DOI : 10.3745/JIPS.2008.4.1.001

Utilizing Fragmented Bandwidth in a Staggered Striping Multimedia System

Wen-Chi Hou*, Yang Pan*, and Dunren Che*

Abstract: In this paper, we discuss the use of fragmented bandwidth to improve the performance of staggered striping in a multimedia system. It is observed that potential disruptions can occur when non-consecutive idle disks are used for displaying multimedia objects. We have identified useful retrieval patterns and shown that with proper selections of fragmented disks and a simple buffering scheme, disruptions can be easily eliminated.

Keywords: *Disk Striping, Multimedia System, Bandwidth*

1. Introduction

Recent advances in networking and multimedia technology have made it possible to provide concurrent services, such as on-line shopping, on-line auction, etc., to hundreds or even thousands of customers. Unlike conventional video rental stores, which can serve only a very limited number of viewers simultaneously, a multimedia-on-demand (MOD) system is intended to provide on-line, concurrent, convenient, and faster service to a large number of viewers. In addition to the entertainment business, an MOD system can also be applied to other areas, such as education and library information systems. For example, with an MOD system, educational videos can be displayed and viewed by students over cable TV systems continuously and reliably. Besides videos, new multimedia applications, such as graphical modeling of nearby 3D objects in architectural buildings, urban city models, scientific visualization, etc., [1] can also benefit from the technology of MOD systems.

In an MOD system, hundreds or even thousands of multimedia objects are stored on a storage server and are ready to be played out upon requests. Multimedia storage servers are generally connected to clients via high-speed networks so that continuous media such as video and audio can be played on users' stations. Viewers can also access multimedia objects simultaneously on the server. The goal of an MOD server is to serve as many viewers concurrently as possible. A typical MOD system is shown in Figure 1.1.

A tertiary storage architecture is usually employed on a large-scale MOD server [2], in which the server combines highly cost-effective tertiary storage devices, such as tapes and optical disks, with high-performance, but perhaps more expensive, magnetic disks. Magnetic disks are used to store frequently accessed objects (e.g., high-demand objects), while tertiary storage devices are used for the remaining objects.

Manuscript received July 6, 2007; first revision December 8, 2007; second revision February 26, 2008; accepted February 27, 2008.
Corresponding Author: Dunren Che

* Dept. of Computer Science, Southern Illinois University, Carbondale, Illinois, USA ({hou, ypan, dche}@cs.siu.edu)

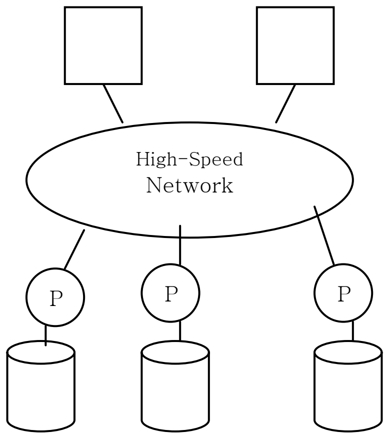


Fig. 1.1. Architecture of an MOD system.

Multimedia data can have varying and sometimes very high bandwidth requirements. For example, a 27 MBps (i.e., megabytes per second) bandwidth is required by NTSC for "network-quality" video [2], while a bandwidth of approximately 81 MBps is required for an HDTV-quality image [2]. Although the bandwidth of a modern magnetic disk drive can reach approximately 20 MBps [3], multimedia data are expected to have much higher bandwidths in the future [4].

Compression techniques [5], such as MPEG-1 and MPEG-2, have been applied successfully to multimedia objects like movie videos, to reduce the bandwidth requirements. For example, MPEG-2 movie video objects require only a bandwidth of 0.42 MBps [2]. However, these compression techniques generally are lossy and can result in loss of important information. Thus, they may not be acceptable to objects like computer programs, scientific data, medical data, etc., which require high accuracy. For example, a slight change in the structure and formation of cells in biology, crystals in material science, phenomena in space, can lead to severe errors or loss of important observations [4, 6, 7]. An MOD system is expected to cope with compressed as well as non-compressed high-bandwidth. Thus, one of the major challenges in building an MOD system is to support varying and perhaps high-bandwidth data with low-bandwidth disk drives.

Although individual disks may have large enough storage capability [8] and speed to display some compressed video objects, a bottleneck can easily arise when there are simultaneous requests for videos stored on the same disk. Since a bottleneck can result in a significant delay in display, it may not be desirable to store an entire video object on a single disk. Thus, how to place the data on disks to balance the workload and serve more requests without much delay has also emerged as an important issue.

RAID (Redundant Array of Inexpensive Disks) is often used as a high-bandwidth secondary storage device by allowing simultaneous access to an array of disks. However, when used in a multimedia system, its lack of control over the placement of data [6] may pose some problems. For example, if the data retrieved from RAID is faster than display, it may overflow the memory buffer of the display station. In addition, it is hard to display several objects simultaneously with one disk controller.

Replication and striping techniques [4, 6, 9, 10, 11] were studied by researchers as alternatives. In replication approaches [6, 9], objects are replicated to support simultaneous displays of the same video objects. However, the bottleneck problem may still exist when the number of requests exceeds the number of replicas of objects. Moreover, objects can take up a lot of space due to the replication.

Disk striping [4, 12, 13] effectively overcomes the bottleneck problem by declustering objects across multiple disks and using the aggregated bandwidth of disks to serve simultaneous requests. In addition, servers in the striping approach can store more video objects than in the replication methods.

Staggered striping [4] is a variant of the striping. It is a promising technique due its flexibility and capability in storing and displaying multimedia objects of variable bandwidths. However, as requests arrive and are served, the system could be left with dispersed variable-sized "holes" of idle disks, termed fragmented disks [4], during display. This situation is similar to the memory fragmentation in the traditional operating systems. When there is no single hole that is large enough to meet the bandwidth requirement of the requested object, the request can not be served, even though the total number of idle disks in the system is enough. Berson et al. [4] pointed out potential bandwidth waste due to this bandwidth fragmentation. They have shown an example of how portions of subobjects, which are contiguous portions of an object, can be retrieved using fragmented disks and buffered until the entire subobjects are put together. However, there were no discussions and analysis on how fragmented bandwidth can be utilized in general. In fact, we have found that the use of fragmented bandwidth is much more complex than the example they showed. Specifically, disruptions, termed as "hiccups" [14, 4], can occur not only at the beginning of the display but also in the middle of display. Moreover, the buffer space needed to store incomplete pieces of subobjects can be very large if we do not take advantage of the properties of retrieval patterns. To the best of our knowledge, there have not been any in-depth studies on the utilization of fragmented bandwidth in the staggered striping. In this paper, we propose a method to solve the disruption problems when non-consecutive idle disks are used for display. We identified repeated patterns of retrieval that can be utilized to smooth the display along. With proper selection of idle disks and a simple buffering scheme, fragmented bandwidth can be fully utilized and disruptions can be eliminated.

Ghandeharizadeh, et al. [15] also discuss placement of data on disks, but its underlying assumptions are different from ours. They assume a subobject (of the object) is placed on a single disk while we assume a subobject (of an object) is striped across a number of disks. Moreover, we allow objects to require different bandwidth, which causes the fragmented bandwidth problem on which our study focuses.

The rest of the paper is organized as follows. A review of related work is presented in Section 2. In Section 3, we demonstrate the disruption problems in the staggered striping and point out the existence of potential disruption patterns. In Section 4, a formal analysis of the disruption patterns is conducted and a solution to the disruption problems is presented. Section 5 is the conclusion.

2. Related Work

In this section, we review two main approaches in storage and retrieval of video data from disks and discuss their problems.

2.1 Terminology

A multimedia object here, for simplicity, is referred to as an object. A subobject represents a contiguous portion of an object and is intended to be a unit of transfer between the server and the viewer's station. In generally, a cluster of disk drives is accessed simultaneously to fulfill the bandwidth requirement of objects. The degree of declustering of an object X, denoted M(X), is defined to be the number of disk drives, across which a subobject of X is declustered. Let $B_{Display}(X)$ be the bandwidth required for displaying an object X, and B_{Disk} be the effective bandwidth of a disk. Then, $M(X) = B_{Display}(X)/B_{disk.}$ Let D be the number of disks in the system, which are numbered from 0 to D-1. Let R be the number of clusters into which the disk drives are partitioned. Then, R = D/M(X).

For simplicity, in this section, we shall assume all objects have the same bandwidth requirement, unless otherwise stated.

2.2 Replication Methods

The replication approach [6] partitions the disk drives into R clusters and assigns a replica of an object to at least one of the clusters. Frequently accessed objects can be replicated in as many clusters as needed. In the example shown in Fig. 2.1, the disk drives are partitioned into 2 clusters, each of which has 3 disk drives. An object X, whose M(X) = 3, is stored in one cluster and replicated in another cluster. The object X is partitioned into blocks X_0, X_1, …., and X_5, and these data blocks are assigned to disk drives in a round-robin manner within a cluster. The system can display the first portion of this object while continuously retrieving the next portion (i.e., X_3, X_4, and X_5) from disk drives of the same cluster (i.e., pipelining). The replication method enables the system to support R simultaneous displays of the objects.

Chan's rotational mirrored declustering (RMD) [9], as shown in Fig. 2.2, is a variant of the replication approach, which uses a set of disk arrays to store objects. Given a set of rn disks, numbered 0, 1, …. , rn – 1, where n is the number of disks in a disk array, and r is the number of the disk arrays, the i^{th} partition of the j^{th} replica of an object is placed on the disk *d*, according to the following formula:

$$d = (i + \lfloor i/n \rfloor * (j - 1)) \bmod n + (j - 1) * n$$

An example of the data placement in RMD is shown in Figure 2.3. An object X, consisting of X_0, …, X_5, is first assigned to a disk array, and then a replica is properly rotated and declustered on another disk array. A predetermined replication threshold is used to determine the number of replicas. This method allows replicas and disks to be added online and was found effective in dealing with load balancing problem. The method is also fault tolerant [9].

In general, a replication approach can support a higher number of simultaneous displays because of the multiple copies of data. It is also more fault tolerant because the traffic to a failed disk can be directed to other disks containing the replica. However, all replication approaches require additional disk drives to store multiple copies of the objects, and the cost can be very high.

2.3 Striping Methods

The basic idea of the striping approach is to stripe the objects across clusters of disks so that the aggregate bandwidth of a cluster can match the bandwidth requirement of objects. In this section, we discuss two methods in this category.

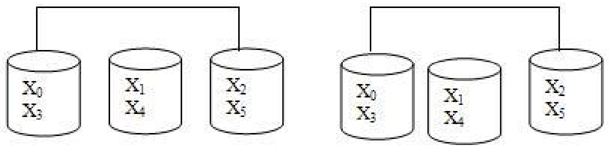


Fig. 2.1. Data Replication

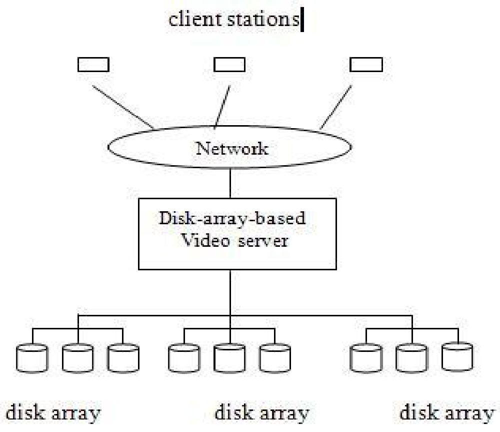
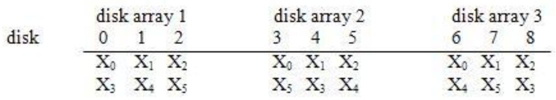


Fig. 2.2. A Disk-array-based Video Server



	disk array 1			disk array 2			disk array 3		
disk	0	1	2	3	4	5	6	7	8
	X_0	X_1	X_2	X_0	X_1	X_2	X_0	X_1	X_2
	X_3	X_4	X_5	X_5	X_3	X_4	X_4	X_5	X_3

Fig. 2.3. Data Placement in RMD

2.3.1 Simple Striping

Figure 2.4 illustrates the placement of an object X on disks using the simple striping technique [4]. Assume that a bandwidth of 60 MBps (i.e., $B_{Display}(X)$= 60 MBps) is required for displaying X and there are 9 disk drives (i.e., D = 9) in the system, each with 20 MBps transfer rate (i.e., B_{Disk} = 20 MBps). Since the aggregated bandwidth of 3 disk drives (i.e., M(X) = $B_{Display}(X)/B_{Disk}$) is needed to match the bandwidth of the object, disk drives are organized as 3 clusters (R = D/M(X)). The object X is divided into a sequence of equi-sized contiguous subobjects X_0, X_1, …, X_n. These subobjects are assigned to clusters from the first available one, say cluster 0, in a round-robin manner [4, 16, 17], as shown in the upper part of Figure 2.4.

A subobject, which is made of 12 data blocks in the example, is further striped across its cluster in a round-robin manner. Figure 2.4 shows how X_1, consisting of data blocks 12 to 23, is interleaved among the disks of cluster 1 [18]. The set of blocks assigned to a disk, though not consecutive in number, constitutes a fragment. That is, data blocks 12, 15, and 18 of X_1, and 21 of X_1 constitute the fragment $X_{1.0}$ and data blocks 13, 16, 19, and 22 make the fragment $X_{1.1}$, and so on. At each time interval, all fragments of a subobject are transferred from individual disks to the network at the same time. In this way, the system can display a subobject while retrieving it from the disk drives (i.e., multi-input pipelining) [4, 19].

When a system is to display X, it starts reading from a cluster, say C_i, that contains the first subobject X_0. Next, it employs $C_{i+1 \bmod R}$ to display X_1. The system iterates over the clusters until X is completely displayed.

Figure 2.5 demonstrates how the system serves concurrent requests for three objects, X, Y, and Z. Assume after retrieving subobject X_{i+2} at time interval 2, the display of object X is completed. Thus, cluster 0 becomes idle at time intervals 3 and 6, while clusters 1 and 2 become idle at time intervals 4 and 5, respectively. If a request for an object, whose first subobject resides on cluster 0, arrives at or before time interval 3, cluster 0 can begin serving the new request at time interval 3; otherwise, this request would have to wait until cluster 0 becomes idle at time interval 6.

Since the size of cluster is fixed and all disks in a cluster are operated synchronously, simple striping can not effectively deal with objects of varied bandwidths. In order to serve objects of different bandwidths, a naive approach [4] is to form the disk clusters based on the highest bandwidth requirement, which clearly would waste a large portion of available disk bandwidth.

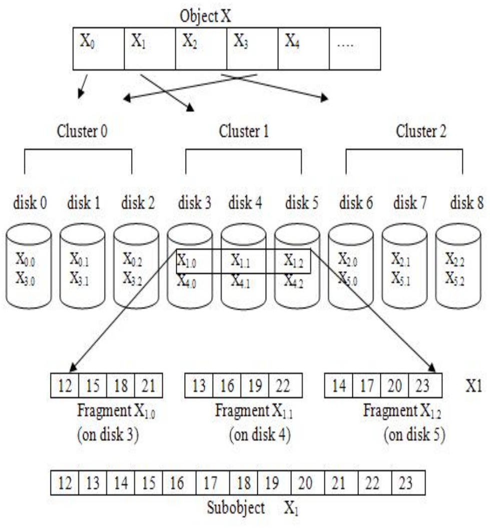


Fig. 2.4. Fragment Layout on Disks

time		Cluster 0	Cluster1	Cluster2
t	1	read Z_{k+1}	read X_{i+1}	read Y_{j+1}
i	2	read Y_{j+2}	read Z_{k+2}	read X_{i+2}
m	3	idle	read Y_{j+3}	read Z_{k+3}
e	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}

Fig. 2.5. Simple Striping with 3 Clusters

2.3.2 Staggered Striping

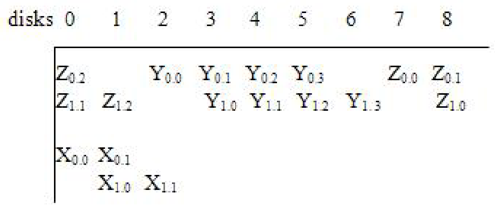
Staggered striping [4] is a refinement of the simple striping. In the staggered striping, disk clusters are formed logically, instead of physically, as in the simple striping. Moreover, the requirement that the assignment of two consecutive subobjects of X, say X_i and X_{i+1}, be on non-overlapping clusters in the simple striping is removed. The staggered striping can alleviate the wasted bandwidth problem due to the varying bandwidth requirements of media objects in the simple striping. The staggered striping is described as follows.

Stride k is defined to be the distance, in terms of the number of disk drives, between the first fragments of two consecutive subobjects $X_{i.0}$ and $X_{(i+1).0}$. Notice that in the simple striping, k is M(X), while in the staggered striping, k can vary in value from 1 to D. Staggered striping is indeed a generalization of the simple striping.

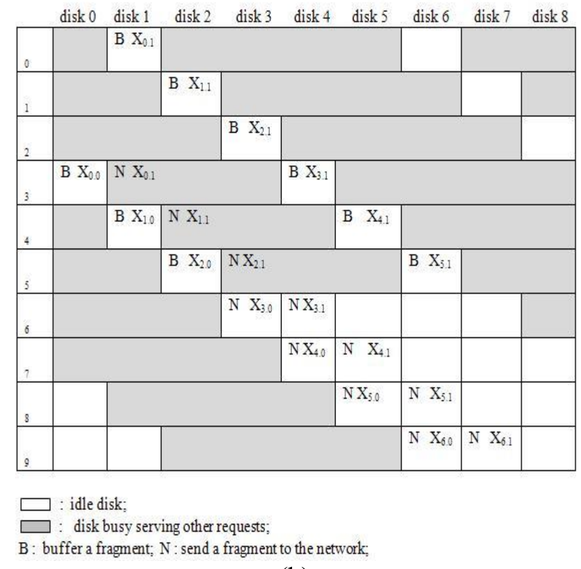
Objects, maybe of different bandwidths, are assigned to disks independently with the same stride. Figure 2.6(a) illustrates a possible placement of objects X, Y, and Z with bandwidth requirements of 40, 80, and 60 MBps, respectively, in a system with 9 disks (with B_{disk} = 20 MBps). X_0 is stored on disks 0 and 1 (M(X) = 2), Y_0 is on disks 2, 3, 4, and 5 (M(Z) = 4), and Z_0 is on disks 0, 7, and 8 (M(Y) = 3). Note that the strides for all objects are 1.

In order to display an object, say X, the system must first locate the M(X) adjacent disk drives that contain subobject X_0 (i.e., disks 0 and 1). If these disk drives are idle, X_0 can be retrieved and displayed immediately; otherwise, the display has to be delayed until these disks become idle. At the next time interval, it reads from the next M(X) disk drives (i.e., disks 1 and 2), by shifting k (i.e., 1 in this example) disks to the right.

As requests arrive and are served, the system is left with dispersed "holes" of idle disks, similar to the memory fragmentation problem in the traditional operating system. Since in the striping approach, all fragments of a subobject must be retrieved from disks at the same time. If the disks containing the requested subobject are not all idle at the same time, bandwidth fragmentation can occur. For example, consider the placement of objects in Figure 2.6(a) again. Assume the system is to retrieve Y_0 and Z_0 at the moment and a request for X has just arrived. While there


disks 0 1 2 3 4 5 6 7 8
$Z_{0.2}$ $Y_{0.0}$ $Y_{0.1}$ $Y_{0.2}$ $Y_{0.3}$ $Z_{0.0}$ $Z_{0.1}$
$Z_{1.1}$ $Z_{1.2}$ $Y_{1.0}$ $Y_{1.1}$ $Y_{1.2}$ $Y_{1.3}$ $Z_{1.0}$
$X_{0.0}$ $X_{0.1}$
$X_{1.0}$ $X_{1.1}$

(a)


disk 0 disk 1 disk 2 disk 3 disk 4 disk 5 disk 6 disk 7 disk 8
0 1 2 3 4 5 6 7 8 9
B $X_{0.1}$
B $X_{1.1}$
B $X_{2.1}$
B $X_{0.0}$ N $X_{0.1}$ B $X_{3.1}$
B $X_{1.0}$ N $X_{1.1}$ B $X_{4.1}$
B $X_{2.0}$ N $X_{2.1}$ B $X_{5.1}$
N $X_{3.0}$ N $X_{3.1}$
N $X_{4.0}$ N $X_{4.1}$
N $X_{5.0}$ N $X_{5.1}$
N $X_{6.0}$ N $X_{6.1}$
□ : idle disk;
▨ : disk busy serving other requests;
B : buffer a fragment; N : send a fragment to the network;

(b)

Fig. 2.6. Buffer Utilization for Staggered Striping with 9 Disks

are two disks (1 and 6) idle, a request for object X cannot be served until the display of Z completes, because disks 0 and 1 are not available at the same time. These free but non-consecutive idle disks, like 1 and 6, are called fragmented disks [4].

To utilize the bandwidth of fragmented disks, additional buffer space is used in [4]. Fragments on those "fragmented" disks are read and stored in the buffer until the rest of pieces of the subobjects are ready to transfer. That is, when all the pieces of a subobject are either having been buffered or residing on currently free disks, they are transmitted to the network (using a pipelining scheme).

In Figure 2.6(b), the white regions indicate that the corresponding disks are idle while the shaded regions indicate that disks are busy serving other requests. Assume that a request for an object X arrives at t_0 and there are two fragmented disks (disks 1 and 6) at this moment. Disk 1 is now in position to read fragment $X_{0.1}$, but disk 0 is busy with another request. Therefore, $X_{0.1}$ is read and buffered. At t_3, since disk 0 can directly transfer fragment $X_{0.0}$ onto the network, the buffered fragment $X_{0.1}$ is also transferred to the network. At t_6, we assume that three intervening disks (disks 4, 5, and 6) have completed their service and become free. Therefore, at t_9, the entire X_6 can be delivered directly from disks to the network without buffering.

While it seems simple to utilize fragmented bandwidth, the above example only illustrates a best scenario. In fact, we have found that disruptions of display can occur at any time when fragmented disks are used. To the best of our knowledge, such disruptions (that occur during the display) have not been addressed in the literature. We will describe this problem in detail in Section 3 and discuss the solution in Section 4.

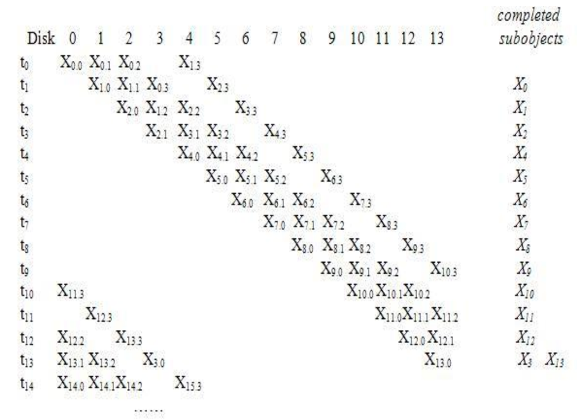
3. Observations on the Disruptions

The quality of a display is severely compromised if disruptions occur during the display. Disruptions can occur when there are not enough idle disks to display objects. However, even with enough idle disks, there can still be disruptions if these idle disks are not consecutive, i.e., fragmented. In this chapter, we show how disruptions can occur and discuss properties associated with disruptions.

3.1 Disruptions in Staggered Striping

As mentioned earlier, there can be variable-sized "holes" of idle disks in the system as disks are allocated to and released by requests for objects of different bandwidths. If non-consecutive idle disks are to be used, subobjects may not be able to be retrieved in their entireties, resulting in potential disruptions.

In Figure 3.1, we show how disruptions can occur in a staggered striping system with k = 1. Assume at t_0, disks 0, 1, 2, and 4 are idle and are designated to serve a request for object X with M(X) = 4. Note that the four idle disks are not consecutive and the rest of disks are busy with other requests. For clarity, only fragments of object X retrieved



Disk	0	1	2	3	4	5	6	7	8	9	10	11	12	13	*completed subobjects*
t_0	$X_{0.0}$	$X_{0.1}$	$X_{0.2}$		$X_{1.3}$										
t_1		$X_{1.0}$	$X_{1.1}$	$X_{0.3}$		$X_{2.3}$									X_0
t_2			$X_{2.0}$	$X_{1.2}$	$X_{2.2}$		$X_{3.3}$								X_1
t_3				$X_{2.1}$	$X_{3.1}$	$X_{3.2}$		$X_{4.3}$							X_2
t_4					$X_{4.0}$	$X_{4.1}$	$X_{4.2}$		$X_{5.3}$						X_4
t_5						$X_{5.0}$	$X_{5.1}$	$X_{5.2}$		$X_{6.3}$					X_5
t_6							$X_{6.0}$	$X_{6.1}$	$X_{6.2}$		$X_{7.3}$				X_6
t_7								$X_{7.0}$	$X_{7.1}$	$X_{7.2}$		$X_{8.3}$			X_7
t_8									$X_{8.0}$	$X_{8.1}$	$X_{8.2}$		$X_{9.3}$		X_8
t_9										$X_{9.0}$	$X_{9.1}$	$X_{9.2}$		$X_{10.3}$	X_9
t_{10}	$X_{11.3}$										$X_{10.0}$	$X_{10.1}$	$X_{10.2}$		X_{10}
t_{11}		$X_{12.3}$										$X_{11.0}$	$X_{11.1}$	$X_{11.2}$	X_{11}
t_{12}	$X_{12.2}$		$X_{13.3}$										$X_{12.0}$	$X_{12.1}$	X_{12}
t_{13}	$X_{13.1}$	$X_{13.2}$		$X_{3.0}$										$X_{13.0}$	X_3 X_{13}
t_{14}	$X_{14.0}$	$X_{14.1}$	$X_{14.2}$		$X_{15.3}$										

......

Fig. 3.1. Retrieval of Object X Using Non-consecutive Idle Disks 0, 1, 2 and 4 with k = 1.

at each time interval are indicated in the figure. Note that each disk stores a set of fragments coming from different subobjects. Fragments on a disk are read sequentially for the display of the object.

The rightmost column "completed subobjects" lists subobjects, all of whose fragments have been completely retrieved from disks at the corresponding time interval. As shown in the figure, a disruption occurred at t_4, and X_3 was not ready until t_{13}. Note that other subobjects, such as X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11} and X_{12}, are all ready before X_3. A naive solution to eliminating disruptions is to delay the retrieval until there are enough consecutive idle disks in the system. Unfortunately, it can result in substantial delay as consecutive idle disks may not be available until one or more current requests are finished.

It is observed from Figure 3.1 that although subobjects retrieved are out of sequence, there are still the same number of subobjects retrieved (i.e., 14 = D) in the first 14 time units as if we were using consecutive idle disks. In addition, the retrieval of fragments is not completely random, that is, no other fragments, except the fragments of the first 14 subobjects are read during the first 14 time units. Another important observation is that the set of disks used at t_0 is used again at t_{14} to retrieve fragments of X_{14} and X_{15} (vs. X_0 and X_1 at t_0). Consequently, the retrieval pattern appearing during the period t_0 to t_{13} is repeated in the next and every successive 14 time units. This observation prompts the idea that if we can buffer a small amount of subobjects (e.g., the first 14 subobjects in the example) in advance, then the display may be able to be smoothed over without disruption. That is, while we are going to retrieve the next 14 subobjects in the next 14 time units, we display the previously buffered 14 subobjects.

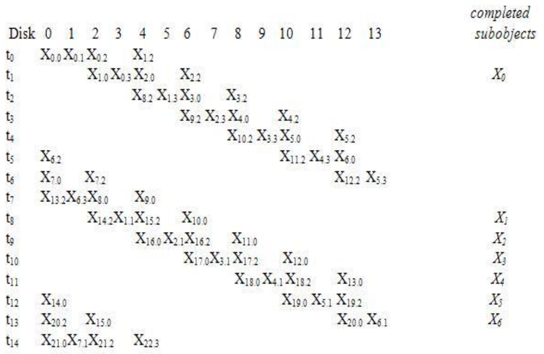
The proposed approach solely relies on the existence of such repeated retrieval pattern. Therefore, in the following, we discuss in what circumstances repeated patterns like above exist.

3.2 Retrieval Patterns

The values of k affect retrieval patterns when non-consecutive idle disks are used. Recalling that in Figure 3.1, where k was set to 1, although a disruption occurred at t_4, all the first 14 subobjects, X_0, X_1, …, X_{13}, were completely retrieved by the end of t_{13}, and a new cycle begins at t_{14}. This repeated pattern can be identified easily and can be very useful in eliminating disruption.

However, the situations may be a little more complex when k is set to other values. Let us consider Figure 3.2, where k is set to 2. Although there are still four fragments of object X read in each time interval, some disks, such as disks 1, 3, 5, 7, 9, 11, and 13, have read only two fragments during the first 14 time units, while others read six fragments. Due to this imbalance in reading, the retrieval of those fragments on odd-numbered disks is considerably delayed. For example, fragments $X_{7.1}$ and $X_{7.3}$, on disks 1 and 3, respectively, were not retrieved until t_{14} and t_{15}, while some other higher-numbered fragments, like $X_{15.2}$, $X_{16.0}$, etc., were read much sooner than in the previous case (i.e., k = 1). Indeed, fragments on even-numbered disks are retrieved sooner, while much later for fragments on odd-numbered disks, prolonging the entire retrieval process by a factor of 2. This implies a very large buffer may be needed if we attempt to eliminate disruptions.

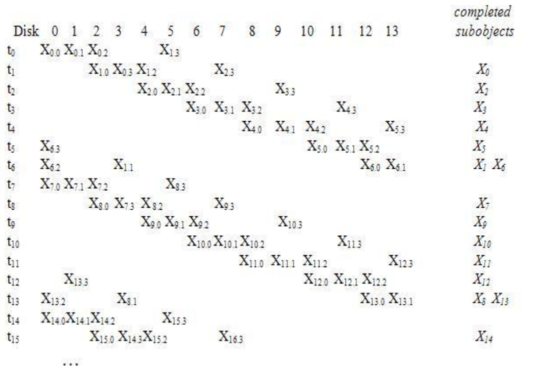
However, if we had used idle disks 0, 1, 2, and 5, assumed available, the situation would have been quite different. As shown in Fig. 3.3, with idle disks 0, 1, 2, and 5 designated at t_0, all the first 7 subobjects are completely retrieved by the end of t_6. A new cycle begins at t_7 and repeats for every 7 time units. This pattern could be useful



Disk	0	1	2	3	4	5	6	7	8	9	10	11	12	13	*completed subobjects*
t_0	$X_{0.0}$	$X_{0.1}$	$X_{0.2}$		$X_{1.2}$										
t_1			$X_{1.0}$	$X_{0.3}$	$X_{2.0}$		$X_{2.2}$								X_0
t_2					$X_{8.2}$	$X_{1.3}$	$X_{3.0}$		$X_{3.2}$						
t_3							$X_{9.2}$	$X_{2.3}$	$X_{4.0}$		$X_{4.2}$				
t_4									$X_{10.2}$	$X_{3.3}$	$X_{5.0}$		$X_{5.2}$		
t_5	$X_{6.2}$										$X_{11.2}$	$X_{4.3}$	$X_{6.0}$		
t_6	$X_{7.0}$		$X_{7.2}$										$X_{12.2}$	$X_{5.3}$	
t_7	$X_{13.2}$	$X_{6.3}$	$X_{8.0}$		$X_{9.0}$										
t_8			$X_{14.2}$	$X_{1.1}$	$X_{15.2}$		$X_{10.0}$								X_1
t_9					$X_{16.0}$	$X_{2.1}$	$X_{16.2}$		$X_{11.0}$						X_2
t_{10}							$X_{17.0}$	$X_{3.1}$	$X_{17.2}$		$X_{12.0}$				X_3
t_{11}									$X_{18.0}$	$X_{4.1}$	$X_{18.2}$		$X_{13.0}$		X_4
t_{12}	$X_{14.0}$										$X_{19.0}$	$X_{5.1}$	$X_{19.2}$		X_5
t_{13}	$X_{20.2}$		$X_{15.0}$										$X_{20.0}$	$X_{6.1}$	X_6
t_{14}	$X_{21.0}$	$X_{7.1}$	$X_{21.2}$		$X_{22.3}$										

…

Fig. 3.2. Retrieval of Object X Using Non-consecutive Idle Disks 0, 1, 2 and 4 when k = 2.



Disk	0	1	2	3	4	5	6	7	8	9	10	11	12	13	*completed subobjects*
t_0	$X_{0.0}$	$X_{0.1}$	$X_{0.2}$			$X_{1.3}$									
t_1			$X_{1.0}$	$X_{0.3}$	$X_{1.2}$			$X_{2.3}$							X_0
t_2					$X_{2.0}$	$X_{2.1}$	$X_{2.2}$			$X_{3.3}$					X_2
t_3							$X_{3.0}$	$X_{3.1}$	$X_{3.2}$			$X_{4.3}$			X_3
t_4									$X_{4.0}$	$X_{4.1}$	$X_{4.2}$			$X_{5.3}$	X_4
t_5	$X_{6.3}$										$X_{5.0}$	$X_{5.1}$	$X_{5.2}$		X_5
t_6	$X_{6.2}$			$X_{1.1}$									$X_{6.0}$	$X_{6.1}$	X_1 X_6
t_7	$X_{7.0}$	$X_{7.1}$	$X_{7.2}$			$X_{8.3}$									
t_8			$X_{8.0}$	$X_{7.3}$	$X_{8.2}$			$X_{9.3}$							X_7
t_9					$X_{9.0}$	$X_{9.1}$	$X_{9.2}$			$X_{10.3}$					X_9
t_{10}							$X_{10.0}$	$X_{10.1}$	$X_{10.2}$			$X_{11.3}$			X_{10}
t_{11}									$X_{11.0}$	$X_{11.1}$	$X_{11.2}$			$X_{12.3}$	X_{11}
t_{12}		$X_{13.3}$									$X_{12.0}$	$X_{12.1}$	$X_{12.2}$		X_{12}
t_{13}	$X_{13.2}$			$X_{8.1}$									$X_{13.0}$	$X_{13.1}$	X_8 X_{13}
t_{14}	$X_{14.0}$	$X_{14.1}$	$X_{14.2}$			$X_{15.3}$									
t_{15}			$X_{15.0}$	$X_{14.3}$	$X_{15.2}$			$X_{16.3}$							X_{14}

…

Fig. 3.3. Retrieval of Object X Using Non-consecutive Idle Disks 0, 1, 2, and 5 when k = 2.

in our attempt to display objects continuously.

From Figures 3.2 and 3.3, we notice that with the proper choice of idle disks, if available, useful repeated patterns with a shorter period (less than D) may be found when k values are different than 1. In the next chapter, we will formally discuss how a pattern is affected by various factors, such as k, D, placement of subobjects, and idle disks, etc.

4. Pattern Analysis

The goal is to find the existence of a period of p time units, within which the next p successive subobjects can be completely retrieved using a given set of non-consecutive disks. For simplicity, we shall not mention the object of concern explicitly in the following discussion

Definition. The storage sequence for the f^th fragments of subobjects m to n, denoted $S_f(m, n)$, is a sequence of disk IDs that stores the f^th fragments of successive subobjects from m to n.

Recall that each subobject is striped across a number of disks and the part of the subobject on each disk is called a fragment. Let j be the ID number ($0 \le j < D$) of the disk containing the f^th fragment of the 0^th subobject of X (i.e., $X_{0.f}$). By definition of the stride k, disk $(j + k \times n) \bmod D$ must be the disk containing the f^th fragment of the n^th subobject. Thus, $S_0(0, n)$ consists of disks j , $(j + k \times 1) \bmod D$, …, $(j + k \times n) \bmod D$.

Definition. The retrieval sequence beginning with disk i for the period from t_m to t_n, denoted as $R_i(t_m, t_n)$, is the sequence of disk IDs beginning with disk I that are used in successive time intervals from t_m to t_n to retrieve the object in the staggered striping.

Let i be the ID number ($0 \le i < D$) of one of the disks designated to serve a request for a certain object X at t_0. Then at t_n, we can infer that disk $(i + k \times n) \bmod D$ must be a disk serving X, because of the stride k. Consequently, $R_i(t_0, t_n)$ is made of disks i, $(i + k \times 1) \bmod D$, …, $(i + k \times n) \bmod D$. Note that each disk designated to serve X at t_0 has its own retrieval sequence.

Let ω be the largest common divisor of D and k. Then, both a storage and a retrieval sequence can be rewritten as i, $(i + k \times 1) \bmod D$, …, $(i + k \times D/\omega) \bmod D$, $(i + k \times (D/\omega + 1)) \bmod D$, …, etc. Since $(k \times D/\omega) \bmod D = 0$, $(i + k \times D/\omega) \bmod D$ becomes $(i + k \times 0) \bmod D$, $(i + k \times (D/\omega + 1)) \bmod D$ becomes $(i + k \times 1) \bmod D$, and so on. As a result, there can be only D/ω distinct disks in each storage or retrieval sequence of length greater than D/ω, and they are reused every D/ω time units.

Let i_m, $0 \le m < M(X)$, be a set of M(X) idle disks chosen to serve X at t_0. As mentioned earlier, each disk stores a set of fragments from different subobjects (of an object) and those fragments on a disk are retrieved one at a time sequentially when the disk is used to display the object. It can be conceived that if the disks appearing in the sequences $R_{i_m}(0, D/\omega)$, $0 \le m < M(X)$, also appear the same number of times in sequences $S_f(0, D/\omega)$, $0 \le f < M(X)$, and vice-versa, then the first D/ω subobjects, nothing more and nothing less, are retrieved during the first D/ω period. Note that the sequences $S_f(0, D/\omega)$, $0 \le f < M(X)$, represent the first D/ω objects. Moreover, a new and identical cycle begins for each successive D/ω time interval because all subobjects appearing in preceding D/ω interval are completely retrieved during that interval. In the following, we will discuss how to find such set of idle disks, if available, so that this retrieval pattern exists.

When $\omega = 1$, i.e., no common divisor between k and D, $D/\omega = D$. Let i_m, $0 \le m < M(X)$, be an arbitrarily set of M(X) idle disks chosen at t_0. It can be observed that at the end of t_{D-1}, every disk in the system appears exactly once in any of the retrieval sequences $R_{i_m}(0, D-1)$, $0 \le m < M(X)$. Meanwhile, all disks will also appear exactly once in each $S_f(0, D-1)$, $0 \le f < M(X)$. Thus, at the end of t_{D-1}, all fragments of the first and only the first D subobjects (i.e., $D \times M(X)$ fragments) are retrieved using arbitrarily chosen M(x) idle disks. Moreover, a new cycle begins for the next and successive D time units because no partially retrieved subobjects from previous D time units are left to be completed in the next D time units.

When $\omega \ge 2$ (obviously, $k \ge 2$), the situation is a bit more complex. For each disk i, designated to serving object X at t_0, only D/ω ($< D$) distinct disks appear in its retrieval sequence, i.e., i, $(i + k \times 1) \bmod D$, $(i + k \times 2) \bmod D$, $(i + k \times ((D/\omega - 1)) \bmod D$, and they are reused every D/ω time units. For example, consider Figure 3.2 again, where $k = 2$ and $D = 14$. Since $\omega = 2$, only 7 distinct disks are used in each retrieval sequence. For idle disk 0 chosen at t_0, $R_0(0, 6)$ consists of only disks 0, 2, 4, 6, 8, 10 and 12, while for disk 1, $R_1(0, 6)$ is made of 1, 3, 5, 7, 9, 11, and 13. Similarly, for each disk j storing the f th fragment, $0 \le f < M(X)$, of the 0th subobject, the storage sequence $S_f(0, D/\omega - 1)$ consists of 7 disks, j, $(j + k \times 1) \bmod D$, …, and $(j + k \times (D/\omega - 1)) \bmod D$.

It can be observed that if i is an idle disk chosen at t_0 and it appears in the sequence $S_f(0, D/\omega)$ (i.e., j, $(j + k \times 1) \bmod D$, …, i, $(i + k \times 1) \bmod D$, …, or $(j + k \times (D/\omega - 1)) \bmod D$), then $R_i(0, D/\omega)$ contains the same set of D/ω distinct disks as $S_f(0, D/\omega)$, except that the order of the disks appearing in the sequences may be different. Thus, if i_m, $0 \le m < M(X)$, are the idle disks chosen at t_0, each of them appearing in a distinct storage sequence $S_f(0, D/\omega)$, $0 \le f < M(X)$, then at $t_{D/\omega - 1}$, then the same set of disks will appear the same number of times in both $R_{i_m}(0, D/\omega - 1)$, $0 \le m < M(X)$, and $S_f(0, D/\omega)$, $0 \le f < M(X)$. That is, the first (and only the first) D/ω subobjects will be completely retrieved during the first D/ω time units using idle disks i_m, $0 \le m < M(X)$. The pattern repeats for every D/ω time units as no partially retrieved subobjects from previous D/ω interval need to be completed in the next D/ω interval and the set of disks used at t_0 is used again at $t_{n\times(D/\omega)}$, $n > 0$. The following theorem follows.

Theorem. If each of the idle disks chosen to serve an object is from a distinct storage sequence of the object, then each successive D/ω subobjects from the beginning can be completely retrieved within each successive D/ω time interval.

In the following, we use a simple example to illustrate how idle disks, if available, are chosen to avoid potential disruptions.

Example. Consider a system with 14 disks and a stride $k = 4$. Assume an request for object X ($M(X)=3$) has just arrived and currently disks 5, 7, 8, and 10 are idle. We further assume that the fragments of the first subobject of X are stored on disks 2,3, and 4.

The storage sequences are:

S_0 = 2, 6, 10, 0, 4, 8, 12, 2, 6, …
S_1 = 3, 7, 11, 1, 5, 9, 13, 3, 7, …
S_2 = 4, 8, 12, 2, 6, 10, 0, 4, 6, …

Notice that S_0 and S_2 are essentially the same sequence. As a result, we can choose disks 8 from S_0 (or from S_2), 5 from S_1, and 10 from S_2 (or from S_0). Another possible combination could be 8, 7, and 10.

Disruptions can be eliminated by taking advantage of these retrieval patterns. Two buffers, each of which can hold D/ω subobjects, may be needed. That is, when the system is displaying previously retrieved D/ω subobjects from one buffer, it retrieves next D/ω subobjects into another buffer. The two buffers are used for input and output alternately.

4.2 Discussions

Disruptions are eliminated using a pair of buffers of size D/ω subobjects each. As conceived, the larger the ω value, the smaller the buffer is required, and the shorter the waiting time (i.e., the time to fill the buffer at the beginning). However, when ω gets larger, it may become more difficult to find suitable idle disks. As readers may have noticed that the disks in the system are in fact divided into ω disjoint groups, each of which has D/ω disks. The ω groups are (i, $(i + k) \bmod D$, …, $(i + (D/\omega - 1) \times k) \bmod D$), ($i + 1$, $(i + 1 + k) \bmod D$, …, $(i + 1 + (D/\omega - 1) \times k) \bmod D$), …, ($i + \omega - 1$, $(i + \omega - 1 + k) \bmod D$, …, $(i + \omega - 1 + (D/\omega - 1) \times k) \bmod D$). In order to guarantee the appearance of previously mentioned patterns, idle disks must be chosen from groups corresponding to the storage sequences of the object. Note that more than one disk may need to be chosen from a group when $M(X) > \omega$. The larger the number of groups, the more difficult it is to find a match. Therefore, a trade-off has to be made by the administrator.

In summary, when $\omega = 1$, any arbitrarily chosen M(X) idle disks can be used to display object X. It is most flexible, however, with a bit longer delay of D time units and a bit larger buffer. Note that there is also a delay in the original staggered striping, where only consecutive idle disks are used for display. The delay is the average amount of time waiting for all idle disks to shift to the right positions to read the first subobject. When ω > 1, each idle disk chosen must appears in a distinct storage sequence of the object. It may be more restrictive, but it uses smaller buffers and spends less time in waiting.

5. Conclusion

Striping techniques can effectively deal with the bottleneck problem by declustering objects across multiple disks and using the aggregated bandwidth of disks to display objects. In general, they can serve more requests concurrently and store more objects than the replication methods. The staggered striping is probably the most promising technique in the striping category for MOD systems because of its flexibility in storing multimedia objects and the capability of displaying objects of variable bandwidths.

Bandwidth fragmentation can occur when requests (for objects of varying bandwidths) arrive and are completed. It renders the systems with many small holes of idle disks, which are not large enough to serve requests, and thus can degrade the performance of the system considerably. In this paper, we addressed the problems of potential disruptions in display when fragmented disks are used. We have analyzed the retrieval patterns and identified patterns that can be useful in eliminating disruptions. We have shown that with proper choice of idle disks and a simple buffer scheme, disruptions can be eliminated. Currently, we are investigating further improvement on the waiting time, which is D/ω now.

References

[1] Santos, J. R., Muntz, R., "Design of the RIO (Randomized I/O) Storage Server", Technical Report 970032, Computer Science Dept., UCLA, 1997.

[2] Gemmel, D. J., Vin, H. M., Kandlur, D. D., Rangan, P.V., Rowe, L. A., "Multimedia Storage Servers: A Tutorial", IEEE Computer, May 1995, pp. 40 – 49.

[3] Ng., S., "Advances in Disk Technology: Performance Issues", IEEE Computer, May 1998, pp. 75 – 81. [ORS 97] Ozden, B., Rastogi, R., Silberschatz, A., "Periodic Retrieval of Videos from Disk Arrays", IEEE 1997, pp.333 – 343.

[4] Berson, S., Ghandeharizadeh, S., Muntz, R, Ju, X., "Staggered Striping in Multimedia Information System", Proc. of the 1994 ACM SIGMOD, vol. 23 no 2, June 1994, pp.79 – 90.

[5] Raghavan, S. V., Tripathi, S. K., "Networked Multimedia Systems: Concepts, Architecture, and Design", Prentice Hall, 1998.

[6] Ghandeharizadeh, S., Ramos, L., "Continuous Retrieval of multimedia data using parallelism", IEEE Transactions on Knowledge and Data Engineering, vol. 5, no. 4, 1993, pp. 658 – 669.

[7] Freedman, C., DeWitt, D. J., "The SPIFFI Scalable Video-on-Demand System", SIGMOD' 95, pp. 352 – 361.

[8] Luther, A.C., "Digital Video in the PC Environment", New York: McGraw-Hill Book Company, 1989.

[9] Chen,M., Hsiao, H., Li, C., Yu, P., "Using Rotational mirrored Declustering for Replica Placement in a Disk-array-based Video Server", Multimedia Systems, vol. 5, December 1997, pp. 371 – 379.

[10] Ghandeharizadeh, S., Kim, S. H., Shi, W., Zimmermann, R., "On Minimizing Startup Latency in Scalable continuous Media Servers", Proceedings of Multimedia Computing and Networking conference, Feb. 1997.

[11] Shahabi, C., "Scheduling the Retrievals of Continuous Media Objects", USC Computer Science Ph.D. Thesis, Aug. 1996.

[12] Al-Marri, J. A. M., "Variable Bit Rate Continuous Media Servers", USC Computer Science Ph.D. Thesis, Aug. 1998.

[13] Ghandeharizadeh, S., Zimmermann, R., Shi, W., Rejaie, R., Ierardi, D., Li, T., "Scalable Continuous Media Server", Multimedia Tools and Applications Journal, Kluwer Academic Publishers, vol. 5, issue 1, July 1997, pp.79 – 108.

[14] Dashti, A.E., Ghandeharizadeh, S., "On Configuring Hierarchical Storage Structures", Joint NASA/IEEE Mass Storage Conference, March 1998.

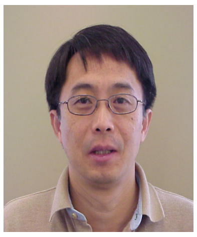
[15] Ghandeharizadeh, S., Kim, S., "Design of Multi-user Editing Servers for Continuous Media" J. of Multimedia Tools and Applications, Vol. 11, pp. 101-127, 2000.

[16] Zimmermann, R., Ghandeharizadeh, S., "Continuous Display Using Heterogeneous Disk Subsystems", ACM Multimedia Conference, Nov. 1997.

[17] Zimmermann, R., "Continuous Media Placement and Scheduling in Heterogeneous Disk Storage Systems", USC Computer Science Ph.D. Thesis, Dec. 1998.

[18] Shi, W., "Data Sharing in Interactive Continuous Media Servers", USC Computer Science Ph.D. Thesis, Aug. 1998.

[19] Ghandeharizadeh, S., Dashti, A., Shahabi, C., "Pipelining Mechanism to Minimize the Latency Time in Hierarchical Multimedia Storage Managers", Computer Communications, vol. 18, issue 3, March 1995, pp. 170 – 184.



Wen-Chi Hou
He received a Ph.D. degree in Computer Sci. & Eng. from Case Western Reserve Univ., Cleveland Ohio, in 1989. He is an Associate Professor at Southern Illinois University Carbondale, USA. His research interests include database & data mining.

Dunren Che
He received his Ph.D. in Computer Science from Beijing University of Aeronautics and Astronautics, China, in 1994. He is currently an Assistant Professor at Southern Illinois University Carbondale, U.S.A. His main interests are in database and data mining.