Staggered Striping in Multimedia Information Systems*

Steven Berson†
UCLA Computer Science Department
Richard Muntz
UCLA Computer Science Department

Shahram Ghandeharizadeh
USC Computer Science Department
Xiangyu Ju
USC Computer Science Department

Abstract

Multimedia information systems have emerged as an essential component of many application domains ranging from library information systems to entertainment technology. However, most implementations of these systems cannot support the continuous display of multimedia objects and suffer from frequent disruptions and delays termed *hiccups*. This is due to the low I/O bandwidth of the current disk technology, the high bandwidth requirement of multimedia objects, and the large size of these objects that almost always requires them to be disk resident. One approach to resolve this limitation is to decluster a multimedia object across multiple disk drives in order to employ the aggregate bandwidth of several disks to support the continuous retrieval (and display) of objects. This paper describes staggered striping as a novel technique to provide effective support for multiple users accessing the different objects in the database. Detailed simulations confirm the superiority of staggered striping.

1 Introduction

During the past decade, information technology has evolved to store and retrieve multimedia data (e.g., audio, video). Multimedia information systems utilize a variety of human senses to provide an effective means of conveying information. Already, these systems play a major role in educational applications, entertainment technology, and library information systems. A challenging task when implementing these systems is to support a continuous retrieval of an object at the bandwidth required by its media type [6, 10, 16]. This is challenging because certain media types, in particular video, require very high bandwidths. For example, the bandwidth required by NTSC[1] for "network-quality" video is about 45 megabits per second (mbps) [8]. Recommendation 601 of the International Radio Consultative Committee (CCIR) calls for a 216 mbps bandwidth for video objects. A video object based on the HDTV (High Definition TeleVision) quality images requires approximately 800 mbps bandwidth. Future demands for even high bandwidth are expected. These bandwidth requirements have to be compared with the typical 20 mbps bandwidth of a magnetic disk drive[2], which is not expected to increase significantly in the near future [11]. A standard technique to support a continuous display of multimedia objects is to sacrifice the quality of data using a lossy compression technique (results in some loss of data). While it is effective, there are applications that cannot tolerate loss of data (e.g., medical data, the video signals collected from space by NASA [2]). As an alternative, one may use a *lossless* compression technique (e.g., Huffman, Lempel Ziv, etc.). While a good estimate for reduction in size with these techniques is anywhere from a factor of 2 to 15, with lossy techniques it ranges from a factor of 10 to 500 [3]. In any case a range of bandwidth may be required in a system: from a fraction of disk bandwidth to several times the bandwidth of a single disk.

In order to simplify the discussion, we assume a hierarchical storage architecture that consists of a tertiary storage device accessible to a group of disk drives as our hardware platform. We assume that the stations used to display objects are independent of both the tertiary storage device and the disk drives. The database resides permanently on the tertiary storage device and its objects are materialized on the disk drives on demand (and deleted from the disk drives when the disk storage capacity is exhausted). In this study, we focus on the I/O bottleneck phenomena, and assume that the bandwidth of both the network and the network device driver exceeds the bandwidth requirement of an

*This research was supported in part by grants from AT&T/NCR/Teradata, Hewlett-Packard, IBM grant SJ92488, and NSF grants IRI-9110522, IRI-9203389, and CDA-9216321

†Now with USC Information Sciences Institute

[1] The US standard established by the National Television System Committee.

[2] The concepts described in this paper are applicable to other secondary storage devices.

object. This assumption is justified considering the current technological trends.

Assuming a fixed bandwidth for each disk (B_{Disk}) in the system, and a database consisting of objects that belong to a single media type with bandwidth requirement $B_{Display}$, we must utilize the aggregate bandwidth of at least $M = \lceil \frac{B_{Display}}{B_{Disk}} \rceil$ disk drives to support a continuous display of an object[3]. This can be achieved by a method we call *simple striping* as follows. First, the D disk drives in the system are organized into $R = \frac{D}{M}$ disk clusters. Next, each object in the database (say X) is organized as a sequence of n equi-sized **subobjects** (X_0,X_1,...,X_n). Each subobject X_i represents a contiguous portion of X. When X is materialized from the tertiary storage device, its subobjects are assigned to the clusters in a round-robin manner, starting with an available cluster. In a cluster, a subobject is declustered [12, 9, 4] into M pieces (termed **fragments**), with each fragment assigned to a different disk drive in the cluster. When the system displays X, it starts by using the cluster that contains X_0 (say C_i) to display the first portion of X. Next, it employs $C_{i+1 \ mod \ R}$ to display X_1. The system iterates over the clusters until X is displayed in its entirety, employing a single cluster at any given point in time.

To illustrate, assume that object X requires a 60 mbps bandwidth to support its continuous display ($B_{Display}(X)$=60 mbps). Moreover, assume that the system consists of 9 disk drives, each with a bandwidth of 20 mbps (B_{Disk}=20 mbps). Thus, we need the aggregate bandwidth of 3 ($M = \frac{60}{20}$) disk drives to support the continuous display of X. Figure 1 demonstrates how the simple striping technique organizes the subobjects of X. In this figures, the disk drives are partitioned into 3 clusters ($\lfloor \frac{D}{M} \rfloor$), each consisting of 3 (M) disk drives. Each subobject of X (say X_1) is declustered into 3 fragments (denoted $X_{1.0}$, $X_{1.1}$, $X_{1.2}$). A request to retrieve object X results in the system employing cluster 0 to display X_0. Subsequently, cluster 1 is employed to display X_1, etc. Hence, the display of X employs only a single cluster at each time interval, enabling the system to support three simultaneous displays.

The fragments of a subobject (say X_1) are constructed using a round-robin assignment of its blocks to each disk drive (see Figure 1), allowing the system to overlap the display of X_1 with its retrieval from the disk drives (multi-input pipelining [5, 6]). This minimizes the amount of memory required for buffering the data. However, in practice, some memory is needed per disk drive to eliminate hiccups that may arise due to disk seeks incurred when the system switches from one cluster to another (described further in Section 3).

[3]For simplicity our initial discussion assumes that the minimum bandwidth allocated to a request is B_{Disk}. In a later section we show how fractions of B_{Disk} can be allocated.

We further generalize the simple striping scheme (and name it *staggered striping*) to support a database that consists of a mix of multimedia objects, each with a different bandwidth requirement. The rest of this paper is organized as follows. In Section 2, we survey work related to this study. Section 3 presents both simple and staggered striping techniques. Section 4 presents simulation results that compare striping with an earlier technique termed virtual data replication. These results verify the superiority of staggered striping. Our conclusion and future research directions are contained in Section 5.

Parameter	Definition
$B_{Display}$	Bandwidth needed to display object
tfr	Transfer rate of a single disk drive
B_{disk}	Effective bandwidth of a single disk
$B_{Tertiary}$	Bandwidth of tertiary storage device
M_X	Degree of declustering for object X, $M_X = \lceil \frac{B_{Display}(X)}{B_{Disk}} \rceil$
D	Number of disk drives in the system
R	Number of disk clusters in the system
T_{switch}	The delay incurred when the system switches from one cluster to another
T_{sector}	Time required by a disk to transfer a sector worth of data to memory
$S(C_i)$	Service time of a disk cluster
f_size	Size of a fragment
k	Disks between $X_{i.0}$ & $X_{i+1.0}$

2 Related Work

Striping [14, 11] and declustering [12, 9, 4] are two popular techniques employed by both general purpose multi-disk I/O subsystems (e.g., RAID [11]) and parallel database management systems (e.g., Gamma [1], DBC/1012 [18]). Staggered striping extends these concepts to multimedia information systems, and is novel because its design enables either a multi-disk or a parallel system to guarantee a continuous retrieval of an object at the bandwidth required to support its display.

In [5, 6], we extended the concept of declustering to a parallel multimedia system based on the shared-nothing architecture [17]. In [7], we extended this architecture with a tertiary storage device and described: 1) *virtual data replication* as a mechanism to support multiple users, and 2) three dynamic techniques to manage the physical replica of objects. Virtual data replication partitions the D disk drives in the system into $R = \lfloor \frac{D}{M} \rfloor$ disk clusters, and declusters an object across the disk drives of a single cluster (i.e., assigns an object to a single cluster). To avoid the cluster that contains a frequently accessed object from becoming the bottleneck for the system, dynamic techniques were introduced to detect and replicate the frequently

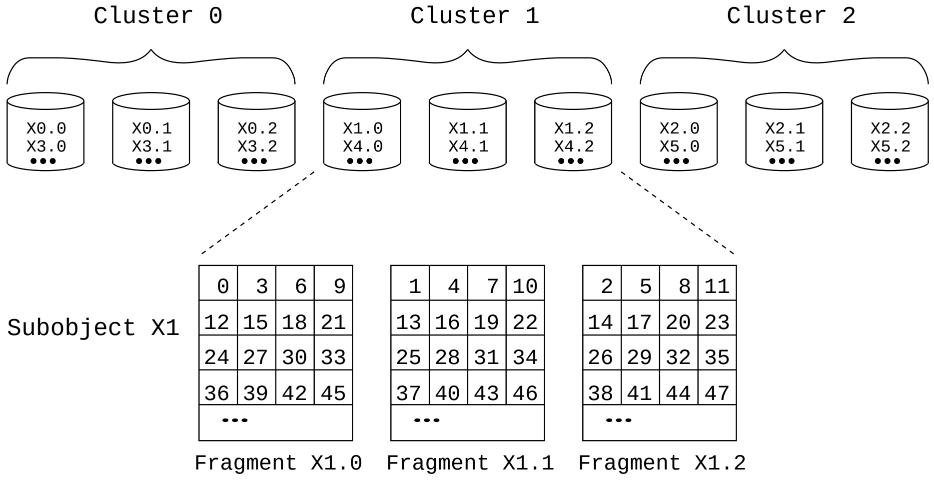

Figure 1: Fragment layout on disks

accessed objects across multiple clusters. While virtual data replication was a significant first step to provide support for a continuous display, the staggered striping method is a major improvement because it avoids formation of bottlenecks by striping an object across the clusters instead of replicating it. This enhances the overall performance of the system by enabling a larger number of objects to become disk resident. This observation is demonstrated further in Section 4.

Several terms will be used in the remainder of the paper and are defined here.

Degree of Declustering (M_X): the number of disk drives a subobject is declustered across; $M_X = \lceil \frac{B_{Display}(X)}{B_{Disk}} \rceil$.

Fragment: unit of data transfered from a single disk drive. Constructed by declustering a subobject (X_i) across M_X disk drives.

Subobject: a stripe of an object. It represents a contiguous portion of the object. Its size is defined as $M_X \times f\_size$.

Disk cluster: a group of disk drives that are accessed concurrently to retrieve a subobject (X_i) at a rate equivalent to $B_{Display}(X)$.

Stride (k): the distance (i.e., number of disk drives) between the first fragment of subobject X_i and the first fragment of X_{i+1}.

3 Two Striping Techniques

For the two striping techniques described in this section, we assume the following:

- each object has a constant bandwidth requirement.
- the display stations have a limited amount of memory. This means that the data has to be produced at approximately the same rate as its consumption rate at a display station.
- the network delivers the data to its destination both reliably and fast enough; consequently, it is eliminated from further consideration by this paper.
- the bandwidth requirements of the objects exceeds the bandwidth of both the tertiary storage device and a single disk drive. This assumption is relaxed in a later section.

We start by completing the description of the simple striping technique which was briefly introduced in Section 1. Next, we describe the limitations of this technique for a database that consists of a mix of media types, and present staggered striping as a solution to these limitations. Subsequently, we describe how staggered striping materializes objects from the tertiary storage device and supports rewind along with fast forward features. Other interesting features of the allocation of disk bandwidth to requests are also discussed.

3.1 Simple Striping

Assuming that all the objects in the database belong to a single media type with a fixed bandwidth requirement, Section 1 described the simple striping technique. Figure 1 illustrates the placement of object X with this technique. When the request service displays X, it employs a single cluster at each time interval. However, when the system switches from one cluster (C_0) to another (C_1), the disk drives that constitute C_1 incur the seek and latency times associated with repositioning their heads to the location containing the referenced fragments. To eliminate hiccups that might be attributed to this factor, simple striping computes the worst case delay (termed T_{switch}) for C_1 to reposition its heads and, relative to the consumption rate of a display station, produces the data such that a station is busy displaying T_{switch} worth of data when the switch takes place (see Figure 3).

Assuming some memory is allocated for each disk drive, the protocol for this paradigm is as follows. Upon activation of the disk drives in a cluster, each disk drive performs the following two steps:

1. Each disk repositions its head (this takes between 0 to T_{switch} seconds)
2. Each disk starts reading its fragment (it takes T_{sector} seconds to read each sector)
3. When all disks have read at least one sector, the synchronized transmitting of data to the display stations is started
4. Disks continue reading of the complete fragment overlapped with transmission to the display station

T_{switch} represents the maximum duration of the first step. T_{sector} is the time required to read a sector into memory. The minimum amount of required memory is a function of these two times and is defined as:

$$B_{disk} \times (T_{switch} + T_{sector}) \tag{1}$$

Simple striping splits time into fixed length intervals. A **time interval** is the time required for a disk drive to perform its four steps, and constitutes the service time of a cluster (denoted $S(C_i)$). The duration of a time interval is dependent on the physical characteristics of the secondary storage device (its seek and latency times, and transfer rate), and the size of the fragments.

To illustrate, recall the physical layout of X in Figure 1. Once a request references X, the system reads and displays X_0 (using C_0) during the first time interval. The display of the object starts at step 3 of this time interval. During the second time interval, the system reads and displays X_1 (using C_1). The display time of the cached data eclipses the seek and latency time incurred by C_1 (step 1), providing for a continuous retrieval of X (see Figure 3). This process is repeated until all subobjects of X are displayed.

Figure 2 further illustrates this object delivery scheme for three requests referencing three different disk resident objects: X, Y, and Z. This figure demonstrates the scheduling of clusters at midpoint of retrieval. Subobject X_{i+2} is the last subobject of X. Thus, disk cluster 0 does not read a subobject during both time intervals 3 and 6, while disk clusters 1 and 2 do not read subobjects during time intervals 4 and 5, respectively. If a request were to arrive before time interval 3 referencing an object whose first subobject resides on

time	CLUSTER 0	CLUSTER 1	CLUSTER 2
1	read Z(k+1)	read X(i+1)	read Y(j+1)
2	read Y(j+2)	read Z(k+2)	read X(i+2)
3	idle	read Y(j+3)	read Z(k+3)
4	read Z(k+4)	idle	read Y(j+4)
5	read Y(j+5)	read Z(k+5)	idle
6	idle	read Y(j+6)	read Z(k+6)
	⋮	⋮	⋮

Figure 2: Standard striping with 3 clusters

cluster 0, then these idle time intervals would be used to service the new request.

The fragment size is a parameter that has to be decided at system configuration time. The larger the chosen fragment size, the greater the effective disk bandwidth. This is because after the initial delay overhead to position the read heads (T_{switch}), there is little additional overhead no matter how much data is read. More formally if tfr is the transfer rate of a single disk, then the effective disk bandwidth B_{disk} can be computed as:

$$B_{disk} = tfr \times \frac{f\_size}{f\_size + (T_{switch} * tfr)}$$

There is also a tradeoff between the effective disk bandwidth and the time to initiate the display of an object. At the instant of arrival of a new request referencing an object X, the cluster containing X_0 might be busy servicing another request while the other clusters are idle. In this case, the request has to wait until the cluster holding X_0 becomes available. For example, if a system consists of R disks clusters and is almost completely utilized servicing $R-1$ requests, then in the *worst case* the latency time for a new request is $(R-1) * S(C_i)$. In summary, as one increases the size of a fragment, the display latency time increases (undesirable) while the effective disk bandwidth increases (desirable).

To illustrate, a typical 1.2 gigabyte disk drive [13] consists of 1635 cylinders, each with a capacity of 756000 bytes. Its peak transfer rate is 24.19 mbps. Its minimum, average, and maximum disk seek times are 4, 15, and 35 milliseconds respectively. Its average and maximum disk latency times are 8.33 and 16.83 milliseconds respectively. Typically a cylinder can be read with an overhead of one seek and one latency. Thus the time to read one cylinder is 250 milliseconds, while the highest overhead due to seeks and latency is 16.83+35 = 51.83 milliseconds. If the size of a subobject is chosen such that each of its fragments are one cylinder

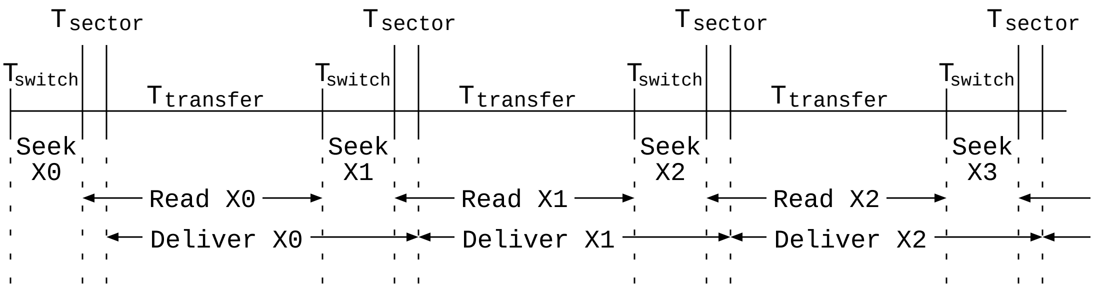

Figure 3: Continuous data transfer from disk clusters

in size (i.e., size(subobject) = M_X * $size(cylinder)$) then $S(C_i)$= 301.83 msec. Thus, on the average, 17.2 percentage of disk bandwidth is wasted due to the seek and latency times. If two consecutive cylinders are transfered, $S(C_i)$ = 555.83 and the wasted bandwidth will be only about 10 percent. In a typical system of 90 disks divided into 30 clusters of 3 disks, the worst case transfer initiation delay would be about 9 seconds in the case of 1 cylinder transfers and 16 seconds in the case of 2 cylinder transfers.

Without loss of generality, and in order to simplify the discussion, for the rest of this paper we assume that the size of a fragment for each object i is two cylinders (size(subobject)=M_i * 2 * $size(cylinder)$). This is a reasonable assumption because: 1) it wastes only about 10% of the disk bandwidth, and 2) the advantages of transfering more than 2 cylinder from each disk drive is marginal because of diminishing gains in effective disk bandwidth beyond 2 cylinders.

When the database consists of a mix of media types each with a different bandwidth requirement, the design of simple striping should be extended to minimize the percentage of wasted disk bandwidth. To illustrate, assume that the database consists of two video objects: Y and Z. The bandwidth requirement of Y is 120 mbps (M_Y = 6) and that of Z is 60 mbps (M_Z = 3). A naive approach to support these objects might be to construct the disk clusters based on the media type that has the highest bandwidth requirement, resulting in 6 disks per cluster (assuming B_{Disk} = 20 mbps). This would cause the system to employ a fraction of disks in a cluster when servicing a request that references object Z, sacrificing 50% of the available disk bandwidth. Staggered striping, described in the next section, is a superior alternative as it minimizes the percentage of disk bandwidth that is wasted.

3.2 Staggered Striping

Staggered striping is a generalization of simple striping. It constructs the disk clusters logically (instead of physically) and removes the constraint that the assignment of two consecutive subobjects of X (say X_i and X_{i+1}) be on non-overlapping disks. Instead, it assigns the subobjects such that the disk containing the first fragment of X_{i+1} (i.e., $X_{i+1.0}$) is k disks (modulo the total number of disks) apart from the disk drive that contains the first fragment of X_i (i.e., $X_{i.0}$). The distance between $X_{i.0}$ and $X_{i+1.0}$ is termed *stride*. If a database consists of a single media type (with a degree of declustering M_x) and D is a multiple of M_x then staggered striping can implement simple striping by setting the stride to equal to the degree of declustering of an object[4] ($k = M_X$).

Figure 4 illustrates both the logical and physical assignment of the subobjects of X with staggered striping (with stride k=1). As compared with simple striping, the display of X with staggered striping differs in one way: after each time interval, the disks employed by a request shift k to the right (instead of M_X with simple striping). When the database consists of a mix of media types, the objects of each media type are assigned to the disk drives independently but all with the same stride. Figure 5 demonstrates the assignment of objects Z, X, and Y with bandwidth requirement of 40, 60, and 80 mbps, respectively (M_Z=2, M_X=3, M_Y=4). The stride of each object is 1. In order to display object X, the system locates the M_X logically adjacent disk drives that contain its first subobject (disks 4, 5, and 6). If these disk drives are idle, they are employed during the first time interval to retrieve and display X_0. During the second time interval, the next M_X disk drives are employed by shifting k disks to the right.

With staggered striping, it is easy to accommodate objects of different display bandwidths with little loss of disk bandwidth. The degree of declustering of objects varies depending on their media type. However, the size of a fragment (i.e., unit of transfer from each disk drive) is the same for all objects, regardless of their media type. Consequently, the duration of a time interval is constant for all multimedia objects. For example, in Figure 5, the size of subobject Y_i is twice that of subobject Z_i because Y requires twice the bandwidth of object Z. However, their fragment size is identical because Y_i is declustered across twice as many disks as Z_i.

When displaying an object, staggered striping uses

[4] Virtual data replication can be implemented by $k = D$.

Subobject	Disk 0	1	2	3	4	5	6	7	8
0	X0.0	X0.1	X0.2						
1		X1.0	X1.1	X1.2					
2			X2.0	X2.1	X2.2				
3				X3.0	X3.1	X3.2			
4					X4.0	X4.1	X4.2		
5						X5.0	X5.1	X5.2	
6							X6.0	X6.1	X6.2
7	X7.2							X7.0	X7.1
8	X8.1	X8.2							X8.0
9	X9.0	X9.1	X9.2						
10		X10.0	X10.1	X10.2					
11			X11.0	X11.1	X11.2				
12				X12.0	X12.1	X12.2			
13					X13.0	X13.1	X13.2		
14						X14.0	X14.1	X14.2	
15							X15.0	X15.1	X15.2
⋮									

(a) Logical disk layout

Disk 0	1	2	3	4	5	6	7	8
X0.0	X0.1	X0.2	X1.2	X2.2	X3.2	X4.2	X5.2	X6.2
X7.2	X1.0	X1.1	X2.1	X3.1	X4.1	X5.1	X6.1	X7.1
X8.1	X8.2	X2.0	X3.0	X4.0	X5.0	X6.0	X7.0	X8.0
X9.0	X9.1	X9.2	X10.2	X11.2	X12.2	X13.2	X14.2	X15.2
	X10.0	X10.1	X11.1	X12.1	X13.1	X14.1	X15.1	
		X11.0	X12.0	X13.0	X14.0	X15.0		

(b) Physical disk layout

Figure 4: Staggered striping with 8 disks

Subobject	Disk 0	1	2	3	4	5	6	7	8	9	10	Disk 11
0	Y0.0	Y0.1	Y0.2	Y0.3	X0.0	X0.1	X0.2	Z0.0	Z0.1			
1		Y1.0	Y1.1	Y1.2	Y1.3	X1.0	X1.1	X1.2	Z1.0	Z1.1		
2			Y2.0	Y2.1	Y2.2	Y2.3	X2.0	X2.1	X2.2	Z2.0	Z2.1	
3				Y3.0	Y3.1	Y3.2	Y3.3	X3.0	X3.1	X3.2	Z3.0	Z3.1
4	Z4.1				Y4.0	Y4.1	Y4.2	Y4.3	X4.0	X4.1	X4.2	Z4.0
5	Z5.0	Z5.1				Y5.0	Y5.1	Y5.2	Y5.3	X5.0	X5.1	X5.2
6	X6.2	Z6.0	Z6.1				Y6.0	Y6.1	Y6.2	Y6.3	X6.0	X6.1
7	X7.1	X7.2	Z7.0	Z7.1				Y7.0	Y7.1	Y7.2	Y7.3	X7.0
8	X8.0	X8.1	X8.2	Z8.0	Z8.1				Y8.0	Y8.1	Y8.2	Y8.3
9	Y9.3	X9.0	X9.1	X9.2	Z9.0	Z9.1				Y9.0	Y9.1	Y9.2
10	Y10.2	Y10.3	X10.0	X10.1	X10.2	Z10.0	Z10.1				Y10.0	Y10.1
11	Y11.1	Y11.2	Y11.3	X11.0	X11.1	X11.2	Z11.0	Z11.1				Y11.0
12	Y12.0	Y12.1	Y12.2	Y12.3	X12.0	X12.1	X12.2	Z12.0	Z12.1			
⋮												

Figure 5: Staggered striping with 12 disks

the ideal number of disk drives per display during each time interval and thereby does not waste the bandwidth of the employed disks. However, it may cause a fraction of the disk drives to remain idle even though there are requests waiting to be serviced. This occurs when the idle disk drives are not adjacent due to the display of other objects. This limitation is termed *bandwidth fragmentation*. To illustrate, consider the assignment of X, Y, and Z in Figure 5. Assume an additional object (say W) with the same bandwidth requirement as object Y is disk resident ($M_W = 4$). Suppose the placement of W_0 starts with disk 4 ($W_{0.0}$ is stored on the same disk drive containing $X_{0.0}$). If the system is busy servicing three displays referencing objects X, Y, and Z, then there are three disk drives that are idle. Assume that a new request arrives referencing object W. It would have to wait because the number of idle disks (3) is less that its required number of disk drives (4). If the display of object X now completes then there would be a total of six disks available. However, the system is still unable to display object W because the available disk drives are not adjacent to each other (they are in groups of three separated by the display of Y and Z). The system cannot service displays requiring more than three disks until the display of either Y or Z completes.

It is interesting to note that the allocation of disk resources to satisfy requests now has fragmentation problems similar to those of dynamic memory allocation. In the remaining subsections, we discuss the bandwidth fragmentation problem as well as other design parameters and how to provide other functionality such as fast forward and rewind.

3.2.1 Fragmentation

Bandwidth fragmentation can be alleviated by careful scheduling of jobs, but cannot be completely eliminated. However, with additional memory for buffer space and additional network capacity, the bandwidth fragmentation problem can be solved. To accomplish this, assume that a fragment can be read from the disk into a buffer in one time interval and, in a subsequent time interval, the same processor node can concurrently transmit to the network both (a) the previously buffered fragment, and (b) a disk resident fragment (using the pipelining scheme outlined earlier). In this section, we show how to use buffers to utilize a set of disks that are not adjacent to deliver an object. We also show how to dynamically coalesce bandwidth fragmented disks as intervening busy disks become available.

Figure 6 illustrates an example of how our approach works. In the figure, the white regions indicate which disks were available for serving new requests while the shaded regions are disks busy serving other requests. A request arrives at time 0 for an object X with a degree of declustering equal to 2. Further the stride is 1 in this example and the initial subobject X0 is stored on disks 0 and 1. There are 2 free disks, but they are not consecutive; there are 2 intervening busy disks. Disk 1 is free and is in position to read fragment X0.1, but disk 6 which is also free will not be in position to read fragment X0.0 until time interval 2. In order to support bandwidth fragmented delivery of object X, disk 1 can keep fragment X0.1 in memory until time 2 when it can be delivered along with fragment X0.0. Thus at time 2, fragment X0.0 is pipelined directly from disk 0 to the network, while node 1 transmits fragment X0.1 from its buffers (while disk 1 is concurrently servicing another request). Similarly disk 2 reads fragment X1.1 at time 1, and buffers it until time interval 3 when both X1.0 and X1.1 can be delivered.

Figure 6 also illustrates how bandwidth fragmented requests can be dynamically coalesced. Suppose at time interval 5, the 2 intervening disks have completed their service and become free. At that point, the bandwidth fragmented request can be completely coalesced so that the disks supporting the transmission of object X are adjacent (depending on how many disks become free, a bandwidth fragmented request may be only partially coalesced). By the start of time interval 5, fragments X3.1 and X4.1 are already buffered, and have to be delivered before reading recommences. During time intervals 5 and 6, fragments X3.1 and X4.1 are delivered from buffers while fragments X3.0 and X4.0 are delivered directly from disk. Starting at time 7, the coalescing has been completed and the 2 consecutive disks pipeline the fragments directly from the disk to the network.

3.2.2 Stride

The choice of a value for the stride (k) must be determined at system configuration time. It may vary in value from 1 to D since a value (say i) greater than D is equivalent to i modulo D. The choice of k and D is important as particular combinations of values for k and D can result in very skewed load on the disks, both storage capacity and bandwidth capacity.

We illustrate the above points by first considering the two extreme values for k (1 and D). Assume a system with 10 disk drives (D=10) and a large object X consisting of hundreds of cylinders worth of data. Assume that the degree of declustering for each subobject of X is 4 (M_X=4). If k=1, then the number of unique disks employed is 10 (D), 4 at a time and for $S(C_i)$ duration before moving to a new set of 4 disks. If k=D, then all subobjects of X are assigned to the same disk drive. Hence, the number of unique disks employed to display X is M_X, each for the entire display time of X ($\frac{size(X)}{B_{Display}(X)}$). Assume requests for objects X and Y arrive to both systems (one system with k=1 and the other with k=D) and where $X_{0.0}$ and $Y_{0.0}$ reside on the same disk. Assume that the requests arrive in the following order: X followed by Y. In this case, with k=1, Y observes a delay equivalent to $S(C_i)$ (typically less than a second). With k=D, Y observes a delay equivalent to the display time of X which is very much larger and generally unacceptable. To prevent data skew, the subobject size of every object in the system must be a multiple of the GCD (Greatest Common Divisor) of D, the total number of disks, and k, the stride. In particular, a stride of 1 guarantees no data skew. Similarly, any choice of D and k such that D and k are relatively prime guarantees no data skew.

Note that with k=D, the display of each object is very efficient because the system can cluster the different subobjects of X on adjacent cylinders in order to minimize the percentage of disk bandwidth that is wasted (saves of less than 10% of the disk bandwidth). Section 4 demonstrates the tradeoffs between these two alternative values for k. The results show that saving of less than 10% of the disk bandwidth as compared to the high probability of collisions is not beneficial.

When k ranges in value between 1 and D, the size of an object X determines the number of disk drives employed to display X because the size of each fragment is fixed (a cylinder in our case). To illustrate, assume D=100 and an object X consist of 100 cylinders (M_X = 4). With $k = M_X$ (i.e., simple striping), X is spread across all the D disk drives. However, with k=1, X is spread across 28 disk drives. In this case, the expected display latency with $k = 1$ is higher than with $k = M_X$.

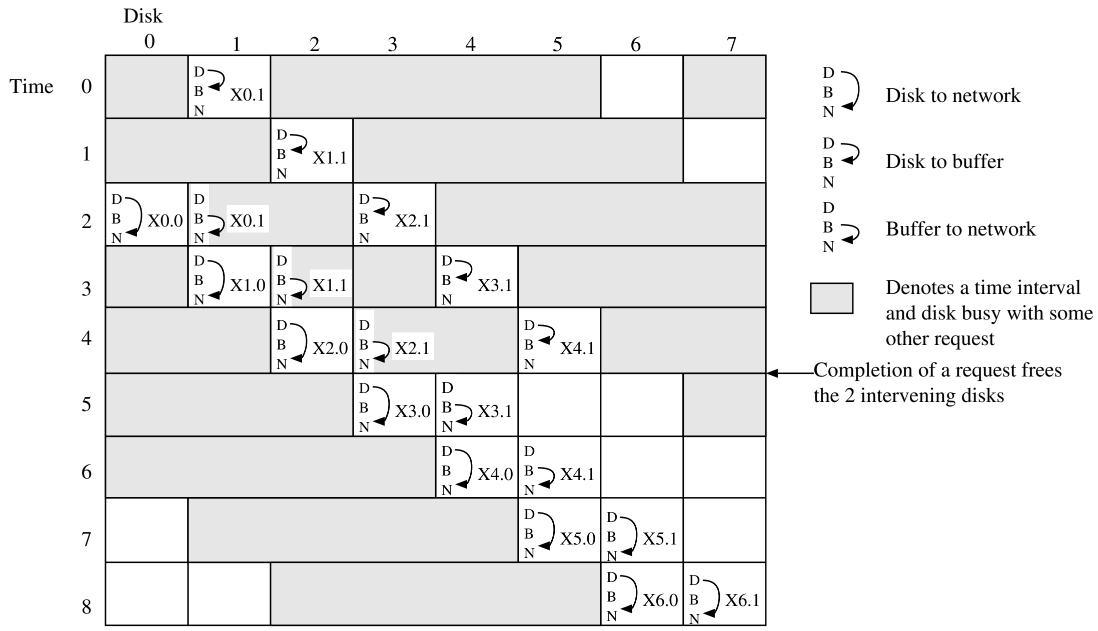

Figure 6: Utilizing bandwidth fragmented disks

3.2.3 Low Bandwidth Objects

There are object types for which $B_{Display} < B_{Disk}$. These might include certain types of audio or slow scan video. Similarly, there are objects whose bandwidth requirement is not an exact multiple of B_{Disk}. In these cases there will be wasted disk bandwidth due to the request to use an integral number of disks. For example, an object requiring 30 mbps when $B_{Disk} = 20$ would waste 25 percent of the bandwidth of the two disks used per interval. Staggered striping can be used to more efficiently support these low bandwidth objects at a cost of some additional buffer space. To efficiently use disk space and disk bandwidth, we propose that subobjects of 2 or more low bandwidth objects be read and delivered in a single time interval. Consider two subobjects X_i and Y_j, each of which has $B_{Display} = 1/2\ B_{Disk}$, and which are to be read during the same time interval. The data in subobject X_i needs to be delivered during the entire time interval including the time when Y_j is being read. An additional buffer can be used to store part of X_i while subobject Y_j is being read. Similarly, part of Y_j needs to be buffered while X_{i+1} is being read during the next time interval. Note that we are again assuming that a node can concurrently transmit from a main memory buffer and from a disk using the pipelining scheme.

Figure 7 illustrates how this is accomplished. During the first half of the first time interval subobject X_0 is read, and the first half of X_0 (represented as X0a) is transmitted using pipelining. The second half of subobject X_0, X0b, is buffered for transmission during the second half of the time interval. In the second half of the first time interval, subobject Y_0 is read and both Y0a and X0b (from the buffer) are transmitted. Y0b now needs to be buffered for transmission during the first half of the second time interval. This process continues until the objects complete transmission.

This scheme effectively divides each disk into two logical disks of approximately one half the bandwidth of the original disk. This scheme can also be beneficial in reducing the overhead due to use of an integral number of disks. In effect, the request is now that an integral number of logical disks be allocated to a request. For example, an object that has $B_{Display} = 3/2\ B_{Disk}$ can be exactly accommodated with no loss due to rounding up the number of disks. In general, the waste due to rounding is reduced.

3.2.4 Materializing Objects from Tertiary Store

This study assumes that the bandwidth of tertiary store is lower than the bandwidth required to display an object. When materializing an object X, the tertiary device cannot produce an entire subobject during each time interval to write to a disk cluster (it produces $\frac{B_{Tertiary}}{B_{Display}(X)} \times size(subobject)$). If an object is stored in a sequential manner on the tertiary store, then the bandwidth of both the disk and the tertiary store will be wasted. This is due to the layout mismatch between the organization of data on tertiary store and that on the disk drives: the organization of an object on the disk drives is not sequential. When materializing object X, this mismatch will cause the system to write $\frac{B_{Tertiary}}{B_{Display}(X)}$ of subobject X_0 to M_X idle disk drives in

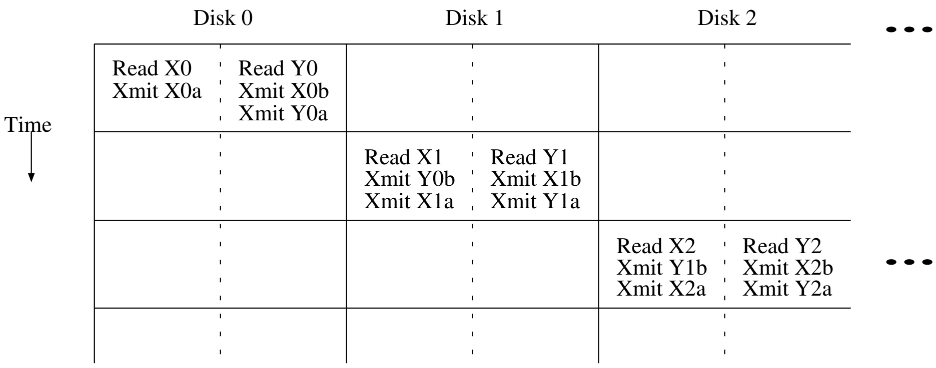

Figure 7: Staggered striping with low bandwidth objects

the first time interval. In the second time interval, the system moves k disks to the right, requiring the tertiary device to produce $\frac{B_{Tertiary}}{B_{Display}(X)}$ of X_1. This would require the tertiary store to reposition its disk head. This reposition time is typically very high for tertiary storage devices and may exceed the duration of a time interval. In this case, the system would be required to materialize a different subobject every other time interval with the tertiary spending a major fraction of its time repositioning its head (wasteful work) instead of producing data (useful work).

One approach to resolve the mismatch between the tertiary store and the disks is to write the data on the tape in the same order as it is expected to be delivered to the disks. To illustrate, assume an object X with bandwidth requirement of 80 mbps. If the bandwidth of the tertiary store is 40 mpbs and the bandwidth of each disk drive is 20 mbps, then the fragments of X could be stored on the tertiary based on the organization of the fragments across the disk drives: $X_{0.0}$, $X_{0.1}$, $X_{1.0}$, $X_{1.1}$, $X_{2.0}$, $X_{2.1}$, etc. The materialization of object X would employ two disk drives in each time cycle. During the first time cycle it writes the first two fragments of the subobject X_0 ($X_{0.0}$, and $X_{0.1}$), while during the second time cycle, it moves k disks to the right and materialized the first two fragments of X_1 ($X_{1.0}$, and $X_{1.1}$) without repositioning its head. This process is repeated until X is materialized in its entirety. A limitation of this technique is as follows. If the bandwidth of a disk relative to the tertiary store were to change, then all the data on the tertiary would have to be re-recorded.

3.2.5 Rewind and Fast-Forward Features

Thus far, we have described the delivery of an object at a constant bandwidth ($B_{Display}$). Other features such as rewind, fast-forward, and fast-forward with scan may also be desirable. Rewinding or fast-forwarding to any spot in the data can be accomplished by waiting for the set of disks servicing the request to advance to the appropriate position. Alternatively, if the appropriate number of disks that contain the referenced location in an object are idle, then the system can employ them to service the request immediately. Even though there is a bandwidth/layout mismatch, the user will not observe hiccups since the system displays no data.

Fast-forward with scanning is more complicated because there is a bandwidth/layout mismatch and images need to be displayed. This is because the data is laid out for normal speed delivery, but there is an occasional demand for fast delivery of only a fraction of the data. For example, typical fast forward scans of VHS video display approximately every sixteenth frame. In order to provide this functionality, our approach stores a *fast forward replica* object for each object in the system. This replica would be a small fraction of the size of a subobject. When fast forward scan is invoked, the system uses this replica to support the display instead of the normal speed object data. When a request for the fast forward replica arrives, disks at (or close to) the correct point in the fast forward replica can start displaying the replica instead of the normal speed object. If excess bandwidth is not available, the system may incur a transfer initiation delay when switching to the fast forward replica (and back to the normal speed replica). This should not be a serious problem because exact synchronous delivery is not expected when switching between normal speed delivery and fast forward scanning.

4 Performance Evaluation

We have implemented staggered striping using the CSIM [15] simulation language. In this section we compare simple striping (special case of staggered striping, implemented by setting $k{=}M_X$) with an earlier introduced technique termed virtual data replication [7] (described in Section 2). We start by describing the simulation model employed for both strategies and their parameters. Next, we summarize the performance results obtained. These results clearly demonstrate the superiority of staggered striping.

Disk Parameters	
Storage Capacity	4.54 gigabyte
Number of Cylinders	3000
Storage Capacity of a Cylinder	1.512 megabyte
Average Latency Time	8.33 msec
Maximum Latency Time	16.83 msec
B_{Disk}	20 mbps
Minimum Seek Time	4 msec
Maximum Seek Time	35 msec
Average Seek Time	15 msec
Database Parameters	
Number of objects	2000
Number of Subobjects/object	3000
$B_{Display}$	100 mbps
Degree of Declustering (M)	5
System Parameters	
Number of Disks (D)	1000
Number of Tertiary Devices	1
Stride (k)	5
$B_{Tertiary}$	40 mbps

Table 1: Important simulation parameters

# Display Stations	*Distribution of Access*		
	10	20	43.5
16	5.10%	2.15%	114.75%
64	11.06%	131.86%	508.79%
128	52.67%	350.73%	469.94%
256	126.10%	602.49%	413.10%

Table 2: Percentage improvement in throughput (number of displays per hour) with simple striping as compared to virtual data replication.

4.1 Simulation Model

The simulation model consists of four main modules: the Display Station, Centralized Scheduler, Disk, and Tertiary Storage. Each Display Station consists of a terminal that generates the workload of the system. Each Disk provides a 20 megabit per second bandwidth (B_{Disk}=20 mbps) and consists of 3000 cylinders, each with a capacity of 1.512 megabytes. Thus, the capacity of each disk drive is 4.5 gigabyte[5]. The Tertiary Storage device provides a 40 mbps bandwidth. The Centralized Scheduler implements an Object Manager, a Disk Manager, and a Tertiary Manager. The Object Manager maintains the availability of different objects on the disk drives. Once the storage capacity of the disk drives is exhausted and a request references an object that is tertiary resident, it implements a replacement policy that removes the least frequently accessed object with the referenced object. The Disk Manager keeps track of the different disks and their status (busy or idle) for each time interval. The simulation model was configured to consist of 1000 disks. The Tertiary Manager maintains a queue of requests waiting to be serviced by the tertiary storage device.

The database consists of a single media type. The bandwidth requirement of this media type is 100 megabits per second ($B_{Display}$ = 100 mbps, M=5). All objects are equi-sized and each consists of 3000 subobjects. The size of each fragment is equivalent to the size of a cylinder ($size(subobject) = 5 \times size(cylinder)$). Hence, the display time of each object is 1814 seconds (30 minutes and 14 seconds). The size of the database is approximately ten times the available disk storage capacity.

Both simple striping and virtual data replication construct 200 disk clusters ($\frac{D}{M}$). Virtual data replication assigns an object to a single disk cluster. With the chosen simulation parameters, at most one object can be assigned to a cluster (the storage capacity of the cluster is exhausted by one object). The simulation was configured to detect and replicate the frequently accessed objects in order to avoid the formation of bottlenecks[6]. Simple striping also allows 200 objects to become disk resident, however, it stripes each object over all the available clusters.

A user employs a display station to request an object. We assume that a display station can display only one object at a time. In our experiments, we varied the number of display stations from 1 to 256 in order to vary the system load. We assumed a closed system where once a display station issues a request, it does not issue another until the first one is serviced. We also assume a zero think time between the requests. This parameter was chosen in order to stress the system and compare striping with virtual data replication in the worst case scenario.

We varied the distribution of access to the objects from uniform to skewed in order to analyze the performance of the different techniques with various working set sizes. In each case the object reference probabilities were modeled by a (truncated) geometric distribution. The mean was varied to model different reference patterns from highly skewed to more uniform. We analyzed three different values for the mean: 10, 20, and 43.5 (resulting in approximately 100, 200, and 400 unique objects to be referenced, respectively).

4.2 Performance Results

[5] At the time of this writing, 4 gigabyte disk drives are commercially available (e.g. from Digital) at a cost of \$0.82/megabyte of storage.

[6] It was configured with the Minimum Response Time (MRT) state transition diagram [7].

Figure 8: Obtained results

Figure 8 presents the performance of both virtual data replication and simple striping for various system loads, with the results for each distribution appearing in a different graph. In general, as the distribution of access to the objects becomes more uniform, the throughput of the system with both techniques decreases. This is because the probability of a request referencing an object that is not disk resident (and incurring the overhead of materializing the object from the tertiary storage device) increases.

For a low number of display stations (one or two), both techniques provide approximately the same throughput. However, as the system load increases, simple striping outperforms virtual data replication by a wider margin. Table 2 shows the percentage improvement in throughput with simple striping. With a skewed distribution of access to the objects (see Figure 8.a), simple striping outperforms virtual data replication because by striping it prevents a frequently accessed object from causing a cluster to become the bottleneck for the system (virtual data replication detects and resolves bottlenecks by replicating the frequently accessed objects). When the distribution of access becomes more uniform, simple striping continues to provide a superior performance because it allows a larger number of unique objects to become disk resident (by replicating the frequently accessed objects, virtual data replication reduces the number of unique objects that are disk resident).

As the distribution of access becomes more uniform (Figure 8.b), the tertiary storage device starts to become the bottleneck and determines the overall processing capability of the system. This reduces the percentage improvement observed with simple striping.

5 Conclusion and Future Research

This paper describes staggered striping as a novel technique for the placement of data in parallel multimedia information systems. Its design enables the system to guarantee a continuous retrieval of an object at the bandwidth required to support its display. Moreover, it provides effective support for multiple users and a database consisting of a mix of media types where each media has a different bandwidth requirements. The evaluation section of this paper demonstrates the superiority of staggered striping as compared to an earlier introduced technique termed virtual data replication.

We believe that staggered striping is a promising area of research and intend to extend this study in several ways. First, we intend to extend its design to avoid the use of maximum seek and latency times. This requires further simulation and analytical studies to investigate the role of larger buffers to increase our effective bandwidth. Second, we plan to investigate scheduling policies for multiple users to address questions such as: 1) Should a request with a small display time have priority over other requests? 2) Should a requests referencing disk resident objects have a higher priority than those requests referencing tertiary resident objects? 3) How can multiple requests be scheduled fairly in order to avoid starvation? Some of these questions have analogues in the area of memory management, however, it is not clear if the solutions are identical. And finally, we intend to investigate an implementation of staggered striping. In particular, the limitations imposed by the bandwidth of a network needs further investigation (e.g., the aggregate bandwidth of a system consisting of 100 disks is one gigabit per second).

References

[1] DeWitt, D., Ghandeharizadeh, S., Schnei-

der, D., Bricker, A., Hsiao, H., and Rasmussen, R. The Gamma database machine project. *IEEE Transactions on Knowledge and Data Engineering 1*, 2 (March 1990).

[2] Dozier, J. Access to data in NASA's Earth observing system (Keynote Address). In *Proceedings of SIGMOD '92.*

[3] Fox, E. A. Advances in Interactive Digital Multimedia Sytems. *IEEE Computer* (October 1991), 9–21.

[4] Ghandeharizadeh, S., and DeWitt, D. A multiuser performance analysis of alternative declustering strategies. In *Proceedings of Data Engineering '90.*

[5] Ghandeharizadeh, S., and Ramos, L. Continuous retrieval of multimedia data using parallelism. *IEEE Transactions on Knowledge and Data Engineering 1*, 2 (August 1993).

[6] Ghandeharizadeh, S., Ramos, L., Asad, Z., and Qureshi, W. Object Placement in Parallel Hypermedia Systems. In *Proceedings of VLDB '91.*

[7] Ghandeharizadeh, S., and Shahabi, C. Management of Physical Replicas in Parallel Multimedia Information Systems. In *Proceedings of the 1993 Foundations of Data Organization and Algorithms (FODO) Conference* (October 1993).

[8] Haskell, B. International standards activities in image data compression. In *Proceedings of Scientific Data Compression Workshop* (1989), pp. 439–449. NASA conference Pub 3025, NASA Office of Management, Scientific and technical information division.

[9] Livny, M., Khoshafian, S., and Boral, H. Multi-disk management algorithms. In *Proceedings of SIGMETRICS '87.*

[10] Maier, D., Walpole, J., and Staehli, R. Storage System Architectures for Continuous Media Data. In *Proceedings of the 1993 Foundations of Data Organization andAlgorithms (FODO) Conference* (October 1993).

[11] Patterson, D., Gibson, G., and Katz, R. A case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of SIGMOD '88.*

[12] Ries, D., and Epstein, R. Evaluation of distribution criteria for distributed database systems. UCB/ERL Technical Report M78/22, UC Berkeley, May 1978.

[13] Sabre Inc. *IMPRIMIS Sabre Eight Inch Module Drive User's Manual*, 1990.

[14] Salem, K., and Garcia-Molina, H. Disk striping. In *Proceedings of Data Engineering '86.*

[15] Schwetman, H. CSIM: A C-Based Process-Oriented Simulation Language. Technical Report PP-080-85, Microelectronics and Computer Technology Corporation, 1985.

[16] Stonebraker, M., Agrawal, R., Dayal, U., Neuhold, E., and Reuter, A. DBMS Research at a Crossroads: The Vienna Upda te. In *Proceedings of VLDB '93.*

[17] Stonebraker, M. R. The case for Shared-Nothing. In *Proceedings of Data Engineering '86.*

[18] Teradata. *DBC/1012 database computer system manual release 2.0*, 1985.