m data objects in the sweep, and that we use our seek optimization, this results in $m+1$ seeks per reading cycle. It is easy to show that the worst case seek latency occurs when the actuator starts at one extreme cylinder, makes $m+1$ equidistant stops finally ending at the other extreme. Then the maximum total seek latency $S(m)$ in a reading cycle with m streams per cohort is:

$$S(m) = \begin{cases} (m+1)\frac{u_1}{1000} + d_{max}\frac{v_1}{1000} & \text{if } m < \lfloor \frac{d_{max}}{b} \rfloor \\ (m+1)\left(\frac{u_2}{1000} + \frac{v_2}{1000}\sqrt{\frac{d_{max}}{m+1}}\right) & \text{otherwise.} \end{cases}$$

Finally we are left to determine the video slice size. We calculate the worst case time, denoted $T(m, \beta)$, required for reading m data objects of size β. Within an n disk RAID 5 or SID organization, each data object will be a slice of size β; in a RAID 3 organization, each data object will be of size $\beta/(n-1)$. Accordingly the number of streams supported within the RAID 5 or SID organization is nm and within the RAID 3 organization the number is exactly m. We determine T by summing the worst case seek, transfer, and rotational latencies; consequently these expressions all have the same form. There are three new terms within these expressions: t_r denotes the worst case disk rotational latency in milliseconds, r_t denotes the minimum transfer rate in kilobytes per second, and finally w_{min} denotes the minimal track size in in kilobytes (KB). Table 3 summaries these definitions.

- RAID 5 or SID without failure:

$$T(m,\beta) = S(m) + m\left(\frac{t_r}{1000} + \frac{\beta}{r_t} + \left\lceil \frac{\beta}{w_{min}} \right\rceil \frac{t_{min}}{1000}\right)$$

- RAID 5 with failure:

$$T(m,\beta) = S(2m) + 2m\left(\frac{t_r}{1000} + \frac{\beta}{r_t} + \left\lceil \frac{\beta}{w_{min}} \right\rceil \frac{t_{min}}{1000}\right)$$

- SID with failure:

$$\begin{aligned} T(m,\beta) = S(2m) &+ m\left(\frac{t_r}{1000} + \frac{\beta}{r_t} + \left\lceil \frac{\beta}{w_{min}} \right\rceil \frac{t_{min}}{1000}\right) + \\ &+ m\left(\frac{t_r}{1000} + \frac{\beta}{q\,r_t} + \left\lceil \frac{\beta}{q\,w_{min}} \right\rceil \frac{t_{min}}{1000}\right) \end{aligned}$$

- RAID 3 with or without failure:

$$T(m,\beta/D) = S(m) + m\left(\frac{t_r}{1000} + \frac{\beta}{D\,r_t} + \left\lceil \frac{\beta}{D\,w_{min}} \right\rceil \frac{t_{min}}{1000}\right)$$

The S expression denotes the worst case seek latency, the term $t_r/1000$ denotes the worst case rotational latency, the term β/r_t denotes the worst case transfer time, and finally $\lceil\beta/w_{min}\rceil\, t_{min}/1000$ denotes the time to do track-to-track moves within a slice. The last three expressions are all multiplied by m within the RAID 5 or SID without failure expression. Within the RAID 5 with failure expression, these expressions are multiplied by $2m$ since each disk will be providing exactly twice as many slices. The SID with failure expression has two expressions beyond the seek cost; one is identical to SID without failure and the second has fragment size β/q rather than β sized transfers. Finally, the RAID 3 with or without failure is similar to the RAID 5 without failure except the stripe width of D+1 and the size β/D of each accessed object. The RAID 3 organization differs from the others in that each disk contributes to each stream; all seeks and rotational latencies contribute to the length of the reading cycle. This feature makes RAID 3 a less attractive data organization.

To avoid starvation, the slice size β must be large enough so the time to display a slice β/r_c is no smaller than the time to access the next slice; here r_c denotes the display consumption rate in kilobytes per second. For SID and RAID 5, we must determine for a given number of streams m, the smallest β such that

$$\beta \geq T(m,\beta)\, r_c.$$

Similarly, for RAID 3, we must determine the smallest β such that

$$\beta \geq T(m,\beta/D)\, r_c$$

We shall refer to each of these inequalities as the *continuity condition*. Once we obtain a suitable value of β, we know the buffer size per stream: 2β. This selection of β will minimize the buffering requirements. The reading cycle length c_t is β/r_c seconds. By allocating a buffer of 2β kilobytes for each stream, we can guarantee that neither buffer starvation nor overflow occurs. From the requirement for non-starvation (β large enough) and the equations listed above, we can obtain lower bounds on β for the various cases. For example, for RAID 5 or SID without failure we get:

$$\beta \geq \frac{S(m) + m\,\frac{t_r}{1000}}{\frac{1}{r_c} - m\left(\frac{1}{r_t} + \frac{1}{w_{min}}\,\frac{t_{min}}{1000}\right)}$$

The search for β begins at this lower bound; the value of β is doubled until $\beta \geq T(m, \beta) r_c$. Then we do a binary search to obtain the smallest acceptable β.

The model described above applies to constant bit rate (CBR) compression where the consumption rate r_c is fixed. Variable bit rate (VBR) compression can be accommodated as well by making some adjustments. We set the consumption rate used in our model to the maximum consumption rate, and compute the slice size β as described above. Movies are stored on the disks exactly as in the CBR case (i.e. a round robin distribution of equal size slices). This will result in a variable display time for slices. To address this issue, during each reading cycle, *at most* one slice of size β is read for each stream. If the buffer for a stream already contains more than β bytes, no slice is read; otherwise, a slice is read. Recall that the buffer size per stream is 2β. Clearly, this policy does not suffer from buffer overflow. Buffer starvation does not occur even if no slice is read since this happens only if the buffer for the stream already contains β bytes which is enough data to display until the next replenishment.

Our performance study determines the buffer requirements per video stream for SID as well as RAID 3 and 5. The following figures give the results of our calculations for some typical SID as well as RAID 3 and 5 configurations. The disk parameters were based on the performance characteristics of Quantum Atlas II, Seagate Cheetah, and IBM UltraStar disks. Table 2 contains our parameter values.

15728	r_t	Minimum transfer rate per disk (KB/s)
6.00	t_r	Maximum rotational latency (ms)
6926	d_{max}	Maximum seek distance
0.8	t_{min}	Track to track seek time (ms)
12.2	t_{max}	Full excursion seek time (ms)
5.4	t_{ave}	Average seek time (ms)
80	w_{min}	Minimum track size (KB)
2.068082	u_1	Seek time coefficient
0.001463	v_1	Seek time coefficient
0.709040	u_2	Seek time coefficient
0.090960	v_2	Seek time coefficient
9	d_c	Disk capacity (GB)
1385	b	Boundary between the square root and linear portions of the seek time

Table 2: Disk Model Parameters

D	Number of disks in a parity group (for RAID 3)
r_c	Consumption rate per stream (KB/s)
c_t	Length of a reading cycle (sec)
β	Video slice size (KB)
$S(m)$	Maximum total seek latency when reading m slices (sec)
$T(m, \beta)$	Maximum time (sec) required to read m slices of size β
q	SID dispersal factor

Table 3: Video Server Model Parameters

Figure 10 shows how the buffering requirement per stream varies with the total number of concurrent streams for a disk array with 12 disks and a video consumption rate of 4 Mbits/sec. The figure presents the performance of three data organizations: RAID 3, RAID 5, and SID. The redundancy ratio for all three organizations is 1/4 (i.e. the RAID 3 and RAID 5 layouts consist of three parity groups of size four, and the SID layout has a dispersal factor of 3.) Striping of size β (slice sized), is utilized in RAID 5 and SID; striping within RAID 3 has size β / D. Accordingly, for SID and RAID 5, each point is for a multiple of 12 streams; for RAID 3, each point is for a multiple of 3 streams. The poor performance of the RAID 3 organization (both with and without failure) and the RAID 5 organization under failure is noted and we return to this comparison in the next section. In both Figures 10 and 11, the perceived discontinuites for RAID 3 arise because the points are so close together; with the wider spacing (more streams per ensemble) the "discontinuites" are less pronounced.

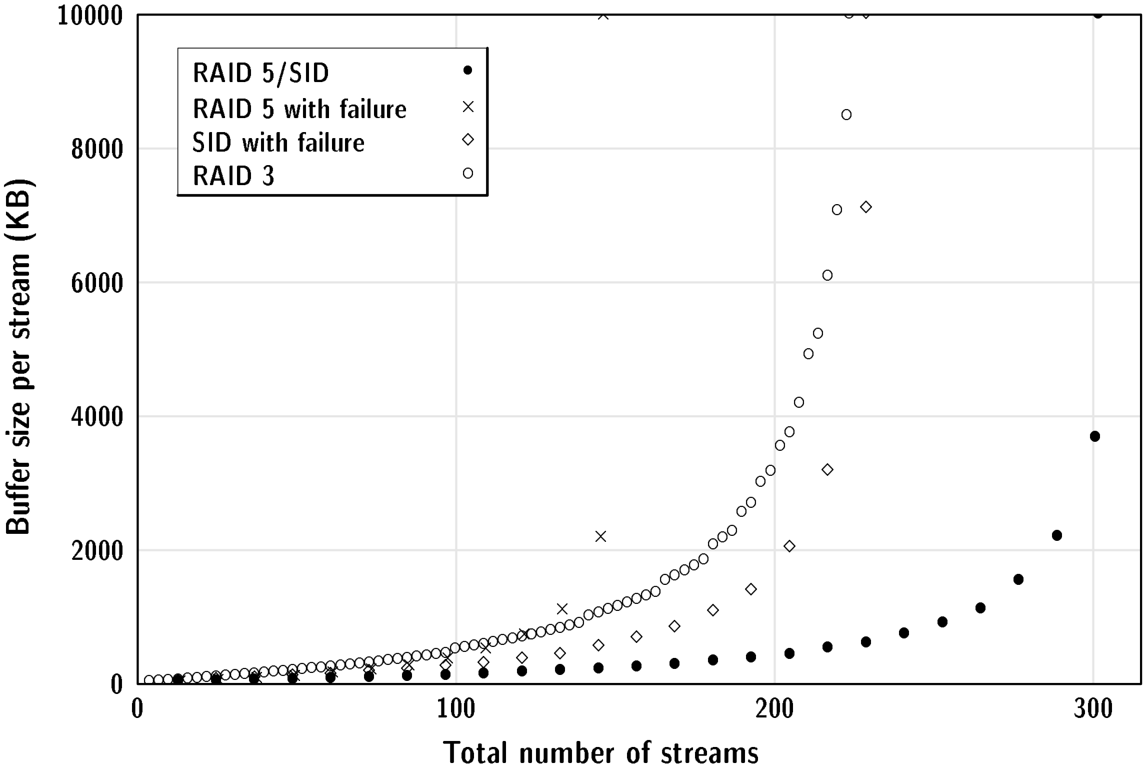

Figure 10: SID: $n = 12$, $q = 3$, vs. RAID 5 and RAID 3: three parity groups of size four; redundancy is 1/4, video consumption is 4 Mbits/sec

We note that both RAID 3 and RAID 5 provide a higher degree of fault-tolerance in this configuration since they protect against one disk failure per stripe while SID protects against one failure in the entire disk array. For a fixed level of fault-tolerance, SID requires a higher redundancy rate than RAID 3 or RAID 5, but it provides a significantly higher level of performance under failure. The same level of fault-tolerance as that of RAID 3 or RAID 5 can be achieved by dividing the disk array into SID groups in the same manner that RAID 3 and RAID 5 arrays are divided into parity groups. Since a large number of disks is desirable for performance reasons regardless of the data organization used, and since current and future disks offer a very high capacity, trading some amount of storage space for a significantly higher level of performance under disk failure is very acceptable.

Figure 11 shows the performance of a disk array with 90 disks. The redundancy rate for these data organizations is 1/9 (i.e. the RAID 3 and RAID 5 layouts consist of ten parity groups of size nine, and the SID layout has a dispersal factor of 8.) The impact of using SID in this case is considerably larger than in the previous case because of the higher dispersal factor. For RAID 5 and SID, each point is a multiple of 90 streams, and for RAID 3, each point is a multiple of ten streams. In this situation, if we limit the buffer size per stream to 10 MB, the SID data organization supports up to 1980 streams under failure, while RAID 5 only supports at most 1080 streams.

The impact of the dispersal factor is illustrated in Figure 12. We see, as expected, that performance improves as the dispersal factor increases, but diminishing returns are obtained once the dispersal factor is high enough to make latencies other than the transfer time dominate. In our disk model, the RAID 5 data organization services at most 12 streams per disk under failure.

An important advantage of SID when compared to BIBD layouts is the guarantee of contiguous reconstruction reads. Figure 13 shows the impact of discontiguity on the performance. The different curves correspond to different numbers of accesses performed during the reconstruction of a video slice. With SID, only one access is performed, but we studied the performance for larger numbers of accesses in order to determine the impact of the discontiguity exhibited by BIBD layouts. We see that even a small amount of discontiguity seriously affects the performance. Once the number of accesses per reconstruction data object

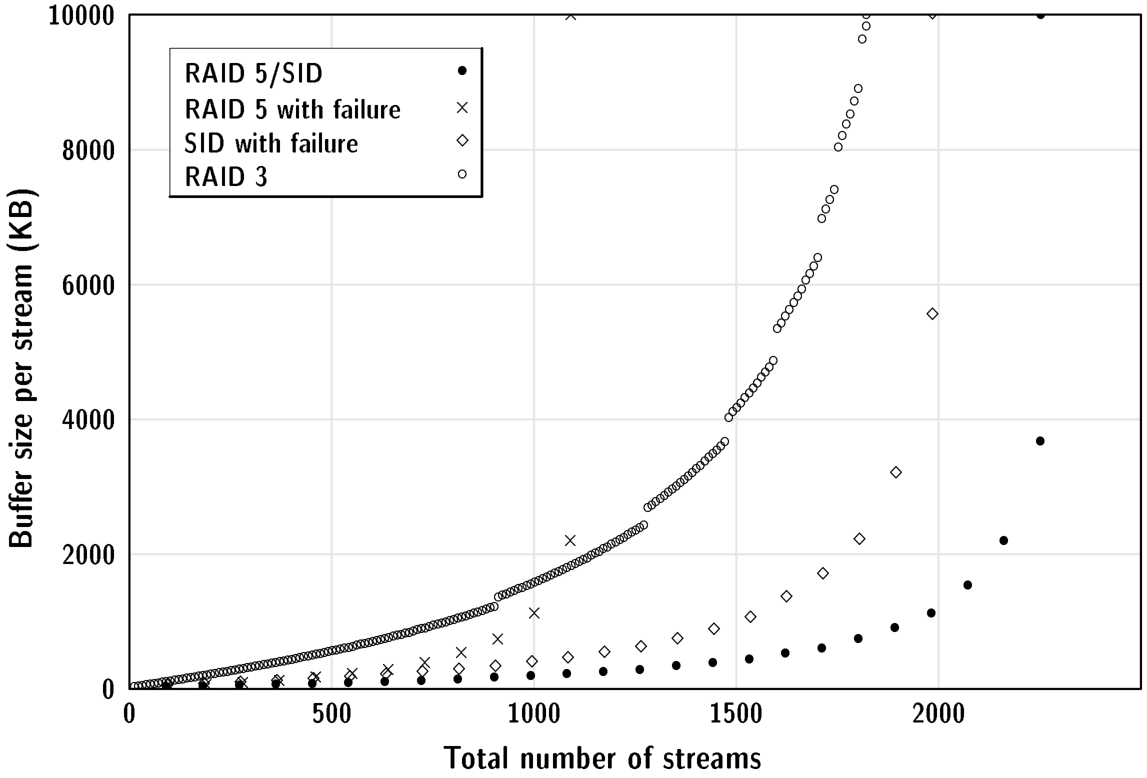

Figure 11: SID: $n = 90$, $q = 8$, vs. RAID 5 and RAID 3: ten parity groups of size nine; redundancy is 1/9, video consumption is 4 Mbits/sec

increases beyond 4, the performance becomes even worse than that of RAID 5 (which requires reading larger, but contiguous, pieces of data).

5 Video Server Design

A video server should support a pre-specified number of concurrent, independent streams with a low cost. We present SID-based cost optimal designs for a 1000 stream video server for different varieties of technologies. One design is based on circa 1996 disk drive technology and the other is based on circa 1998 technology. Thus, SID-based video server designs can successfully accommodate a range of disk technologies.

A similar RAID 5 based design was presented by Ozden et al. [19] using circa 1996 component technologies and costs. We begin by reviewing their results where they obtain a cost optimal RAID 5 video server for 1000 streams containing 44 disks in which each disk services up to 23 clients. We calculate, using similar analysis, that the reading cycle c_t is approximately 844 ms and the buffer space per stream per reading cycle β is 159 KB. Our analysis uses a more realistic (and larger) assessment of the total seek time per reading cycle. We determine that each (of 46 disks) will be able to service up to 22 clients per reading cycle. Consequently, our values for c_t and β differ slightly from theirs. This fault-free configuration costs \$81553 with a component cost of \$1500 per 2GB disk and \$40 per MB of RAM. This RAID 5 design cannot accommodate the inevitable disk failures gracefully; with a single disk failure, this design will fail to service the 1000 clients in a timely fashion. Our calculations show that by increasing the buffer space per stream, the cost optimal design contains 84 disks with $\beta = 259$ KB. This configuration costs \$146415. The greater cost provides additional reliability; our cost assessment is independent of the RAID stripe width. With smaller stripe widths, we gain additional reliability due to the additional redundant data stored but at the same time lose client data storage capacity. Ozden et al. [20] have considered single failure tolerant architectures for continuous media servers which employ BIBD declustered data layouts. They conclude their article with results for a 32 disk fault tolerant video server. The stripe width varies over 2,4,8,16, and 32 and the buffer space size is either 256 KB or 2 GB. They show that with 2 GB of buffer space, their server supports less

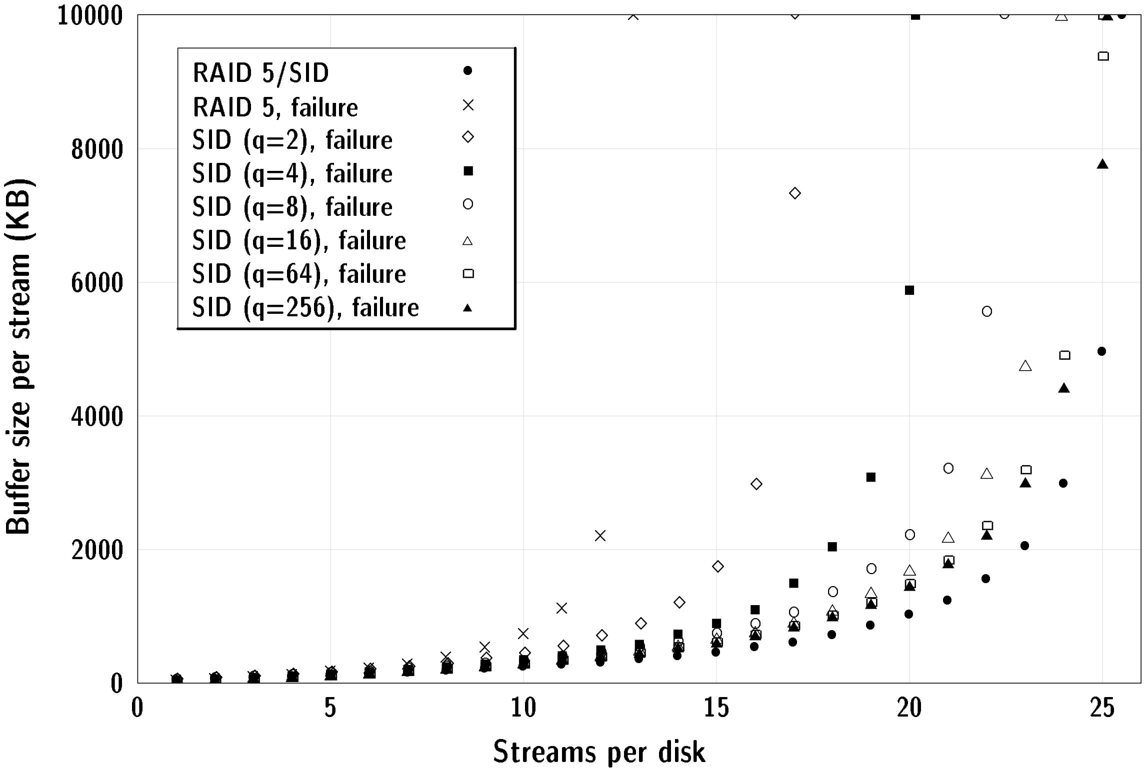

Figure 12: Dispersal factor impact: video consumption=4 Mbits/sec

than 700 streams in degraded mode.

Using circa 1996 technologies, we present cost optimal SID based data layouts for a 1000 client server. For fault-free operation, our analysis for c_t and β will be the same as given above since SID and RAID 5 behave identically in this mode. The server configuration cost will ultimately be determined by the dispersal factor q which we will let range from 2 to 8. We determine β to meet the continuity condition as well as select a suitable SID design in Table 1. These results are summarized in Table 4. The costs continue to be

q	n	β	cost $
2	67	231	118617
3	59	276	110107
4	59	217	105523
5	56	240	102880
6	53	280	101547
7	53	264	100289
8	53	254	99502

Table 4: SID cost-optimal designs

q	n'	g	β	cost $
5	30	2	198	105778
4	20	3	217	107292
3	15	4	276	111993
3	13	5	276	115425
3	10	6	276	112425
3	10	7	163	118371
2	5	14	231	123949

Table 5: SID design variations

reduced for larger q but we cannot meet our necessary condition $n \geq q^2 + 1$. The designs within Table 1 indicate that a (53,6)-SDS design is possible. Any of these hardware configurations is less expensive than the RAID 5 configuration. Remaining issues include the reliability and video storage capacity. Each of our video server configurations presented above tolerates a single failed disk. We can improve the reliability by using $g > 1$ stripes within the disk array, each of which is a SID-based design. We may stray slightly from optimal cost; however, the additional cost buys reliability. For each q, we select n' to be the smallest value to meet our necessary condition as well as a multiple gn' that is the smallest value at least equal to the number of disks in the q dispersal cost optimal layout. Table 5 contains some such configurations together with their costs. The resulting designs have improved reliability since with g groups, the video server will

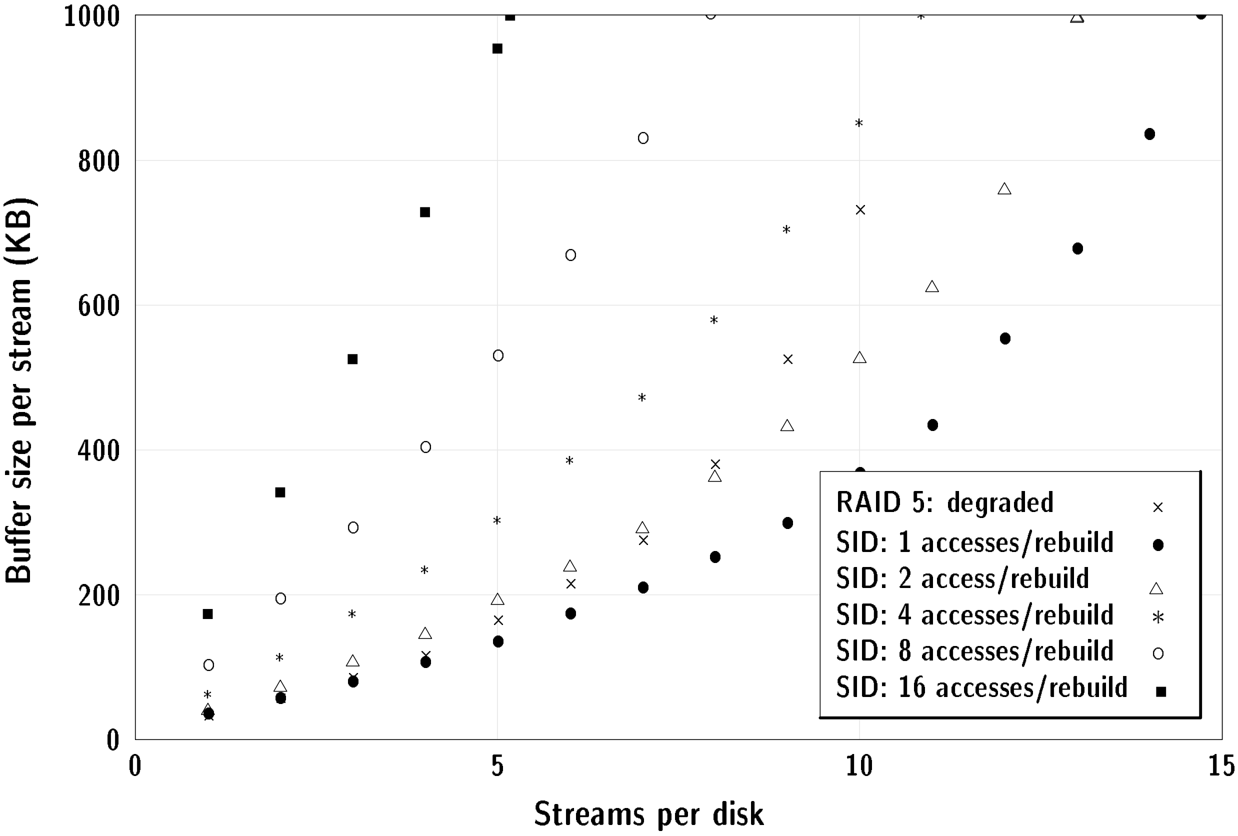

Figure 13: Discontiguity impact: $q = 3$, video consumption=4 Mbits/sec

tolerate some configurations of up to g concurrent disk failures. Finally we note the video storage capacity is $nd_c(q-1)/q$.

Now we consider a video server design for 1000 clients using the circa 1998 disk drives. First we present the cost optimal fault-free SID design which contains 42 disks, β is 1097 KB and the reading cycle period is 2142 ms; the cost is \$33719 with component costs for RAM of \$2 per MB and \$700 per 9GB disk. This configuration will suffice for either a RAID 5 or SID data layout. Second, we present several cost optimal single failure tolerant SID designs for various dispersal factors. Table 6 summarizes these designs. The $q = 1$ entry designates the RAID 5 configuration. Since n is small in these cost optimal designs and we must meet the continuity condition, only a few q values need be considered. As an aside, slightly larger q values would give rise to only slightly less costly component configurations in Table 6. Table 7 contains several feasible variation designs with $g > 1$ groups having costs close to optimal.

q	n	β	cost \$
1	84	1097	63118
2	63	1415	49675
3	59	1013	45269
4	53	1507	43029
5	53	1114	41485
6	50	1357	40304

Table 6: SID cost optimal designs

q	n'	g	β	cost \$
1	12	7	1097	63118
2	7	9	1415	49675
3	15	4	1013	46036
4	18	3	1507	43839
5	30	2	1115	46965

Table 7: SID design variations

Finally, we demonstrate the reliability and storage capacity for a pair of these designs: the design (A) consists of nine (7,2)-SDS data layouts and (B) seven stripe width twelve RAID 5 data layouts. The redundancy ratio for (A) is 1/3 and for (B) 1/12; the storage capacity for (A) is $63 \times 2/3 \times d_c = 42d_c$ MB and for (B) is $84 \times 11/12 \times d_c = 77d_c$ MB. We note the mean time to data loss (MTTDL) for the SDS based data layout is $0.076487/\lambda$ which is greater than the MTTDL for the RAID based design which is $0.049974/\lambda$; here λ designates the failure rate for a single disk. The MTTDL values are determined assuming that no repair is done; none of our design analysis allows for reconstruction activity.

We continue with two configurations that have identical storage capacities. Suppose we utilize K_A instances of the (A) configuration and K_B of (B). Then we require that

$$K_A \times 42 = K_B \times 77.$$

In other words, we have $K_A = 11/6K_B$; we consider an ensemble in which K_A is 11 and K_B is 6. The eleven (A) system configuration costs $11 \times \$49674 = \$546414$; each disk can service up to 16 streams for a total of $16 \times 63 \times 11 = 11088$. The six (B) system configuration costs $6 \times \$63118 = \$378708$; each disk can service up to 12 streams for a total of $6 \times 84 \times 12 = 6048$. The per stream cost for the (A) system configuration is \$49.27 and for the (B) system configuration \$62.61. We calculate the MTTDL for these systems: the Markov model representing the eleven (A) system configuration is given within figure 14. The Markov model representing the six (B) system configuration is fiben within figure 15. The individual disk

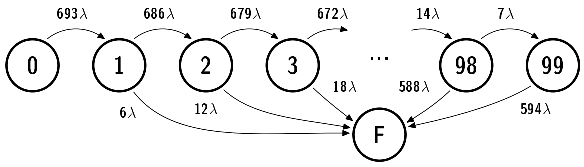

Figure 14: Eleven × nine (7,2) SDS Markov model

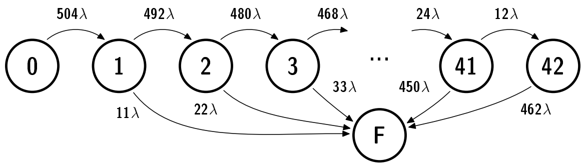

Figure 15: Six × seven (12,1) RAID 5 Markov model

failure rate is λ. The horizontal rows of states labeled **I** designate **I** groups each containing one failed disk; state **F** designates that at least one group contains at least two failed disks. $MTTDL_A$ is $0.02049/\lambda$ and $MTTDL_B$ is $0.01780/\lambda$. The almost identical MTTDLs for these configurations are roughly 2% of the mean time to failure (MTTF) for a single disk. In this comparative example, the actual video capacity of the two configurations is identical, the MTTDL is essentially the same, and the cost per stream is much less within the SID based data layout.

6 Conclusions

The SID data layout provides excellent performance for video servers as well as any other task characterized by large, fixed-size and constant rate sequential accesses. A slice is large provided the sum of the rotational and seek latencies is less than the transfer time for the slice data object. The SID fault-free run-time performance is identical to that of RAID 5; since the access rate is constant, our performance measure is the minimal buffer size per stream required to maintain a given number of streams. The SID degraded mode run-time performance is much better than either that of RAID 3 or 5. One limiting aspect of SID run-time performance is the sum of the seek and rotational latencies. By raising the dispersal factor, we can diminish the data transfer times during degraded mode operation but eventually the seek and rotational latencies dominate.

The benefits of contiguous data layout have been noted as well. We observe a severe penalty per stream as the number of accesses per parity fragment increases. Other data layout models, such as BIBD, often necessitate a non-contiguous data layout with resulting poor performance.

We have considered RAID 5 and SID data layouts with (as much as possible) identical redundancy ratios for our "performance results." SID extends the domain of possible fault tolerant video server designs considerably. SID obtains contiguous data layout and has excellent fault-free and degraded mode run-time performance.

Even with disk drives increasing their storage capacity, it will probably not be possible to store all desired videos within the video servers. But since our video service is a read-only operation, we could include a tertiary near-line archive that would transfer videos to the server. We could permanently allocate some space to each cohort or have two modes of operation: "normal" as before and "almost normal" in which the number of clients serviced would be slightly diminished to allow for migration of video material from the archive. This variety of migration would be very straightforward without dynamic contingencies. Another approach would be "just in time" delivery of the video material which would require the tertiary archive to be capable of operating at a sufficiently rapid rate. In both situations, a key decision is what to overwrite. This topic is the subject for an expanded study.

Acknowledgements

We wish to thank the anonymous referees for their detailed suggestions regarding our presentation.

References

[1] Alvarez, G.A., W.A. Burkhard, and F. Cristian, Tolerating Multiple Failures in RAID Architectures with Optimal Storage and Uniform Declustering. *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pp. 62-72, 1997.

[2] Alvarez, G.A., W.A. Burkhard, L.J. Stockmeyer, and F. Cristian, Declustered Disk Array Architectures with Optimal and Near-optimal Parallelism. *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pp. 109-120, 1998.

[3] Berson, S., L. Golubchik, and R.R. Muntz. Fault Tolerant Design of Multimedia Servers. *SIGMOD Record*, **24**(2), pp. 364–375, 1995.

[4] Berson, S., R.R. Muntz, S. Gandeharizadeh, and X. Ju. Staggered Striping in Multimedia Information Systems. *SIGMOD Record*, **23**(2), pp. 79-90, 1994.

[5] Bitner, J.R. and C.K. Wong, Optimal and Near-Optimal Scheduling Algorithms for Batched Processing in Linear Storage. *SIAM Journal of Computing, 8*, pp. 479-498, 1979.

[6] Chen, P. and D.A. Patterson. Maximizing performance in a striped disk array. In *Proceedings of the 17th International Symposium on Computer Architecture*, pp. 322-331, 1990.

[7] Cohen, A. and W.A. Burkhard. Segmented Information Dispersal (SID) for Efficient Reconstruction in Fault-Tolerant Video Servers. In *Proceedings of the ACM International Multimedia Conference*, pp. 277-286, 1996.

[8] Cohen, A. and W.A. Burkhard. Segmented Information Dispersal—A New Design with Application to Erasure Correction. *submitted,* 1997; also `http://www-cse.ucsd.edu/groups/gemini/papers.html`

[9] Cohen, A. *Segmented Information Dispersal.* Ph.D. Dissertation, Department of Computer Science & Engineering, University of California, San Diego, 1996.

[10] Cohen, A., W.A. Burkhard, and P.V. Rangan. Pipelined Disk Arrays for Digital Movie Retrieval. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 312–317, 1995.

[11] Furino, S., Maio, Y., Yin, J. **Frames and Resolvable Designs: Uses, Constructions, and Existence**, CRC Press, Boca Raton, 1996.

[12] Gary, J.B., B. Horst, and M. Walker. Parity Striping of Disc Arrays: Low Cost Reliable Storage with Acceptable Throughput. In *Proceedings of the 16th VLDB Conference,* pp. 148-161, 1990.

[13] Haskell, B.G. and A. Puri and A.N. Netravali. **Digital Video: An Introduction to MPEG-2**, Chapman & Hall, New York, 1997.

[14] Holland, M., and G.A. Gibson. Parity Declustering for Continuous Operation in Redundant Disk Arrays. In *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 23–35, 1992.

[15] Krunz, M. and H. Hughes. A Traffic Model for MPEG-Coded VBR Streams. In *Proceedings of the Joint International Conference on Measurement & Modeling of Computer Systems*, pp. 47-55, 1995.

[16] Le Gall, D. MPEG: A Video Compression Standard for Multimedia Applications. In *Communications of the ACM*, volume 34, pp. 46-58, 1991.

[17] Muntz, R.R., and J.C.S. Lui. Performance Analysis of Disk Arrays Under Failure. In *Proceedings of the 16th Conference on Very Large Data Bases*, pp. 162–173, 1990.

[18] Ng, S.W., and R.L. Mattson. Maintaining Good Performance in Disk Arrays During Failure via Uniform Parity Group Distribution. In *Proceedings of the 1st International Symposium on High-Performance Distributed Computing*, pp. 260–269, 1992.

[19] Ozden, B., R. Rastogi, and A. Silberschatz. Disk Striping in Video Server Environments. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, June 1996.

[20] Ozden, B., R. Rastogi, P. Shenoy, and A. Silberschatz. Fault-tolerant Architectures for Continuous Media Servers. In *Proceedings of the ACM Conference on Management of Data*, pp. 79–90, June 1996.

[21] Patterson, D.A., G.A. Gibson, and R.H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the ACM Conference on Management of Data*, pp. 109–116, 1988.

[22] Reddy, A.L.N., J. Chandy, and P. Banerjee. Design and Evaluation of Gracefully Degradable Disk Arrays. *Journal of Parallel and Distributed Computing*, **17**, pp. 28–40, 1993.

[23] Ruemmler, C., and J. Wilkes. An Introduction to Disk Drive Modeling. In *IEEE Computer*, pp.17-28, 1994.

[24] Schwabe, E.J., and I.M. Sutherland. Improved Parity-Declustered Layouts for Disk Arrays. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 76–84, 1994.

[25] Schwarz, T.J.E., J. Steinberg, and W.A. Burkhard. Permutation Development Data Layout (PDDL) Disk Array Declustering. In *Proceedings of the Fifth International Symposium on High-Performance Computer Architecture*, pp. 214–217, 1999, HPCA'99; Technical report CS98-584, CSE, UCSD, April 1998.

[26] Tobagi, F.A., J. Pang, R. Baird, and M. Gang. Streaming RAID: A disk storage system for video and audio files. In *Proceedings of ACM International Multimedia Conference*, pp. 393-400, 1993.

[27] Triantafillou, P., and C. Faloutsos. Overlay Striping and Optimal Parallel I/O in Modern Applications. http://www.ced.tuc.gr/Research/Reports/HERMES/Reports/htm, #TR8, 1997.

[28] Vin, H.M., P.J. Shenoy, and S. Rao. Efficient Failure Recovery in Multi-disk Multimedia Servers. In *Proceedings of the 25th International Symposium on Fault-Tolerant Computing*, pp. 12-21, 1995.