EFFICIENT STORAGE AND RETRIEVAL METHODS FOR MULTIMEDIA INFORMATION

Kwok Chung Tse

This thesis is presented in fulfilment of
the requirements of the degree of
Doctor of Philosophy

School of Communications and Informatics

Victoria University

Australia

1999

Declaration

This thesis contains no materials that have been accepted for the award of any other degree or diploma in any university. The materials presented in this thesis are the product of the author's own independent research under the supervision of Professor Clement Leung.

Philip Kwok Chung Tse

August, 1999

Abstract

The input/output performance has always been the bottleneck problem of computer systems, and with multimedia applications, the problem has been intensified. Hierarchical storage systems provide extensive storage capacity for multimedia data at very economical cost, but the long access latency of tertiary storage devices makes them not attractive for multimedia systems. In this thesis, we present new storage and retrieval methods to handle multimedia data on hierarchical storage systems efficiently. First, we create a novel hierarchical storage organization to increase the storage system throughput. Second, we enhance the data migration method to reduce the multimedia stream response time. Third, we design a new bandwidth based placement method to store heterogeneous objects. Fourth, we demonstrate that disk performance is significantly enhanced using constant density recording disks. We have quantitatively analysed and compared the performance of magnetic disks and hierarchical storage systems in serving multimedia streams of requests. We have also carried out empirical studies which confirm our findings. Our new storage and retrieval methods are able to offer significant advantages and flexibility over existing methods.

Acknowledgment

I am deeply indebted to my supervisor, **Professor Clement Leung**, for his guidance, constant direction, and countless hours of time. I thank him for his patience and trust that allow me perform at full potential. I also appreciate his encouragement and support through the darkest days. I have benefited immeasurably from his tutelage.

I gratefully acknowledge **Associate Professor Neil Barnett**, Head of the School of Communications and Informatics, for his approvals to the generous scholarship and financial support. I would like to thank other staff of the School of Communications and Informatics for being supportive and cooperative. I would like to thank my friends, **Simon So**, **David Taniar**, **Dwi Sutanto**, **Tony Sahama**, and **Raymond Summit** for many hours of technical discussions.

I deeply owe my appreciation to my wife, **Peky**, for her consistent support with trust and love in these years that I was able to do this research. I miss the moments that I could otherwise spend with my two sons, **Joshua** and **Jonah**, who understand their father's reason for returning home late. I also thank **my parents** who provide me with education and direction since I was very young.

Last but not least, I praise **GOD** for leading me to pursue this dream, answering all my prayers, and fulfilling all my needs during this study.

List of External Publications

1. **Philip K. C. Tse** and Clement H. C. Leung, "Improving Multimedia Systems Performance Using Constant Density Recording Disks", *ACM Multimedia Systems Journal*, Springer-Verlag, Volume 8, Number 1, pages 47-56, 2000.

2. Clement H. C. Leung, Simon W. W. So, Audrey Tam, Dwi Sutanto, and **Philip K. C. Tse**, "Visual Information Indexing for Content-Based Search and Retrieval", to appear in Michael Lew (Ed.), *Multimedia Search: State of the Art*.

3. David Taniar, Simon W. W. So, **Philip K. C. Tse,** and Clement H. C. Leung, "Parallel Content-Based Image Retrieval: A Case of Inter-Image Parallelism vs. Intra-Image Parallelism", *Proceedings of the sixth Annual Australasian Conference on Parallel And Real-Time Systems* (PART '99), pages 273-283, 29th Nov. to 1st Dec., 1999.

4. **Philip K. C. Tse**, Audrey Tam, Clement H. C. Leung, and W. W. S. So, "Efficient Storage Organization for the Execution of Visual Queries in Large Object Databases", SPIE Vol. 3846, *Proceedings of Multimedia Storage and Archiving Systems IV Conference*, Boston, MA, pages 351-359, 19th to 22nd Sep., 1999.

5. Clement H. C. Leung and **Philip K. C. Tse**, "Storage Organization and Performance for Multimedia Data", *Handbook of Multimedia Computing*, CRC Press, Chapter 24, pages 507-524, 1999.

6. **Philip K. C. Tse** and Clement H. C. Leung, "A low latency hierarchical storage system for Multimedia Data", *IAPR International Workshop on Multimedia Information Analysis and Retrieval*, Springer-Verlag Lecture Notes in Computer Science 1464, pages 181-194, August, 1998.

7. Simon W. W. So, **Philip K. C. Tse,** and Clement H. C. Leung, "A Comparative Evaluation of Algorithms Using Compressed Data for Image Indexing and Retrieval", *Proceedings of International Conference on Computational Intelligence and Multimedia Applications*, pages 866-872, February, 1998.

8. **Philip K. C. Tse**, "Tertiary Storage for Multimedia Systems", *Proceedings of CSPSC Conference*, RMIT, Australia, pages 174-183, 10^{th} to 11^{th} December, 1996.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objectives

Advanced technology in recent years increases the availability of video and audio information for computers. This trend leads to the merging of the computer and entertainment industries for the purpose of efficiently handling multimedia information in computer applications. This capability is able to support many new multimedia applications, such as video-on-demand systems, interactive television, and video conferencing.

The main objective of this research is to discover new knowledge and techniques in building storage systems for managing multimedia data. Efficient techniques to store, retrieve, and migrate multimedia data will be designed and evaluated in this thesis.

The storage and retrieval characteristics of multimedia data are very different from those of traditional data, and multimedia data place significantly heavier demand on all components of computer systems. In particular, multimedia data require high capacity, fast access time and high transfer rates from the storage and memory. Data compression techniques using international standards such as

JPEG, MPEG, have helped to reduce the storage size and bandwidth, but new data placement techniques are necessary to meet the continuous retrieval requirements of multimedia data.

Although secondary storage devices can provide some storage capability for multimedia data, however, their size limitation and cost constraints present a significant obstacle for their large scale adoption. Tertiary storage devices are necessary to provide permanent storage for multimedia data. However, the performance of the current generation of tertiary storage devices is highly inadequate. It is widely felt that tertiary storage devices in combination with hierarchical storage management will provide the most workable approach. New knowledge on hierarchical storage systems will need to be found to make them efficient for storing multimedia data.

Multimedia data require large bulky storage space so that tertiary storage is inevitable in many large systems. Traditional storage organizations of tertiary storage devices are tailored for sequential data access and are inefficient in accessing multimedia data. Performance of these storage devices will degrade by the high switching overheads and latency of tertiary storage devices. We provide the design of a new storage organization for tertiary storage device to enhance their performance for multimedia data retrievals.

Traditional data migration methods are inefficient for multimedia data. Although the pipelining method minimizes the response time, it also limits the display options. We shall present a new pipelining method, which minimizes the response time but at the same time is able to support various user functions.

There is a recent trend to use disks based on constant density recording (CDR) structure, which have different storage formats and performance characteristics. The performance of multimedia requests on CDR disks so far has not been studied in detail. We provide a detailed analysis of CDR disks in handling multimedia data, present a new efficient method to place heterogeneous multimedia objects on these disks.

1.2 Scope of Research

Research studies in computer storage system performance is often divided into the following areas.

1. the architecture of the computer system,
2. the storage organizations, and
3. the data retrieval mechanisms.

In this thesis, we shall focus on the storage organizations aspects while taking consideration the other two aspects.

- **System Architecture Design**

A computer system is built by connecting CPU, system bus, memory, disks, keyboard, and CRT display together. It may also be connected to networks, storage libraries, and other electronic peripherals. Since the storage system is often the slowest but frequently accessed part of a computer system, the overall performance of these systems is constrained by the performance of the storage system. Hence, the choice of storage devices is critical to system performance.

We shall investigate the impact on the performance of multimedia requests when disks based on constant density organization are used in place of traditional organizations. The result of this analysis will have a direct bearing on future multimedia system design.

- **Storage Organizations**

Storage organizations determine the physical location of data placed at the storage devices. Since most overheads in storage devices are spent in moving to the physical locations of the accessed data, the storage organization is critical in determining the system performance. We develop two new storage organizations for multimedia data based on constant density recording structure and tertiary storage systems.

- **Data Retrieval Mechanism**

The data retrieval mechanism, in particular the scheduling algorithm, is often ignored in designing high performance systems design. However, retrieval overheads in some devices may be reduced when several requests are being served together using appropriate scheduling algorithms. Data migration forms part of the data retrieval mechanism that moves data across storage systems. We present a new pipelining method that will enable efficient migration of multimedia information across different levels of the storage hierarchy.

1.3 Contributions

This thesis makes four principal contributions. The relationships between our contributions and the scope of research are shown in Figure 1.1.

- **New Bandwidth Based Placement**

We design a novel bandwidth based placement method to store heterogeneous multimedia objects on constant density recording disks. This method of placement has the advantage of maintaining the balance between the space and bandwidth of disk zones to secure high bandwidth zones for high bandwidth objects.

- **New High Concurrency Placement**

We design a novel storage organization for the efficient execution of multimedia data requests on tertiary storage systems. This novel high concurrency placement method allows the overheads of concurrent multimedia data requests to be shared out, and increases the system throughput by means of data migration. Since the tertiary storage system performance is enhanced by application of the high concurrency placement method, large multimedia systems can be made more efficient through this new storage organization.

- **New Segment Based Pipelining**

We also design a new segment based pipelining method that moves data between storage hierarchy. This pipelining method reduces the response time in serving interactive user functions and it also allows data previewing prior to display.

- **Performance Model and Analysis**

We analytically demonstrate that constant density recording structures are superior to traditional disks in serving multimedia data requests. We also provide insight to the design of these storage structures.

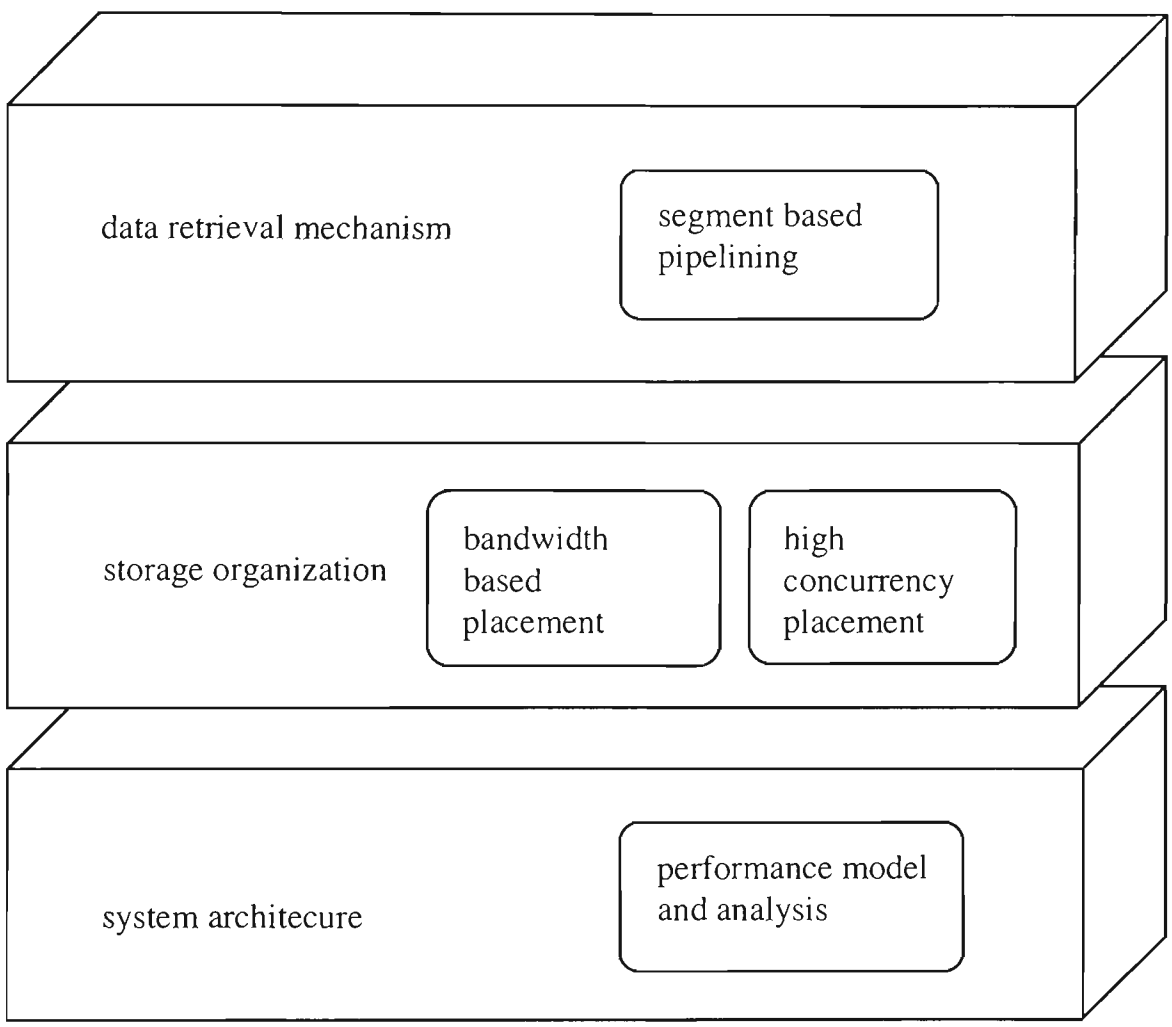


Figure 1.1. Contributions within scopes of research

1.4 Thesis Organization

The organization of this thesis is as follows. In Chapter 2 we review and evaluate the design requirements of multimedia storage systems and data storage organizations. In Chapter 3 we present our new hierarchical storage organization design. In Chapter 4 we analyse the performance characteristics of CDR disks, and we study the disk request service behaviour for multimedia data accesses. In Chapter 5 we prove the feasibility of accepting and processing multimedia heterogeneous streams, and present our performance model of tertiary storage systems and data migration methods. We also analyse and compare the tertiary storage system performance in large multimedia storage servers. Chapter 6 provides the conclusion to this thesis.

Chapter 2

Multimedia Storage Systems

Performance Characteristics

2.1 Introduction

Multimedia systems are required to store and manipulate a variety of media types. The types of media include text, graphics, images, sound, audio and video. Because of the types of media being handled, multimedia information systems are able to produce striking visual and audio impact on the user. Applications, such as video-on-demand systems, interactive television, and video conferencing, are expected to be widely available.

The storage components of multimedia systems may comprise video servers, audio servers, and image servers [10, 28], where each server tends to be optimized to retrieve one particular type of data. A collection of multimedia servers will need to balance the load of various types of media data in order to deliver an acceptable level of performance.

2.2 Problems of Existing Storage Systems

There is a substantial performance gap between storage systems and other parts of the computer system. Main memory access time is of the order of 10^{-8} second, current disks support an access time of 10^{-2} second, and tertiary storage devices takes minutes to access data. Such performance gap already exists for conventional structured data and text based applications, but they are widened in multimedia applications. Conventional databases mainly focus on Online Transaction Processing (OLTP) applications and these typically require no more than a handful of rows from a relational table. Online Analytic Processing (OLAP), data warehouses, decision support applications, and data mining admittedly require a higher level of data intensity, but their performance constraint is much less stringent than that for real-time multimedia applications.

A multimedia file system needs to store a pool of multimedia objects, and is different from a conventional file system due to the support required for continuous retrieval of the multimedia streams and for efficient browsing and indexing of the data [135]. A multimedia object can consist of data from several media streams such as audio and video. Stored information of a multimedia object may consist of compressed media streams, lengths of data blocks, the inter-media synchronization information, the real-time access bandwidth, the display bandwidth, and the resolution of the objects. All these information must be maintained by the file system [88].

Traditional storage organizations are tailored for small transfers on magnetic disks. With the advent of the new multimedia age, these storage organizations become inefficient in accessing multimedia data. It remains a

significant research area to improve the performance of multimedia storage systems.

Two approaches are commonly found in designing multimedia disk storage. In the first approach, the file organization is highly structured and optimized for storing audio and video files [99, 115, 142]. Some storage organizations have been proposed for multimedia data on magnetic disks: disk arrays provide large storage capacity; staggered striping reduces disk contentions; constraint allocations lower seek overheads to meet continuous display requirement; interleaving allocation supports concurrent streams. In the second approach, disk scheduling algorithms are specialized for accessing multimedia data efficiently [2, 55, 122, 142]. Both approaches aim to reduce the overheads of accessing multimedia data from disks.

Unfortunately, these storage methods that works well in magnetic disks are not directly applicable to tertiary storage in large multimedia systems. Current works on tertiary storage are also inadequate for accessing multimedia objects, resulting in low throughput and long response time. New storage organizations and data migration methods that provide efficient multimedia data retrievals from tertiary storage devices are an important research areas.

Multimedia data require large bulky storage space that the use of tertiary storage is inevitable in many large systems. Traditional storage organizations on tertiary storage devices are tailored for sequential data access and are inefficient in accessing multimedia data. Performance of these storage devices is degraded by the high switching overheads and latency of tertiary storage devices. It remains an

important research problem to create new storage organizations on tertiary storage for efficient multimedia data retrievals.

Traditional data migration methods are inefficient for multimedia data. The pipelining method minimizes the response time, but it also limits the display options. Hence, the pipelining method should be enhanced to increase flexibility.

2.3 Characteristics of Storage Devices

Computers access data from memory, magnetic disks, optical disks, optical jukeboxes, and magnetic tape libraries. Storage devices are often classified by the access mechanism into three types: random access, direct access, and sequential access [102, 119].

2.3.1 Random Access Storage Device

Random access storage devices are memories that any particular location of small size (either a bit or a byte) has a unique, physically wired-in addressing mechanism. These memory chips are mounted permanently on the mother boards of the computer.

Random access memory (RAM) and read only memory (ROM) are two most commonly used random access storage devices. However, RAM and ROM are also the most expensive storage devices and their storage capacities are the smallest. Data on RAM would also be lost when the power is gone. Therefore, random access storage devices are suitable for storing data that are small, transient, temporary, dynamic, and recoverable.

Since all stored locations are physically wired, data on these memories can be read and written without any mechanical movements. Time to access any given location is the same for all locations. Since the data access time is of the order of 10^{-8} seconds and this is much faster than other storage devices, we assume that the bandwidth of random access storage devices are unbounded in this thesis.

2.3.2 Sequential Access Storage Device

Sequential access storage device are memories that all the storage locations can only be accessed sequentially. Magnetic tapes, optical tapes and tape-like storage devices belong to this category.

The major advantage to sequential access storage device is their low media cost. The reason for such low cost is the separation of media surface from the heads assembly. All the array wirings in random access storage devices are eliminated and replaced with only one assembly of read/write heads to access the entire media surface.

The second advantage to sequential access storage device is their large storage capacity per physical volume. Since the storage surface is concealed by wrapping around a reel, whereas the platter surfaces are exposed in disks. The storage capacity per physical volume of a tape is larger than that of a disk. Thus, tapes provide one order of magnitude larger in storage capacity per volume than disks.

The complete separation of media surface from the heads assembly also allows for the most flexible choice in the number of head assemblies. When low data bandwidth is already enough, only one drive can be used. When higher data

bandwidth is necessary, more drives can be added. The necessary bandwidth can easily be multiplied by the number of heads assembly being used.

Unfortunately, tape drives have a much greater price range starting from several hundred dollars to hundreds of thousand dollars though tape storage are cheaper than disk storage. The high end tape drives are very expensive, and a few drives are often used for a large number of tapes. This price factor makes tapes mainly used for storing large amount of low activity data. Thus, tapes are used in many write-once and seldom read type of applications, like archival of data. Since its introduction, magnetic tape has been used as the medium for mass storage.

Due to the long access latency and high transfer rate, the tape drive throughput is low for random access but high for sequential access. Hence, tapes are often used in sequential data access type of applications, such as data archival and merging of sorted records.

2.3.3 Direct Access Storage Device

Direct access storage devices (DASD) are memories that require both a direct operation and a sequential operation in accessing data. The direct operation goes directly to one area of the total storage and the sequential operation moves over a smaller portion of the storage area [95, 102, 119].

Magnetic hard disks, floppy disks, compact disks (CD), and magneto-optical disks are common DASD. The digital versatile disks (DVD) emerge as the new storage medium for multimedia data. Floppy disks, CD, DVD, and magneto-optical disks are often exchangeable. Magnetic hard disks are often fixed inside the computers though some exchange boxes are commercially available.

Magnetic disks contain several circular platters connected together at the centre spindle. Data are recorded on the surface on the platters. The read/write heads are mounted on arms that move in parallel to the surface of the platters. The disk motor rotates the spindle and the platters at a fixed rotation speed. Each read/write head then hovers at a circular *track* above the surface of the platter. Each track is then divided into a number of *sectors*. The tracks on all platters are grouped together to form a *cylinder*.

(a) Disk Formats

Magnetic disks can be classified according to the recording density divided into Variable Density Recording (VDR) disks and Constant Density Recording (CDR) disks.

Traditionally, the same amount of data are recorded in each track of the disk no matter the track is close to or far from the centre. Since the outer tracks are longer than the inner tracks, data are packed together at inner tracks and spaced apart at outer tracks. The recording densities of inner tracks are then higher than the recording density of outer tracks (Figure 2.1). As the recording density is not uniform, these traditional disks are also called *Variable Density Recording* (VDR) disks.

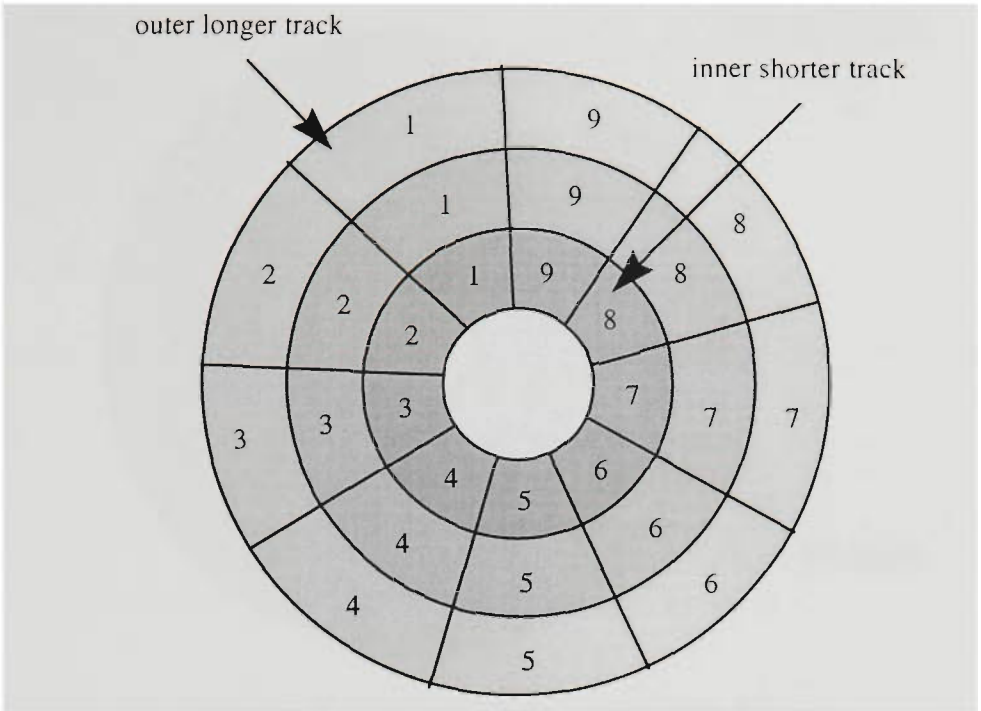


Figure 2.1. Variable density recording

VDR disks use simple fixed motor speed and fixed linear recording speed. However, the maximum amount of data that can be recorded on each track are limited by the length and recording density at the innermost track. As the longer outer tracks can store more data using the same recording density as the innermost track, storage capacities of the outer tracks are somewhat wasted. Hence, these disks suffer from low recording density and low storage efficiency at outer tracks.

As the need for disk storage capacity increases, a method in disk design now gaining wide acceptance is *Constant Density Recording* (CDR). The recording density of these CDR disks is maintained to be uniform across the disk surface [83]. This is achieved using either the *Constant Angular Velocity* (CAV) or the C*onstant Linear Velocity* (CLV) approach.

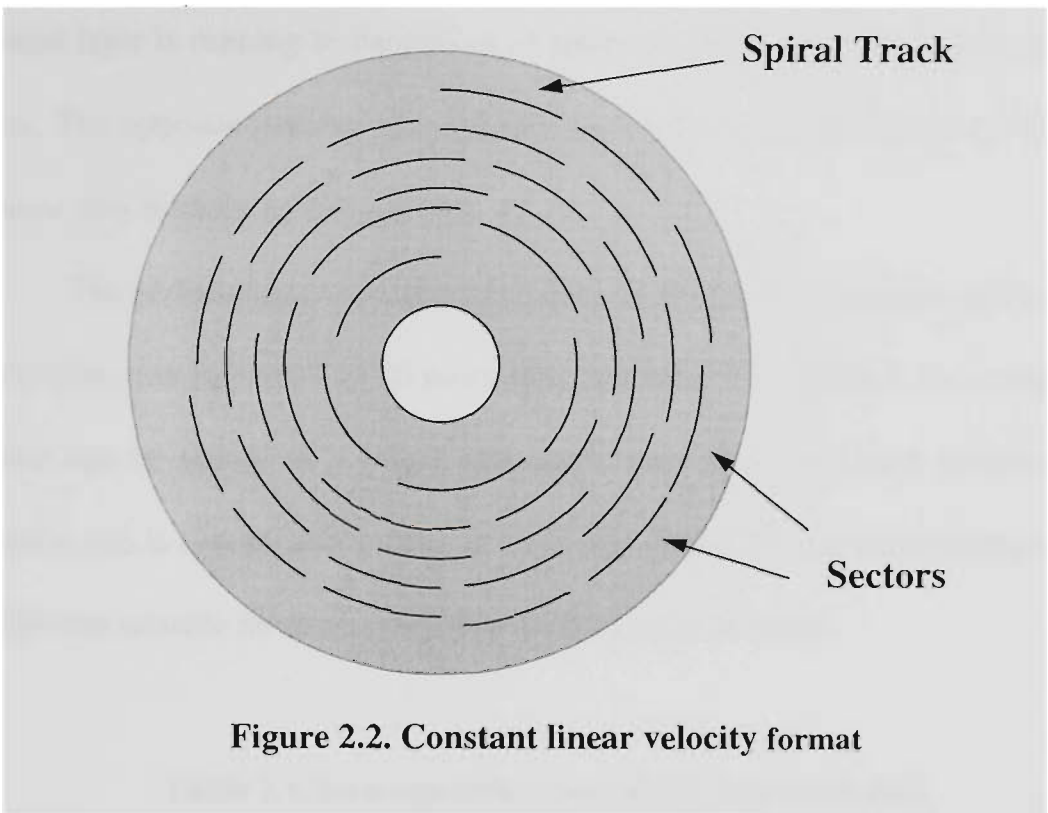


Figure 2.2. Constant linear velocity format

- **Constant Linear Velocity Approach**

Optical disks, such as CD and DVD, use the CLV approach. When the read/write heads move towards the centre of the disks, the drive motor accelerates to a higher speed. When the read/write heads move away from the centre, the drive motor slows down. The read/write heads are then maintained at a fixed linear velocity above the disk track. Data are written on the sectors of a long continuous spiral track on each side of the disk surfaces (Figure 2.2). As the read/write heads traverse the disk surface at a fixed linear speed, the same amount of data are then transferred in a fixed period of time. Hence, their data transfer rates on different tracks are the same whereas their track capacities vary [132].

A DVD may have one or two layers on each side of the disk. Each layer has one spiral track running from the inside to the outer rim of the disk. A single layer DVD can store up to 4.7 gigabytes of data on each side. A dual layer DVD has two spiral tracks on each side of the disk, one on each layer. The track on the

second layer is running in parallel or in opposite directions with that on the first layer. The opposite running spiral tracks are used to provide seamless jump and reverse play features in video access.

The performance and capacity of optical disks are increasing rapidly [113]. DVD disks can provide 7 to 26 times the capacity of CD (Table 2.1). A two-hour movie can be stored on a single side single layer disk, and their standard data transfer rate is comparable to that of a 9X CD-ROM. These characteristics make DVD very suitable for storing multimedia data, such as video.

Table 2.1. Some specifications of CD and DVD disk

	CD	DVD-5	DVD-9	DVD-10	DVD-17
Layers	1	1	2	1	2
Sides	1	1	1	2	2
Capacity (gigabytes)	0.65	4.7	8.4	9.5	17
User data rate (10^6 bits/sec)	0.15	11.08	11.08	11.08	11.08

- **Constant Angular Velocity Approach**

The CAV approach is often used in magnetic disks. Each disk surface is divided into several concentric zones by grouping tracks of similar radii together. The same number of sectors are written to each track within the same zone [83]. As outer zones have longer perimeters, they contain more bits and sectors than inner zones (Figure 2.3).

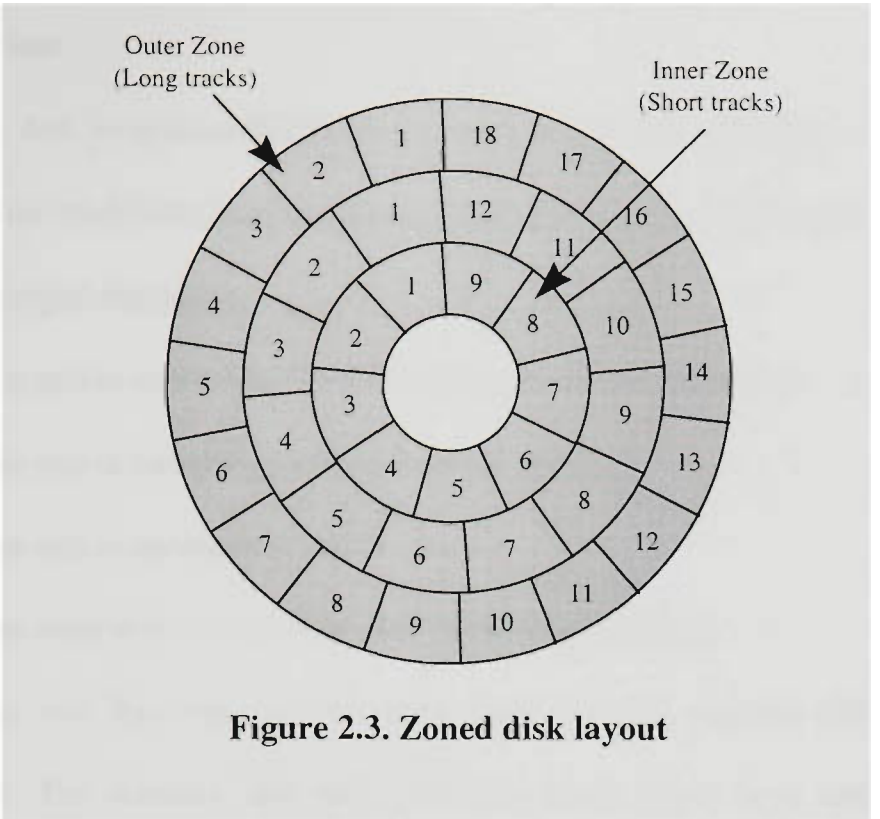


Figure 2.3. Zoned disk layout

Recording densities in each zone are optimized and are nearly equal. These disks are also called *zoned disks*. The disk platters rotate at a fixed speed and the read/write heads traverse data faster in the outer zones than in the inner zones. Hence, the data transfer rate is higher in the outer zones.

(b) Disk Drive Mechanism

The disk drive serves data requests using many steps. A simple model is to add the time components of the mechanical movements, including seek time, rotational latency and data transfer time, together as the disk request service time. Although more detailed model may include head switching time, channel waiting time and others [126], we do not include them since they are rather fixed and small.

- **Seek Time**

When the disk receives a SEEK command from the disk controller, it moves the arms and the read/write heads to the required cylinder. The seek action consists of the following components:

1. The arm is accelerated until it reaches the maximum speed.
2. The arm is travelling at the maximum speed.
3. The arm is decelerated until it stops.
4. The read/write heads are settled on the required track.

The heads will then stay on this track until the disk receives another SEEK command. The distance that the read/write heads travel in a seek action is commonly referred to as the *seek distance*. If all the seek destinations are random on VDR disks, the mean seek distance is $(b\text{-}a)/3$, where a and b are the radii of the innermost and outermost tracks respectively [95].

Seek time is the time required for the seek action and it depends on the seek distance. Frequently, the seek time is taken to be a linear function of the seek distance. The relationship between seek time and seek distance is generally non-linear. When the seek distance is short and the arm starts to decelerate before it reaches the maximum travelling speed, the seek time is proportional to the square root of the seek distance. When the seek distance is long and the arm needs to travel at the maximum speed, the seek time increases proportionately with the increase in seek distance [119]. The seek time s is shown in [126] to be

$$s = \begin{cases} a_1 + a_2 d, & d > a_5 \\ a_3 + a_4 \sqrt{d}, & d \le a_5 \end{cases}, \tag{2.1}$$

where d is the seek distance, and a_1, a_2, a_3, a_4, a_5 are fixed parameters.

- **Rotational Latency**

After the disk completes the seek action, it waits for the required data block to rotate under the read/write head. *Rotational Latency* is the time required for the starting position of the required data block to rotate under the read/write head. In most cases, rotational latency is assumed to be uniformly distributed and it is estimated as half of a disk revolution time. Letting T be the disk revolution time and l be the rotational latency, we have

$$l = \frac{T}{2}. \tag{2.2}$$

However, this is inaccurate for two reasons.

First, the rotational latency for fixed block size is not uniformly distributed because data transfer can only begin at fixed block starting positions and fixed angular positions. The sum of seek time plus rotational latency, hence, must necessarily be in integral multiples of basic block duration for consecutive accesses under busy conditions [93].

Second, some disks use *Rotational Positional Sensing* (RPS) to detect the rotational position of the disk platters. RPS gives an early warning when the data will rotate under the read/write heads. The I/O controller then looks for a free path from the disk to the memory. If such a path cannot be established within the available time, an RPS miss occurs. When this happens, it will need to wait for another full rotation before data transfer can start. The rotational latency is then increased by a full disk revolution.

- **Transfer Time**

After the I/O path is established and the desired data come under the read/write head, the head starts to transfer data. The read/write head sends data over a bit-serial wire to the I/O controller. The speed of transferring data must be the same as the speed of data coming under the head. Hence, the speed of data transfer is directly proportional to the recording density and rotation speed of the disk.

Data transfer time is the time required for the required data to be read. The data transfer time, τ, to transfer a data block of size D from a VDR disk can easily be shown to be

$$\tau = \frac{D \cdot T}{2\pi a \cdot \varepsilon}, \tag{2.3}$$

where T is the disk revolution time, a and ε are the radius and recording density of the innermost track respectively. Therefore, the data transfer rate, ρ, is

$$\rho = \frac{2\pi a \cdot \varepsilon}{T}. \tag{2.4}$$

- **Disk Throughput**

Many researchers use the term disk throughput or disk bandwidth to indicate the disk performance. Disk throughput is often defined as the speed of accessing data from the disks. The disk throughput, β, is given by

$$\beta = \frac{D}{s + l + \tau}. \tag{2.5}$$

or

$$\beta = \frac{1}{\frac{(s+l)}{D} + \frac{1}{\rho}}. \tag{2.6}$$

We can see from Equation (2.6) that disk throughput increases with increase in data block size D. Unfortunately, increasing data block size would also increase the buffer size. It is a tradeoff between disk throughput and available memory. Very often, the block size used in traditional disks ranges from half to a few kilobytes.

2.3.4 Disk Scheduling

Traditionally, requests are served in First-In-First-Out (FIFO) or First-Come-First-Serve (FCFS) orders. When requests arrive at the disk, they are placed in a waiting queue. When the disk is free, the request that arrives earliest is served. The disk often serves each request in an non-preemptive way without interruption from other requests.

The request that requires the shortest service time is served first in the Shortest Job First (SJF) method. Since a long request increases the waiting time of all waiting requests more than a short request, SJF scheduling chooses the shortest request in the waiting queue to be served. The total waiting time of all waiting requests is then minimized. However, long requests may wait indefinitely long when the disk is very busy.

Round-robin scheduling method places requests to priority queues. One or a few requests are served from each queue. This scheduling method tries to strike a balance between request priority and service arrival time.

The requests are served in the sequence of data location on the disk surface in SCAN scheduling or elevator scheduling algorithms. When disk requests are served, the service time of each request depends on the seek distance from the

previous request. The disk heads traverse across the disk surface and serve requests that accesses data on its way. This scheduling method efficiently reduces the total seek distance in serving a number of requests. The complexity of the SCAN algorithm is also low.

2.3.5 Disk Arrays

In order to store more data on a system, multiple disks can be used. The disks may serve requests in parallel or independently. When multiple disks are used as a disk array, data are striped across the disks. When data are accessed, each disk is issued a request. All the requests are then served simultaneously. Each request retrieves a fraction of the data block using shorter data transfer time. Hence, large data transfers are served efficiently.

Unfortunately, the mean time to disk failure shortens when many disks are used. In order to be able to recover data after disk failure, redundant data are encoded and stored. Data on the failed disks can then be recovered from data on other disks. This arrangement of disks forms a redundant disk array.

Redundant array of inexpensive disks (RAID) is an array of low capacity disks that store encoded redundant data to increase data reliability and data security. When a single disk fails, data on the failed disk are recovered from data on the remaining disks. There are at least six RAID levels [24, 63, 80].

RAID 1: Mirrored disks. The disks are arranged in pairs and each disk in the pair contains the same data. Only half of the available disk capacity is utilized for data storage and this is the most expensive option.

RAID 2: Bit interleaved array. Several correction disks are added to the group of data disks in a way similar to RAM chips. A single parity disk can detect a single error, but at least three disks are needed to correct an error. Larger group sizes yield a reduced overhead for fault tolerance, but smaller group means less I/O events per second. Since the whole group must be accessed to validate the correction codes, this is inefficient for small transfers.

RAID 3: Parity disk. Data are interleaved bit-wisely or byte-wisely across the data disks. Disk controller can detect the failed bit position and a parity disk contains the parity of the data disks. It is possible to recover data on any single lost disk by reading the contents of the surviving disks, and re-computing the parity. The disk array performance is similar to a RAID2 with a single correction disk.

RAID 4: Block interleaved. Every individual block is stored on a single disk. As the data are interleaved between disks at the block level instead of the bit level or byte level. The new parity is calculated as new parity (= old data xor new data xor old parity). A small write uses two disks to perform four accesses. Since all writes access the parity disk, contentions at the parity disk would suffer.

RAID 5: Rotated Parity. Parity blocks are interleaved among the disks in a rotating manner called Left-Symmetric. Two writes can take place in parallel as long as the data and parity blocks use different disks. This disk array performs better for small and large transfers, making it the most widely accepted level for transaction processing workloads [89].

RAID5 tolerates single disk failure in each parity group of disks. Data are lost only when multiple disks in the same group of disks fail. Gibson used Mean-Time-To-Data-Loss to measure the reliability of disk arrays and showed that RAID5 can increase data reliability [63].

RAID 6: **Two dimensional parity**. The disks are arranged into a 2-dimensional array and a parity disk is added to each row and each column of the array. This disk array can survive any loss of two disks and many losses of three disks. The only exception for three loss disks is that the data disk and both the parity disk and the column disk of this data disk fail at the same time. Since every logical write needs three disks and six accesses, the impacts on I/O performance are significant. Hence, this disk array is only acceptable for very highly fault-tolerant applications.

In most data storage on disks, data are not differentiated into read-write or read-only type. Read-only data are static and cannot be modified by the applications. Read-write data are dynamic and are frequently modified by the application. Read-only data are easily recoverable from elsewhere, such as tertiary storage. RAID addresses the problem of losing data under the conditions of disk failures. Under the condition that read-only data are recoverable easily, these redundant information could waste storage capacity and bandwidth.

2.3.6 Hierarchical Storage System

Hierarchical Storage Systems (HSS) are storage systems such that their storage devices are grouped into at least two hierarchical levels. The lower level of storage devices provide fast data retrieval at higher media storage cost whereas the higher level of storage devices provide cheap media storage at a slower data retrieval rate. In general, any two types of storage devices may form a storage hierarchy. Typical storage hierarchies include cache memory/RAM, RAM/magnetic disk, magnetic disk/optical jukebox, and magnetic disk/tape library. When more than two levels of storage hierarchy are used together, the storage devices are divided into primary, secondary, and tertiary storage devices.

A hierarchical storage system often refers to a storage system that uses RAM, magnetic disks, and a robotic storage library to form its storage hierarchy (Figure 2.4). The robotic storage library often consists of a magnetic tape library or an optical jukebox that exchanges tapes and optical disks automatically.

When data exhibit the spatial locality reference property, data in the neighbourhood of recently accessed data have a higher chance of being accessed. Hence, data are accessed in blocks of appropriate size in order that the consecutive accesses can be served together. When data exhibit the temporal locality reference property, data being accessed recently are likely to be accessed again. Data in the lower level should contain the data that are recently accessed to reduce the number of necessary accesses to the higher level. Therefore, the lower level storage devices are often used as cache of the higher level storage devices in order to provide an efficient hierarchical storage system.

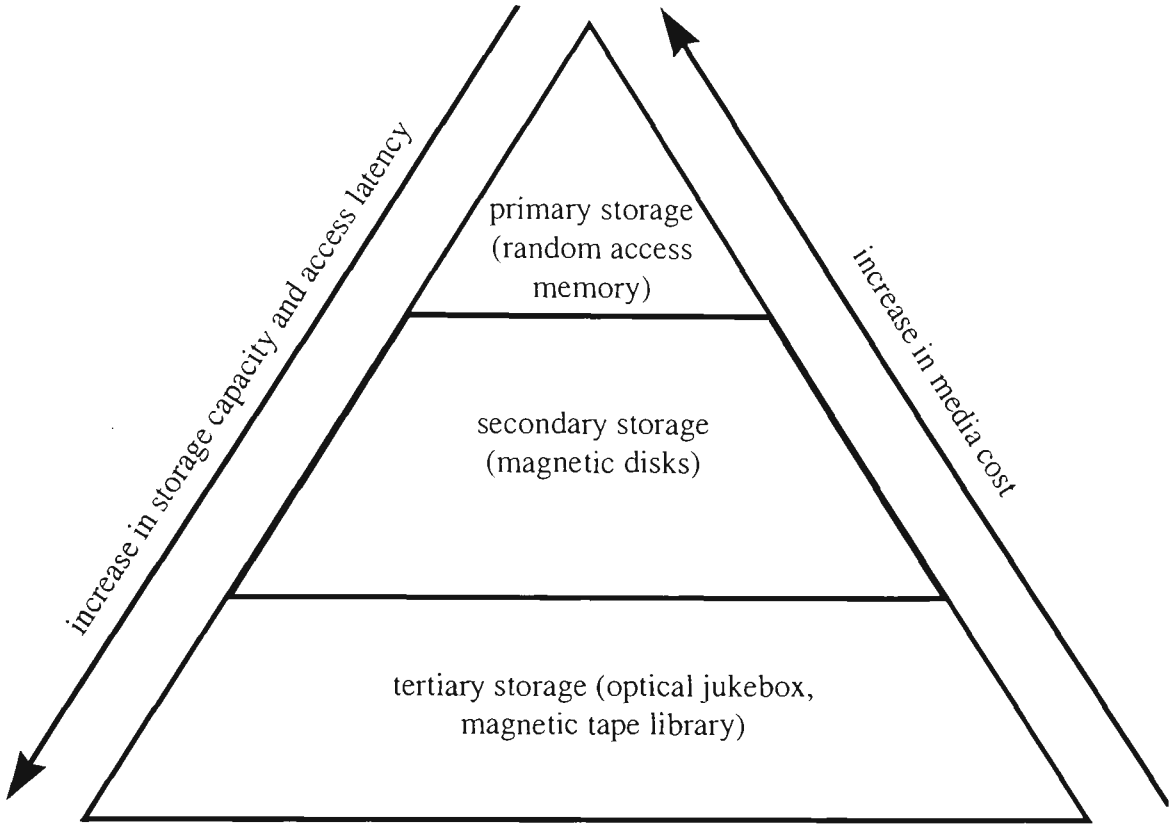


Figure 2.4. Hierarchical storage system

In order to be useful, each level of storage devices differs from the next level of storage devices by at least one order of magnitude in storage capacity, media cost, and access latency. The media cost, unit capacity and access latency of different types of storage devices are compared in Table 2.2 [72]. These values only show the order of magnitude instead of the actual range of current values.

Table 2.2. Comparison of types of storage devices

Storage device	Media cost	Unit capacity	Access time
random access memory	10^1 to 10^2 \$/MB	10^1 to 10^3 MB	10^{-8} sec
magnetic disks	10^1 to 10^2 \$/GB	10^{-1} to 10^1 GB	10^{-2} sec
optical jukebox	10^0 to 10^1 \$/GB	10^0 to 10^1 GB	10^1 sec
magnetic tape library	10^0 to 10^1 \$/GB	10^0 to 10^1 GB	10^2 sec

In order to provide sufficiently large storage capacity in tertiary storage libraries, data are stored on removable media units, such as tape cassettes or optical disks. The media units can be exchanged automatically using robot arms or motor driven handling units.

(a) Removable Media Units

A reason that tertiary storage libraries can provide huge capacity is that some media units are not directly accessible at the drives. Since each drive can retrieve data on only one media unit at a time, it exchanges and loads the media unit before data on the required media unit can be accessed.

The tape cells in magnetic tape libraries are often arranged in a two dimensional grid or cylindrical manner. Optical disks in jukeboxes are often arranged in a stack. CD changers arrange disks in a round table manner. Although each CD changer can access only a few disks, an array of CD changers may access data from many media units.

(b) Exchange Mechanism

Exchange devices in typical tertiary storage libraries include robotic arms of tape libraries, exchange devices of optical jukeboxes, and turntables of CD changers. When data on a particular media unit are required, four steps are performed:

1. The new media unit is moved from the cell containing it to the drive.
2. The old media unit in the drive is ejected or unloaded.
3. The new media unit is loaded at the drive.
4. The old media unit is moved to an empty cell.

Exchange devices can be dedicated to a drive or shared among drives. Magnetic tape library often uses only one or two robot arm for all drives. One exchange device is often shared among several optical drives in jukeboxes, whereas one exchange device is dedicated to each drive in a CD changer.

When the exchange device is a dedicated one, all drives can exchange media units concurrently. When an exchange device is shared among several drives, some drives may request for exchange while other drives are exchanging media. At this moment, the drive that issues new exchange requests need to wait for the exchange device becomes free. Since the drive cannot serve any requests while waiting for exchange, this contention for exchange device wastes substantial drive throughput.

(c) Time to Exchange Media

Media exchange time is the time required for the media unit to be exchanged. The exchange devices and the drives execute several mechanical movements. It takes time for the exchange device to fetch and move the media units, and for the drive to unload the old media unit and load the new media unit.

A plausible model is to add the time components of the mechanical movements together. We do not include other time components since they are much smaller than these mechanical movements. Typical times to perform exchange requests are shown in Table 2.3. The HP jukebox is the HP SureStore Optical Jukebox [75] and other columns are tape libraries [39, 67].

Table 2.3. Typical exchange service times (seconds)

Movement	Exabyte	StorageTek	Ampex	HP Jukebox
mean rewind time	75	30	13	-
drive eject time	17	5	4	3.0
robot exchange	21+22	15+15	<20	10
drive load time	65	7	5	5.5
exchange service time	200	72	42	18.5

2.3.7 Data Migration

Random access latency on tapes is three to four orders of magnitude slower than that on disks. Hillyer and Silberschatz studied I/O scheduling methods on serpentine tapes. They proposed the LOSS (a greedy algorithm for asymmetric travelling salesman) algorithm which performs better than other scheduling methods except the NP-hard optimal algorithm [72].

Although data migration requests can be scheduled using disk scheduling methods, their performance may become worse off due to the presence of large media exchange overheads. Therefore, data migration methods on tertiary storage devices are studied separately.

Less data on a storage level reduce the storage cost, but more requests are directed to the higher levels where requests are served slowly. Conversely, more data on a storage level allow more requests to be satisfied at that level, but storage cost is inevitably raised. Because of these conflicting requirements for inexpensive storage and high performance retrieval, efficient techniques are needed to control the migration of data between storage levels. This is viewed as a file migration problem or a multilevel cache management problem [73].

When data are placed on a storage hierarchy, tertiary storage devices provide permanent storage of all files at the lowest cost. Using a process of *staging* in traditional systems, files are copied to the disks when required; these data on disks become accessible to users and can be deleted later.

Traditional tertiary storage libraries migrate files across the storage hierarchy by the following steps:

1. (Start of stage one) Data are copied from tertiary storage devices to primary memory.
2. Old data are erased from disk cache to reclaim space if necessary, data are copied from primary memory to disk cache, and data are erased from primary memory to free primary memory for other use.
3. (Start of stage two) Data are copied from disks to primary memory.
4. Data are consumed from primary memory.

Since data are available in memory after stage one, the materialization time is the earliest time when data become accessible. The materialization time or time to complete stage one is given as

$$w + \chi + \frac{O}{\phi}, \tag{2.7}$$

where w is the queuing time for free drive, χ is the sum of media exchange and reposition time, O is the object size, and ϕ is the data transfer rate [42].

2.4 Multimedia Storage Systems Characteristics

Optimization of system performance should consider the particular characteristics of the running application in order to deliver an efficient system. Application characteristics that are related to data storage include the data characteristics, access pattern of data requests, and specific requirements for the type of application. The file sizes directly affect the amount of time to transfer data from the storage systems. Request arrival patterns affect the contention of requests for storage devices. The application requirements limit the choices among methods that exhibit different performance characteristics.

Multimedia applications have many different requirements from traditional computer applications [9, 47, 69, 70, 71, 82]. New multimedia applications, including digital library, video-on-demand system, multimedia process control system, visual information system, multimedia information systems, and others, require to store multimedia data on multimedia storage systems. Multimedia storage systems are different from traditional computer storage systems in several aspects [49, 132, 143, 145].

2.4.1 Multimedia Data

The storage requirements of many applications are huge and most data kept in these applications are multimedia data. For example, the Thomas Jefferson National Accelerator Facility is expected to generate roughly 250 *terabytes* (250 x 10^{12} bytes) of raw and experimental data each year. A total of one *petabyte* (10^{15} bytes) will be stored on-line by the year 2000 [13]. Researchers in NASA have projected their storage requirements to be 2 petabytes in 1999 [127]. The only

practicable and economical solution to store these multimedia data is to use tertiary storage devices. We first describe the multimedia objects and their data sizes in part (a).

Multimedia objects should always be compressed and kept in the compressed format because of their large object size. Many methods are already available to compress multimedia data. Lossless compression techniques, such as Hoffman encoding, LZW, and wavelet, can reduce images data to one third of original data size. Lossy compression standards like JPEG (Joint Photographic Experts Group) and MPEG (Motion Picture Experts Group) standards can compress multimedia data to a much smaller size without much noticeable effects on novice users. Typically, the JPEG standard can compress images in the ratio of 15:1 [49]. We briefly describe the MPEG compression standards for video in part (b).

Even though compressions can reduce the original size by one to two orders of magnitude, multimedia objects are still too bulky to be handled in its entirety. Hence, they should be partitioned into smaller units. Multimedia data can be striped while keeping either the display time or data size constant. These two approaches to perform data striping are compared in part (c).

(a) Data Size

Image object size increases linearly with each dimension of the image resolutions and the colour depth. High resolution images are much larger than low resolution images. A full screen colour image may require 1024x864x24 bits or 2,654,208 bytes to represent.

Audio objects store the information of recorded sound waves. The size of an audio object is proportional to the resolution, the frequency, and the duration of the object. A 60-minutes of 16-bit stereo audio is represented with 60x60x44100x2x16 bits or 605 MB.

Video objects are composed of a series of image frames displaying at a fixed rate. Each frame contains an image object and the frame size depends on the image resolution and the colour depth. The video object size also depends on the frame rate and the display duration. A two-hour colour PAL video may be represented with 576x384x24x25x60x60x2 bits or 119,439,360,000 bytes.

Since the data requirement increases linearly with each dimension of the resolutions, high resolution objects require many times more data storage space than the low resolution and preview objects.

(b) Compression

Since the amount of data to represent multimedia objects are so large, compression is inevitable in all levels of processing multimedia data. In MPEG compression, an object is organized as a video sequence. A video sequence consists of a sequence header followed by a number of Group of Pictures (GOP). Each GOP is composed of a GOP header followed by a number of pictures. Each picture can be an intra-frame (I-frame), a predicted frame (P-frame) or a bi-directional predicted frame (B-frame).

An I-frame is actually the discrete cosine transform (DCT) based compressed image of the picture. A P-frame contains the motion compensation vectors that predict their values from the previous I-frame or P-frame. A B-frame

contains the vectors that predict its value from both the previous and the following I-frame or P-frame. Hence, a B-frame is always placed after the previous and the following I-frame and P-frame within the coded stream [49]. If the ratio of I:P:B frames is 1:3:8, then this sequence has one I-frame, three P-frames, and eight B-frames in every 12 frames. Since P-frames and B-frames contain motion compensation vectors that show only changes in the image content, the P-frames and B-frames are often smaller in size than the I-frames. The video data can be compressed by factors of 30:1 to 200:1 and audio data can be compressed by factors of 5:1 to 10:1 using MPEG compression [49].

Other video compression methods, such as H.261 and the new XYZ compression, could also reduce the data storage requirement by one to two orders of magnitude [51].

Compression techniques definitely can reduce the size of multimedia objects. Despite multimedia objects can be compressed to two orders of magnitude lower than that of its original size, the compressed objects are still very large. Their storage requirements in primary, secondary, and tertiary memories are vast, necessitating huge storage capacity on all storage devices [13, 49, 71, 127].

Multimedia data have the characteristics that portions of data are required at a later times during retrieval. This is commonly known in sequences of video and audio. It is also exhibited in compression methods such as, progressive JPEG and multi-resolution video, that progressively increase the object resolutions [43, 121].

In the progressive compression technique, the system initially displays a low resolution image. It then displays a higher resolution object and the image resolution keeps increasing until the full resolution object is displayed.

(c) Data Striping

As compressed multimedia objects are still large, each object is often partitioned into smaller units called segments before they can be handled efficiently. Each segment is further subdivided into smaller units called data stripes. Each data stripe is accessed by one data requests. Data striping can be applied using two approaches, the Constant Data Length (CDL) approach or the Constant Time Length (CTL) approach [17].

If the CDL approach is used to create fixed length data stripes, it is difficult to predict the display time of each data stripe. The fixed data stripe size does not always form a multiple of the atomic unit for consumption, such as a frame. If an atomic unit spans several data stripes, data consumption may only start after all data stripes of the unit are transferred.

If the CTL approach is used, the display time of every data stripe is fixed. Each request is separated from the previous request of the same stream with a fixed unit of time. Since the access latency already fluctuates from request to request, it is not worse off to vary the data size and transfer time as well. Since the variable data stripe size is an atomic unit for display, the display can always proceed after a data stripe is retrieved. Although this variable data rate property increases the maximum buffer size, the buffer size can still be averaged out by sharing with other streams. Therefore, the CTL approach is more preferable.

When the display duration of a data segment is long, more data stripes are created. The sizes of each data stripe vary from kilobytes to megabytes. If each MPEG frame forms one data stripe, the size of data stripes fluctuates depending on the type of frame. If each data stripe contain one GOP, the size of each data stripe depends on the amount of movements within that GOP. Hence, the size of data stripes can be more steady when data stripes are larger [87].

2.4.2 Display Modes

Multimedia applications often provide an interface to control the presentation of multimedia data over the time dimension like a Video Cassette Recorder (VCR). Hence, multimedia applications provide functions, such as play, fast forward, fast reverse, pause, stop, and slow, like a VCR. We shall describe how our design can support these display modes in Chapter 3.

(a) VCR like functions

When users display multimedia objects in normal mode, data of the object are consumed in a fixed rate. Consecutive data stripes of the object are consumed one after another. This speed in displaying data is normally the same as the speed in recording data. When a user displays multimedia objects in the fast forward mode, he skips certain parts of the data object to search the start of the desired part of the object in a short time. Sometimes, the user may not be sure of how far is the desired part from the current displaying position. He displays data in a fast mode so that he can pause or stop displaying the object near the desired position. In order to support fast forward mode, data are supplied in one of the ways below.

1. The data object is supplied in a faster speed. This requires higher bandwidth in accessing the object.
2. A fast forward object is retrieved instead of the normal data object. This would require extra data storage space.
3. A low temporal resolution part of the object is accessed. This low temporal resolution part is displayed in a normal speed. Multi-resolution compression methods provide this low temporal resolution objects as part of the compressed objects.

No matter which method is chosen, the user can search the object and resume display at the desired position.

After the user has finished searching data using either fast modes, the user would resume to normal display mode. In some designs such as pipelining, there is a delay in resuming back to the normal mode if only part of the object has been transferred. This resume latency can be similar in magnitude to the latency of pipelining the object.

If the user wishes to skip to an earlier part of the object, he displays data in a fast reverse mode. If data can be retrieved in an reverse sequence, the fast reverse mode can be handled in a way similar to the fast forward mode. Otherwise, a reverse object has to be created and stored.

Since the pause and stop functions can be served without additional data from the storage system, we assume that they can be handled by the operating system alone. Other VCR functions, such as slow motion and frame advance functions, consume data at a slower rate than that in the normal mode, it is

reasonable to assume that these display modes can also be supported with data buffering.

(b) Browsing Interface Functions

Many multimedia applications need to provide a browse mode for users to have a brief view before watching the details [41, 114]. This browse mode could reduce the workload on the storage system by skipping unnecessary details.

For continuous media, the beginning part is often played as a preview of the object. This method can only give the user a very rough idea of the object contents. When many objects begin with similar contents, the preview data does not provide enough distinguishable information. Alternatively, a preview method with some data from the content of the object is more informative. Multi-resolution compressed objects can provide such preview with their low temporal resolution segments without much additional data [43].

After the user has browsed the multimedia object, he may select one of the low resolution segments for displaying. The system should then display the high resolution objects beginning at the selected segment. In this situation, the storage system needs to jump to and retrieve data beginning at the appropriate location.

2.4.3 Access Pattern of Multimedia Data Requests

Multimedia data are accessed differently from traditional computer data. Since storage system throughput depends heavily on the data access pattern, the access pattern of multimedia request streams should be investigated before system performance can be optimized.

(a) Access Frequency

Several studies characterized user access pattern on video-on-demand systems according to the Zipf's Law [28]. Contrary to these research results, Griwodz *et al.* showed that the access probabilities of multimedia objects are sometimes less skewed [68]. Nevertheless, Zipf's distribution is still the most popular way to model multimedia object access frequencies.

In Zipf's Law, the probability of choosing the nth most popular of M objects is taken to be C/n where C is a constant. We list the Zipf distribution for 1000 objects grouped by 100 objects in Table 2.4. The skew column shows the ratio between the lowest to the highest probability within each group. Although the probabilities of the first 100 objects are highly skewed, the probabilities for the last 800 objects at the tail of the Zipf distribution are much less skewed. If the first 200 objects are resident on disks, then requests for only the last 800 objects are directed to the tertiary storage. Thus, access probabilities of objects in the tertiary storage are less skewed.

Multimedia objects are commonly divided according to the frequency of access into hot, warm, and cold objects. Hot objects are frequently accessed by a number of users. Hot objects are likely to be accessed again by new users before it is completely displayed to existing users. Warm objects are accessed less frequently. They are less likely to be accessed again before its display duration. Cold objects are seldom accessed. Griwodz *et al.* showed that data temperature of movies increases quickly at the time of release and drops slowly over an extended period of time [68].

Table 2.4. Zipf distribution of 1000 objects in groups of 100

Group	Highest	Lowest	Skew
1 to 100	13.3592%	0.1336%	0.0100
101 to 200	0.1323%	0.0668%	0.5050
201 to 300	0.0665%	0.0445%	0.6700
301 to 400	0.0444%	0.0334%	0.7525
401 to 500	0.0333%	0.0267%	0.8020
501 to 600	0.0267%	0.0223%	0.8350
601 to 700	0.0222%	0.0191%	0.8586
701 to 800	0.0191%	0.0167%	0.8763
801 to 900	0.0167%	0.0148%	0.8900
901 to 1000	0.0148%	0.0134%	0.9010

- **Read/Write Frequency**

Multimedia objects are commonly created by specific equipments such as video camera and audio recorder. Before multimedia objects are presentable, clips of the objects are cut and pasted together to produce the final product. While professional expertise is often required in the post-production process, novice users may retrieve objects. Thus, more users would retrieve multimedia objects whereas fewer users would modify them. While only read operation is required in displaying objects, both read and write operations are required in modifying objects. Multimedia objects are hence more often being read than being written.

Once data are placed on the storage devices, performance of all object retrievals is affected by the efficiency of the storage system. Therefore, efficiency of retrieving multimedia objects is more important than that of modifying them in many applications.

(b) Stream of Requests

In traditional computer systems, data requests are discrete and discontinuous, and requests are normally independent of each other. On the contrary, multimedia data requests appear like an aperiodic stream. When a multimedia object is requested to display, a new stream is created. The stream generates new data requests that often access adjacent data stripes of the object.

- **Real Time Constraints**

Multimedia requests are bounded by real time constraints. Since multimedia data are displayed continuously to users, the supply of data should be continuous. If data are fed slower than the consumption rate, starving will occur and jitters will appear. If data supply is too fast, buffer may overflow. If data supply is discontinued, the display freezes. These artifacts undesirably affect the quality of service to users.

In particular, the execution time of requests is hence limited by the continuous display requirement such that the retrieval of a data request should complete before the previous data stripe is displayed. Let s be the seek time, l be the latency, M be the media block size, ρ be the data transfer rate, N be the number of disks in an array, and δ be the display bandwidth. In order to support real-time continuous display of the media stream from disks, the retrieval time of one media block from each disk must be less than the display time of N media blocks. The continuous display constraint for N disks to support a single stream is

$$s + l + M\rho < N\delta. \tag{2.8}$$

If the maximum seek time and the maximum latency are used, this constraint provides a *hard guarantee* for the streams; i.e. the data retrieval time is 100% guaranteed against starving.

- **Read-ahead buffering**

A mechanism to handle variations of data retrieval time called *read-ahead buffering* is proposed in [36]. A number of read-ahead buffers are first filled before display or consumption begins. If k read-ahead buffers are used, then buffers for $N+k+1$ data stripes would be required. Increasing the number of read-ahead buffers per stream enables the system to absorb higher variability in disk response time. The same study showed that triple buffering is already sufficient to achieve the best cost-performance.

- **Requests are separated evenly**

Since each stream issues request periodically, consecutive requests are separated evenly from each other. Hence, the arrival pattern of requests is more regular than the random arrival pattern.

The Poisson distribution is often used to describe the random arrival pattern in traditional systems. Unfortunately, it becomes a less accurate model for multimedia systems. Although the streams arrival pattern is random and can be described by the Poisson distribution, the request arrivals are more regular than the random arrivals. Since there are no accurate models available, we still use the Poisson distribution for request arrivals to find a plausible waiting time.

- **Access neighbouring data**

When data are displayed in the normal mode, data stripes are accessed according to the temporal dimension. Consecutive requests of the same stream access adjacent data stripes of the same object. They may access data stripes with a fixed time gap from each other when data are displayed in the fast mode. The storage location for neighbouring data stripes of an object is hence considered by many researchers to create highly structured storage organizations.

- **Continuous over an extended period of time**

A stream may issue requests over an extended period of time. Short video streams may be 5 to 10 minutes in length. Long video streams, such as movies, can be over two hours. Security video cameras may even record data continuously with indefinite length. Requests that access images being compressed using progressive compression techniques are also issued continuously over a short period of time.

(c) Streams Synchronization

Multimedia composite objects are made up of several component objects [21, 111, 112]. Each component object may belong to a different media type. An example is given in Figure 2.5. These media components are separate at input and arrive at the storage devices as different media streams. They are usually retrieved from the storage devices at similar times and merged prior to consumption.

The request streams for the component objects need to be synchronized before display. Otherwise, artifacts such as video of a talking person without lip

synchronization may result. Thus, the storage system that can handle multiple streams are more suitable for multimedia composite objects.

When synchronized objects are stored on storage system, the data stripes can be placed in an interleaving manner. Each stream can still be served independently. When several objects are retrieved at the same time, the retrieval efficiency is increased by merging the streams of requests. The feasibility of merging streams on optical disks is studied in [122, 151].

When synchronized objects are accessed in parallel, the objects can be retrieved in shorter total time by optimizing performance of accessing multiple objects together. [5, 57]. Since some buffers may already be required to perform synchronization, no additional buffers would be required to access them in parallel. Hence, parallel retrievals may not use much additional resources in accessing synchronized objects.

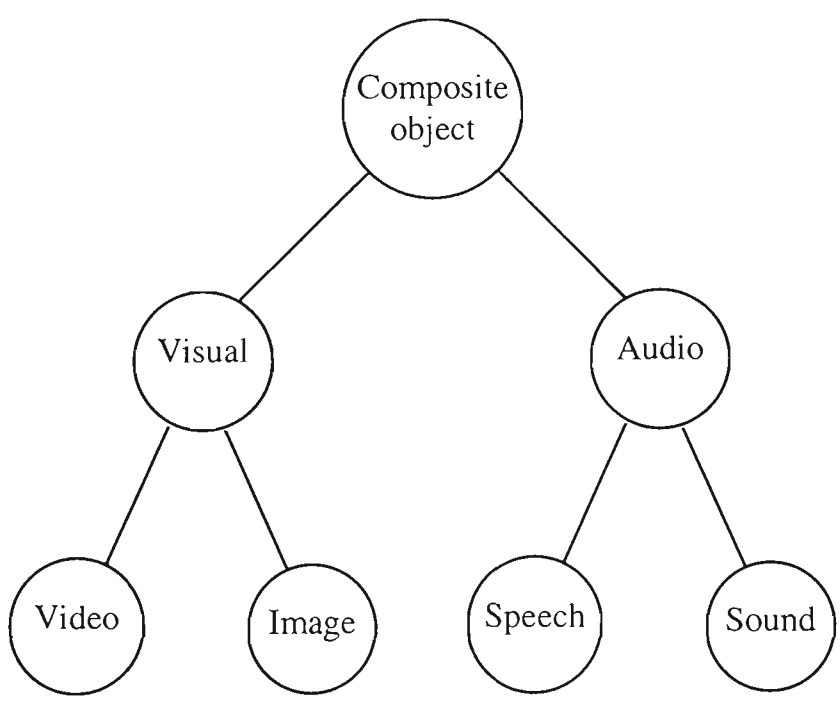


Figure 2.5. Composite multimedia object

- **Requests Scheduling**

Instead of providing hard guarantee that limits disk throughput by the maximum seek, some researchers provide *soft guarantee* for multimedia streams. The requests are queued for disk accesses, and the waiting request is chosen according to a scheduling policy that is aware of the deadline of the waiting requests. The disks may achieve higher throughput, but the quality of streams can sometimes be lowered.

The Earliest Deadline First (EDF) algorithm serves the request with the earliest deadline first in order to satisfy its real-time constraint. Since requests may arrive in any order, seek overheads are significant. The SCAN-EDF algorithm combines seek optimization techniques and EDF together. Requests with the earliest deadline are served first. Requests that have the same deadline are served according to their track locations on the disk following the SCAN algorithm [123, 124].

The Group Sweeping Scheduling (GSS) algorithm partitions each round of requests into groups and each stream is assigned to one group. The groups are served in a fixed sequence within each round. The requests within each group are served according to the SCAN algorithm. The round length is hence traded off with the latency of successive stream retrievals by optimally deriving the number of groups [18, 22, 55].

Some researchers proposed the piggyback technique to merge request streams for the same object. They use slow drifting to merge requests that arrive at slightly different times. The earlier request is slowed down and the latter request is speeded up until both requests are served together. Using this method, the

workload on the storage system can be reduced [1, 66]. This piggyback merging technique is hence suitable for merging concurrent streams on the hot objects.

2.5 Storage Organizations on Disks

Many intelligent storage organizations, or data placement methods, have been designed for traditional data files and database systems. Traditional file placement methods are grouped into the following strategies [90]:

1. **Naïve placement:** Each data file is split into blocks and the file blocks are randomly placed on any storage location. This is the simplest strategy to handle random accesses to file blocks.
2. **Contiguous placement:** Each data file is stored to contiguous physical locations. This strategy performs best when the entire file is accessed by consecutive requests. However, fragmentation prohibits the placement of large files.
3. **Type based placement:** Files containing the same type of data are grouped to the same category. Files of the same category are placed close to each other. This strategy trades off the seek distance of consecutive requests on data of the same type with that on data of different types.
4. **Frequency based placement:** Files are sorted according to their stationary probabilities and placed in an organ-pipe manner with the most frequently accessed files in the middle of the tracks. This strategy needs to record the access frequency of files in order to reorganize the data files.

5. **Markovian Placement:** Data files with the highest transition probabilities are stored to consecutive locations. This strategy optimizes the seek distance of requests according to the access history.

Many storage organizations and data placement methods that are specifically designed for multimedia data can be found in the literature. We also group them into the following strategies.

1. **Naïve placement:** Data stripes are stored randomly. This simple method is used for comparison only.
2. **Contiguous placement:** Objects are physically stored to consecutive physical locations.
3. **Log structure placement:** All write operations are appended to the end of a log file.
4. **Statistical placement:** Objects are stored according to the stationary or transition probabilities.
5. **Fault tolerant:** Redundant codes are encoded and stored to increase data reliability and security.
6. **Replication:** Objects are fully or partially replicated to increase availability of data.
7. **Striping:** Objects are partitioned to allow round robin or parallel retrievals.
8. **Constraint Allocations:** The physical storage locations are limited to reduce the maximum overheads in serving consecutive requests.

These data placement methods are described in the following sections.

2.5.1 Naïve Placement

This placement method places data blocks to random locations on disks. The only benefit of this method is simple. The system can find any empty space to store the data blocks. However, each request's response time can be very long and the system throughput can be very low.

2.5.2 Contiguous Placement

Some researchers placed multimedia objects contiguously on disks. Since the entire object can be accessed with only one seek action, the overheads in data access time are very low when objects are accessed in its entirety [2, 26].

Unfortunately, the disk becomes fragmented after deletions and insertions. New objects cannot be inserted due to lack of contiguous disk space. Data need to be reorganized to create sufficiently long contiguous empty space. If objects are modified, extra copying of unmodified portions is required. The copying overhead erodes disk throughput when multimedia objects are modified [99]. Also, multimedia objects are large and bulky. If the object is accessed in its entirety, the memory buffer must be very large in order to store the entire object. Each request also takes a long time to finish and may create long queues of short requests. If requests of concurrent streams are served in an interleaving manner, the disk heads would move to-and-fro between physical locations of the objects and excessive seek overheads are incurred.

2.5.3 Log Structure Placement

Some researchers store multimedia objects using log structure files. The disk layout consists of an index table and an append-only log table. The index table stores management information describing the log table status that is composed of a log head pointer and a logical log address pointing to physical disk block map table. Each log file is organized as a simple list of data blocks written contiguously to disk [99].

Since the log structure files are stored contiguously on disks, the seek overheads in writing them are low. Unfortunately, this placement method cannot guarantee performance on reading modified data blocks. Since all modified data blocks are appended to the tail of the log table, they require extra seek time overheads during retrieval. Therefore, this placement method is limited to multimedia applications without much editing.

2.5.4 Statistical Placement

In order to reduce the average seek time of disk requests, access probabilities have been considered in designing optimal file locations [45, 138]. Since the access frequencies or data temperatures of multimedia data can be obtained from prediction or access history, some researchers distribute movie data among disks following the access frequencies [98]. Some researchers store multimedia data objects on constant density recording disks according to the access frequencies [27, 136, 147].

The hottest object is placed at the optimal position that has the minimum random access time from all positions on the disk. The next hottest objects are

then placed at the next optimal position available. The objects are stored following a skewed organ-pipe pattern (Figure 2.6). When all objects are independently accessed and in its entirety, the mean data access time is minimal.

If the access frequencies of objects are obtained from access history, extra disk storage space is required to store the access history. Fortunately, the data temperatures of objects in some multimedia applications can be predicted. However, their data temperature dynamically changes over time whereas the placement methods are static [68]. In order to maintain the optimal condition, data need to be reorganized frequently.

This strategy assumes that objects are independently accessed. However, a stream of requests sequentially accesses data stripes of the same multimedia object. When there are several concurrent streams, the disk heads traverse to-and-fro between storage locations of objects. Therefore, this strategy should be refined to a finer granularity in order to handle concurrent multimedia streams.

When multimedia objects are not accessed independently, the transition probabilities must be considered. The Markov chain is used to model the access patterns for browsing graphs with low connectivity. A heuristic algorithm has been proposed to place the objects [20]. The running time of the proposed heuristic algorithm is $O(n^3)$.

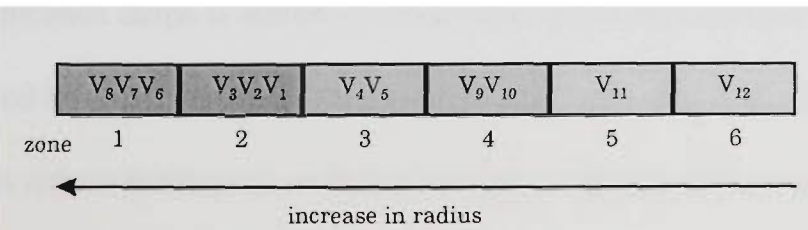


Figure 2.6. Frequency based placement

When objects are stored on tertiary storage and temporarily placed on staging buffers, the objects are written once and then read back once. After these two data accesses, the objects can be deleted to release disk space. The number of data accesses of each object on the disk is hence different from the object temperature in the placement. Moreover, the individual access frequency of these objects may be very low, but the staged buffers are accessed more frequently than other resident objects. If these buffers are allocated at the two ends of the disk, the mean seek distance would be very long. Therefore, these placement methods need to be refined for finding the optimal locations of staging buffers.

Similar to frequency based placement, extra storage space is also required to store the access history. The presence of concurrent streams and the continuous display requirement render that these statistical placement methods should be enhanced to handle streams of requests for multimedia data.

2.5.5 Fault Tolerant

Some researchers investigated the storage of redundant information on disks to cater for data recovery after disk crash [32, 137]. RAID is widely accepted in recent years and some researchers proposed the Streaming RAID to serve multimedia streams. In Streaming RAID, every data block is split into a number of data stripes and each stripe is stored to consecutive physical locations on one disk. The Segmented Information Dispersal method adds check data to the data contents of the disks to reduce the impact on disk performance during data recovery.

A limitation of RAID is its lack of control on the placement of multimedia data in storage. Hence, the data bandwidth cannot be effectively controlled and large variations in data bandwidth require more read-ahead buffer.

Storing redundant information obviously increases the security of data during disk failure. Although the security of multimedia data should not be neglected, proper backing up and archiving data can also achieve the data security. Since most multimedia data are not frequently modified and storing the redundant information would reduce the data bandwidth, redundant codes can be considered when surplus bandwidth is available or other means to provide data security are not available.

2.5.6 Data Replication

Several researchers replicate data to achieve better disk performance. In the Lancaster continuous media storage server, object files are replicated according to their distance from the originating site [100]. Ghandeharizadeh *et al.* proposed to migrate request with data replication across disk clusters in order to reduce start up latency [60]. Chang and Molina replicated the leaders of multimedia objects on a separate disk to reduce start up latency in constraint allocation methods [19]. Korst proposed to duplicate objects on randomly selected disks according to the access frequency of the objects [86].

Ghandeharizadeh and Ramos also proposed data replication to avoid disk multitasking that could reduce disk throughput [57]. When multimedia objects are declustered across a group of disks, all the disks in the group must be accessed simultaneously in order to retrieve the objects in real-time. In order to maintain the

disk retrieval throughput at the desired level, the number of disks to decluster each multimedia object is limited. Due to the limitation of disk throughput, each disk can only support a limited number of streams. The contention of streams for disk bandwidth could reduce the throughput. In [57], data are replicated in other disks to reduce this contention. An example is given in Figure 2.7.

A general requirement of all data replication methods is that extra storage space is used. When the disk array is bandwidth bound, the usage of vacant space to raise throughput is possible. This strategy is however limited by the amount of free space available.

Unfortunately, multiple data copies should be synchronized while they are modified. The synchronization of multiple data copies on disks increases program complexity and workloads. The selection of data to replicate and the selection of disks to place the replica should be optimized to achieve the sufficient gain against the extra workloads.

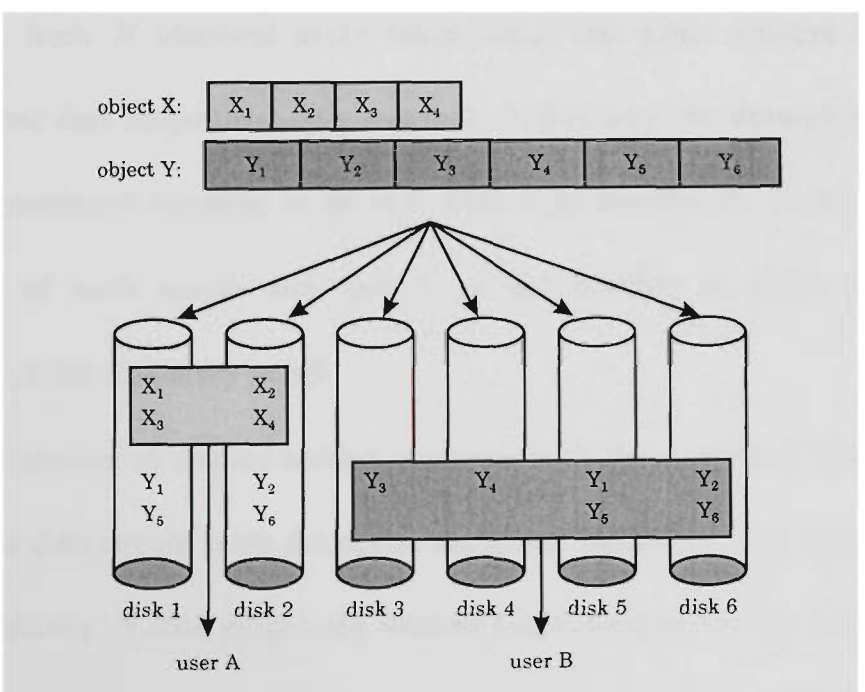


Figure 2.7. Data replication

2.5.7 Striping

Since each stream needs continuous data supply, the aggregate data bandwidth of many streams imposes very high demand on bandwidth of multimedia storage servers. A strategy to provide higher disk throughput is to store data stripes across multiple disks and access them back in parallel. This *disk striping* or *data striping* technique is similar to data striping in RAID except that no redundant data are stored [31, 76]. When an object is striped across N disks, the first data stripe is placed on disk 1, the second data stripe is placed on disk 2, and so on. In general, the ith data stripe is placed on disk $1 + (i\text{-}1) \bmod N$ (Figure 2.8).

When the data stripes are accessed, one disk request is sent to every disk in the cluster at the same time. While the first disk is repositioning its read/write heads to the desired location, the second to the last disks are also repositioning their read/write heads to the desired locations. One data stripe is then transferred back from each disk to the memory buffers. Hence, the time required to retrieve N data stripe from N identical disks takes about the same amount of time as retrieving one data stripe from only one disk. In this way, the throughputs of all N disks are combined together to provide high data bandwidth. Letting β be the throughput of each single disk and N be the number of disks, the overall throughput of the disk array is $N\beta$.

The amount of display buffers increases with the number of disks. In order to retrieve a data stream from the disks, minimum buffer for N+1 data stripes are required. Initially, N data stripes are fetched from the disks to the N buffers. The (N+1)th buffer is started to fill from disk while data in the first buffer is being consumed. After the first buffer is exhausted, the (N+2)th data stripe is started to

fill and the second stripe is consumed. After the jth data stripe is exhausted, the $(N+j)$th data stripe is retrieved while the $(j+1)$th data stripe is consumed. Thus, the whole stream can be retrieved using $N+1$ buffers and the time to fill N buffers is the initial start up latency.

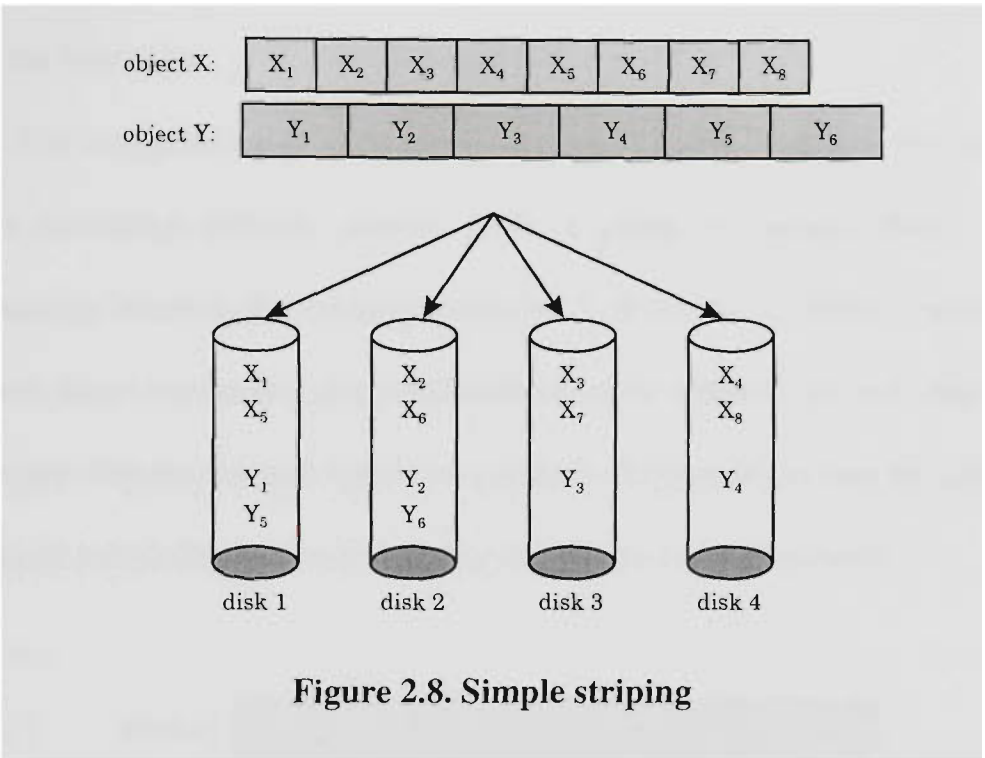


Figure 2.8. Simple striping

(a) Staggered Striping

Berson *et al.* proposed a *staggered striping* method to generalize the simple striping method across disk clusters [5]. It removes the constraint that two consecutive sub-objects must be assigned on non-overlapping disks. It can accommodate objects of heterogeneous display bandwidth with little loss of disk throughput.

A multimedia object is partitioned into a number of sub-object X_i. Each sub-objects X_i are then placed to a cluster of k disks. The number of disks in a cluster is chosen in a way that it can support the required bandwidth of X. The next cluster of disks is selected as the next k disks being shifted by r disks, where

r is called the *stride*. The sub-object X_{i+1} is then placed in the next disk cluster (Figure 2.9).

While an object is being retrieved in parallel, the cluster of disks in use changes from time to time. Hence, each disk becomes free periodically. As long as a new stream can be served within the time gap, another object can be retrieved within the time gaps.

The staggered striping method provides effective support for multiple streams accessing different objects from a group of striped disks, and it automatically balances the loading among disks. However, it suffers from several problems. Since continuous disk bandwidth must be obtained, the disk bandwidth can become fragmented and rejects new streams. This problem may be alleviated by efficient scheduling methods that alter the service order of requests.

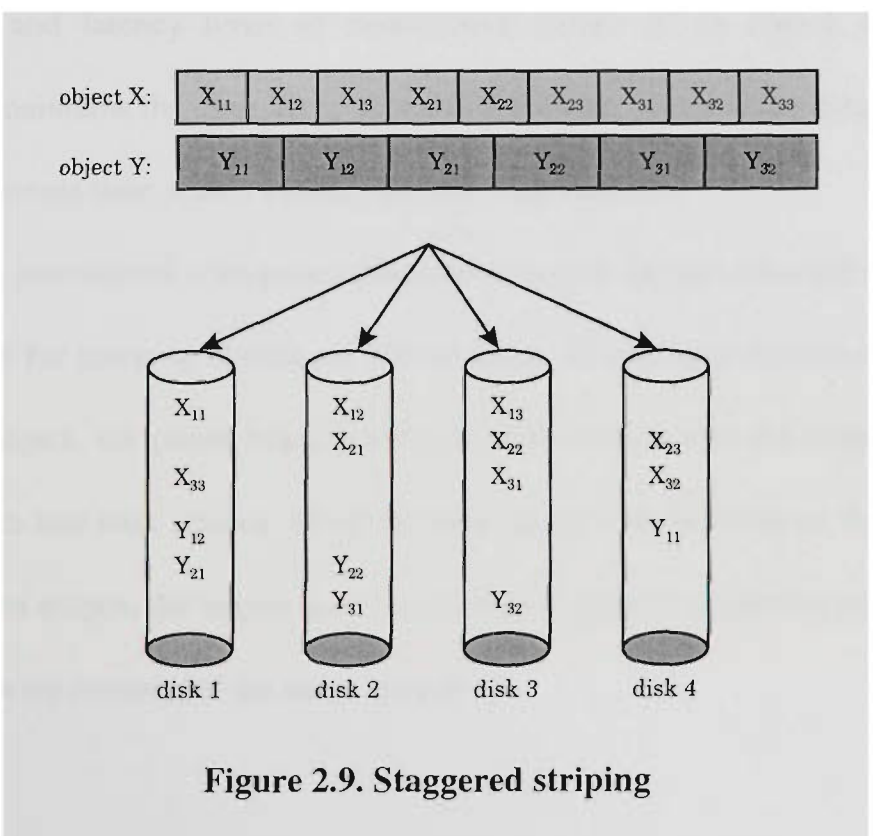


Figure 2.9. Staggered striping

Although data on tertiary storage devices may also mismatch the staggered striping arrangement on disks, data on tertiary storage can be pre-arranged to alleviate this problem. Unfortunately, the objects are not always presented in the normal display rate. When an object is presented in fast forward mode or rewind mode, data are retrieved in a different rate from the cluster of disks. Berson *et al.* suggested to create replica to support objects retrieved at abnormal rates. However, each rate would require an extra replica and the system is obviously limited to a small number of display rates.

2.5.8 Constraint Allocation

Most existing storage servers store data stripes on random disk tracks. Separations between data stripes of an object are not constrained enough to guarantee bounds on access and latency times of consecutive stripes of an object. Constraint allocation maintains the distance in separation between consecutive data stripes to bound the access time within media playback requirements.

The interleaved contiguous placement in part (a) provides the constraint requirement for merging objects on optical disks. When many streams access the same hot object, the phase based constraint allocation in part (b) supports more streams with less seek actions. Since the data access time depends on the distance between data stripes, the region based allocation in part (c) limits the longest seek distance among requests of the same stream.

(a) Interleaved Contiguous Placement

Since multimedia objects are recorded and retrieved using request streams with real-time media playback requirements, they can be stored with gaps between data stripes. The total storage space of several objects is reduced if some objects can be stored at the gaps of others. Some researchers proposed the *interleaved contiguous placement* to place data stripes of multimedia objects on CLV disks in an interleaving manner [122, 142, 151].

Each stream is characterized by a storage pattern composing of two parameters M and G, where M is the number of data blocks of each data stripe, and G is the number of gap blocks between two consecutive data stripes. A storage pattern can satisfy the continuous display requirement if

$$\frac{M+G}{\rho} \leq \delta, \tag{2.9}$$

where ρ is the disk retrieval bandwidth and δ is the display time for each data stripe. That is, the time to skip over the gap and retrieve the next successive data stripe is less than the time to display a data stripe. Since the left hand side of the inequality is the time to skip over a gap and read the next media block from the device, and this is less than the display time of one media block, there is sufficient time to retrieve the next successive media block while the current media block is displayed. Continuing likewise, the whole data stream can be served and displayed without delay.

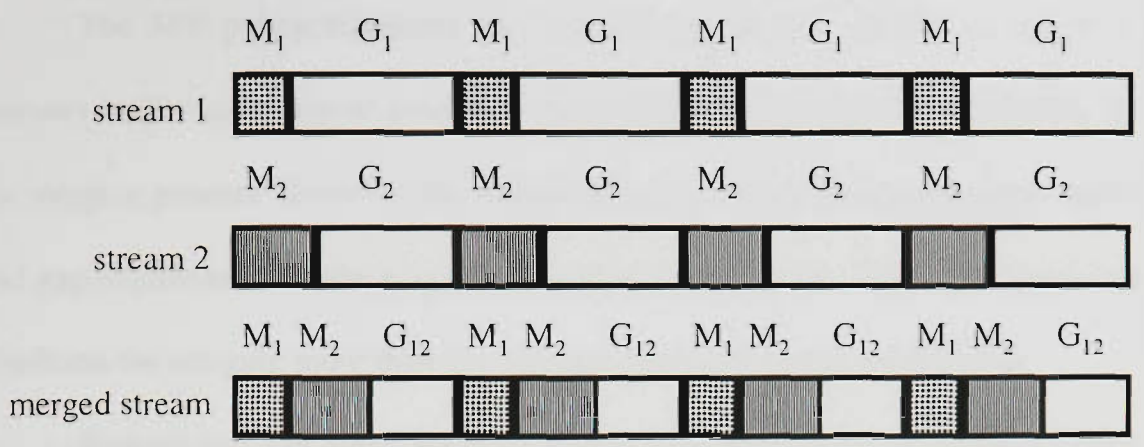


Figure 2.10. Interleaved contiguous placement

Two or more data streams may be placed together by a process of *merging* (Figure 2.10). Let M_i be the number of data blocks in the ith data stream and G_i be the number of space blocks in the ith data stream. They considered whether data streams (M_1, G_1), (M_2, G_2), ... , (M_k, G_k) can be merged while maintaining the continuous display requirement.

Two policies, the Storage Pattern Preserving Policy (SPP policy) and the Storage Pattern Altering Policy (SPA policy) have been studied to decide the feasibility conditions of merging. The SPP policy maintains the storage pattern of each data stream unchanged after merging. The SPA policy maintains the average storage pattern during the merging process.

Yu *et al.* considered the possibility of merging two objects while maintaining their original storage patterns [151]. Two media streams can be merged if and only if their greatest common divisor satisfies the condition:

$$M_1 + M_2 \leq \text{GCD}(M_1 + G_1, M_2 + G_2), \tag{2.10}$$

where GCD denotes the greatest common divisor of the list of numbers enclosed in brackets.

The SPP policy maintains the continuous retrieval conditions with both memory buffer and retrieval time for every component media stream unaltered by the merging process. However, the merged stream may not fulfil the regular media and gap requirements of the original individual media stream. Thus, the feasibility condition for merging more than two streams cannot be easily generalized.

Rangan and Vin allow the storage pattern to alter in the SPA policy [122]. They maintained the average gap size over a number of media blocks and store extra data blocks in read-ahead buffers. Although this policy requires more memory buffers to maintain the continuous display requirements, the merging of streams can be generalized. Two streams that cannot be merged under the SPP policy may now be merged under the SPA policy.

The number of streams that can be merged can now be generalized from only two streams to k streams. It is shown that data streams (M_1, G_1), (M_2, G_2), ... , (M_k, G_k) can be merged if and only if

$$\frac{M1}{M1+G1}+\frac{M2}{M2+G2}+\ \ldots\ +\frac{Mk}{Mk+Gk}\le 1. \qquad (2.11)$$

When several interleaving objects are retrieved, the disk can retrieve the objects at higher throughput since all the interleaving data stripes can be accessed using sequential reads without extra seek actions. Hence, this placement method is very suitable for composite objects whose component objects must always be synchronized. If the probability of several objects being concurrently served is high, then interleaving these objects could also raise the disk throughput. We shall switch the interleaving from the space domain to the temporal domain to allow for the feasibility condition to be used on general storage organizations in Chapter 5.

(b) Phase Constraint Allocation

Since the access pattern of multimedia objects are very much skewed in some applications, many users may request one object at slightly different times, particularly in near video-on-demand applications. Özden *et al.* proposed the *phase-constraint allocation* to serve multiple streams synchronously on the same hot object [116, 117].

Each multimedia object is organized as a (m x n) matrix of data stripes and these stripes are placed on m disks. Consecutive stripes are ordered sequentially from disk 1 to disk m and so on. Each column of n stripes are stored contiguously on a disk. All the streams on the object are divided into n phases with each phase of streams being separated from the previous phase at a fixed time period. All streams of the same phase are merged together and the retrieved data are broadcast to all streams of the same phase. With only one disk seek action, all n phases of streams on the same object are served (Figure 2.11).

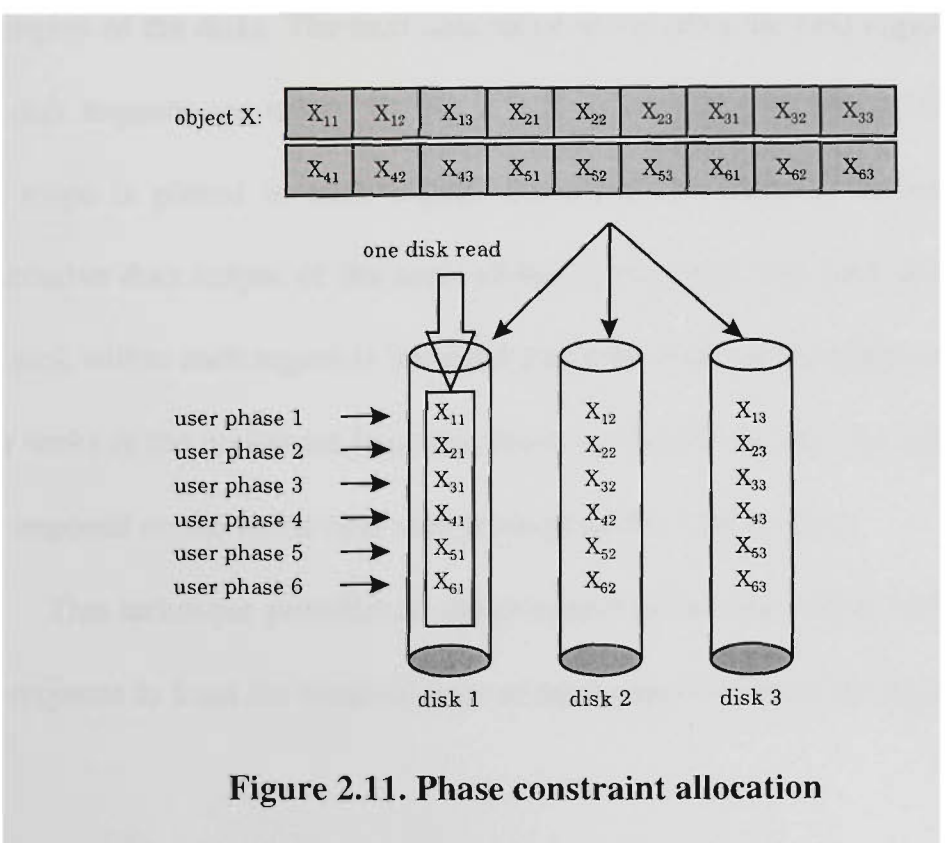


Figure 2.11. Phase constraint allocation

The advantage of this placement method is that several streams on the same object can be served with only one seek action. The maximum bandwidth requirement on any object is then limited by the number of phases instead of the number of streams. This could alleviate the disk bandwidth contention on the hottest multimedia objects.

Unfortunately, the start up latency is being traded off. Since there is a gap between the start of any two phases, new streams must wait for the beginning of the next phase before they are served. On average, new streams wait for half of the phase period. Keeping the first data stripe of every hot object from each disk resident on memory may alleviate this start up latency problem.

(c) Region Constraint Allocation

The region based allocation technique partitions each disk into several regions similar to zones in constant density recording disks. One data stripe is stored to one region of the disks. The next data stripe is stored to the next region and so on. The disk requests are served by the SCAN scheduling [59, 101, 115]. Since one data stripe is placed in each region, the maximum distance between any two consecutive data stripes of the same object is bounded. The seek distance of the first seek within each region is bounded above by twice of the region size whereas other seeks in the region are bounded above by the region size. An upper bound is then imposed on the worst case seek time on all the seek actions.

This technique provides an upper bound on the maximum seek distance of disk requests to limit the retrieval time of each requests below the displaying time.

The seek time overheads are bounded; more concurrent streams are acceptable; and the system throughput is raised.

Although new streams may only start when the region containing the first data stripe is being scanned, the maximum start up latency of new streams is limited to the period of traversing all regions once. Since the start up latency is proportional to the number of regions, Chang and Molina proposed to replicate data to reduce this start up latency [19]. This constraint allocation method is particularly suitable for multimedia objects on disks.

2.6 Hierarchical Storage Systems

There are two approaches to managing data on tertiary storage devices. The traditional approach is to use a *Hierarchical Storage Manager*. Such systems usually operate at the granularity of a file. The whole file is a unit of migration from tertiary storage to disks or memory.

The new approach is to embed the tertiary storage devices into the database system. The tertiary storage device thus forms a part of the database system, and data are moved between layers of the storage hierarchy within the process of executing a query. Myllymaki and Livny studied the tradeoffs between memory and disk requirements and the execution time of a join [105]. Sarawagi and Stonebraker exploit dynamic reordering to match execution order to the optimal data fetch order [128, 129]. The performance of the query optimizer is enhanced using information caching, query optimization, and query scheduling in [128, 152]. Issac describes a prototype that integrates a database management system with a storage management, allowing the storage of database on hierarchical storage systems [77]. Chen *et al.* investigate an algorithm to partition datasets based on data access patterns and storage device characteristics [25]. These tertiary databases include multimedia data together with other binary and textual data. Our design would be similar but not restricted to this new approach.

Several issues must be considered in designing hierarchical storage systems. These include

1. the total cost of the storage system,
2. the tertiary storage organizations,

3. the data migration methods, and

4. the cache replacement methods.

If it is possible to store all required objects on disks at lower system cost, there is no need to use a hierarchical storage system. Storage space usage can be divided according to the duration of data occupancy into resident storage and temporary buffers. While performance of accessing resident data is mainly affected by the storage organizations, performance of accessing data in temporary buffers is mainly influenced by the cache replacement methods. Since the access latency of tertiary storage is high, the stream response time is mainly determined by the data migration methods.

2.6.1 System Cost

The storage system cost consists of the data storage cost and the bandwidth cost. If only magnetic disks are used, the system cost can be prohibitively high for large multimedia systems. Tertiary storage is an essential component of large multimedia servers due to its low media cost and large storage capacity. Despite the low cost of storage, the bandwidth cost of tertiary drives are high. Some researchers found that the bandwidth of tertiary storage libraries is inadequate for online multimedia data accesses in some applications [28]. Fortunately, the bandwidth cost can be reduced in several ways.

Firstly, increasing the number of subsystems in a distributed environment can reduce the cost per stream. A multimedia storage server can then be built more cost effectively by coupling several smaller subsystems together [37]. Ford *et al.* coupled multiple small tertiary libraries together to create large tertiary storage

systems following the popular RAID method. This RAIL method achieves better cost/performance ratio than conventional tertiary storage systems [46].

Secondly, playing multimedia streams directly from higher storage level may reduce the cost per stream of low bandwidth tertiary drives. Staging or pipelining can be chosen dynamically based on the relative load on the tertiary versus secondary storage device [81, 118].

Thirdly, storage hierarchy becomes more economical by sharing bandwidth among multimedia streams in the placement strategy. Whether storage hierarchy should be used or not depends on the storage devices cost/bandwidth and cost/capacity ratios [92].

Fourthly, efficient cache replacement methods can increase the hit ratio on disk cache. This would reduce the workload on the tertiary storage; resulting in more efficient use of tertiary storage bandwidth and disk cache size. Therefore, efficient cache replacement method can also reduce the system cost.

It is found that the choice of data on the different levels depends on the fraction of time when a stream of an object is active. It is suggested in [81] that objects having an active stream less than a third of the time should reside on tertiary storage. Since large multimedia systems may require vast data storage, the use of hierarchical storage systems becomes inevitable.

2.6.2 Tertiary Storage Organizations

Despite the vast amount of researches on disk based placement methods, only a few data placement methods that are proposed for tertiary storage can be found in the literature. We group them into five strategies.

1. **Contiguous:** Whole object is stored to consecutive physical locations.
2. **Log Structure:** All writing operations are appended to a large log.
3. **Frequency Based:** Objects are stored following the access frequencies.
4. **Striping:** Data are divided into stripes and retrieved in parallel.
5. **Resident Leaders:** The starting parts of each objects are resident on disks.

We discuss these storage organizations in the following paragraphs.

(a) Contiguous Placement

In traditional systems, tertiary storage libraries store each file to consecutive physical blocks. When the same strategy is applied to multimedia objects, each object is stored like a file. The drive throughput is high since the whole file or object is retrieved with only one media exchange overhead followed by one search operation.

Triantafillou and Papadakis changed the sequence of data stripes on tertiary storage devices according to the number of data stripes that can be retrieved within the display time of one data stripes [139]. The whole object is still stored contiguously on the media units. The data stripes that can be displayed from memory directly after retrieval are interleaved with the data stripes that are staged to disks, resulting in less staging data and lower disk bandwidth.

(b) Log Structured Placement

Kohl *et al.* developed a hierarchical storage system using the log-structured file system. The access ranges within a file is tracked to determine whether a file is often accessed sequentially or randomly. Sequential files are stored contiguously and randomly accessed files are stored to many locations [85]. Log structured files are optimized for writing data, instead of reading data. This method is efficient in some applications where data are more often being modified than retrieved.

(c) Frequency Based Placement

Christodoulakis *et al.* used the frequency based placement method to minimize the average access time in tertiary storage libraries. Placing as many hot objects as possible together on one media unit minimizes the number of media exchanges. Placing the objects in an organ-pipe manner minimizes the tape reposition distance [30].

Unfortunately, there is a potential problem associated with grouping all hot objects on one media unit. If too many requests are directed to only one media unit and handled by the loaded drive, the drive becomes a bottleneck. If the requests are not served fast enough, the long queue can grow indefinitely on one drive. In the extreme case when all requests are waiting for service at only one drive, the requests are served serially. If the hot objects are distributed across more media units, the requests can then be served by multiple drives in parallel. Therefore, the two level view of the placement method delivers only local optimal solutions.

Nemoto *et al.* use foreground and background migrations of media units to balance the loading among tertiary storage libraries. A wagon is used to move media units across the libraries. The desired locations of media units are dynamically adjusted according to the current access frequency and drive workload [107].

Similar frequency based strategy can be applied to placing media units within cells. Since media units may be placed in cells close to or far from the drives, the cell arrangement can be optimized according to the frequency of access. More frequently accessed media can be placed nearer to the drive which should be placed near the center of all media cells. Although we have not found any researches on this application, we intuitively assume that the strategy is applicable in similar way as other frequency based placement methods.

(d) Striping

Striping data over multiple media units allows several drives to perform I/O in parallel. The times to retrieve data from multiple media units are then overlapped to reduce the response time of each request. Hence, requests are served with shorter data transfer time.

Drapeau and Katz investigated the benefits and disadvantages in applying parallel striping in tape arrays. For a small library composed of many tapes, a few drives and a single robot arm, contention for the small number of drives limits the value of striping under heavy workload. For a large library with fast robots and high throughput drives, the exchange time predominates and penalizes striped systems for all but the largest accesses at very light workload. Striped systems

generate more cartridge switches than non-striped systems. Striping increases contention for drives, resulting in an artificial increase in resource utilization [39, 40].

Golubchik *et al.* studied the impact of stripe width on striped tape arrays and showed that the optimal stripe width is dependent on the system's workload [67]. However, it is difficult to choose an optimal stripe width that can vary according to the dynamically changing system workload.

Chiueh proposed the triangular striping technique that stores data stripes following the order of media exchange. When several drives simultaneously exchange media using only one robot arm, the media units are exchanged to one drive at a time. Some data stripes are retrieved from the already exchanged drives while other drives are still pending for exchange. All the drives are arranged to complete the data retrieval at the same time. The data retrieval time is hence overlapped with robot exchange time to reduce request service time [29].

Since striping methods separate objects to several media units that are accessed by multiple drives, they increase the number of media exchanges to access each object. Although several drives may access one object in parallel to reduce response time under light workload, the throughput of tertiary drives drops and the response time increases under medium to heavy workload. Hence, new methods that can share the media exchange overheads under medium to heavy workload becomes a necessity for striping to succeed on tertiary storage.

Although parallel striping can increase the chance that some requests may share a media exchange, this is unlikely with the low access probability of each media unit. In addition, the worst case condition must be assumed in multimedia

system to provide guarantee on the continuous display requirement. Hence, the parallel striping methods cannot deliver a higher throughput than the non-striping method under heavy loads.

Instead of considering parallel striping method for general data, we focus on striping of only multimedia data in this research. Instead of parallel striping to only a few media units, we striped the objects across all media units so that all concurrent streams can definitely share the media exchange overheads. We also remove the synchronization of drive exchanges to avoid contentions at the robot arm.

(e) Resident Leaders Placement

The resident leaders placement methods store multimedia objects partially on disks and partially on tertiary storage. They place the initial part of each object, called *leader*, on disks and the rest of the object, called *tail*, on tertiary storage. The two parts are connected together via data buffering. When an object is accessed, the object is started to display leaders from the disks while the tail are being copied from tertiary storage. Data are alternatively supplied from both disks and tertiary storage [110, 133, 138].

Nishimura *et al.* retrieved data synchronously from magnetic disks and optical jukeboxes. One fixed size data stripe is transferred to each user within a fixed period of time. The leaders are kept resident on magnetic disks to reduce the start up latency of cold objects [110].

Similarly, each object is divided into fixed time units in multiple readout method. Each time unit is further split into one leader and one tail. Leaders of all

objects are resident on magnetic disks. The tails are stored within one media unit. While the leaders of m concurrent streams are displaying, the tails are being retrieved from m media units to magnetic disks [133].

These resident leader methods are useful only when the multimedia streams are retrieved using the pipelining method. If the whole object is retrieved prior to display, the copying time is only reduced slightly. These methods significantly reduce the maximum start up latency in displaying multimedia objects from the hierarchical storage system.

Unfortunately, keeping data resident on disks increases the disk buffer size. Since the number of retrieving streams is much less than the number of objects on tertiary storage, more disk space is required. As the leaders are resident on disks, we compose them in the way that can provide an overview of the multimedia object. Leaders can hence facilitate browsing and searching the objects without any extra preview file.

2.6.3 Data Migration Methods

Multimedia objects can be stored and staged across the storage hierarchy similar to traditional files. After objects are materialized on disks, they can be accessed randomly from disks to support any user display modes. The tertiary drives are totally free from any references to the objects while the objects are staying on disks. Frequently accessed objects may also stay longer for repeated references. If objects are scheduled in advance, the objects can be staged to disks prior to the presentation time [10, 125].

Unfortunately, the staging method is inefficient in response time and disk space consumption. Firstly, users have to wait for a long time of copying the entire object to view it. Since the system responds after the completion of the first stage and the transfer time of large objects is long, the user needs to wait for a long time even when he is only previewing the objects.

Secondly, the disk buffer space to contain the entire object is reserved for a long time. While the object is staged, the disk buffer size grows slowly to the full object size during the first stage. If the entire object stays on disk buffer for consumption until the object is no longer required, the full object size is reserved during the entire consumption period. Hence, the staging buffer size is large during both the staging and consumption period.

(a) Time-Slice Scheduling

Lau *et al.* proposed the time-slice scheduling method to reduce response time of concurrent streams. Instead of staging an object in its entirety, an object is retrieved as multiple slices. New waiting streams may start earlier if the streams can be served concurrently. The average response time is hence reduced [91]. Unfortunately, exchange overheads are increased and they undesirably lower the drive throughput. They increase the waiting time and response time under heavier workload. Therefore, the time-slice scheduling is efficient under very light workloads only.

(b) Pipelining

For certain applications such as on-demand systems, users can only view multimedia objects from beginning to end. Since the time to stage objects from tertiary store to disks are very long, the system may start to display the data to users before the entire object is completely copied to disks. Ghandeharizadeh and Shahabi investigated this *pipelining* technique to minimize the response time of multimedia streams [58, 146].

The pipelining technique groups the data blocks of X into s logical slices ($X_1, X_2, \ldots, X_s$) such that the display time of X_1 is longer than the retrieval time of X_2, the display time of $(X_1 + X_2)$ is longer than the retrieval time of $(X_2 + X_3)$ and so on. The Production Consumption Ratio, PCR, is defined as

$$\text{PCR} = \frac{\gamma}{\delta}, \tag{2.12}$$

where γ is the tertiary bandwidth and δ is the display bandwidth. If PCR < 1, then the waiting time is minimal when the last slice consist of a single block. In this case, the size of the first slice is

$$O - \lfloor \text{PCR } (O - 1) \rfloor, \tag{2.13}$$

where O is the object size.

Intuitively, this result is applicable to any contiguous segments of multimedia objects. A multimedia stream pipeline on a segment of multimedia object can be set up if the same fraction of the segment is already copied to or resident on disks. We shall use this result to access segments continuously from tertiary storage.

Since the disk buffer filling and consumption periods overlap with each other, the duration of a slice staying on disk buffer is reduced. Some slices may

even be played directly from the memory without being copied to disks when the time of retrieval and time to display are close together. As a result, smaller disk buffer space for each object is required in pipelining than staging.

Wang *et al.* proposed to use a circular buffer to reduce the buffer space. The leaders or the first slices are cached to disks and the tails are loaded on demand. A circular buffer is used to retrieve and display the tail portion. This circular buffer shrinks when the size of the slices decrease [146]. The disk buffer space is maximized when it contains the first two slices. Hence, the required disk buffer space for an object of size O is estimated as

$$(1 - PCR^2)O. \tag{2.14}$$

Unfortunately, the pipelining methods limit the user display options. Users are prevented from freely referencing other locations within the object as in the staging method. Objects can either be displayed in normal mode or be rewound to previously displayed locations. When the user displays in fast mode or jumps behind the current location, another leader needs to be retrieved before display can resume. The user waits for this latency until sufficient data are copied to disks.

Tavanapong *et al.* studied the choice of blocks to be transferred over a low bandwidth network during the start up time of the pipelining method. An object is split into many fixed sized fragments in the Two Phase Service Model (2PSM). The leaders of all fragments are transferred in the first phase. After all the leaders are transferred, the user can start to preview or display the object. The tails of any fragment are transferred while the user is viewing the object. User may also preview the leaders of the object to interrupt the transfer of the object [134].

Tavanapong *et al.* provided three strategies, linear, spreading, and binary-tree, to select the leaders of fragments.

These strategies are good at improving preview quality over longer start up time. The fragments are uniform in size and easy to be analysed. His method also reduces the latency in supporting VCR-like functions. Unfortunately, some short scenes may not be present in the leaders. Fast forward, fast reverse, and searching from only the leaders could miss some important contents.

We have performed independent studies prior to the publication of Tavanapong's paper. Our method differs from their method in several ways. First, we perform logical segmentation prior to data striping. The logical segments contain the low temporal resolution segments to provide sufficient preview information. Second, we divide each logical segment into data stripes with decreasing sizes instead of just one fixed size. Hence, the data retrieval time after fast forward and fast reverse operations overlaps with the displaying time using the pipelining techniques. Third, we include the media exchange time and disk bandwidth so that our method can be applied to more realistic and practical situations, and the continuous display requirement can always be guaranteed. Our method is described with details in Chapter 3.

Kienzle *et al.* stated that the choice of whether staging or pipelining for minimal storage system cost should base on the ratio of tertiary storage throughput and display bandwidth [81]. Pang used a dynamic retrieval algorithm to choose between staging and pipelining methods based on the relative load on the tertiary versus secondary storage device [118]. Hence, pipelining methods will inevitably be used in future multimedia systems.

2.6.4 Cache Replacement Methods

Tertiary storage devices provide lower storage cost but higher bandwidth cost than disks. Increasing the number of streams from the tertiary storage raises the disk cache cost, resulting in higher total system cost [37]. After data are copied from tertiary storage to disks, they should be kept in disk cache as long as possible for repeated accesses to minimize the number of streams to tertiary storage. Therefore, objects should be removed from disk cache only when storage capacity is required.

Traditionally, the selection of objects to be deleted is based on the access history and the frequency of access. Simple algorithms, such as Least Recently Used (LRU) algorithm, have been used for many years to manage memory cache. Data of multimedia objects are consumed sequentially. Removing the leaders of cold objects increases the variance of start up latency, while removing the tails of hot objects can reduce the cache hit ratio. Hence, efficient cache replacement methods are required to select multimedia data objects to remain on cache under the constraint of these conflicting requirements.

(a) Frequency Based Cache Replacement

When more data requests are served at the lower storage level, fewer requests are directed to the higher storage level. When a small amount of material is heavily accessed and used, this skew is then exploited by using a storage hierarchy that stores the less frequently used data in lower cost tertiary storage and the more frequently used data in higher cost secondary storage. As hot objects are frequently accessed, placing them on the lower level reduces many more requests

to the higher level than placing cold objects on the lower level. Hence, many studies use the access frequencies to select the objects to stay in the lower storage level [10, 28, 37, 42, 81, 133, 140].

Some researchers proposed a cache management algorithm that uses a priority mechanism. The priority value is determined by the expected time to next requests on the object. These priority values may vary with time and they are evaluated periodically. Low priority objects are replaced with high priority objects until only the highest priority objects are in the disk caches [10, 42, 125]. In this system, detailed access statistics are stored to determine the access frequency of objects and all user requests must be scheduled in advance.

While data placement methods are static for resident data objects, cache replacement methods can dynamically determine the objects that remain on the disk buffers. Hence, cache replacement methods can adjust itself dynamically to suit changes. If some animated objects may have higher access frequency at different times of the day, then these objects may have different priorities that depend on the current time.

Unfortunately, cold objects that are resident on the tertiary storage are less skewed since their access frequencies belong to the tail of the Zipf distribution. If all accesses are highly skewed, storing the most frequently accessed objects on disks can significantly reduce the number of accesses to the tertiary storage. Otherwise, recently accessed objects are overwritten well before they are accessed again. Hence, the frequency based cache replacement methods are less useful for cold objects on tertiary storage.

When a large cold object is placed on a light loading disk, there is little disk space left to store other objects. This would erode disk bandwidth due to limitations on the disk space constraint. Likewise, when a small hot object is placed on a heavy loading disk, there is little bandwidth left to serve other objects. This would then erode disk space due to the limitations on the disk bandwidth constraint. Some researchers proposed a cache replacement method that balances the space and bandwidth of objects on disks [35].

Brubeck and Rowe proposed a selection algorithm that chooses the disks with minimum load [10]. The available load, L, on a video file server is defined as

$$L = L_{\text{cache}}\, L_{\text{stream}}\, L_{\text{net}} \text{ and } 0 \leq L \leq 1, \tag{2.15}$$

where L_{cache} is the cache load, L_{stream} is the stream load, and L_{net} is the network load. The disks with the minimum load and sufficient storage is chosen to store the object. Similarly, Dan and Sitaram proposed to use the bandwidth-to-space ratio to select the staging disk. The disk with the lowest bandwidth to space ratio is chosen to store a staged object [35]. These methods can only balance disk loading on an object level since an object is either placed to a disk or to another disk.

(b) Latency Based Cache Replacement

Some researchers proposed to delete only partial objects according to the impact on the variance of response time. The leaders are often given higher priority than the tails of each object [104, 110, 133]. Ghandeharizadeh and Shahabi designed a Partial Replacement Technique (PIRATE) to delete partial objects from the disk cache in order to maintain the start up latency low when any objects are accessed again. The leaders of cold objects may stay while tails of hotter objects are removed from the disk cache [8, 58].

These methods use the position of the data stripe from the start of the object to choose the data for removal. Although more requests would unfavourably be directed to tertiary storage than the frequency based methods, the maximum response time of any streams is bounded.

Unfortunately, these latency based methods undesirably lower the cache hit ratio and increase the number of requests to tertiary drives. When more requests are served, drive contentions occur more frequently. The waiting time for free drives can significantly increase the response time of requests. Hence, the average response time is traded off with the variance of response time. Ghandeharizadeh and Shahabi provided a parameter to tune the relative importance of average response time and variance of response time [58].

2.7 Other Related Works

The access speed of the tertiary storage devices is often considered as the bottleneck of hierarchical storage systems [25]. Chan and Tobagi performed sensitivity analysis of various system parameters on the start up latency. The analysis are based on contiguous object placement, staging method, and LRU cache replacement algorithm. The disk bandwidth, the disk capacity, the number of tertiary drives, the tertiary drive bandwidth, and the tertiary drive exchange latency are adjusted to reveal their influence on the waiting time [16]. This model analyses only on the staging method and its results cannot apply directly to the pipelining method.

In [79], a query model is developed for accessing files from tertiary storage devices. The model caters for batches of file requests that behave differently from multimedia data request streams.

In [65], a method that provides preemptive tape drive unloads is studied. Gniewek shows that robot service time can be reduced under some application conditions. Parallel striping methods become more attractive when the robot service time is shorter.

2.8 Discussion

- **Storage Device Characteristics**

Traditional disks under-utilize recordable surface area on outer tracks and waste disk bandwidth. Constant density recording uses recordable surface area more efficiently. Constant linear velocity approach, however, wastes the data transfer rate by reducing the revolution speed at outer tracks and increases the overheads in changing motor speed. Intuitively, using a constant angular velocity approach on constant density recording disks is the most efficient in supporting random accesses on both storage capacity and data bandwidth.

Disk array provides large storage capacity and high storage system throughput. RAID encodes redundant data to increase data reliability and security. The SCAN scheduling methods can reduce seek overheads on random requests according to the data access locations on disk surface.

- **Multimedia Storage System Characteristics**

Large multimedia systems that require data storage of the order of petabytes make the inexpensive magnetic disks neither economical nor practical. Hierarchical storage systems utilizing the large capacity of tertiary storage devices could be considered to meet the size demand in such systems. The media exchange time of hierarchical storage systems is so long that it significantly lowers the performance of these systems. Traditional data migration methods also delay the response time significantly.

Multimedia storage systems differ from traditional storage systems in many aspects. Large multimedia objects should be maintained in compressed

format for as long as possible. Progressive and multi-resolution compression techniques allow coarse views using partial objects. Bulky multimedia objects should be partitioned into data stripes for efficient retrieval using the constant time length approach.

User may view multimedia objects in various display modes. VCR-like functions change the sequences and deadlines in accessing data stripes. Some data stripes may be skipped over in fast modes. User may browse and search multimedia objects. The storage system should support at least these types of data retrieval patterns.

Access frequencies of multimedia objects are often predicted with the Zipf distribution and multimedia objects are more often retrieved than modified. Multimedia streams generate requests periodically and continuously over an extended period of time. Requests of the same stream that access neighbouring data in the object are separated evenly. Each request should finish within a real-time deadline, and variations in request response time can be absorbed using read-ahead buffers. Composite multimedia objects need to synchronize the retrieval time of their component streams prior to display. These multimedia data access patterns make multimedia storage systems behave very differently from traditional computer storage systems.

The SCAN-EDF and GSS algorithms are the disk scheduling currently tailored for multimedia streams. However, they can only provide soft guarantee that may reduce the quality of service. When multiple streams on the same object are served, they may merge using the piggyback technique.

- **Storage Organizations on Disks**

Traditional intelligent file placement methods are not designed to serve multimedia request streams. The random placement can only be used as a reference for comparison. The contiguous placement strategy is optimized for object based retrieval. Fragmentation, copying overheads, and the presence of concurrent streams, however, reduce the successful application of this technique.

The log structure placement strategy removes the overheads on consecutive write operations, but it does not guarantee performance on accessing modified data blocks. The statistical placement strategy minimizes the average response time of requests. The object based methods would place each object contiguously since they always have the same probabilities. This strategy should be refined to a finer granularity to optimize the performance of concurrent streams. Redundant codes increase data reliability and data security. Unfortunately, the updating and retrieving redundant codes increase the system workload. Multimedia data are less volatile and are usually recoverable from other sources. Data replication strategy increases the availability of data by using alternative storage devices. The need for extra storage space yet lowers the effectiveness of this strategy. Data striping strategy partitions bulky multimedia object for efficient data retrievals. Staggered striping removes the contentions on hot disks and balances the disk load automatically. Constraint allocation strategy effectively guarantees the continuous display requirement of streams at the small expense of start up latency. Phase constraint allocation is mainly suitable for very hot objects in near-video-on-demand environment, but its benefits may be replaceable with the more dynamic piggyback merging method. Region based constraint allocation bounds the

maximum seek overheads of consecutive requests and interleaving contiguous placement handles heterogeneous streams on optical disks effectively. These methods should be extended to heterogeneous streams on other storage devices, such as magnetic disks, tapes, and multiple storage devices. We shall establish the feasibility condition for accepting concurrent streams by multiple independent devices in Chapter 5.

- **Tertiary Storage Organizations**

Traditional hierarchical storage managers operate at the granularity of a file. The new approach migrates data automatically across the storage hierarchy in the database system. Large multimedia systems may require storage capacity so huge that magnetic disks are not practical. Smaller multimedia systems that place data according to access frequency also have a large number of less skewed and cold objects that are best stored on tertiary storage. Although tertiary bandwidth cost is more expensive than disk bandwidth cost, storage hierarchy can be more economical in some systems.

Contiguous placement strategy stores objects on the fewest media units. Log structure placement strategy is suitable for storage systems that are bounded by write operations. Frequency based strategy is applied on several placement problems to minimize the average retrieval time. This strategy is also applied to migrate media units among libraries. These methods are efficient for object based retrieval, but multimedia objects are too bulky to be accessed in its entirety.

Unlike striping on disks, parallel striping methods on tertiary storage are less successful. The presence of long media exchange time erodes the tertiary

bandwidth for small objects under heavy workload. The optimal stripe width is bounded by the number of drives. Triangular placement helps to reduce half of the waste on synchronized robot exchange period. We suggest two ways to reduce the overheads:

First, media exchanges at the tertiary drives should be asynchronous, instead of synchronized, to avoid robot contentions. Second, the media exchange time should be shared among concurrent streams to reduce average overheads per stream. These are described in Chapter 3.

A typical method to store multimedia objects using storage hierarchy follows the frequency based placement strategy with resident leaders. Multimedia data are ranked according to their access frequencies, or object temperatures. Hot multimedia data are disk resident. Very cold data are entirely stored on tertiary storage devices. Warm multimedia data are divided into leaders and tails; the leaders are resident on disks whereas the tails are retrieved from tertiary storage on demand.

Traditional staging methods are not optimized for materializing multimedia objects across the storage hierarchy. Time-slice scheduling is applicable under very light load conditions. The simple pipelining methods optimize the start up latency in new streams, but they also limit the user display options. Tavanapong *et al.* extended the pipelining methods to provide preview operation over low bandwidth networks. Independently, we extend the pipelining methods differently to minimize the latency in supporting VCR-like, browsing, and previewing functions.

Frequency based cache replacement methods optimize the cache hit ratio to reduce the average response time. Latency based cache replacement methods maintain the presence of leaders in cache to minimize the variance of the response time at the expense of the average response time.

2.9 Chapter Summary

We have evaluated the characteristics of multimedia storage systems in this chapter, and highlighted the issues that need to be resolved, which will be addressed in the remainder of this thesis.

Traditional storage devices are tailored for small amount of data retrieved by discrete random requests. Random access memory is expensive and small, though data can be accessed instantly. Sequential access storage device provide large storage capacity at very low cost, but data can only be retrieved after very long exchange latency. Direct access storage devices provide inexpensive storage at fixed storage capacity to bandwidth ratio. With the emergence of the new multimedia age, these storage devices become inefficient in accessing multimedia data. It remains a significant research challenge to improve the performance of multimedia storage systems.

There is a recent trend to use constant density recording disks. These disks have different storage formats and access performance characteristics. The performance of multimedia requests on CDR disks needs to be investigated with suitable models so that their performance can be analysed quantitatively. Requests to different zones in CDR disks perform differently. Higher throughput is expected from the outer zones. This variable throughput should be considered in

efficient data storage organizations. New placement methods are hence required in placing data on CDR disks.

Many storage organizations have been proposed for multimedia data on magnetic disks. Disks array provides large storage capacity. Staggered striping reduces disk contentions. Constraint allocations lower seek overheads to meet continuous display requirement. Interleaving allocation supports concurrent streams. These placement methods are efficient on direct access storage devices.

The requirements of multimedia data storage in new applications are in the order of hundred terabytes to petabytes. Disk only systems would waste valuable bandwidth in storing large amount of infrequently accessed data. Even though the cost of disks can be considered inexpensive, the only practicable and economical solution to store these multimedia data would be to use tertiary storage devices.

Although some researches show that storage hierarchy may not reduce the total system cost, it has been found that storage hierarchy can be more economical by sharing bandwidth among multimedia streams. It is also discovered that playing multimedia streams directly from higher storage level could also reduce the cost per stream. Therefore, new efficient storage and retrieval methods for multimedia data on hierarchical storage systems are required in order that large multimedia systems may become more economical and practical. While the pipelining method minimizes the response time, it also limits the display options. Hence, the pipelining technique needs to be enhanced to minimize the delays in supporting interactive functions.

Chapter 3

New Efficient Storage Organizations for Multimedia Data

We have shown in the previous chapter that new efficient storage organizations are required for multimedia data, we present our novel storage organizations in this chapter. We first consider performance measures on multimedia storage system.

3.1 System Performance Measures

In order to compare the efficiency of different storage and retrieval methods, it is necessary to measure the amount of resources being used. Quantitative measures, including system throughput, request response time, and buffer size, are commonly used in traditional computer storage systems. The criteria to measure performance are different in multimedia storage systems. We consider a multimedia storage system to be efficient if:

1. the continuous display requirement is guaranteed,
2. the system throughput is high,

3. the response time of new streams is short, and

4. the necessary buffers are small.

(a) Continuous display requirement

When the storage system continuously supplies data at a faster rate than the display rate, the display buffer overflows. This problem can sometimes be rectified by suspending the streams of data requests until some data are displayed from the buffer. When the storage system supplies data at a slower rate than the display rate, the display stream starves. Artifacts, such as jitters and hiccups, will then occur and they adversely affect the quality of service. Therefore, the continuous display requirement must be guaranteed in designing multimedia storage systems [53].

(b) Throughput

Throughput of most storage systems are determined by their data access speed. It increases in proportion to the rate of requests being served. When requests are served quickly with a corresponding effect on the waiting queue, the request service time significantly affects the storage system throughput.

Multimedia storage systems only accept new streams if they can serve the streams without violating the continuous display requirement. The number of concurrent streams is hence limited by the performance of storage devices. When more concurrent streams are accepted, the system can serve more streams concurrently to quickly reduce the waiting queues. Hence, throughput of

multimedia storage systems determines the storage system's capacity in accepting new streams.

When system load is heavy, new arriving streams experience long waiting time in queue to be accepted. The response time of the new streams could be increased indefinitely by this long waiting time. Unless some streams are served, the waiting streams cannot be served. This waiting time, hence, depends very much on the throughput of the system. The system should quickly complete the service of the accepted streams so that waiting streams can be started. Hence, higher system throughput reduces the waiting time and response time under heavy loads. Therefore, storage methods that deliver high throughput under heavy loads are desirable.

(c) Response Time

When user starts a multimedia stream, the response time that appears to the user is the response time of the new stream instead of the response time of individual requests. Hence, stream response time is the external characteristics that manifest the performance of the multimedia storage system to users. New streams should respond fast in efficient systems. Hence, the stream response time provides a measure of the quality of service to interactive users.

The response time of new streams consists of the following delays:

1. the waiting time of the stream for acceptance,
2. the copying time if data are migrated from tertiary storage devices,
3. the network delays if data are accessed from remote sites, and
4. the initial start up latency to fill the read-ahead and display buffers.

These delays may overlap with each other during processing. For example, some data may be migrated from tertiary storage devices for waiting requests if the tertiary storage devices are free. These delays significantly affect the stream response time under light loads.

If the system load is light, the system is free to accept new arriving streams. The new stream does not have to wait and its response time is significantly influenced by the start up latency. The system should serve the accepted streams with short start up latency so that the streams can respond quickly. Therefore, storage methods that respond quickly under light loads are desirable.

(d) Buffers

Buffers on RAM and disks are used in multimedia systems. When data are accessed from disks, a display buffer is required on RAM to keep temporary data prior to display. A few read-ahead buffers may also be required on RAM to cater for variations in disk throughput. When data are retrieved from tertiary storage, an additional staging buffer is required on disks. The staging buffer keeps the intermediate data to prevent from overflowing the display buffer or suspending requests to tertiary storage [55]. Large buffers will raise the storage cost of the system. Hence, buffer size measures the amount of resources being used in the storage system and efficient storage systems should use small buffers.

To summarize this Section, a multimedia storage system is considered efficient if this storage system can store and retrieve data at high throughput without violating the continuous display requirement and it can respond quickly to new streams while utilizing small buffer space.

3.2 System Architecture

The architectures of storage systems being considered in this research are multimedia systems that store multimedia data using a storage hierarchy (Figure 3.1). One or more tertiary storage library may be used. The tertiary storage library consists of a media exchange device, several tertiary drives, and many media units.

Data are transferred through the system bus among various storage devices. As the system bus bandwidth is often over four times the data bandwidth from the tertiary storage level, the system bus is assumed to be of sufficient bandwidth in this thesis.

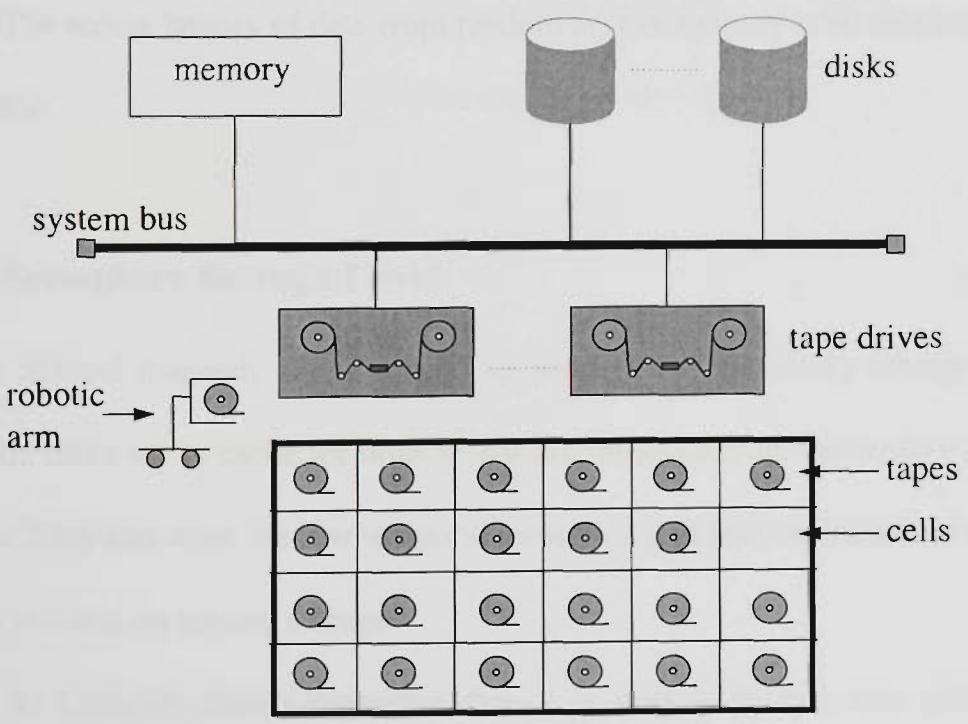


Figure 3.1. Typical storage system architecture

Multimedia data are stored at three levels in the hierarchical storage system. The primary storage level acts as cache for data in the secondary storage level and the secondary storage level acts as cache for data in the tertiary storage level. All data migrated between the secondary and tertiary storage levels should also go through the primary storage level.

3.2.1 Primary Storage Level

Random access memories should be used in the primary storage level. Apart form acting as cache for data in the secondary storage level, all data from disks and tertiary storage are directly transferred to the memory. The memory would also keep working data and running programs. We assume that the bandwidth of the random access memory is sufficiently large that it is considered as infinite in this thesis. The access latency of data from random access memory is so small that it is negligible.

3.2.2 Secondary Storage Level

One or several magnetic disks should be used at the secondary storage level. Magnetic disks act as cache for objects that are retrieved from the tertiary storage devices. They also store the low temporal resolution part and the leaders of objects that are resident on tertiary storage.

As Constant density recording disks can store more data than traditional VDR disks of the same size, CDR disks at constant angular velocity must be used. This is necessary for multimedia data which are large in size. Apart from having more data in outer zones, there is also a skew in the access probability of the

zones. As the tracks in the outer zones are longer in size, these tracks have a higher chance of being accessed. The read/write heads hence have a higher probability of staying around the outer zones. Intuitively, the expected seek distance of random accesses are shorter than that of the VDR disks.

When CDR disks at fixed angular velocity are used, more data pass under the read/write heads of CDR disks than that of VDR disks. Hence, CDR disks transfer data at a faster rate. We therefore expect CDR disks to retrieve bulky multimedia data quickly. Although CDR disks retrieve data at variable data transfer rates and increase the difficulty in buffering data from different zones, this variation in disk throughput can easily be controlled by using read-ahead buffers.

Multiple disks can be used together to provide large storage space that is needed for multimedia data. The disks can be accessed in parallel using striping techniques to achieve high disk throughput. We assume that the disk throughput is many times higher than the tertiary storage devices throughput.

3.2.3 Tertiary Storage Level

Optical jukeboxes, robotic tape libraries or array of DVD changers can be used as the tertiary storage servers. All multimedia objects are permanently stored at the tertiary storage level. We consider to use tertiary storage devices for the following reasons:

1. Large storage systems need to store petabytes of data and most of the data stored in these systems are multimedia data such as images, audio and video. Tertiary storage devices are the only practical devices that can

provide such huge storage capacity. Hence, tertiary storage devices are inevitably used in large multimedia storage systems.

2. Before any system designs can become commercially successful, the system cost must be considered. Although the bandwidth cost of tertiary storage devices is high, some multimedia systems have large amounts of cold data that can be more economically stored on tertiary storage devices than on magnetic disks.

3. Tertiary storage devices may be required to create copies on exchangeable media units for purposes such as updating, backup, archival, and distribution of data. It is reasonable to utilize already available resources to increase data bandwidth from the storage system.

Tertiary storage devices may be used to access distributed multimedia data in a single user environment. They may also be used to concurrently supply multimedia data from large multimedia storage servers to a number of users.

3.3 Data Striping on Tertiary Storage

Large contiguous multimedia objects are too bulky to be accessed from the tertiary storage as a whole. They should be partitioned into shorter data stripes in order that they can be handled efficiently. We investigate the method that divides the multimedia objects into stripes and places these data stripes on secondary and tertiary storage devices in this Section.

Each multimedia object consists of a number of media blocks such that each media block is a unit for display and retrieval. The multimedia objects are partitioned using a two level data striping method. The first level of data striping in Section 3.3.1 divides objects into logical segments of limited length. The second level of data striping in Section 3.3.2 further divides logical segments into shorter data stripes.

Data can be striped using either the data length or the time length approach. Since other components of data access time already vary from request to request, it is not worse off to vary the data stripe size and transfer time as well. Therefore, we choose the time length approach since the display time of each data stripe is fixed and it is simpler to meet the continuous display requirement.

3.3.1 First Level

The first level of data striping divides objects into logical segments using two steps. In the first step, a number of low resolution data segments are found. In the second step, long logical segments are divided into short ones.

The low temporal resolution data segments of a multimedia object are composed of the rough view of the object in the time dimension. For example, all

ten-second clips in the beginning of every 5 minute video may form the low temporal resolution data segments of an two-hour video. In the first step, the low temporal resolution data segments are first identified from the object. 3D-wavelet and multi-resolution video compression methods already exist to provide low temporal resolution data. The R-frames for MPEG compressed objects can also be used as the low temporal resolution data. These low temporal resolution data are interleaved with the high temporal resolution data in the object. Hence, the object when viewed as a number of continuous data blocks is already divided by the low resolution data into logical segments at the change cut positions.

If the data compression method cannot provide low temporal resolution data, automatic data segmentation methods that divide objects into data segments have already been studied by many researchers. If automatic methods are also not available, the whole object can be treated as one long logical segment in this step.

The low temporal resolution data segments provide a temporal preview of the content of the object. The size of these low temporal resolution data is assumed to be much smaller than the object size. The low temporal resolution data segments divides an object X into (X_1, X_2, ... , X_m) as shown in Figure 3.2.

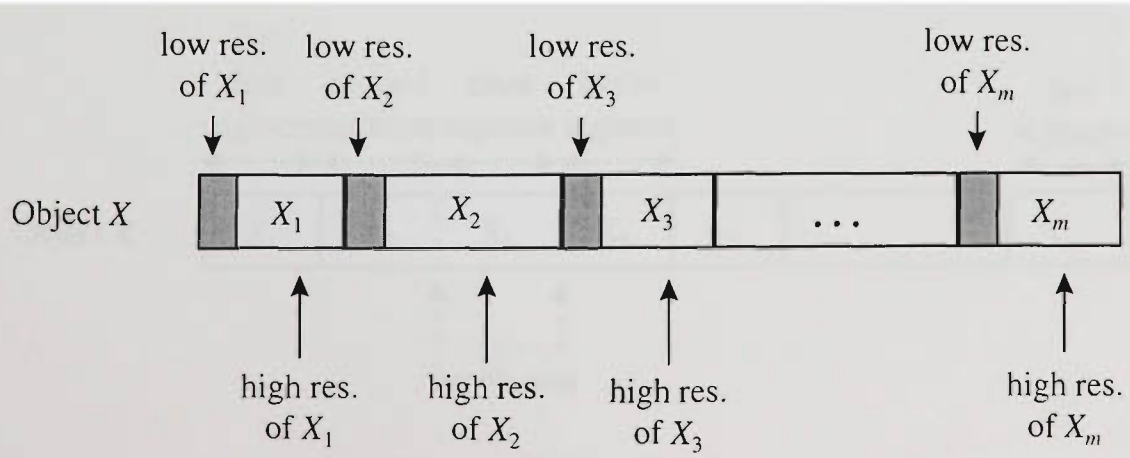


Figure 3.2. Low temporal resolution data segments

In the second step, a number of *length cuts* are found from the logical segments. The length cuts are the logical positions to split long logical segments exceeding a certain length into shorter ones.

The list of change cuts and the list of length cuts are then merged to form a list of logical positions to split the object. An object is then split into logical segments at these positions. Letting c be the number of low resolution data segments, s be the number of length cuts, we have the number of logical segments, m, is equal to

$$m = s + c. \tag{3.1}$$

Additional logical segments are created from long segments at length cuts in Figure 3.3. Without finding the low temporal resolution data segments, the low resolution data may miss out important information. Without using the length cuts, some logical segments may be too lengthy that jumps to the middle of the logical segments are desired. Therefore, the merged list provides a list of logical and reasonably spaced positions to start presenting any logical segments. This first level data striping method divides data logically so that the user access pattern of multimedia data can be handled efficiently.

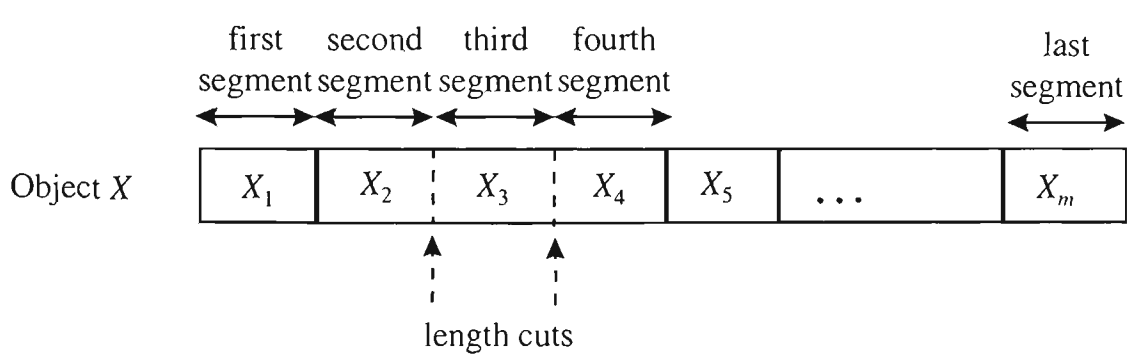


Figure 3.3. Length cuts

3.3.2 Second Level

In the second level of data striping, each logical segment is split into a number of data stripes while maintaining the continuous display requirement. It requires that the display time of any data stripe is not less than the retrieval time of the next data stripe. That is, the display time of a data stripe in any segment is not shorter than the retrieval time of the next data stripe in the same segment. Also, the display time of the last data stripe in each segment is not shorter than the retrieval time of the first data stripe of the next segment.

The logical segments of an object X is divided into data stripes $X_{i,j}$ in Figure 3.4. Since the length of each logical segment is bounded by the length cuts, the number of data stripes, n, is also bounded above. Each logical segment of an object X is then split into n data stripes. The first data stripe is also called the start up data stripe.

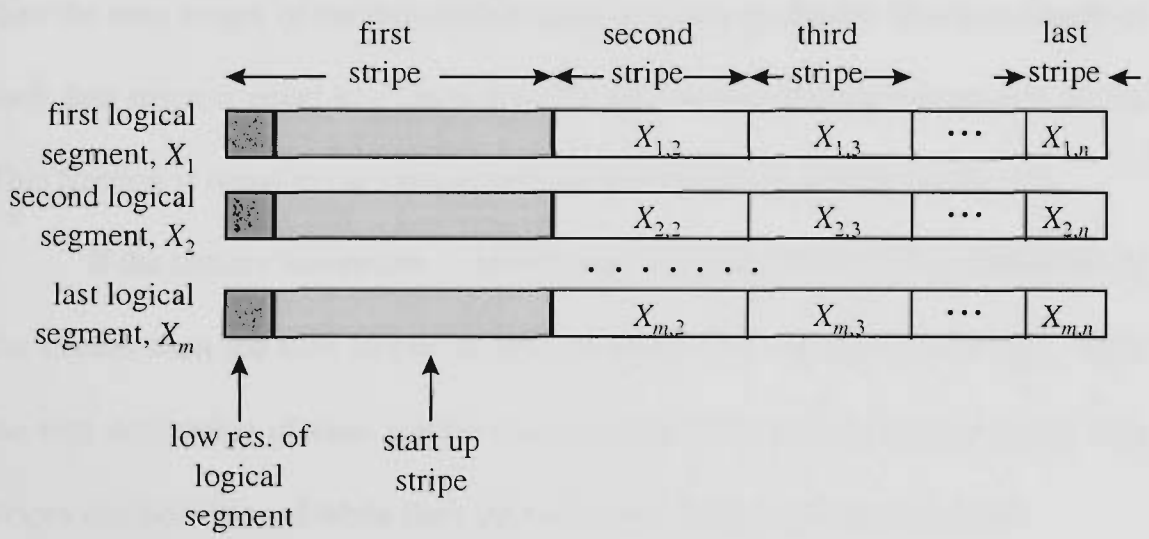


Figure 3.4. Second level data striping

We divide the logical segments into data stripes in order that the data retrieval time of each segment overlaps with the displaying time of the segment itself. Each data stripe should contain an integral number of media blocks. For example, a frame in MPEG compression can always be displayed, a data stripe may contain an integral number of frames. The size of the last data stripe in each logical segment, $X_{i,n}$, should also occupy at least one media block.

After the start up data stripe of each segment is retrieved from tertiary storage, the second data stripe can be retrieved while the start up data stripe is being consumed. While a data stripe is being consumed, the next data stripe is retrieved and so on. The data stripes of each segment are accessed sequentially in a fixed sequence only, such as $X_{1,3}$ after $X_{1,2}$, $Y_{1,4}$ after $Y_{1,3}$, and $Z_{2,2}$ after $Z_{2,1}$. If an object is accessed in the reverse sequence, then a reverse object needs to be created.

If the tertiary bandwidth is lower than the display bandwidth of the stream, then the time length of the data stripes should reduce gradually. The time length of each data stripe is equal to a fraction of the time length of the previous data stripe. This fraction is equal to the ratio of tertiary bandwidth to display bandwidth.

If the tertiary bandwidth is higher than or equal to the display bandwidth of the stream, then the data stripes should be divided in constant time length. After the first data stripe of each segment is retrieved from tertiary storage, other data stripes can be retrieved while their previous data stripe is being consumed.

3.4 Data Placement on Tertiary Storage

We consider placing data stripes on media units using two methods, a high concurrency placement method and a low latency placement method. The choice between these two placement methods depends on the tertiary drives being used.

When high end tertiary drives are used, multiple streams can be served concurrently to reduce average response time. The high concurrency placement method should be employed to raise the system throughput for large number of concurrent streams. We present this placement method in Section 3.4.1.

When low end tertiary drives that can serve only one stream at a time are used, the low latency placement method should be used to reduce the response time of user interactive requests. The low latency placement method is described in Section 3.4.2.

3.4.1 High Concurrency Placement

The high concurrency placement method consists of three parts. Firstly, we describe our approach to partitioning the media units in part (a). Next, we present the sequence of placing data segments among media units in part (b). Lastly, we elaborate the placement within each media unit in part (c).

(a) Media Units Partitioning

In order to store and retrieve data segments efficiently, we evenly partition and assign the media units to the tertiary drives. Since the number of media units is many times more than the number of drives, each drive should be assigned with several media units. A few remaining media units may be left unused. Therefore, we assume that each drive is assigned with the same number of media units. An example to partition twelve media units among two tertiary drives is shown in Figure 3.5.

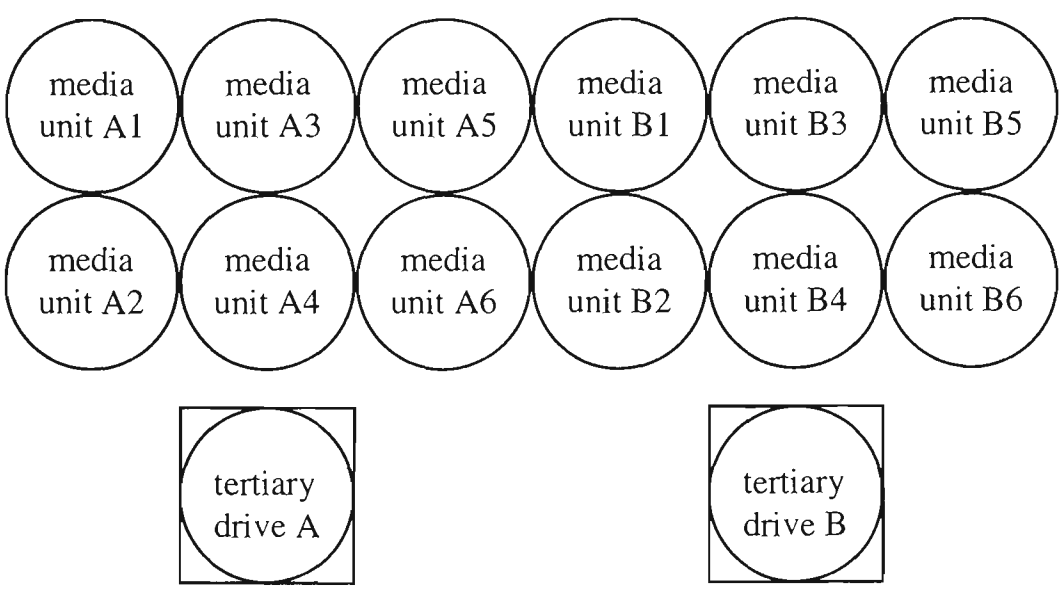


Figure 3.5. Media unit partitioning

There are two reasons that lead us to consider dividing the media units among the drives. First, an array of automatic disk changers or an array of libraries may not be able to share media units between them so that our placement method can be applied directly to these storage systems. Second, contention for a media unit is avoided since each drive would access only the media units assigned to it.

If the times to exchange a media unit to any drive are the same, the media units can be assigned randomly to the drives. Otherwise, the media units should be assigned to the tertiary drive that can quickly access it. The following steps can be used.

1. The media units are sequenced in descending order of average accessing speed from all drives.
2. The first media unit is assigned to the nearest drive that can access this media unit at the shortest time.
3. The next media unit is also assigned to the drive nearest to it.
4. If a drive is assigned with the maximum number of media units, this drive is then removed from the list of drives to be assigned.
5. The above two steps are repeated until all the media units are assigned.

The above steps can distribute evenly the media units to the drive that access them.

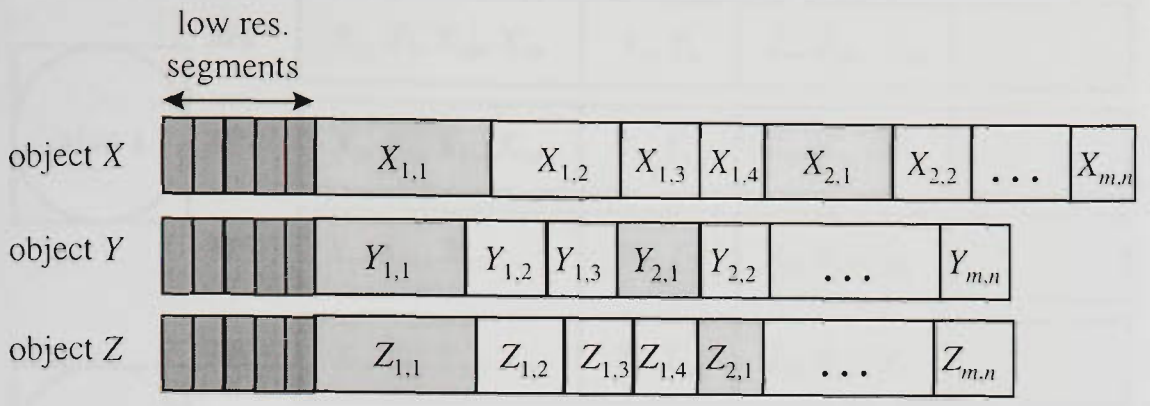


Figure 3.6. Reorder data segments in high concurrency placement

(b) Placing data segments among media units

After an object is divided using the two level data striping method, the data segments should be rearranged to reflect the sequence of data usage. If object preview is required, all low temporal resolution data segments of each object should be stored contiguously. This order reflects the sequence of data segments during the retrieval mechanism. The original order of data segments in the object can easily be restored using a simple linked list. An example of reordering data segments of three multimedia objects, *X*, *Y*, and *Z*, is illustrated in Figure 3.6.

The data segments are then stored on all media units in a cyclic manner. An example of placing data segments on six media units and two drives is shown in Figure 3.7. The low temporal resolution data segment and the first data segment are placed in media unit *M*1 for drive 1. The second data segment is then placed in media unit *M*2 for drive 2 and so on. After a data segment is placed in the last media unit *M*6, the next data segment is placed in media unit *M*1 again. This process is repeated until the entire object is stored.

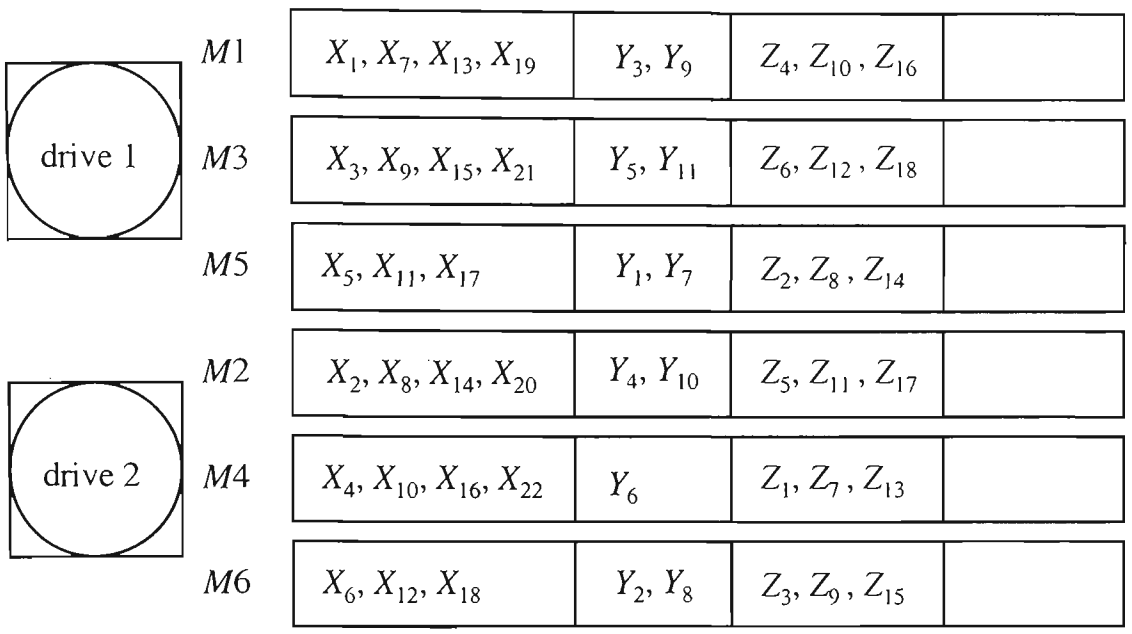


Figure 3.7. Placing data segments among media units

When the number of data segments is more than the number of media units in the tertiary storage library, the whole object is stored on all the media units. More than one data segments belonging to the same object can be stored on the same media unit.

When the number of data segments is less than the number of media units, the object is stored on fewer media units. The media units being chosen to store an object should be media units that are exchanged consecutively in a round, otherwise there would be some gaps while retrieving the object.

When the data segments are placed among the media units, we can distribute them using two strategies. In the first strategy, the first data segments of every object are stored on the first media unit of the first drive. However, the space in the first media unit of the first drive can easily be filled up. When this happens, any long objects that contain large number of data segments cannot be placed to this tertiary library. Short objects could be placed but not following the

same strategy. This would quickly fill up the storage space of the tertiary library. Therefore, we consider this strategy wastes storage space on media units.

In the second strategy, the data segments of all objects are distributed evenly to the media units. The free storage space in each media unit are maintained as even as possible. It would be easy to find consecutive media units to store new objects. Therefore, this strategy that distribute data evenly is better than the previous one.

When a number of objects are placed on the tertiary library, we use a simple method to distribute the data segments evenly. The first data segment of the first object may be placed in the first media unit. If the last data segment of this object is placed in the media unit N, the first data segment of the next object is then placed in the media unit N+1. Using this method, the number of data segments in each media units would roughly be the same.

After some objects are deleted, the objects may no longer be distributed evenly to the media units. Since the data segments are evenly distributed to the media units at first, this problem should not significantly affect the performance. Even when the problem becomes serious, it can simply be solved with data reorganizations.

We place the data segments in this way in order that the requests of all concurrent streams would retrieve data segments following a fixed sequence of media units. After each media exchange, one request of every concurrent stream is served. The media exchange is shared among all concurrent streams. Therefore, the overhead of exchanging media is reduced to raise the throughput.

(c) Placing data segments within media units

Each object should have all its data segments placed together within each media unit so that the sequences of data segments belonging to the same object is preserved on all media units. The data segments are placed according to the access frequency of the object. The data segments of the hottest object should be placed closest to the position after media exchange in the media unit (Figure 3.8). If object X has the highest access frequency, then data segments of object X are always placed before data segments of object Y and object Z on all media units.

For instance, on longitudinal tapes, data segments of hotter objects are placed nearer the loading point after media exchange and data segments of colder objects are placed far from the loading point.

Hence, hotter objects are accessed at a shorter distance from the loading point. Since the access overheads increase monotonically with the travelled distance, the access overheads of hotter streams are traded off with the access overheads of colder streams. Therefore, the average access overheads using this frequency based method are lower than that using the random placement method.

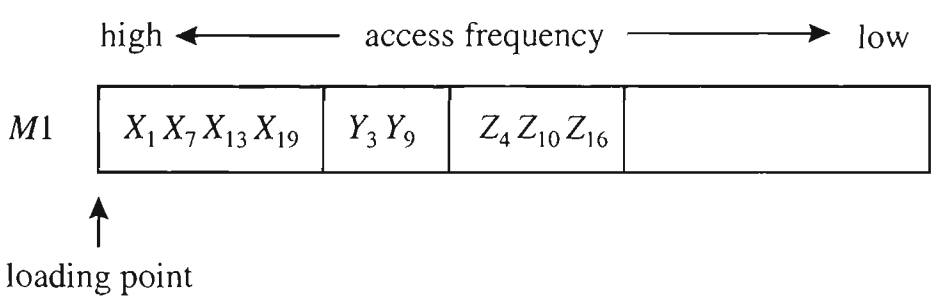


Figure 3.8. Placing data segments within media units

3.4.2 Low Latency Placement

When low end tertiary drives are used, the tertiary bandwidth may not be sufficient to serve multiple streams. The low latency placement method should be used to provide instant response to user interactive requests. The data stripes of an object are arranged in a different sequence as follows.

All low temporal resolution data segments of the object are first kept together. The start up stripe of all logical segments follows the low resolution data segments. The other data stripes are then placed at the end of the object. This order reflects the sequence that data stripes are retrieved. The original order of data stripes in the object can easily be restored using linked lists. An example of reordering data stripes for three multimedia objects X, Y, and Z is shown in Figure 3.9.

If the low resolution segments and start up stripes are pre-loaded to the disks during initialization, the low resolution data segments and the start up stripes of all objects are stored contiguously on a few media units to minimize the time to pre-load these data from tertiary storage. Other data stripes of each object are stored together on one media unit to reduce the need for extra media exchange.

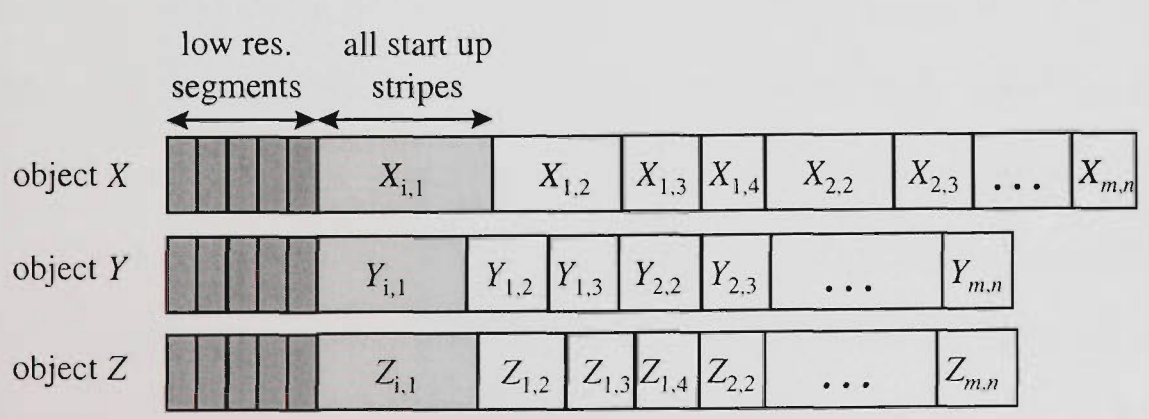


Figure 3.9. Reorder data stripes in low latency placement

If only the low resolution segments are pre-loaded, then the low resolution segments of all objects are stored contiguously on a few media units. Other data stripes of each object are stored together on one media unit.

If there is no pre-loading and any object is loaded on demand, then the entire object is stored contiguously. The low resolution data stripes of all logical segments should first be stored in front. The start up data stripes of all logical segments are then stored next. After that, other data stripes are stored at the end.

The data stripes are stored in these ways in order that the smallest number of media exchanges is required during the system initialization and objects retrieval. This sequence reflects the earliest time that the data stripes would be requested by users.

3.5 Data Placement on Disks

In this Section, we present our novel Bandwidth Based Placement method to place heterogeneous multimedia objects together with binary and textual data on constant density recording disks.

In our tertiary storage organizations, some data are kept resident on disks whereas other data are loaded on demand and cached on the disks. Hence, multimedia data may be stored as resident data or cache data on disks.

Different multimedia objects may have different bandwidth requirements. High bandwidth multimedia objects, such as video, may require more data per second than low bandwidth multimedia objects, such as voice. In order to meet the continuous display requirement of data streams, high bandwidth streams should be served with higher data rates than low bandwidth streams.

Apart from the multimedia objects, binary and textual data are also stored on the same set of disks. These binary and textual data files are accessed using discrete requests as in traditional computer systems. Hence, these requests can be served with arbitrary data rates.

Since the same amount of data are transferred in less time from outer zones than from inner zones of CDR disks, the throughput of accessing data from different zones varies. This variation in data transfer rate is being considered to place the multimedia objects in our bandwidth placement method as follows.

First, the multimedia objects to be stored on disks are grouped according to the bandwidth requirement of the object. Binary and textual data are grouped as an arbitrary bandwidth group. The number of bandwidth groups is found as the

number of groups with different bandwidth requirements. The bandwidth groups are then sorted from the highest to the lowest bandwidth.

Next, each CDR disk is divided into the same number of zone groups as the number of bandwidth groups. Objects belonging to the highest bandwidth group are stored at the outermost zone group on all disks. Objects belonging to the next highest bandwidth group are stored at the next outermost zone group on all disks and so on. After the objects in all bandwidth groups are stored, the binary and textual data are stored at the innermost zone group. A bandwidth based object placement dividing a CDR disk into four zone groups is illustrated in Figure 3.10.

Using this method, multimedia objects are stored together with traditional data files on the same set of disks. Objects with higher bandwidth requirements are stored at zone groups outside objects with lower bandwidth requirements. Binary and textual data are stored at the innermost zones.

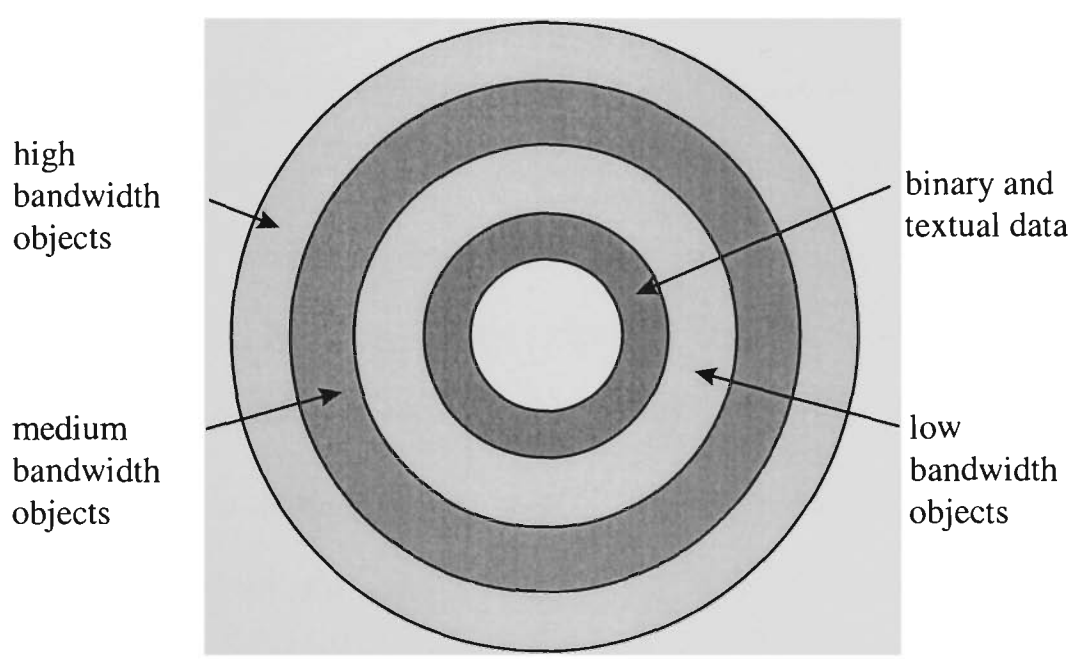


Figure 3.10. Bandwidth based placement

Since high bandwidth data are stored at outer zones than low bandwidth data, the transfer rate of higher bandwidth data are always higher. This would reduce the data transfer time in accessing high bandwidth objects at the expense of longer data transfer time in accessing low bandwidth objects. Therefore, the access time to high bandwidth objects is reduced at the expense of longer access time to low bandwidth objects.

In fact, nothing is lost for the requests on binary and textual data. Since these data are normally accessed in small blocks as in traditional computer systems, only a few kilobytes of data are often sufficient to satisfy each request and disk performance is often measured in number of I/Os per second. The requests on textual or binary data still enjoy a similar number of I/Os per second and they are not worse off.

3.6 Multimedia Data Migration Methods

We consider migrating multimedia data over the storage hierarchy using two methods: the concurrent streams management and the segment based pipelining method.

The concurrent streams management in Section 3.6.1 is designed for the high concurrency placement method. It manages the acceptance of new arriving streams, the scheduling of data requests, and the scheduling of exchange requests.

Pipelining methods overlap the retrieval time with the display time. Systems that use pipelining methods respond to new streams at the shortest start up latency. When low latency placement method is used, multimedia data are pipelined according to the user's requests for logical segments of objects. We describe this segment based pipelining in Section 3.6.2.

3.6.1 Concurrent Streaming for High Concurrency Placement

When multimedia objects are required from the storage system, the user program initiates new streams to access multimedia data objects. The system first checks if there is sufficient disk bandwidth to serve the stream or not. If disk bandwidth is insufficient, then the stream is rejected. Otherwise, the system checks whether the entire object is cached to disks or not. If the whole object is accessible from disks alone, the stream is accepted and the object is accessed from the disks. Service of request streams from disks alone can be controlled by the Group Sweeping Scheduling (GSS) method that is already described in Section 2.4.3 and is not discussed any further.

If part of the object is not cached on disks, data have to be migrated from tertiary storage. The new stream is acceptable only if the continuous display requirement is not violated for any concurrent streams. That is, the stream is accepted if

$$\frac{S_1 + M_1}{\delta_1} + \frac{S_2 + M_2}{\delta_2} + \ldots + \frac{S_n + M_n}{\delta_n} \leq D. \tag{3.2}$$

where S_i are the retrieval overhead times, M_i are the data transfer time, and δ_i are the data display times, n is the number of streams, and D is the number of tertiary drives. We shall prove this feasibility condition to accept heterogeneous streams over multiple devices in Chapter 5.

If a new stream is not accepted, it is placed in a stream queue of the Parallel Stream Controller (Figure 3.11). New streams waiting in the queue are served according to the order of the media unit that contains the first data segment. Since the data segment being accessed by the first request may reside on any media unit, this method does not discriminate against any streams.

Once accepted, the Parallel Stream Controller creates the new stream object. The new stream sends two requests to every tertiary drive and waits. After a drive finishes a request, it sends an access notification back to the stream. The stream then sends the next request to the same drive. In this way, requests are served by all tertiary drives in parallel.

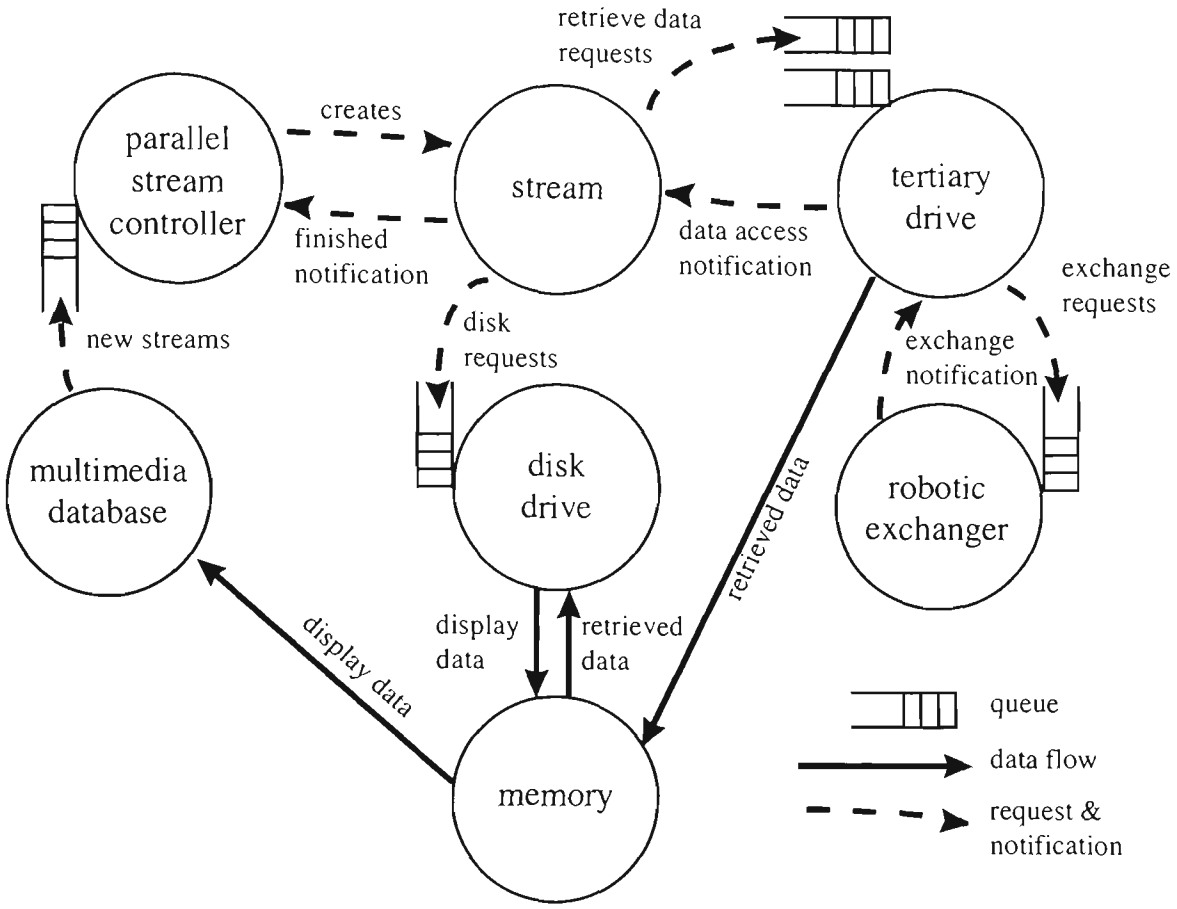


Figure 3.11. Concurrent streams management

An accepted stream starts to display data at the completion of at least one cycle from every tertiary drive. While the first data segment is being displayed, the stream checks to see if the next data segment has been retrieved or not. If the next data segment has been retrieved, the stream then displays the next data segment. Otherwise, the stream starves and waits until the next data segment has been retrieved from tertiary storage.

Only a few data segments are kept in memory. If the number of retrieved data segments is more than a threshold, then the latest retrieved data segments are flushed to disks. This threshold may be adjusted according to the amount of available memory. Before a data segment is consumed, the stream checks to see if the next data segment is present in memory. If the next data segment was retrieved

but flushed to disks, then the stream sends a disk request to access the next data segment back from the disks.

After a stream finishes displaying all data segments, a notification is sent back to the parallel stream controller. If the stream is aborted by the user, it stops sending new data requests to the tertiary drive and data migration will stop eventually.

Since the drives are shared by all the streams to migrate data segments, the scheduling algorithm has an influence on the performance of the requests. We describe how the tertiary drives serve data requests asynchronously in part (a) and how the exchange device serves exchange requests in part (b).

(a) Tertiary Drive Serves Data Requests Asynchronously

Each tertiary drive keeps the waiting requests in two queues. The first queue keeps waiting requests that access data segments on the current media unit and the second queue keeps requests that access data from other media units. After the tertiary drive serves all requests from the first queue, it moves requests for the next media unit from the second queue to the first one and serves them. This process is repeated until both queues are empty.

Since every accepted stream sends two requests to every tertiary drive, the request for data on the next media unit waits in the second drive queue. When the drive accesses the next media unit, there is one outstanding request per stream to be served. The outstanding requests allow queue scheduling to be optimized.

When a new stream arrives at an idle tertiary drive, the drive exchanges the first media unit and retrieves data segment of the first request on this media unit.

After one request per stream is served on the current media unit, the drive exchanges media and serves the next request of each stream on the next media unit. After one request per stream is served for data on the last media unit, the drive then serves requests on the first media unit again. This process is repeated until there is no more outstanding requests (Figure 3.12). Hence, the tertiary drives access the media units cyclically in a round robin manner.

When a drive needs a new media unit, it sends an exchange request to the exchange queue and waits. After the required media unit is exchanged, the drive then serves one request per streams for data on this media unit.

After each media exchange, the tertiary drive moves the read/write heads to the required data position of the first request and starts to transfer data. After that, it moves to the required data position of the next request and starts to transfer data and so on.

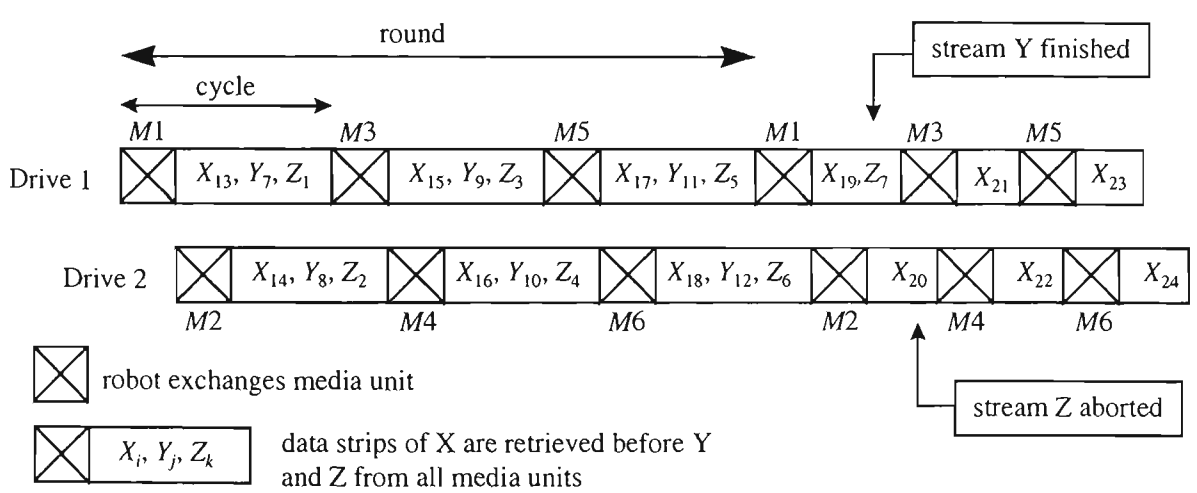


Figure 3.12. Request service order

The SCAN, FCFS, or GSS scheduling methods may be used to schedule the requests on each media unit. The FCFS method maintains the start up latency of new streams short. However, the system throughput is low since excessive search overheads are involved in retrieving data randomly on the media unit. The SCAN scheduling method is an efficient method that achieves near-optimal throughput with low complexity. The GSS method tradeoffs the start up latency with system throughput. We assume that the SCAN scheduling method is used because it achieves the highest system throughput.

(b) Media Exchange Request Service

When the tertiary drives need a different media unit, they send exchange requests to the exchange queue and wait. The exchange device serves the exchange requests in this exchange queue. After it exchanges the media unit on the first drive, it looks for any exchange requests on the second drive waiting in the exchange queue. If a request is found, it serves this exchange request. Otherwise, it serves the first exchange request in the queue.

If one exchange device is dedicated to each drive, the exchange device is idle while its dedicated drive is not exchanging media. The exchange requests, hence, never need to wait and no contention for media exchange will ever occur.

If several drives share one exchange device, there may be several outstanding exchange requests in the exchange queue. The exchange device selects one exchange request from the queue following a modified round robin scheduling policy.

We modify the round robin policy in order that the exchanger device would perform the similar number of exchanges for each tertiary drive. We assumed that requests from the least served drive have the least slack time from its deadline. The exchange request from the drive with the least number of served exchange requests is chosen to be served first. After this exchange request is served, another exchange request is selected similarly from the remaining requests in the queue. Therefore, late exchange requests are served with higher priority to catch up with the display deadline.

3.6.2 Segment Based Pipelining

In our low latency placement method, data stripes of the object are reordered. All the low temporal resolution data segments are stored in front of other data stripes. The start up data stripes of all logical segments are stored in front of the rest of the object. The low resolution data segments and the start up data stripes can be downloaded from tertiary storage during system initialization. Other data stripes are downloaded on demand.

When multimedia objects are required from the tertiary storage, the user program initiates a new stream to access a multimedia object. Since the low latency placement method is designed for single stream, each tertiary drive serves only one stream at a time.

When the tertiary drive serves a stream, the system checks to see whether the low resolution data and the start up data stripes of all logical segments are downloaded to disks or not. If these data stripes are not found on disks, then the low resolution data are first downloaded. After the low resolution data are downloaded, preview is allowed on the load resolution data segment and the start up data stripes of all logical segments are downloaded. After the start up data stripes are downloaded, data consumption begins as in other pipelining methods.

Since different parts of the multimedia object are required in different presentation modes, we describe how the segment based pipelining supports various presentation modes in the following two parts. The VCR-like functions are described in part (a). The browsing interface functions are shown in part (b).

(a) VCR-like Functions

When user plays a multimedia object in normal mode from the beginning to end, the start up data stripe of the first logical segment is firstly retrieved from disk. While this start up data stripe is being displayed, the tertiary drive exchanges media and retrieves the second data stripe to disks. Before the start up data stripe is completely displayed, the second data stripe has already been retrieved. While the second data stripe is being displayed, the third data stripe is being retrieved and so on.

While the last data stripe of the first segment is being displayed, the start up data stripe of the second segment is retrieved from disks. Since data are retrieved directly from either tertiary storage or disks to primary memory and the start up data stripes can be much larger than the last data stripes of a logical segment, the retrieval time of start up data stripe from disks can be comparable to the display time of the last data stripes.

This pipeline process continues until the entire object is displayed. Thus, the user only waits for a short time while the disks are retrieving the start up data stripe of the first logical segment before display begins.

When the user previews an object, the low temporal resolution data of all logical segments are retrieved from disks and displayed in normal speed. After all the start up data stripes are retrieved to disks, resume to normal model is allowed. When the user resumes normal mode at any position of the low temporal resolution data, the tertiary drive then searches for the corresponding logical segment. The start up data stripe of the corresponding logical segment is retrieved from disks for display. While this start up data stripe is being displayed, the

tertiary drive retrieves the second data stripe of this logical segment. Pipelining then continues on this logical segment as in normal mode.

Fast reverse display mode is served similarly. The low temporal resolution data are displayed in the reverse order. The start up data stripe belonging to a previous logical segment is retrieved from disks. The tertiary drive searches to a previous logical segment on the media unit but the data stripes of this logical segment may have already been cached on disks.

(b) Browsing Interface Functions

When the user browses a multimedia object, the low temporal resolution data segments are retrieved from disks and presented to the user. After the user selects to start displaying the object at full resolution beginning at a logical segment, the start up data stripe and the second data stripe of the corresponding logical segment are retrieved from disks and the media unit respectively. Hence, the user only waits while the disks are retrieving the start up data stripe of the required logical segment from disks.

These are the display modes of multimedia data as we have described in Chapter 2, our segment based pipelining method provides short latency in supporting interactive user functions.

3.7 Discussions

Retrieving objects from tertiary storage incur long access latency. Storing data on disks alone waste large storage space of higher media cost. Placement on a storage hierarchy trades off access latency with media cost. Large amount of data can be stored on tertiary storage devices with lower media cost. Small amount of start up data can be retrieved from disks with low latency. Pipelining multimedia data from tertiary storage can hide the long access latency of retrieving data from tertiary storage. Hence, placing multimedia data at both levels in a storage hierarchy and retrieving them using pipelining technique is desirable.

- **High Concurrency Placement and Concurrent Streaming**

When objects are stored using non-striping method on the media units, the number of objects that can be stored together on one media unit is limited. Since many media units may exist in the storage library, the probability that consecutive streams access objects from the same media unit is low. Although this probability can be raised using frequency based placement, the increase is mild since access frequencies on multimedia objects are less skewed on tertiary storage and the access frequency of each object is low. Even when requests for objects on the same media units are served consecutively, the gain is little unless the waiting queue is deep.

When objects are stored using parallel striping method, the segments are striped to a few media units such that the stripe width does not exceed the number of drives. All the drives serve one stream at a time. The entire object is retrieved by all drives in parallel. The service time of each stream is reduced by sharing

among the drives. Since each object resides on several media units, several media units are exchanged at the start of the stream. The drives becomes idle while waiting for exchange, this artificial contention of exchange requests degrades the system throughput.

Although the parallel striping method increases the number of objects that share the space on one media unit, the probability that two requests can share a media exchange overhead is low. Intuitively, this probability is inversely proportional to the number of media units per drive.

Our high concurrency placement method stripes multimedia objects across all media units. In doing so, each object occupies only a fraction of its data on each media unit. Since multimedia objects are large, each object can be striped across many media units. Although the capacities of the media units are limited, each media unit can still store one or more segments of every object.

Since each media unit contains segments belonging to every object, each stream accesses data from every media unit. When more streams are concurrent, the media exchange overheads are shared among more streams. This shared media exchange time per stream can be reduced to less than the time for one media exchange when the number of concurrent streams is sufficiently large. Hence, our new method could reduce the media exchange overheads for a large number of concurrent streams.

When an object is accessed, all the media units are accessed. The system throughput can be wasted by the artificial media exchange overheads. This method seems to be undesirable if only a small number of streams are concurrent. Fortunately, the multimedia data objects are consumed slowly. When data are

retrieved too quickly from tertiary storage, excessive data are copied to disks to save memory space, resulting in large staging disk buffers. When only an adequate amount of data are supplied in time, data are consumed almost directly from memory. Therefore, the reduction in system throughput does not produce much adverse impact on the system performance.

One obvious drawback of our high concurrency placement method is the long start up latency of new streams. Since the tertiary drives access media units in a cycle, they do not immediately respond to new streams that need media units different from the current media units on the tertiary drives. We have attempted to solve this problem in two ways. First, if plenty of tertiary storage space is available, the first round of data can be made available through data replication. Some drives are reserved to access the first round data from these replicated media units. The start up latency for media unit is then reduced. New streams can then be immediately added to the concurrent streams and start to display. Unfortunately, this hybrid method undesirably lowers the system throughput by the reserved drives that access the replicated data. Hence, the hybrid method is not studied any further.

Second, data to be retrieved in the first round are kept resident permanently on the disks. Since the display time for all data in the first round is sufficient for a full round time, new accepted streams can be initiated with disk latency. This resident leader method trades off disk storage with start up latency. Unless the objects are so large that they are retrieved in many rounds, the benefit of this method is minimal.

Thus, our remaining option is to let new streams to wait. This would not be a problem for requests that are scheduled in advance. Even for ad hoc streams, the start up latency is bounded above.

Unfortunately, data segments must be accessed in a fixed sequence in the high concurrency placement method. Although preview is allowed on low temporal resolution data on disks, any interactive user functions can only be supported after the entire object is completely copied to disks.

- **Low Latency Placement and Segment Based Pipelining**

Our low latency placement and segment based pipelining method ensures that user enjoy the shortest latency on interactive operations even though partial objects are retrieved from tertiary storage.

In pipelining methods, the size of the first slice is proportional to the size of the object. In our segment based pipelining, the total size of the logical segments is equal to the size of the object; the aggregate size of start up data stripe of all logical segments is hence comparable to the size of first slice of the object in other pipelining methods. When the tertiary bandwidth is the same, the time to download the start up data stripes of all logical segments in our method is then comparable to the time to download the first slice of the object in other pipelining methods.

The only limitation of the segment based pipelining method is that data must start to display at the beginning of logical segments. User may jump to any logical segment and data stripes can then be displayed continuously.

When the start up data stripes are kept resident on disks, storage space for a fraction of all objects is required on the disks. Since the number of objects is expected to be much more than the number of streams, the total resident disk space is larger than the disk buffer size.

When the segment based pipelining is used, the hierarchical storage system can respond to user requests at disk latency. Since the access latency of magnetic disks is at least an order of magnitude shorter than the access latency of tertiary drives, this method of using storage hierarchy is able to respond to interactive operations much faster than other methods.

- **Bandwidth Based Placement**

The bandwidth based placement approach reduces the variance of request service time by trading off the service time of requests on high bandwidth objects with that on low bandwidth objects.

Requests for data on different zones in CDR disks are served with different transfer rates. The transfer rate at inner zones may be too slow to support the bandwidth requirement of some objects to meet their continuous display requirement. If the high transfer rate storage area at the outer zones is filled with low bandwidth objects, the high bandwidth objects consequently cannot be stored on the disk, and the disk space is then eroded. Therefore, bandwidth is an important concern in placing heterogeneous multimedia objects on CDR disks.

Unlike the temperature of objects, data bandwidth is static and is intrinsic to the multimedia object. Once the objects are placed on the disks, they can stay at the same location since their bandwidths never change. Data re-organization is not required until some objects are replaced with those of different bandwidth.

3.8 Chapter Summary

We have presented our novel storage and retrieval methods for multimedia data on hierarchical storage systems, and we have proposed a two level data striping method to partition multimedia object. When a multimedia storage system needs to support a large number of users, the high concurrency placement method should be used to share the overheads in serving concurrent streams. This storage organization has the ability to share out the media exchange overheads with all concurrent streams. When a multimedia storage system only needs to serve one stream at a time, the segment based pipelining method should be used to reduce the user latency.

We have also established a placement strategy whereby heterogeneous multimedia data on constant density recording disks are placed according to their bandwidth requirement. This bandwidth based placement approach helps to balance the transfer rate with the bandwidth requirement of the objects.

Chapter 4

Disk Storage Structure Analysis

4.1 Introduction

As we have seen in the previous chapter, the timing and type of system demand and usage cannot usually be predicted exactly in advance, therefore probabilistic techniques will need to be employed for predicting and accessing system behaviour and performance. With multiple streams of data requests, there will invariably be resource contention on different components of the system. With probabilistic demands, queueing analysis is the most suitable technique to employ, and contention models based on queueing theory are hence developed to analyse the system performance. This will allow the response time, waiting time, delay and utilization measures to be quantitatively evaluated.

A detailed disk storage model, which is not available before, is developed in this chapter. It is able to predict with a high degree of accuracy the behaviour of multimedia data requests on constant density structures and it is also able to predict the data transfer time for the bandwidth based placement strategy. We use the disk performance model to analyse the service delay of disk requests and to compare the performance of CDR disk requests with VDR disk requests. The

impacts of various disk parameters and the bandwidth based placement method on the request service time are also studied.

4.2 Performance Model of Disk Requests

As we saw in Chapter 2, in a CDR storage structure, the recording density is constant throughout the entire disk and the rotation speed of the disk is fixed. Here, we study their performance using a continuous model, which is customarily used in such analysis [95]. We also assume that the starting positions of data retrieval requests are randomly distributed throughout the disk surface. We adopt the notations given in Table 4.1 for this model.

Table 4.1. Notations in disk performance model

Parameter	Meaning
a	radius of the innermost track
b	radius of the outermost track
s	seek time
D	seek distance
l	rotational latency
T	disk revolution time
τ	data transfer time
k	recording density in bytes per unit length
R	number of bytes to be transferred

Let $P_x dx$ be the probability of having the required data located in track of radii between x and $x+dx$. This probability can be approximated by the probability of having the data within the area of a ring with radii bounded by x and $x+dx$ (Figure 4.1). Hence, we have

$$P_x dx = \frac{\text{Area of ring of radii } x \text{ and } x + dx}{\text{Total disk surface area}}.$$

When dx tends to zero, the area of ring of radii x and $x+dx$ can be approximated with $2\pi x dx$. Since the total surface area for data storage is $\pi(b^2 - a^2)$, we have

$$P_x\, dx = \frac{2x dx}{(b^2 - a^2)}. \tag{4.1}$$

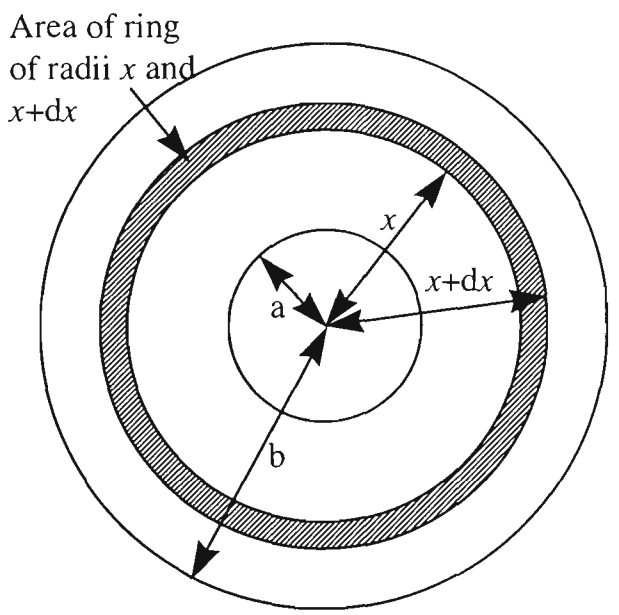


Figure 4.1. Area of track of radius x

4.2.1 Seek time

The time that disk heads spent in seeking the destination track depends on the location of the two tracks in consecutive requests. Seek distance from the starting track x to the ending track y is $|y\text{-}x|$ as shown in Figure 4.2.

Let $P_x dx$ be the probability of data in track of radii x to $x+dx$ and $P_y dy$ be the probability of data in track of radii y to $y+dy$. The mean seek distance is given as $\int_a^b \int_a^b P_x P_y |y-x| \, dy \, dx$. We substitute P_x and P_y from Equation (4.1) to obtain the mean seek distance, $\overline{D}$, as

$$\overline{D} = \int_a^b \int_a^b \frac{2x}{(b^2-a^2)} \frac{2y}{(b^2-a^2)} |y-x| dy dx,$$

which implies

$$\overline{D} = \frac{4}{(b^2-a^2)^2} \int_a^b \left(\int_a^x xy(x-y) dy + \int_x^b xy(y-x) dy \right) dx.$$

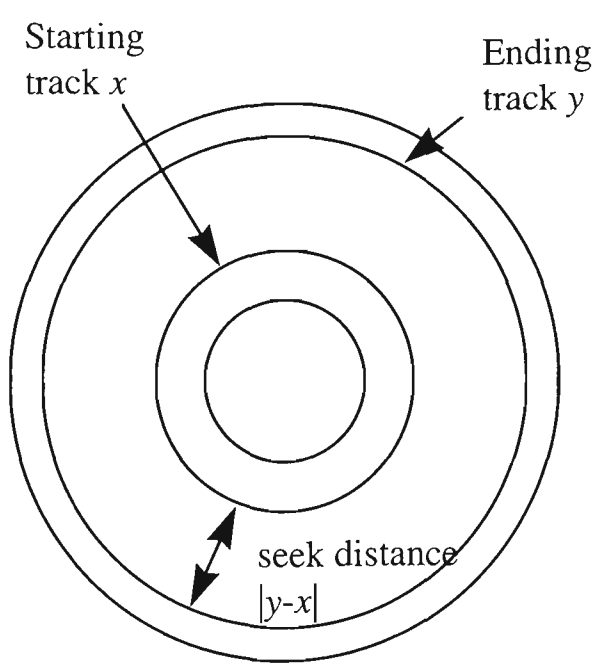


Figure 4.2. Seek distance

After simplification, this yields

$$\overline{D} = \frac{4(b-a)(a^2+3ab+b^2)}{15(a+b)^2}. \tag{4.2}$$

The variance of seek distance is given by $= \int_a^b \int_a^b P_x P_y \left(|y-x|\right)^2 \mathrm{d}y\mathrm{d}x - \left(\overline{D}\right)^2$.

We substitute the mean seek distance from Equation (4.2), P_X and P_Y from Equation (4.1), and we obtain the variance of seek distance, Var[D], as

$$\mathrm{Var}[D] = \int_a^b \int_a^b \frac{2x2y(y-x)^2 \mathrm{d}y\mathrm{d}x}{\left(b^2-a^2\right)\left(b^2-a^2\right)} - \left[\frac{4(b-a)(a^2+3ab+b^2)}{15(a+b)^2}\right]^2.$$

After simplification, we obtain

$$\mathrm{Var}[D] = \frac{(b-a)^2(3a^2+14ab+3b^2)(3a^2+4ab+3b^2)}{225(a+b)^4}. \tag{4.3}$$

We note that the mean and variance of the seek distance is quite different from a conventional disks structure which are

$$\overline{D} = \frac{(b-a)}{3}, \tag{4.4}$$

and

$$\mathrm{Var}[D] = \frac{(b-a)^2}{18}. \tag{4.5}$$

The seek time, s, can hence be found using

$$s = \begin{cases} a_1 + a_2 D, & D > a_5 \\ a_3 + a_4\sqrt{D}, & D \le a_5 \end{cases}, \tag{4.6}$$

where a_1, a_2, a_3, a_4, a_5 are fixed parameters [126].

4.2.2 Rotation latency

Since the starting positions of the data blocks for CDR disks are not fixed relative to the angular position of the disk, it is difficult to predict the starting position of the block relative to the angular position. The rotational latency is often taken to be uniformly distributed from 0 to T.

Let Q_t dt be the probability that the rotation latency is between time t and t+dt, where $0 < t \leq T$, and dt tends to zero. The mean rotational latency is given by $\int_0^T tQ_t \mathrm{d}t$. As the disk is rotating at fixed angular speed,

$$Q_t = \frac{\mathrm{d}t}{T}. \tag{4.7}$$

We have for the mean rotation latency,

$$\bar{l} = \int_0^T \frac{t\mathrm{d}t}{T} = \frac{T}{2}. \tag{4.8}$$

The variance of rotational latency is given by $\mathrm{Var}[l] = \int_0^T t^2 Q_t \mathrm{d}t - (\bar{l})^2$. We substitute Q_t from Equation (4.7) and we obtain

$$\mathrm{Var}[l] = \int_0^T \frac{t^2 \mathrm{d}t}{T} - \left(\frac{T}{2}\right)^2 = \frac{T^2}{12}. \tag{4.9}$$

4.2.3 Data Transfer Time

Data transfer time is the time for all required data in the current track to be read. When the required data are on a track of radius x, the amount of data in this track is given by $2\pi xk$. In one disk revolution, all the data on this track will pass under the head, hence the data transfer time required for R bytes at a track of radius x is given by $\frac{TR}{2\pi xk}$. As the probability of required data in track of radius x is given by $P_x dx$, the mean data transfer time is $\int_a^b P_x \frac{TR}{2\pi xk} dx$, we substitute P_x from Equation (4.1) to get the mean data transfer time,

$$\bar{\tau} = \int_a^b \frac{2xTR}{2\pi xk(b^2 - a^2)} dx = \frac{TR}{k\pi(a+b)}. \tag{4.10}$$

The variance of data transfer time is given by $\mathrm{Var}[\tau] = \int_a^b P_x \left[\frac{TR}{2\pi xk}\right]^2 dx - \bar{\tau}^2$. We substitute P_x from Equation (4.1) and the mean data transfer time from Equation (4.10), and we obtain, after simplification,

$$\mathrm{Var}(\tau) = \frac{T^2R^2}{k^2\pi^2}\left[\frac{\ln(b/a)}{2(b^2 - a^2)} - \frac{1}{(a+b)^2}\right]. \tag{4.11}$$

4.2.4 Data Transfer Time in Bandwidth Based Placement Strategy

Let the CDR disk be partitioned into n zone groups and number 1 to n starting from the innermost zone group. Let z_i be the inner radius of the ith zone group, for $i = 1, 2, ..., n$. Let τ_i be the data transfer time of the ith zone group.

We use Equation (4.10) to find the data transfer time in each zone group by substituting the inner radius and outer radius of the disks with those of the zone group. Hence, we obtain

$$\bar{\tau}_i = \frac{TR}{k\pi(z_i + z_{i+1})}, \qquad \text{where } i=1, 2, ..., n\text{-}1, \tag{4.12}$$

and

$$\bar{\tau}_n = \frac{TR}{k\pi(z_n + b)}. \tag{4.13}$$

Since $0 < a = z_1 < z_2 < .... < z_n < b$, we have

$$0 < \bar{\tau}_n < .... < \bar{\tau}_2 < \bar{\tau}_1. \tag{4.14}$$

Hence, the mean data transfer time of data from the outer zone group are always smaller than the data transfer time of the same amount of data from the inner zone group.

4.2.5 Waiting Time in Queue

When multiple requests arrive at the disk, only one of them can be served while other requests are waiting in the queue. Average arrival rate of streams of requests can easily be found, but the variance of arrival time is however unknown. Intuitively, streams of request should arrive more regularly than random requests. We have not found in the literature any distribution functions that sufficiently describe the arrival patterns of multimedia requests in streams. In addition, a disk may serve requests that access binary and textual data. These binary and textual data requests are random requests and they are mixed together with the multimedia requests in the disk queue. In order to simplify our analysis, we assume that requests arrive randomly to the disks in this model. A Poisson distribution with a fixed rate λ can thus describe their arrival pattern.

The service time of requests is represented by the sum of the seek time, rotational latency, and data transfer time of each request. The mean and variance of these time components are already analyzed in Sections 4.2.1 to Section 4.2.4.

We are interested in conservative systems in which no requests are created or destroyed. Although some multimedia requests can be discarded without too much impact on the system, the quality of service is adversely affected. Hence, we assume all requests are served.

We assume the non-preemptive scheduling condition that each request must wait for outstanding incomplete requests to finish. We also assume the First-In-First-Out (FIFO) scheduling rule, which is most commonly used in traditional systems. Although multimedia requests can also be served according to Earliest-Deadline-First (EDF), SCAN, or Group Sweeping Scheduling (GSS) scheduling

rules, the request performance using these rules depends on the storage organization in use.

We then use the Pollaczek-Khintchine formulae to analyse the performance of streams of data requests [95]. Using these formulae, request waiting time, request response time, queue length and number of acceptable streams are found.

4.3 Analysis of Request Service Time

We directly apply the performance model to find the data access time and disk throughput. The performance model shows us that data access time, and hence disk throughput, depend on a number of parameters. These parameters include innermost diameter, disk size, seek start/stop time, read/write arm moving speed, number of platters, rotation speed, and recording density. The impacts of varying these parameters on the request service time are analyzed below.

We analyze the disk performance using the specifications of five disks with diameters 10.88 inches, 5.25 inches, 3.50 inches, 2.50 inches and 1.90 inches [119]. These disks rotate at 3600 revolutions per minute (rpm) except the 3.50 inches disk that rotates at 4318 rpm. Large disks have higher recording density except the 3.50 inches disk that has a higher recording density than the 5.25 inches disk. We use, as default values, an innermost track diameter of one inch, 20 tracks per cylinder, 10 msec seek start/stop time, and 0.1 inch per msec arms moving speed. We use a block size of 100 kilobytes unless otherwise indicated. We assume that a separate seek is done if more than one cylinder of data is accessed.

We present the impact of various data block size in Section 4.3.1. The relative magnitude of various time components in the total request service time is shown in Section 4.3.2. The influence of data stream size or continuous data chunk size on the throughput is investigated in Section 4.3.3. We analyze the system throughput against various physical disk sizes in Section 4.3.4. The influence of seek start/stop time and arm moving speed on data access time is described in Section 4.3.5 and Section 4.3.6. After that, the influence of the number of disk platters, rotation speed, and recording density is investigated in

Section 4.3.7, Section 4.3.8 and Section 4.3.9 respectively. Lastly, we analyze the throughput of bandwidth based placement method in Section 4.3.10.

4.3.1 Data Block Size

In storage systems, the method of using a large block size to access a large amount of sequential data is often advantageous. Since we obtain more data from the disk each time, we save on the number of data accesses and access overheads. The data access time and the response time hence depend on the block size. We evaluate the effect of block size on disk service time for a continuous stream of requests on a multimedia object of 16 MB. When a small block size is used, multiple requests of the same stream would incur excessive overheads and hence increase the disk service time (Figure 4.3). A logarithm scale is used in the figure because of the large variation in service time. When the block size is above a threshold, the service time becomes almost level and varies only slightly.

However, there are two tradeoffs for using large block size. The first tradeoff is that more memory is required to buffer the data from the disk. The second tradeoff is that the start up latency is longer. Although we can save on the number of I/O's, we also increase the response time of each request. The start up latency of the multimedia data stream is hence adversely affected. Therefore, the smallest block size that can deliver the data in the required time is desirable.

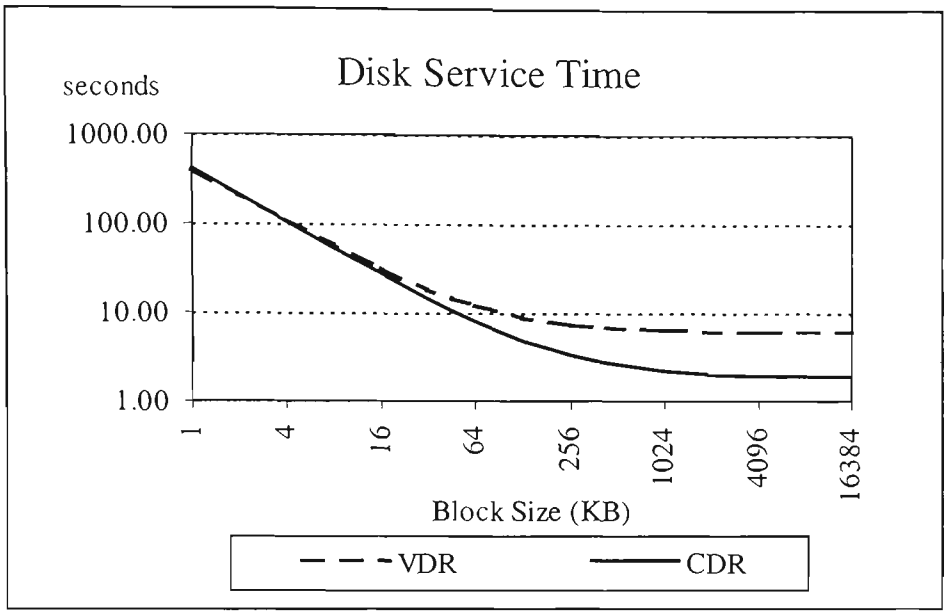


Figure 4.3. Disk service time vs data block size

4.3.2 I/O Time Distribution

The percentage of I/O time distribution of CDR disks is shown in Figure 4.4. The I/O time distribution of VDR disks is similar. When data are accessed by 10 KB or less, most of the time is spent in seeking and rotational latency. Less than 5% of time is spent in transferring data. Both seek time and rotational latency are, however, overheads in accessing data from the disk. When more data are accessed each time, the data transfer time component becomes more prominent.

As the recording density of the outer zones in CDR disks is kept at maximum recording density, up to one whole track of data can be read or written within the same disk revolution. Therefore, a full track block can deliver better performance for large sequential access in multimedia systems. However, the amounts of data stored in each track of CDR disks are different. A full track block for CDR disks requires a variable buffer size that may not make the most efficient use of memory.

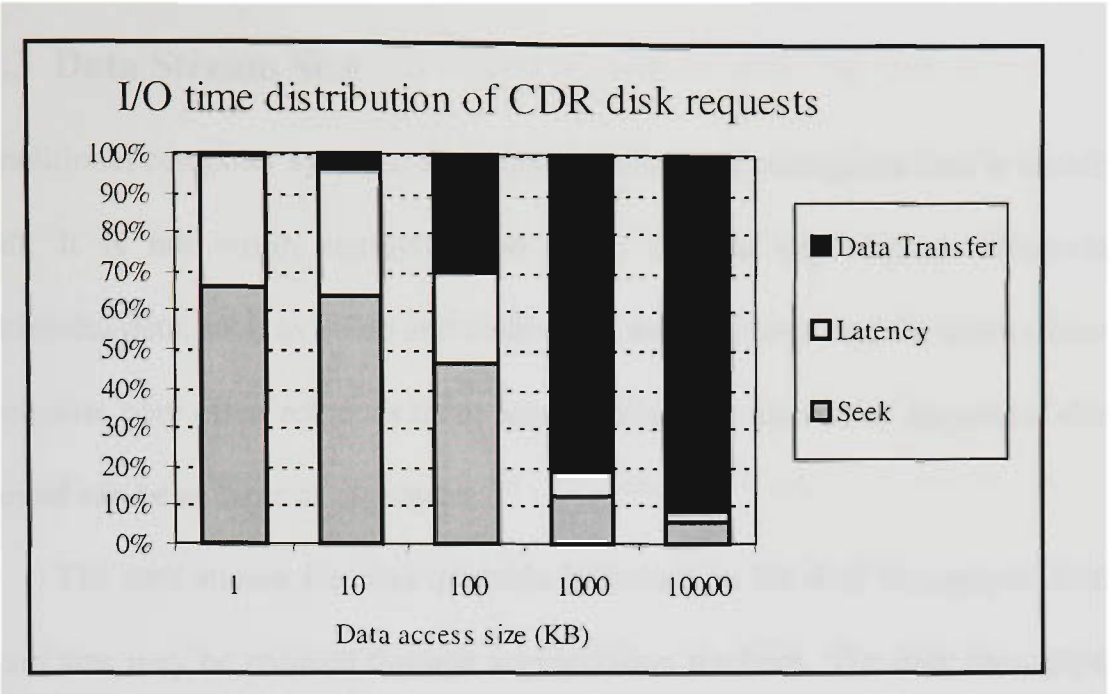


Figure 4.4. I/O time distribution of CDR disk requests

When more data are required, better performance can be achieved by storing consecutive multimedia data in the same cylinder. Some overheads, including rotational latency and head switching time, are still involved, but the seek time is removed. Rotational latency may be eliminated by carefully placing data on the tracks on different platters in the same cylinder. A gap can be used to separate the beginning and end of each track so that the revolution time to pass this gap is longer than the head switching time. In this way, all data on the same cylinder can be retrieved continuously with minimum delay.

4.3.3 Data Stream Size

In traditional computer systems, the required amount of contiguous data is usually small. It is not worth retrieving too much data in one request. However, multimedia data, such as video and audio data, are very large and the same stream of requests very often retrieves them sequentially. The amount of sequential data required can be as large as gigabytes.

The data stream size has dramatic influence on the disk throughput. Data stream size may be reduced through segmentation methods. The disk throughput increases in proportion with the data stream size and saturates slowly to the maximum throughput (Figure 4.5). Throughput of the disks saturates at size of one cylinder of the disks. Hence, the segmentation should only be performed to the size of one cylinder in order to maintain the maximum disk throughput. Although this may vary on disks with different formats, CDR disks always have a larger cylinder size than VDR disks. Hence, larger segments can be used on CDR disks than VDR disks.

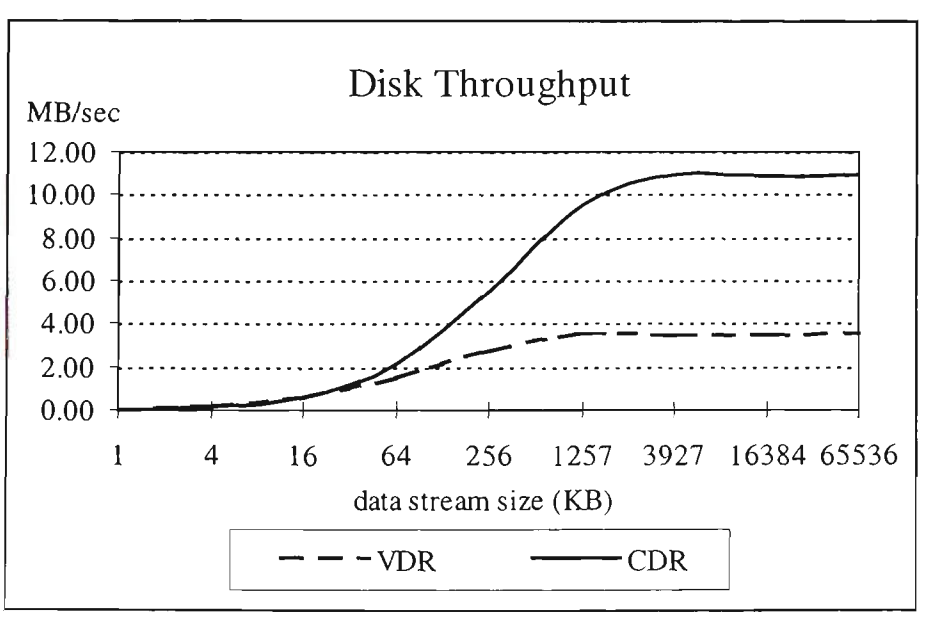


Figure 4.5. Throughput vs data stream size

The throughput of a disk begins to increase when the data stream size increases above 16 KB. A VDR disk can deliver up to 167 MB per minute whereas a similar CDR disk can deliver up to 456 MB per minute. Therefore, the maximum throughput is two times higher by using CDR technology. High resolution images may require more than 2 MB of storage. Thus, for large image databases, CDR disks have considerable performance advantage over VDR disks.

The percentage of reduction in access time varies from 3% to 68% depending on the data stream size (Figure 4.6). For small data stream size and small discrete data accesses, the data access time is reduced by only 3%, mainly due to shorter mean seek distance. For large stream size, a much higher percentage of data access time is reduced mainly due to reduction in data transfer time involved. For very large data stream size, over 80% of time is spent in transferring data when the maximum throughput is achieved. Therefore, CDR disks perform significantly better for multimedia data streams than for traditional data requests.

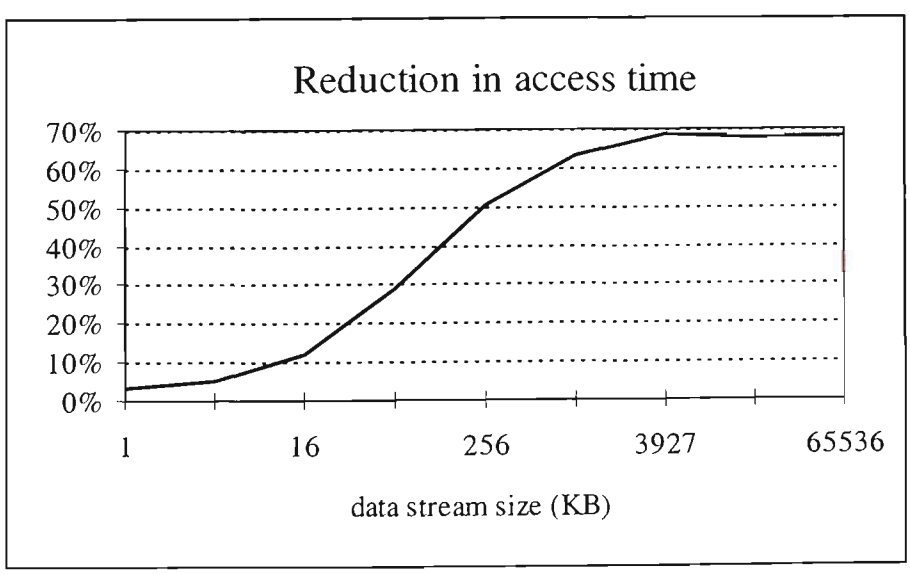


Figure 4.6. Reduction in access time vs data stream size

4.3.4 Disk Size

For VDR disks, the disk capacity, C, is given by $\int_a^b 2\pi ak\,dx$. Hence,

$$C = 2\pi ka(b-a),$$

which implies

$$\frac{dC}{da} = 2\pi k(b-2a). \tag{4.15}$$

When $a = b/2$, $\frac{dC}{da} = 0$, and

$$C = \frac{\pi kb^2}{2}. \tag{4.16}$$

When $a = b/2 \pm \delta$, where δ is small,

$$C = \frac{\pi kb^2}{2} - 2\pi k\delta^2 < \frac{\pi kb^2}{2}. \tag{4.17}$$

Hence, the disk capacity is maximized when the innermost diameter is half of the disk diameter. Therefore, half of the disk size is often chosen as the diameter size of the innermost track in VDR disks.

When the size of VDR disks increases, the mean seek distance, and hence the data access time, also increase (Figure 4.7). When the size of CDR disks increases, the mean seek distance increases proportionately, but the data transfer time decreases. The disk throughput hence rises at small disk size but falls back at larger disk size. Therefore, an optimal disk size exists for CDR disks but not for VDR disks.

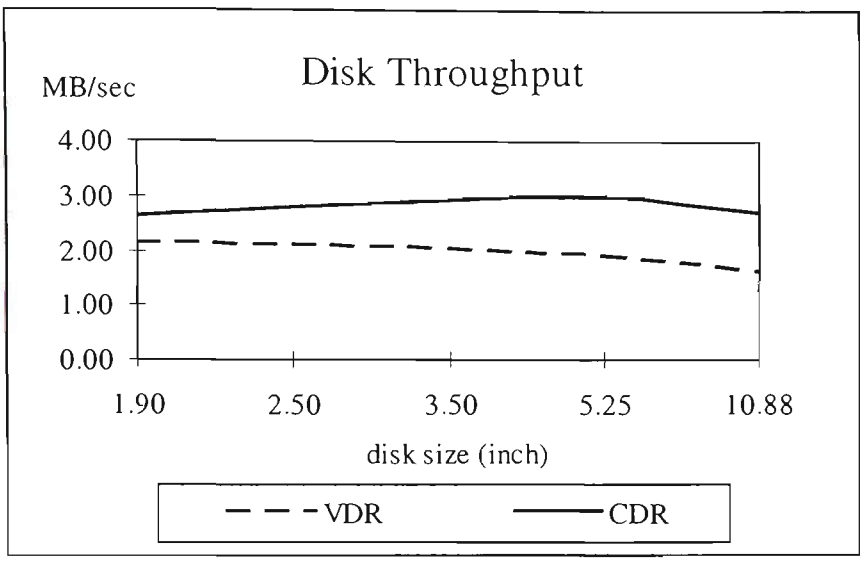


Figure 4.7. Throughput vs disk size

4.3.5 Seek Start/stop Time

The data access time is shown against the seek start/stop time and arm moving speed in Figure 4.8. The seek start/stop time is an overhead in every seek action. Both VDR and CDR disks behave similarly. The seek time increases in proportion to the start/stop time in moving the read/write heads. Hence, the data access time increases linearly as the seek start/stop time.

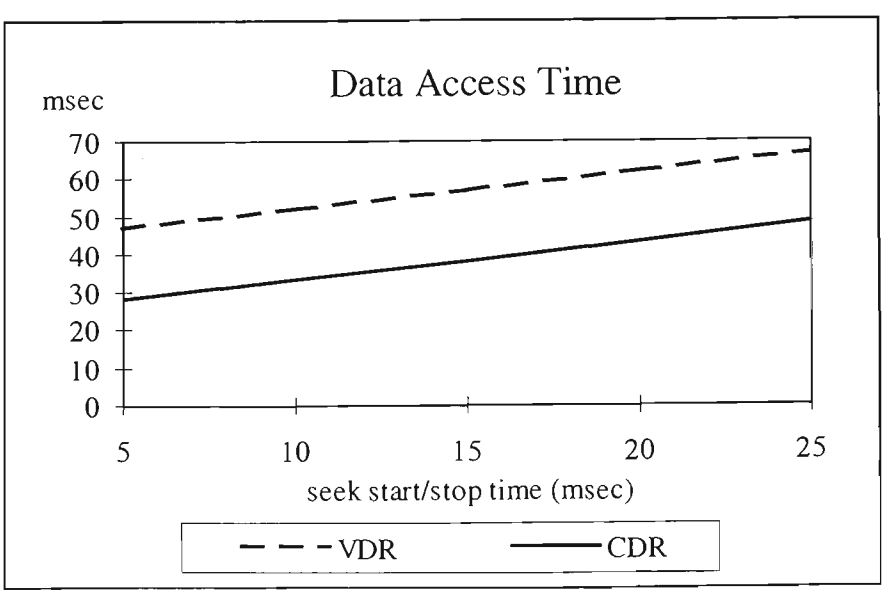


Figure 4.8. Data access time vs seek start/stop Time

4.3.6 Arm Moving Speed

The disk arm moves the read/write heads to the required track of data. As the arm moving speed is increased, the time to reach the required track is reduced (Figure 4.9). Hence, the data access time varies inversely with the arm moving speed. Both VDR disks and CDR disks behave similarly.

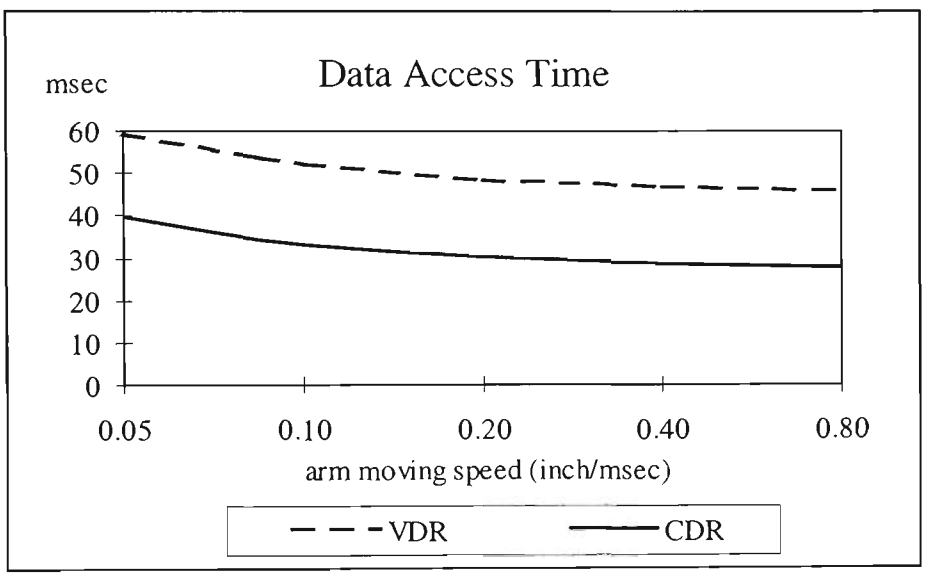


Figure 4.9. Data access time vs arm moving speed

4.3.7 Number of Disk Platters

The number of disk platters determines the size of each cylinder. This applies to both CDR disks and VDR disks. When more data are placed in one cylinder, more data can be retrieved with only one seek action. This does not make any difference when the amount of data being accessed is less than one track. When the amount of data exceeds the size of one track, disk throughput can be increased by accessing more data from different platters (Figure 4.10). Hence, the throughput increases in a stepwise manner. Therefore, more platters allow more data to be stored in each cylinder to avoid extra seek actions in each access.

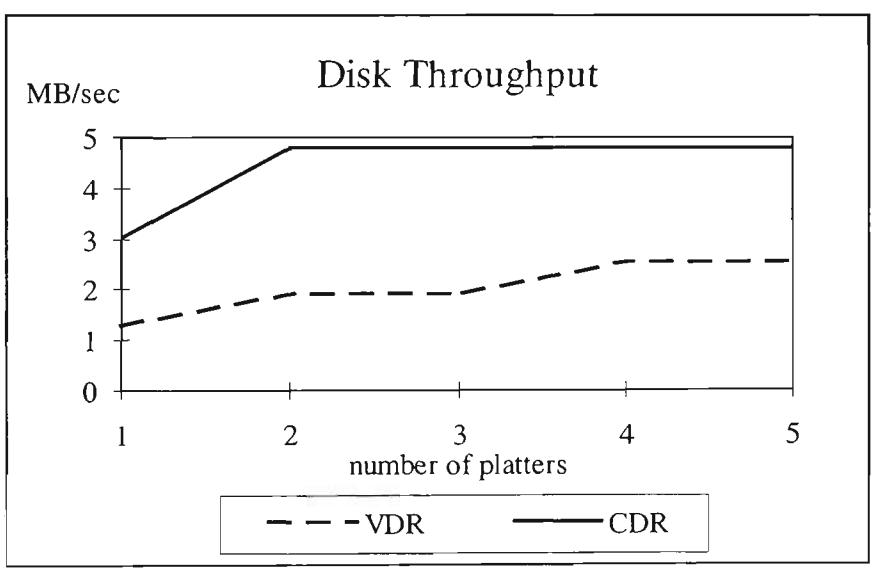


Figure 4.10. Throughput vs number of platters

4.3.8 Rotation Speed

When the disks rotate faster, both the rotational latency and data transfer time are reduced proportionately (Figure 4.11). Since both rotational latency and data transfer time are inversely proportional to the rotation speed, the disk throughput increases linearly. Both VDR disks and CDR disks have similar variations although the throughput of CDR disks is significantly higher.

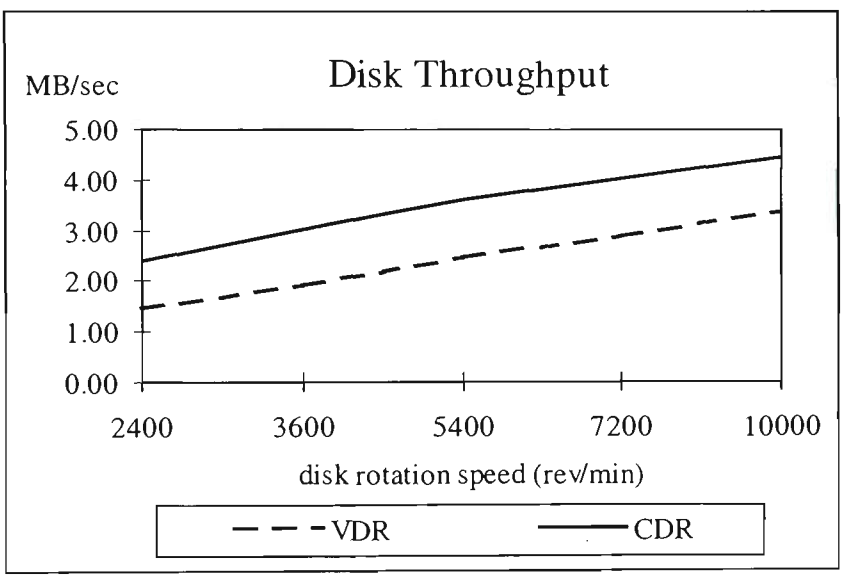


Figure 4.11. Throughput vs rotation speed

4.3.9 Recording Density

Recording density increased at the compound rate of approximately 30% per year in the past. New disks that use Magneto-Resistive heads have increased the recording density substantially and accelerate the rate of increase in recording density. When the recording density of the magnetic disks increases, the data transfer rate and the disk throughput increase proportionately (Figure 4.12).

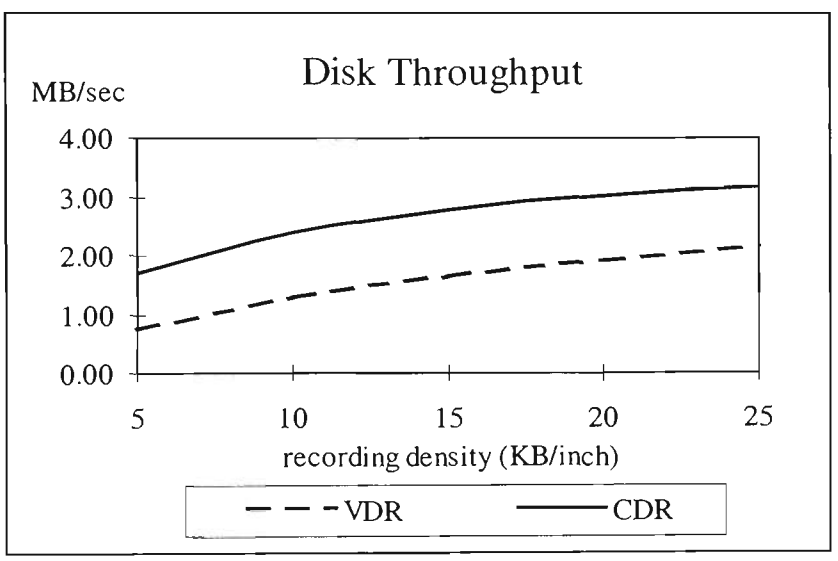


Figure 4.12. Throughput vs recording density

4.3.10 Bandwidth Based Placement

We study a case of partitioning the CDR disks into two zone groups to analyze the benefits of our new bandwidth based placement method. The outer zone group stores multimedia data which are continuous media data such as video and audio. The inner zone group stores traditional data, such as textual data and binary data. The data transfer rate of multimedia data is always higher than the data transfer rate of traditional data.

On the one hand, the multimedia data enjoys a higher throughput on the outer zone group. The large amount of data being retrieved by the same access can support a longer data consumption period. Hence, the throughput of the multimedia data being accessed from the disk access is increased. It may seem that the throughput of multimedia data in the outer zone group is raised at the expense of the throughput of traditional data in the inner zone group. In fact, nothing is lost for the traditional data access, since the access pattern of textual and binary data is normally small and random as in traditional computer systems, and only a few kilobytes of data are often sufficient in each access. From the I/O time distribution in Figure 4.4, we clearly see that most data access time for these access sizes are spent in the overheads. Disk performance is often measured in number of I/Os per second. The textual or binary data still enjoy the same number of I/Os per second and are not worse off.

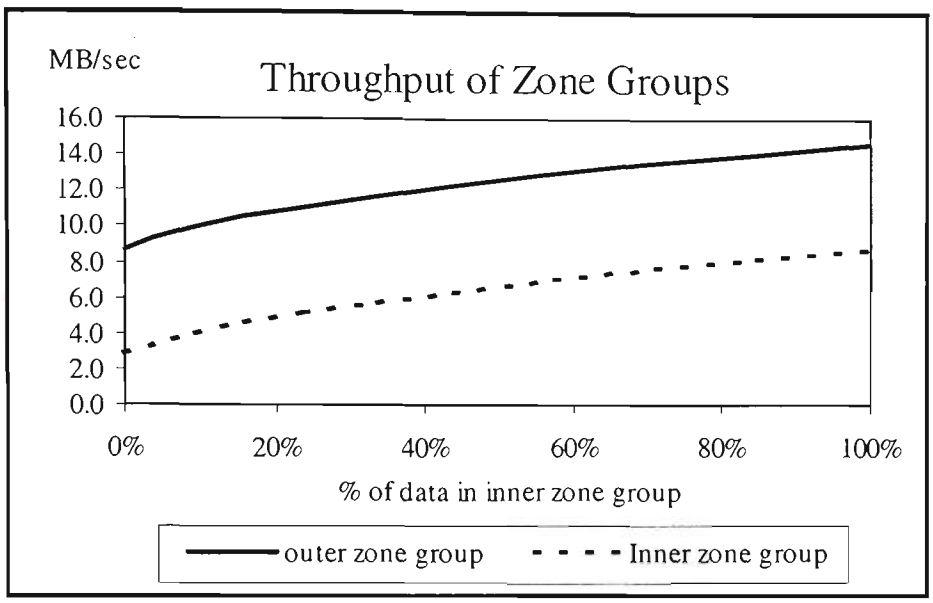


Figure 4.13. Throughput of zone groups

The throughput of the multimedia data in outer zone group and traditional data in inner zone group are found by directly applying the disk performance model (Figure 4.13). The average throughput of the multimedia data ranges from 7 MB/sec to 12 MB/sec. The average throughput of the traditional data ranges from 3 MB/sec to 7 MB/sec. This is particularly suitable for situations when both multimedia data and traditional data are placed in the same CDR disk.

When multiple disks are used, the total storage capacity is large and the aggregate throughput is high. It becomes very useful to partition the disks into several zone groups so that each zone group can have sufficient storage space to contain data that require a similar bandwidth. By choosing the appropriate zone group to store the data, the disk throughput can be adjusted to match the necessary object bandwidth.

4.4 Contention Analysis

The request response time is composed of waiting time in queue and request service time. The waiting time in queue depends on the disk utilization. We first describe the disk utilization in Section 4.4.1. Next, the request waiting time in queue is analyzed in Section 4.4.2. The request response time is then presented in Section 4.4.3. The number of requests waiting in queue is then studied in Section 4.4.4. Lastly, we present the number of outstanding requests in the system in Section 4.4.5.

4.4.1 Disk Utilization

The disk utilization is directly proportional to the request arrival rate (Figure 4.14). The utilization is linked to the probability of finding the system busy [95], and for a given workload, a better performing system is one that gives the lower utilization. When data are accessed from CDR disks, disk utilization is reduced by over 60%. As the disk utilization is reduced, the disks have spare capacity to handle more request streams. The CDR disks become fully utilized until the requests arrive at three times the rate.

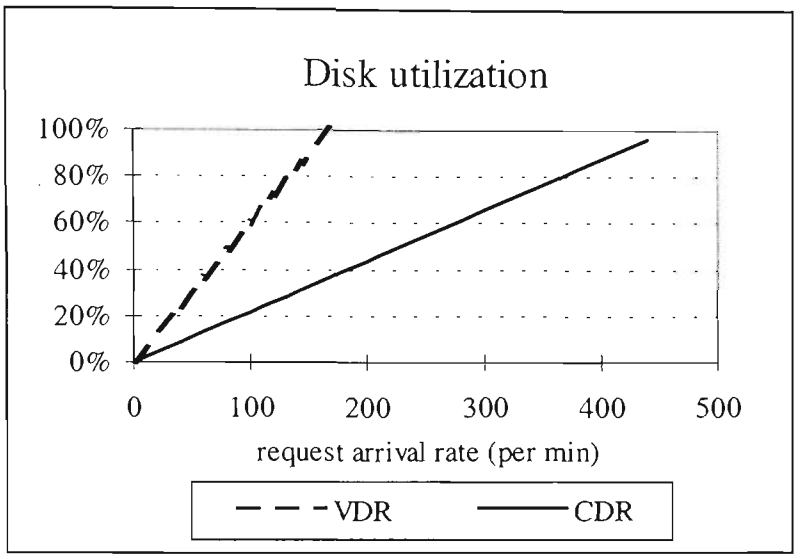


Figure 4.14. Disk utilization vs request arrival rate

In multimedia systems, the number of acceptable streams is controlled by the admission control routines in the operating system to guarantee system performance. The number of acceptable streams should be bounded by the maximum request arrival rate; otherwise the waiting queue is long and requests have a high chance of being discarded. It should also be chosen just below this limit to achieve high utilization and throughput. Hence, CDR disks can accept three times more streams than VDR disks.

4.4.2 Request Waiting Time

Each request to a disk must wait for the existing incomplete I/O request and other requests in the queue. When the request arrival rate is very low, the disk is usually free and there is no waiting required. The waiting time is hence close to zero (Figure 4.15).

When the request arrival rate is between 40 and 120 requests per minute, the queue for VDR disk builds up, and the VDR disk has more time in busy state. The requests for VDR disk spend some time in waiting while the requests for CDR disk can be served almost immediately. The waiting time of requests for CDR disks is less than 20% of that for VDR disks. Hence, the difference in waiting time lies between 80% and 100%.

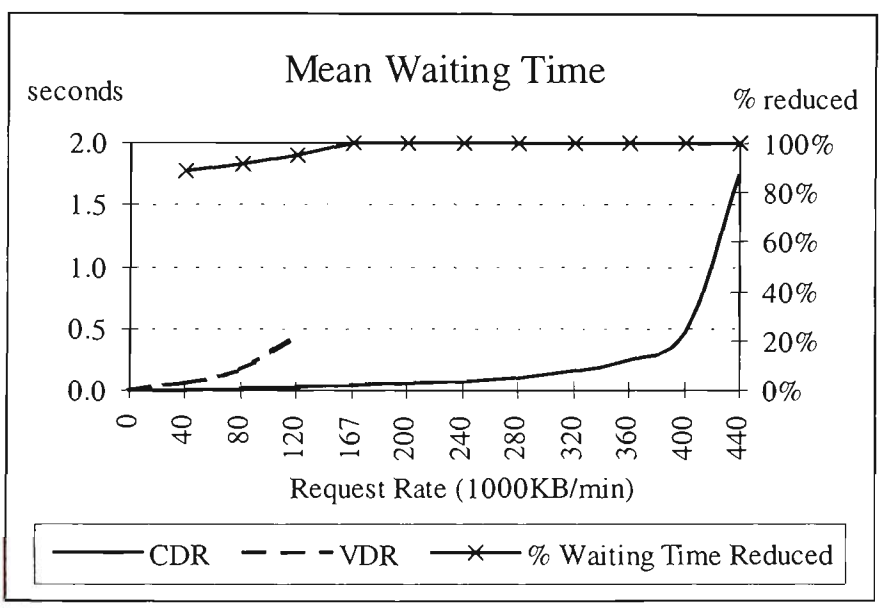


Figure 4.15. Mean waiting time vs request arrival rate

When the request arrival rate increases above 120 requests per minute, the queue for VDR disk builds up quickly, and the VDR disk is seldom free. The waiting time of requests for VDR disks increases sharply to 4000 seconds (not shown) whereas the waiting time of requests for CDR disks is still below 1/5th of a second.

When the request arrival rate increases beyond 167 requests per minute, the VDR disk is always busy. Unless some requests are discarded, the request queue grows indefinitely. The waiting time of the CDR disk remains below one second until it approaches 440 requests per minute. When the request arrival rate increases beyond 440 requests per minute, the waiting queue grows indefinitely in a similar way to the VDR disk at 167 requests per minute.

4.4.3 Request Response Time

Each request to a disk must wait for the existing incomplete I/O request and other requests in the queue. The response time is the sum of queue waiting time and disk service time for a request. When the request arrival rate is very low, both VDR disk and CDR disk respond within a tenth of a second because the disks are normally free and the waiting time is close to zero (Figure 4.16). The response time of requests to CDR disks is 60% faster than similar requests to VDR disks due to the higher throughput of CDR disks.

When the request arrival rate is below 120 requests per minute, the queue for VDR disk builds up, and the disk has more time in busy state. The waiting time for VDR disk increases and it delays the response. Hence, the difference in response time increases from 60% to 80%.

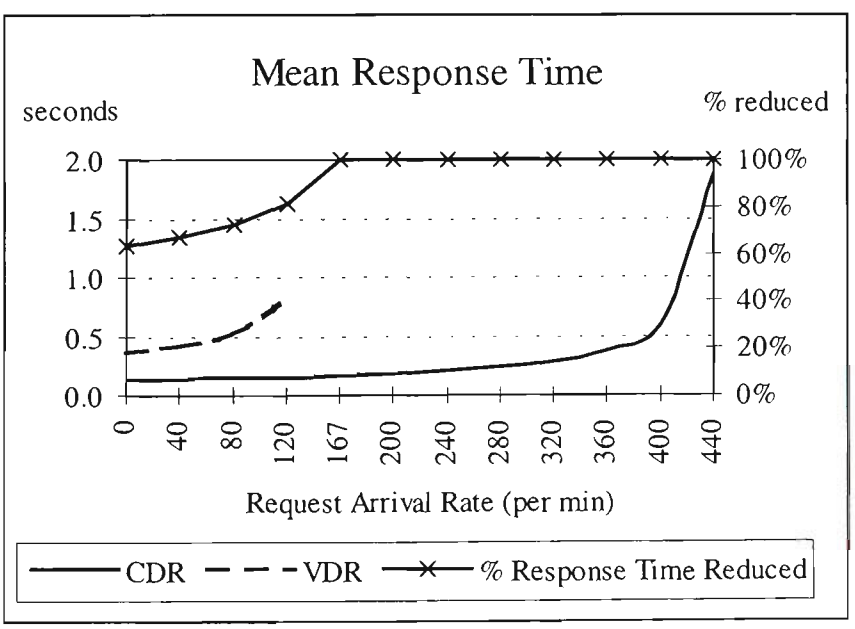


Figure 4.16. Mean response time vs request arrival rate

When the request arrival rate increases beyond 120 requests per minute, the queue for VDR disk builds up quickly, and the VDR disk is seldom free. The waiting time dominates the response time. The response time of VDR disks increases sharply to 4000 seconds (not shown) whereas the CDR disks can still finish requests within 0.17 second. Nearly the whole response time is wasted in waiting for the VDR disk to become free.

When the request arrival rate increases beyond 167 requests per minute, the VDR disk is always busy. Unless some requests are discarded, the requests queue grows indefinitely. The response time of the CDR disk remains below one second until it approaches 440 requests per minute. When the request arrival rate increases beyond 440 requests per minute, the waiting queue of CDR disk also grows indefinitely. We have seen that the mean response time of requests to CDR disks is always lower than that of requests to VDR disks.

4.4.4 Queue Length

In multimedia systems, some requests that cannot be finished within guaranteed time may be discarded. This will cause missing frames in video data or pops in audio data and will adversely affect the quality of service. However, some requests are still allowed to wait for the service of the disk. These requests are placed in a queue and more memory is necessary for longer waiting queues.

When the request arrival rate is high, the mean queue length is long. When the request arrival rate approaches 167 requests per minute, the queue length in VDR disk increases sharply (Figure 4.17). Under similar conditions, the CDR disk maintains a short queue of mean length 0.1 requests. Therefore, smaller memory is required for the waiting queues by using CDR disks.

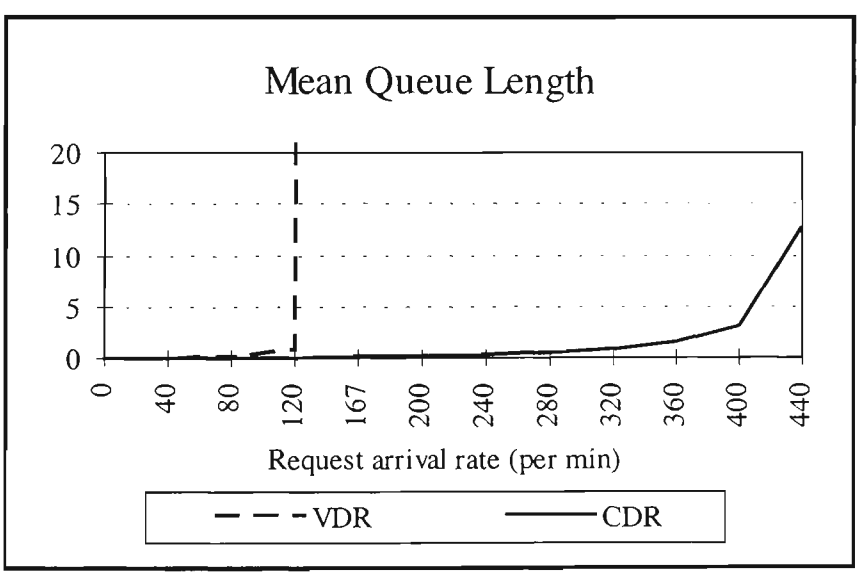


Figure 4.17. Mean queue length vs request arrival rate

4.4.5 Number of Requests

When the request arrival rate is high, more requests are kept in the system. When the request arrival rate approaches 167 requests per minute, the number of requests in VDR disk increases abruptly (Figure 4.18). Under similar conditions, the CDR disk maintains a small number of requests. Therefore, less outstanding requests are present in the system by using CDR disks.

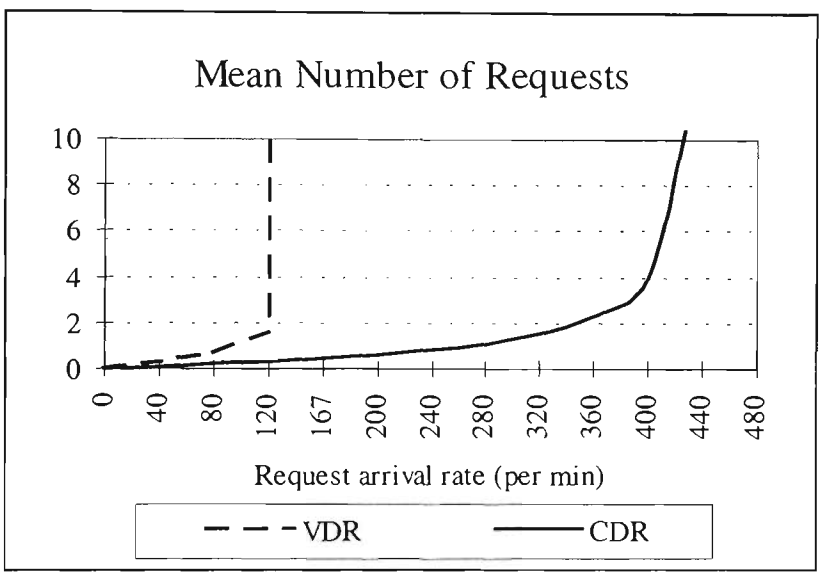


Figure 4.18. Mean number of requests vs request arrival rate

4.5 Disk Storage Structure Simulations

In studying complex systems, it is usually inadequate to use only analytic models, as these may not be able to provide all the numerical measures needed for meaningful performance evaluation. In contention analysis, even single server queues do not admit closed-form solutions under most conditions. Simulation will allow all relevant measures to be obtained numerically by incorporating a model of the system within the simulation program and is a technique used extensively in the study of non-deterministic systems. In addition, simulations can demonstrate the validity of theoretical models, allowing a flexible choice of traffic conditions and parameters. A significant number of simulations have been performed to empirically measure the behaviour of different disk storage structures.

A disk structure simulation program is created which allows the user to simulate the behaviour of either a VDR disk or a CDR disk, using different disk parameters. These disk parameters include the innermost track diameter, disk size, the seek start/stop time, the arm moving speed, the tracks per cylinder, the number of cylinders, the rotation speed, and the disk recording density. The simulation program runs on a Pentium personal computer to generate the multimedia data requests and record their service times. No extra simulation tools are used.

The disk seek time is specified as two functions that depend on whether the seek boundary is exceeded or not. When the seek distance is less than the seek boundary, the seek time uses the short seek function that increases linearly with the square root of seek distance. When the seek boundary is exceeded, the seek time uses the long seek function that increases linearly with the seek distance.

The simulation keeps track of the current positions of the read/write heads, the disk status, and the simulation time. The current head position is composed of the cylinder number, the track number, and the angular position. The disk status can be free, seek, latency, or transfer.

A number of data requests are generated at the specified data access rate. Each request accesses an object of a particular size from a random data position on the disk. When requests are initiated to access data from the disk, they are placed in a waiting queue. The requests are then retrieved from the waiting queue based on a First-Come-First-Serve scheduling policy.

The seek distance is measured on consecutive data requests. The seek time is evaluated as a function of the measured seek distance. The rotational latency is measured against the time being spent in waiting for the required data to come under the heads. The data transfer time is measured against the time necessary for all required data to pass under the read/write heads.

The program also keeps track of the waiting time in queue. The mean waiting time, mean data access time, mean response time, and mean queue length are all measured and stored in a relational database.

4.5.1 Hardware Parameters

We perform simulations on five disks with diameters 10.88 inches, 5.25 inches, 3.50 inches, 2.50 inches and 1.90 inches [119]. These disks rotate at 3600 rpm except the 3.50 inches disk that rotates at 4318 rpm. Large disks have higher recording density except the 3.50 inches disk that has a higher recording density than the 5.25 inches disk. We assume that all of them have an innermost diameter

of one inch, 20 tracks per cylinder, 10 msec seek start/stop time, and 0.1 inch/msec arms moving speed. Since multimedia data are accessed in large blocks, we use a block size of 100 kilobytes unless otherwise indicated.

4.5.2 Data Access Time

We plot the mean data access time being measured in simulations in Figure 4.19. A log scale is used to clearly show the curve due to the large variation in data access times. The predicted and measured values of five different disks are listed in Table A.1 of the Appendix. We have observed that the measured data access times fall exactly on our predicted curves. The simulation results verify that the mean data access times being predicted in our model are sufficiently accurate.

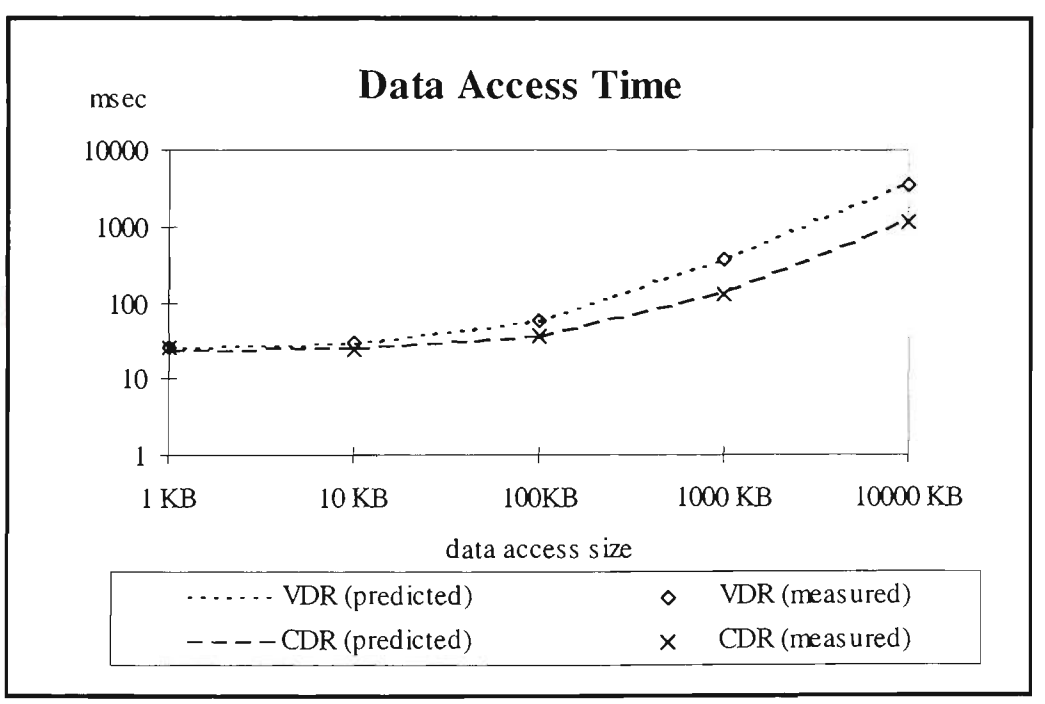


Figure 4.19. Data access time (5.25 inch disk)

4.5.3 Throughput of Partitioned Disk

We have simulated a disk being partitioned into two zone groups. The system throughput of each zone group is measured and plotted in Figure 4.20. The predicted and measured values are listed in Table A.2 of the Appendix. Most of the measurements in the inner zone groups are within the range of 10% of our predictions. All the measurements in the outer zone groups are within 20% below our predicted values. Since the system throughput of the outer zone group is significantly higher than that of the inner zone group, our model is sufficiently accurate in predicting the system throughput of different zone groups in a partitioned disk.

The measured values agree with our model that the inner zone group delivers data at a range of throughput that is lower than the outer zone group, confirming that constant density recording disks can be partitioned into zone groups so that high bandwidth objects on the outer zone group are accessed at a higher throughput than low bandwidth objects on the inner zone group.

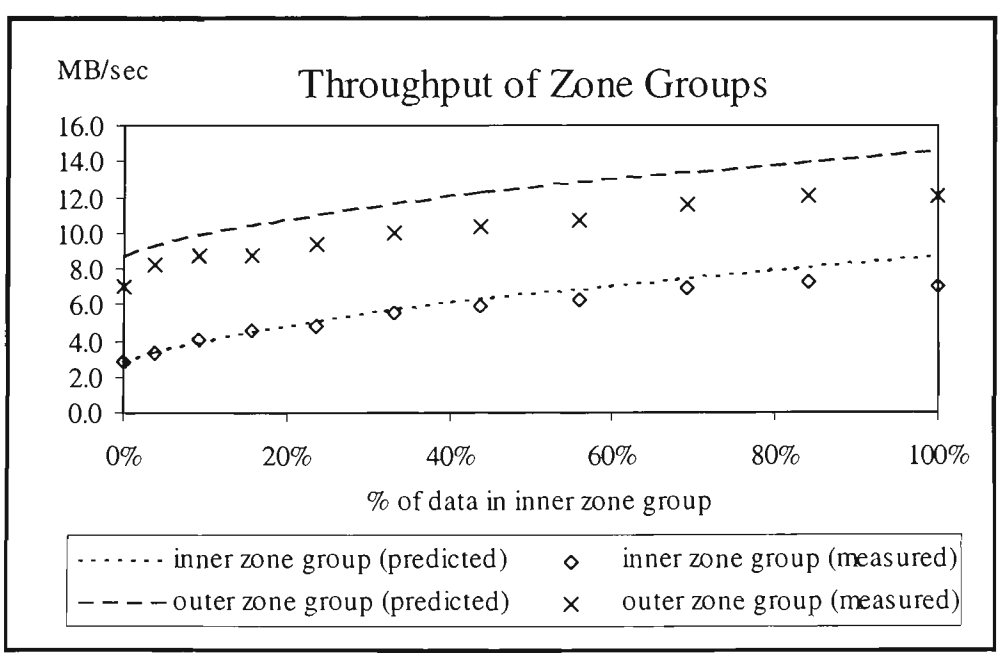


Figure 4.20. Throughput of zone groups

4.5.4 Disk Utilization

We plot in Figure 4.21 the disk utilization being measured. We have found that all the measured values fall on the utilization curves. The simulation results verify that our model is accurate in predicting the utilization of the disks. In addition, the measured disk utilization values also show that utilization of CDR disks is around one-third of the utilization of VDR disks. Thus, the simulation results confirm that CDR disks can accept three times more concurrent streams than VDR disks.

The predicted and measured values are also listed in Table A.3 of the Appendix. Since the queue will tend to be unstable when the utilization approaches 100%, we focus on utilization values up to around 80%. For values approaching 100%, the error tends to diverge as expected.

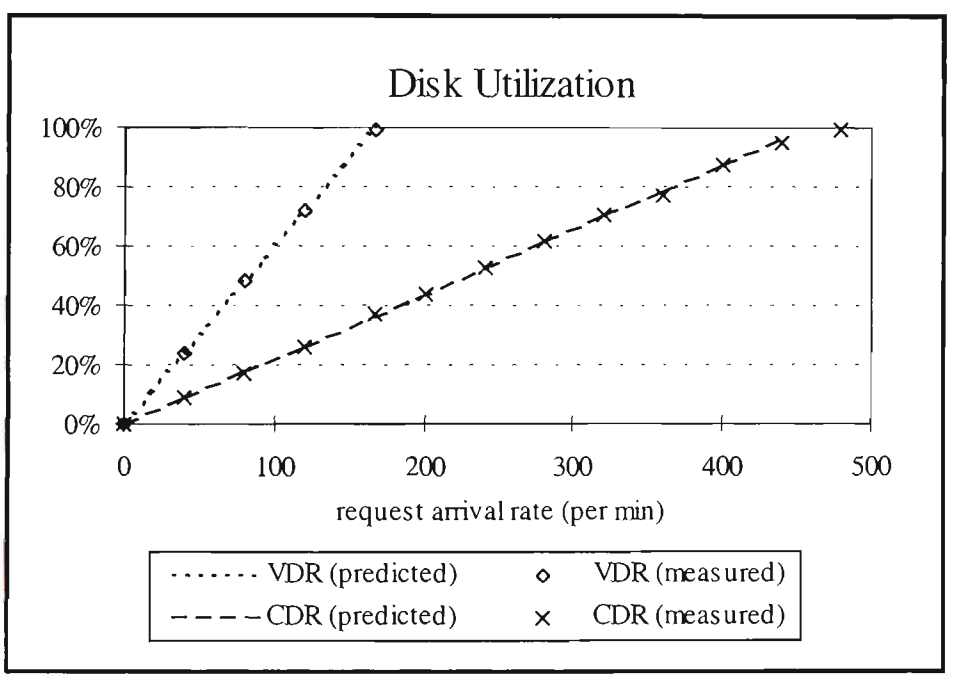


Figure 4.21. Disk utilization

4.5.5 Mean Waiting Time

We have measured the mean waiting times in simulations and plot them in Figure 4.22. We have observed that most measured values fall closely to the predicted curves. The values are also listed in Table A.4 of the Appendix. Since an unstable waiting queue can amplify to a large variation in waiting time when the utilization value is large, the error rate is expected to be high when the system is highly utilized. We focus on stable queues when the utilization values are up to around 80%. The measurements verify that our performance model is reasonably accurate in predicting the mean waiting time. Therefore, the simulation results confirm that the waiting time is significantly reduced when constant density recording disks are used in place of variable density recording disks.

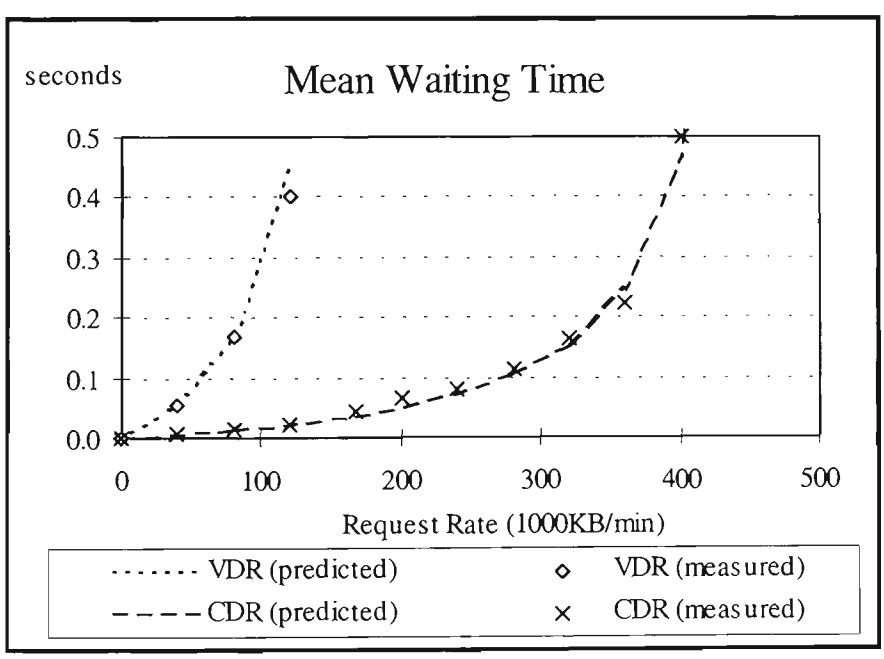


Figure 4.22. Mean waiting time

4.5.6 Mean Response Time

We present in Figure 4.23 the mean response time being measured in simulations. The predicted and measured values are also listed in Table A.5 of the Appendix. We have observed that most of the measured values fall on the predicted curves. Based on the same reason as in the previous Section, we only consider stable queues when the utilization values are up to around 80%. The measurements verify that our model is reasonably accurate in predicting the request response time in the same way as the request waiting time. Thus, the simulation confirms that the mean response time is also significantly reduced when constant density recording disks are used in place of variable density recording disks.

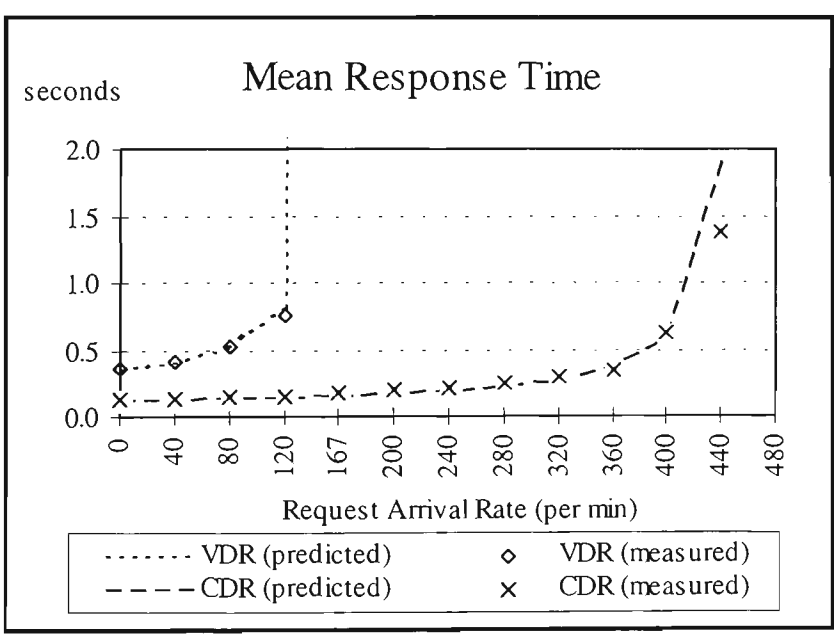


Figure 4.23. Mean response time

4.5.7 Mean Queue Length

We have measured the mean queue lengths in simulations and plot them in Figure 4.24. The values are also listed in Table A.6 of the Appendix. We have observed that most measured values below the 80% utilization fall closely on the prediction curves as in the two previous Sections. Thus, the simulations verify that our performance model is also accurate in predicting the mean queue length.

Since the disk utilization of constant density recording disks reaches 100% at a much faster arrival rate than that of variable density recording disks, the queue starts to grow at much faster arrival rates. This confirms that waiting queues are shorter when constant density recording disks are used in place of the variable density recording disks.

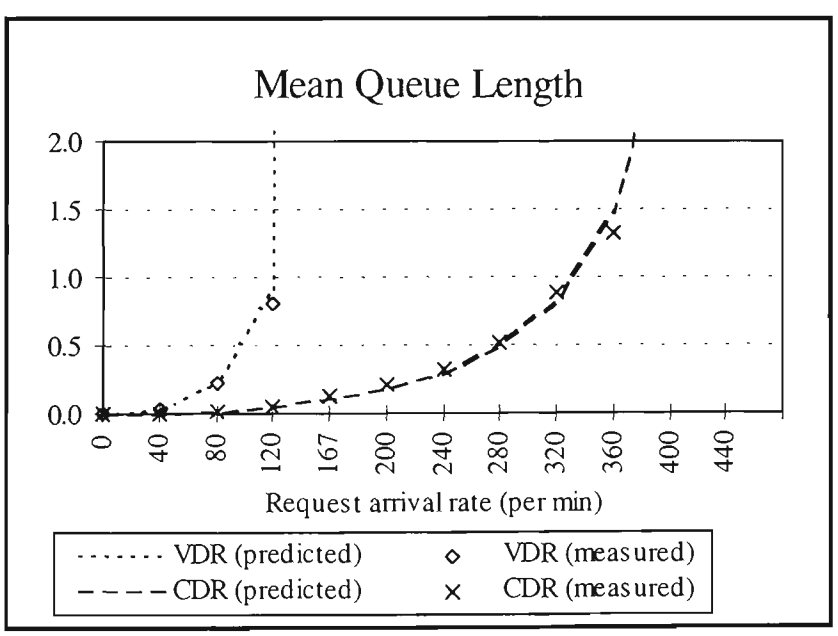


Figure 4.24. Mean queue length

4.5.8 Mean Number of Requests

We have plotted in Figure 4.25 the mean number of requests being measured during simulations. The predicted and measured values are also listed in Table A.7 of the Appendix. We have observed that the measurements fall closely to the curve. Hence, the simulation results have also verified that the mean number of requests being predicted in our model is reasonably accurate.

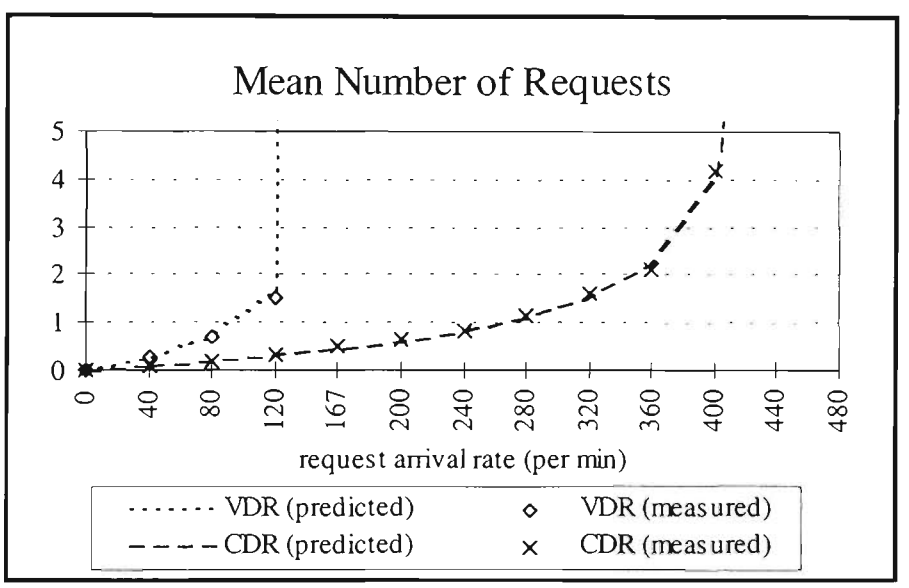


Figure 4.25. Mean number of requests

4.6 Chapter Summary

We have presented a mathematical model of the performance of constant density recording disks. In this model, we have obtained closed form solutions to data access time. This model is used to compare quantitatively the performance of constant density recording disks with that of traditional variable density recording disks.

We have also analyzed the impact of different disk parameters on request service time. The minimum block size that can deliver necessary throughput should be chosen to reduce memory buffer and start up latency. Data stream size can be reduced through segmentation methods to an extent that high disk throughput is still maintained.

We have also analyzed the data transfer time of our new bandwidth based placement method. We then use this model to study the impact on system throughput of various zone groups. We have shown that our new method can increase throughput for multimedia data requests while maintaining similar I/O rate for binary and textual data requests.

For individual data access, CDR disks reduce the data access time by over 60%. For streams of continuous media requests, CDR disks can significantly increase the number of acceptable streams, reduce request response time, and reduce queue length. These results indicate that a storage structure based on CDR significantly outperforms that based on traditional VDR for both multimedia data and traditional data.

The simulation results verify that the mean data access times and the disk utilization being predicted in our model are accurate. This confirms that the bandwidth based placement method performs better than the random placement method for multimedia data requests on constant density recording disks. The simulation results also verify that our predicted waiting time, response time, queue length, and number of requests are sufficiently accurate. These results confirm that performance is significantly enhanced when constant density recording disks are used in place of variable density recording disks.

Chapter 5

Analysis of Hierarchical Storage Organizations

5.1 Introduction

In order to analyse performance of storage systems, mathematical models are used. We have developed a new model to quantify the performance of multimedia data streams on the tertiary storage system.

We first present the feasibility of accepting homogeneous and heterogeneous streams to a storage system while maintaining the continuous display requirement. We prove these feasibility conditions in Section 5.2.

A performance model for processing streams of homogeneous multimedia streams on tertiary storage systems is developed. This model can predict the performance of multimedia data streams in tertiary storage systems. In addition, it allows the comparison and assessment of efficiencies of different storage organizations. This new Queueing model that contains feedback requests and a nested queue is presented in Section 5.3.

When the tertiary bandwidth is low, the segment based pipelining should be used to minimize latency in retrieving objects from tertiary storage. The latency that respond to user interactive functions depends on the size of logical segments. The logical segment size and data stripe size should also be kept within constraints to meet the continuous display requirement of multimedia streams. We present the mathematical model to find these bounds on the latency and segment size in Section 5.4.

We use the performance model to investigate performance of concurrent streams of requests on the tertiary storage system and to compare the performance of our high concurrency placement method, the non-striping method, and the parallel striping method in Section 5.5.

We then analyse the user latency of requests that migrate data from low end tertiary drives in Section 5.6. The latency of our segment based pipelining is compared with that of other data migration methods. The latency is also analysed with respect to changes in tertiary and disk bandwidth. Since the user latency model is rather straight forward, we consider that simulations are not required and can be skipped.

Simulations allow relevant measures to be obtained numerically by incorporating a model of the system within the simulation program and is a technique used extensively in the study of non-deterministic systems. A number of simulations have been performed to validate our theoretical models, and the behaviour of large multimedia storage servers in supporting concurrent streams is empirically measured. The simulation results are presented in Section 5.7.

5.2 Feasibility of Accepting Concurrent Streams

Rangan and Vin investigated feasibility to interleave data blocks of multimedia streams using the Storage Pattern Altering (SPA) policy [122]. They consider only fixed transfer rate over both the media and gap blocks to interleave data in optical disks.

Since magnetic disks and tapes use a format different from optical disks, their performances are different as well. Data on magnetic disks are stored in tracks. Disk heads move across the tracks at a speed that varies according to the seek distance. Gap blocks on tapes are skipped over at higher than reading speed, typically six to ten times faster. Hence, data on storage devices are skipped over at a different rate from data reading or writing.

This is a generalization of their interleaving placement method. We switch from interleaving heterogeneous streams over the space domain to interleaving them over the temporal domain. This allows for the feasibility condition to be used on general storage organizations and arbitrary scheduling methods. We generalize from fixed transfer rate to variable transfer rate over the gaps so that performance characteristics of general storage devices can be included. The parameters in Table 5.1 are used in this model.

Table 5.1. Notations in feasibility conditions

Parameter	Meaning
S	access overheads
M	transfer time
G	gap time
δ	display time

We assume that each stream seeks with an overhead of S seconds and transfers a data stripe using M seconds. After that, the stream suspends data retrieval for G seconds. Each data stripe can display for δ seconds. A multimedia stream (M, δ) is acceptable if and only if it satisfies the continuous display requirement.

$$S + M \leq \delta. \tag{5.1}$$

This continuous display requirement must be maintained over a finite period of time. The requirement can temporarily be violated by satisfying requests in advance and keeping the retrieved data in read-ahead buffers. The average ratio of transfer time to display time must however be maintained over a finite period of time.

5.2.1 Homogeneous Streams

Multimedia streams are considered as homogeneous if all streams have the same display time period δ. Let n streams be characterized by (M_1, δ), (M_2, δ), to (M_n, δ). Let S_i be the access overhead time in serving each stream and G_i be the time gap of the ith stream, for $i = 1$ to n. By definition of the time gap, we have

$$S_i + M_i + G_i \leq \delta. \tag{5.2}$$

Corollary 1: n streams can be concurrently displayed if and only if

$$S_1 + M_1 + S_2 + M_2 + \ldots + S_n + M_n \leq \delta. \tag{5.3}$$

Proof: In order to be able to accept n concurrent streams, request of any stream is served during the time gap of other streams. The continuous display requirement necessitates that k requests are served within a continuous time period $k\delta$ for finite value of k.

If n streams are concurrent, then n streams are served in turn over a finite time period $k\delta$ such that k requests of each streams are served within the time gap of other stream. Hence, we have

$$\sum_{j=1, j\neq i}^{n} k\left(S_j + M_j\right) \leq kG_i, \qquad i = 1, 2, \ldots, n,$$

which implies

$$\sum_{j=1, j\neq i}^{n} \left(S_j + M_j\right) \leq G_i, \qquad i = 1, 2, \ldots, n. \tag{5.4}$$

Summing Equation (5.4) for all streams, we have

$$\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \left(S_j + M_j\right) \leq \sum_{i=1}^{n} G_i ,$$

which implies

$$(n-1)\sum_{j=1}^{n} \left(S_j + M_j\right) \leq \sum_{i=1}^{n} G_i .$$

Substituting G_i from Equation (5.2), this becomes

$$(n-1)\sum_{j=1}^{n} \left(S_j + M_j\right) \leq \sum_{i=1}^{n} \left(\delta - S_i - M_i\right),$$

which implies

$$(n-1)\sum_{j=1}^{n} \left(S_j + M_j\right) \leq n\delta - \sum_{i=1}^{n} \left(S_i + M_i\right). \tag{5.5}$$

Changing the subscript of the left hand side in Equation (5.5) from j to i, we have

$$(n-1)\sum_{i=1}^{n} \left(S_i + M_i\right) \leq n\delta - \sum_{i=1}^{n} \left(S_i + M_i\right),$$

which implies

$$n\sum_{i=1}^{n} \left(S_i + M_i\right) \leq n\delta .$$

Therefore, we have

$$S_1 + M_1 + S_2 + M_2 + \ldots + S_n + M_n \leq \delta,$$

which is the necessary condition. Conversely, we have

$$S_1 + M_1 + S_2 + M_2 + \ldots + S_n + M_n \leq \delta.$$

Since all terms are positive, we have

$$S_i + M_i \leq \delta, \qquad i = 1, 2, \ldots, n.$$

Hence, the continuous display requirement of all streams is fulfilled. Therefore, the n streams can be concurrent. ■

5.2.2 Heterogeneous Streams

Multimedia streams are considered as heterogeneous when their cycle periods are different. Let n streams be characterized by (M_1, δ_1), (M_2, δ_2), to (M_n, δ_n) such that not all δ_i are the same. Let S_1 to S_n be the access overhead time in serving each stream.

Corollary 2: n streams can be concurrent if and only if

$$\frac{S_1 + M_1}{\delta_1} + \frac{S_2 + M_2}{\delta_2} + \ldots + \frac{S_n + M_n}{\delta_n} \leq 1. \qquad (5.6)$$

Proof: If n streams are concurrent, then there exists a finite time period δ that k_j requests of the jth streams are served. By the continuous display requirement, this time period does not exceed the display time of each stream. We have

$$\delta \leq k_j \delta_j, \qquad j = 1, 2, \ldots, n,$$

which implies

$$\frac{1}{\delta_j} \le \frac{k_j}{\delta}, \qquad j = 1, 2, \ldots, n. \tag{5.7}$$

Since the time period δ is the retrieval time of all requests, we have,

$$\sum_{j=1}^{n} k_j \left(S_j + M_j\right) = \delta,$$

which implies

$$\sum_{j=1}^{n} \frac{k_j \left(S_j + M_j\right)}{\delta} = 1. \tag{5.8}$$

Substituting $\frac{1}{\delta_j} \le \frac{k_j}{\delta}$, we obtain

$$\sum_{j=1}^{n} \frac{\left(S_j + M_j\right)}{\delta_j} \le 1,$$

which is the necessary condition. Conversely, we let

$$\delta = \delta_1 \, \delta_2 \ldots \delta_n$$

and let $k_j \in R$ such that

$$k_j = \frac{\delta}{\delta_j}, \qquad j = 1, 2, \ldots, n,$$

which gives

$$\frac{k_j}{\delta} = \frac{1}{\delta_j}, \qquad j = 1, 2, \ldots, n. \tag{5.9}$$

From the necessary condition, we have

$$\frac{S_1 + M_1}{\delta_1} + \frac{S_2 + M_2}{\delta_2} + \ldots + \frac{S_n + M_n}{\delta_n} \le 1.$$

This implies

$$\sum_{j=1}^{n} \frac{\left(S_j + M_j\right)}{\delta_j} \le 1.$$

Substituting $\frac{1}{\delta_j}$ from Equation (5.9), we obtain

$$\sum_{j=1}^{n} \frac{k_j\left(S_j + M_j\right)}{\delta} \leq 1,$$

which implies

$$\sum_{j=1}^{n} k_j\left(S_j + M_j\right) \leq \delta.$$

Hence, we obtain

$$k_i\left(S_i + M_i\right) + \sum_{j=1, j\neq i}^{n} k_j\left(S_j + M_j\right) \leq k_i\delta_i, \quad i = 1, 2, \ldots, n. \tag{5.10}$$

Since all terms are positive, we can take away the term $\sum_{j=1, j\neq i}^{n} k_j\left(S_j + M_j\right)$ from the left hand side of the inequality. Hence, we have

$$k_i\left(S_i + M_i\right) \leq k_i\delta_i, \qquad i = 1, 2, \ldots, n,$$

which implies

$$S_i + M_i \leq \delta_i, \qquad i = 1, 2, \ldots, n.$$

The continuous display requirement of each stream is fulfilled over a finite period of time. Hence, the n streams can be concurrent. ■

5.2.3 Multiple Devices

When multiple devices are available, the devices may serve the streams independently or in parallel. When the streams are served in parallel, the above inequality for a single drive with different access overheads and transfer rate can be used. When the streams are served independently, one request is served by one device each time. We assume that the requests can be distributed evenly to p devices, otherwise some devices can be overloaded while others are under utilized.

Corollary 3: n streams can be concurrent on p independent devices if and only if

$$\frac{S_1 + M_1}{\delta_1} + \frac{S_2 + M_2}{\delta_2} + \ldots + \frac{S_n + M_n}{\delta_n} \leq p . \tag{5.11}$$

Proof: If n streams are concurrently served by p devices, then there exist a finite time period δ such that k_j requests of the jth streams are served by p devices. By the continuous display requirement, this time period should not exceed the display time of each stream. We have

$$\delta \leq k_j \delta_j, \qquad j = 1, 2, \ldots, n,$$

which implies

$$\frac{1}{\delta_j} \leq \frac{k_j}{\delta}, \qquad j = 1, 2, \ldots, n. \tag{5.12}$$

Since the total retrieval time of all requests must be less than the service time of the p devices over the time period δ, we have,

$$\sum_{i=1}^{n} k_j \left(S_j + M_j \right) \leq p\delta,$$

which implies

$$\sum_{j=1}^{n} \frac{k_j\left(S_j + M_j\right)}{\delta} \le p. \tag{5.13}$$

Substituting $\frac{1}{\delta_j} \le \frac{k_j}{\delta}$ from Equation (5.12), we obtain

$$\sum_{j=1}^{n} \frac{\left(S_j + M_j\right)}{\delta_j} \le p.$$

Hence, the necessary part is proved. Conversely, we let

$$\delta = \delta_1 \delta_2 \ldots \delta_n$$

and let $k_j \in R$ such that

$$k_j = \frac{\delta}{\delta_j}, \qquad j = 1, 2, \ldots, n,$$

which implies

$$\frac{k_j}{\delta} = \frac{1}{\delta_j}, \qquad j = 1, 2, \ldots, n. \tag{5.14}$$

Substituting $\frac{1}{\delta_j}$ from Equation (5.14) to the necessity condition, we have

$$\sum_{j=1}^{n} \frac{k_j\left(S_j + M_j\right)}{\delta} \le p,$$

which implies

$$\sum_{j=1}^{n} k_j\left(S_j + M_j\right) \le p\delta. \tag{5.15}$$

Since all terms are positive, we can take away all except the ith term from $\sum_{j=1}^{n} k_j\left(S_j + M_j\right)$. Hence, we obtain

$$k_i\left(S_i + M_i\right) \le p k_i \delta_i, \qquad i = 1, 2, \ldots, n,$$

which implies

$$\left(S_i + M_i\right) \leq p\delta_i, \qquad i = 1, 2, \ldots, n. \tag{5.16}$$

That is, requests of the ith stream can be served within time period δ_i by p devices. As long as the requests are distributed evenly to the devices, the continuous display requirements of all streams are fulfilled. Therefore, the n streams can be accepted to be served concurrently. ■

5.2.4 Application on Storage Devices

The access overhead time, S, is a function that depends on the storage devices, the storage organizations, and the presence of other concurrent streams. We shall consider the following three typical types of overhead functions:

1. fixed value function,
2. linear value function, and
3. monotonically increasing function.

These three types of overhead time functions and their combinations would cover most data access overheads of storage devices.

(a) Fixed Overhead Function

When media units are exchanged, the exchange time is considered as uniformly distributed over a fixed value. Hence, the fixed value function can be used. In effect, the overhead is considered as

$$S = \overline{S}. \tag{5.17}$$

where $\overline{S}$ is the average overhead time.

(b) Linear Overhead Function

When data on magnetic tapes are skipped over, the time to skip over data is proportional to the amount of data being skipped over. Hence, the overhead time is linearly related with the distance being travelled. Therefore, we have

$$S(d) = a_1 + a_2 d, \tag{5.18}$$

where a_1, a_2 are fixed parameters and d is the travelled distance.

We can see that the fixed value function is, in fact, a particular case of the linear overhead function with $a_2 = 0$. The CLV disk format is also a special case of the linear overhead function with $a_1 = 0$ and a_2 = data transfer rate.

(c) Monotonically Increasing Overhead Function

When data on magnetic disks are accessed, the seek time function, as explained in Chapter 2, is characterized as

$$S(d) = \begin{cases} a_1 + a_2 d, & d > a_5 \\ a_3 + a_4\sqrt{d}, & d \le a_5 \end{cases}, \tag{5.19}$$

where d is the seek distance, and a_1, a_2, a_3, a_4, a_5 are fixed parameters. We can easily see that the linear overhead function is a particular case of the monotonically increasing overhead function with $a_5 = 0$.

Based on the motion dynamics of the disk heads, the time to travel a distance in one direction increases monotonically with the length of the distance. According to Equation (5.19), the seek time function increases with seek distance and the parameters being used in [126] for current disks, it is plausible to say that disk seek time is a monotonically increasing function of the seek distance.

From the characteristics of magnetic disks, the seek time for a long seek cannot exceed the sum of seek time of its component seeks. The sum of seek time for two seek actions is maximized when the two seek actions are equal in distance [59]. That is

$$S(d_1+d_2) \le S(d_1) + S(d_2), \tag{5.20}$$

and

$$S(d_1) + S(d_2) \le 2*S(\frac{d_1 + d_2}{2}). \tag{5.21}$$

We can easily generalize the above result for n requests to get

$$S(d_1+d_2+\ldots+d_n) \leq S(d_1) + S(d_2) +\ldots+ S(d_n) \tag{5.22}$$

and

$$S(d_1) + S(d_2) +\ldots+ S(d_n) \leq n*S(\frac{d_1 + d_2 +\ldots+d_n}{n}). \tag{5.23}$$

That is, the seek time for a long seek cannot exceed the sum of seek time of all its component seeks. The sum of seek time for n seek actions is maximized when all n seek actions are equal in distance.

5.2.5 Optimization of Probability Model

Since the throughput performance of a storage organization depends on the concurrent streams being accepted to the system, the system throughput is optimal by accepting the most concurrent streams.

Let Prob(M_1, M_2, ... , M_n) be the probability that M_1, M_2, ... , M_n are concurrent. An efficient method should maximize the probability that the feasibility condition,

$$\frac{S_1 + M_1}{\delta_1} + \frac{S_2 + M_2}{\delta_2} +\ldots+ \frac{S_n + M_n}{\delta_n} \leq 1,$$

can be achieved. That is to say, an efficient storage and retrieval method should reduce

$$\sum \Pr(M_1, M_2,\ldots, M_n)\left(\frac{S_1 + M_1}{\delta_1} + \frac{S_2 + M_2}{\delta_2} +\ldots+ \frac{S_n + M_n}{\delta_n} \right) \tag{5.24}$$

in order to accept the most concurrent streams.

5.3 Performance Model of Large Multimedia Storage Systems

In this Section, we develop the performance model of large multimedia storage systems (LMSS) utilizing various tertiary storage organizations, including high concurrency placement method, the non-striping method, and the parallel striping method. The non-striping method (using contiguous placement) and the parallel striping method are described in part (a) and part (d) of Section 2.6.2 respectively. The assumptions and notations of the model are first described in Section 5.3.1. Next, we show the arrival pattern of streams and the generation of requests in Section 5.3.2. We present the transition states of concurrent streams in Section 5.3.3. The service times of high concurrency placement method are given in Section 5.3.4. In Section 5.3.5 the system throughput is modelled, and we determine the expected stream response time in Section 5.3.6. Lastly, the necessary size of the staging buffer is found in Section 5.3.7.

5.3.1 Assumptions and Notations

We use a queueing model to analyse the service time and the queue contention of the tertiary storage system. We assume a conservative system in which no requests are created or destroyed. We also assume the non-preemptive scheduling condition in which each requests must finish before another request can be served. The notations in Table 5.2 are adopted in this model.

Table 5.2. Notations in LMSS performance model

Parameter	Meaning
λ	arrival rate of new streams
μ	stream service rate
μ_j	stream departure rate at j streams
ρ	utilization
r	number of requests per stream
D	number of tertiary drives
I	combined input rate
I_2	input rate to the exchange queue
A	inter-stream arrival time
J	inter-request arrival time
R	request response time
X	request service time
B	concurrent group service time
S	number of concurrent streams
$\hat{S}$	maximum number of concurrent streams
ω	media exchange time
w	exchange waiting time
χ	exchange service time
α	total reposition time in a group
α_S	reposition time of a request in a group
Y	total reposition distance in a group
L	length of media unit
τ	transfer time of each request
T	total transfer time in a batch
Z	segment size
ϕ	data transfer rate
K	number of media units per drive
δ	display bandwidth

5.3.2 Arrival Pattern

New streams arrive randomly at the system. We shall describe the stream arrival pattern in part (a). Since these streams may generate the requests at different times, the expected inter-request arrival time is explained in part (b). We then describe the variance of inter-request arrival time under three different conditions. The variance of inter-request arrival time when requests arrive independently is presented in part (c). The variance of inter-request arrival time when all requests

are generated on arrival is described in part (d). The variance of inter-request arrival time for our concurrent streaming method in which two requests are generated on arrival is described in part (e).

(a) Stream Arrival Pattern

We assumed that new streams arrive randomly to the tertiary storage system. The stream arrival pattern can suitably be described with the Poisson distribution at a fixed rate λ (Figure 5.1).

When the arrival pattern of the streams is Poisson distributed with a fixed rate λ, the mean time between stream arrivals is $1/\lambda$ and the variance of time between stream arrivals is $1/\lambda^2$.

(b) Inter-Request Arrival Time

In general, every new stream sends r requests to the tertiary storage system. These r requests are distributed to D drives and each drive receives on average r/D requests from each stream. After all the requests of a stream are served, the finished stream exits the system (Figure 5.2).

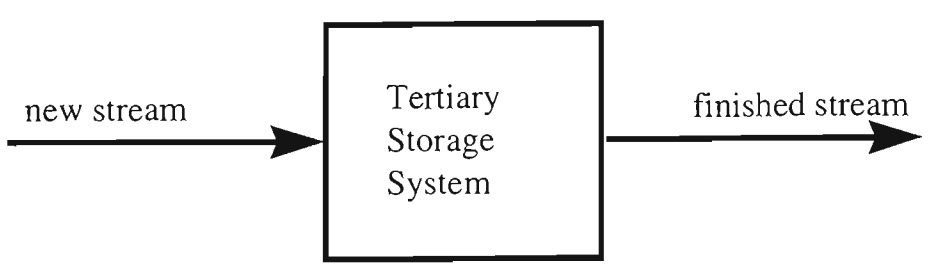


Figure 5.1. Model of streams

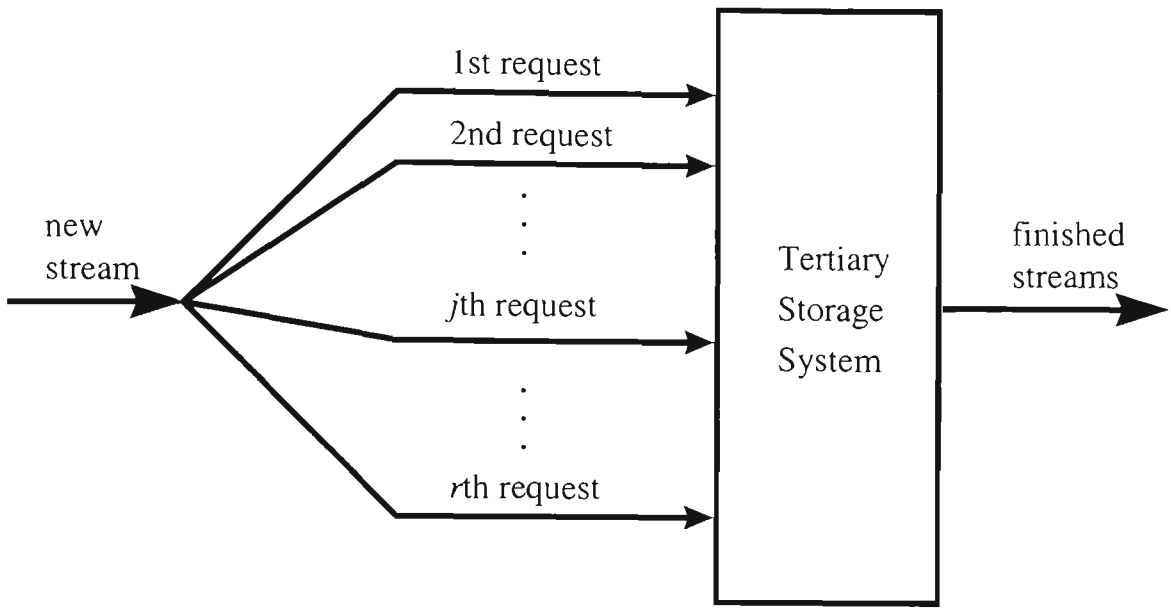


Figure 5.2. Model of request arrival

All these requests are served by the tertiary storage system. The arriving pattern of the requests would be the same as merging the arriving requests. Since the arrival rate of all requests is equal to the arrival rate of the streams multiplied by the number of requests per stream, the arrival rate of all requests to the tertiary storage system is $r\lambda$, where r is the number of requests in each stream.

Let A be the random variable for inter-stream arrival time, and J be the random variable for inter-request arrival time. If we consider any period of time t such that there is the same number of request at the start and end of the time period, there is an average of λt arriving streams. If each stream generates r requests, then it is obvious that there is an average of $r\lambda t$ new requests over the same period of time t. Therefore, we have the mean time between arriving requests, $\overline{J}$, is

$$\overline{J} = \frac{1}{r\lambda}. \tag{5.25}$$

(c) Independent Requests

Every new stream generates a fixed number of requests to each queue of the tertiary drives. Since new streams are Poisson distributed, the arrival pattern of the first request of stream is also Poisson distributed. Likewise, the arrival pattern of the *j*th request of the streams is also Poisson distributed. Hence, we have a number of arriving requests that are all Poisson distributed. If the requests are independent of each other, then the arrival pattern of the merged Poisson streams would also be Poisson distributed. In which case, the arrival pattern of the requests is Poisson distributed.

If all requests arrive randomly, then the request arrival time is Poisson distributed and the variance of request arrival time is $\frac{1}{(r\lambda)^2}$.

Since the standard deviation of Poisson distribution is the same in value as the mean value, the variance of stream arrival time is $\frac{1}{\lambda^2}$. As the second moment is equal to the sum of the variance and the square of mean value, we have

$$\mathrm{E}[A^2] = \frac{2}{\lambda^2}.$$

The second moment is given as $\mathrm{E}[A^2] = \int P(t)t^2 dt$, where $P(.)$ is the probability density of streams arrival time. Thus, the variance of the inter-requests arrival time is

$$\mathrm{Var}[J] = \frac{1}{(r\lambda)^2}. \tag{5.26}$$

(d) All Requests Are Generated On Arrival

If all requests are generated by the stream immediately, then the requests arrival time would be zero for r-1 requests and $1/\lambda$ for one request. Therefore, the second moment of the request arrival time is

$$\mathrm{E}[J^2] = \frac{1}{r}\int P(t)t^2 dt + \frac{r-1}{r}\int P(t)\cdot 0 dt,$$

which implies

$$\mathrm{E}[J^2] = \frac{1}{r}\mathrm{E}[A^2].$$

Hence, we obtain

$$\mathrm{E}[J^2] = \frac{2}{r\lambda^2}. \tag{5.27}$$

Therefore, the variance of the request arrival time is equal to

$$\mathrm{Var}[J] = \frac{2}{r\lambda^2} - \left(\frac{1}{r\lambda}\right)^2 = \frac{2r-1}{(r\lambda)^2}. \tag{5.28}$$

Since $r > 1$, we have

$$\frac{2r-1}{(r\lambda)^2} > \frac{1}{(r\lambda)^2}. \tag{5.29}$$

Therefore, the variance of request arrival time when all requests are generated at stream arrival is greater than the variance of request arrival time when the requests arrive randomly.

(e) Two Requests Are Generated On Arrival

Every new stream generates a fixed number of requests to each queue of the tertiary drives. The initial requests of each stream are generated immediately after the stream is accepted for service. This tertiary storage system is modelled as a queueing network with feedback as in Figure 5.3.

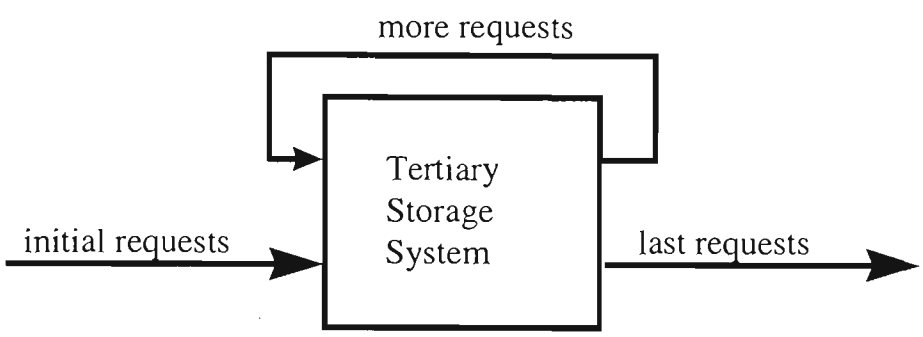
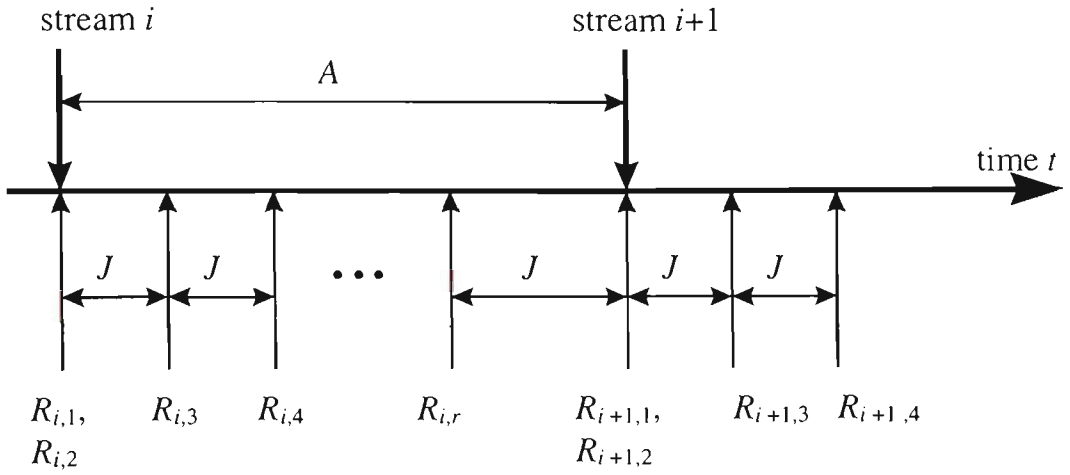


Figure 5.3. Model of request queueing with feedback

Within this system, the initial requests of each stream will arrive externally and the last requests of each stream will depart the system. When other requests are served, they will initiate another requests of the same stream. If the service times of two streams do not overlap with each other, the inter-request arrival time can be found as follows.



$R_{i,j}$ represents jth request of stream i

Figure 5.4. Arrival times of streams and requests

If only the first two requests are generated by the stream immediately and other requests are generated at request completion per our concurrent stream management, then the third to the last request arrive at a time that is equal to the request service time (Figure 5.4). Since the first two requests arrive at the same time, the requests inter-arrival time is

$$J = \begin{cases} A - \dfrac{r-2}{r\mu} & \text{for the first request,} \\ 0 & \text{for the second request, and} \\ \dfrac{1}{r\mu} & \text{for other requests.} \end{cases}$$

Therefore, the second moment of the request arrival time, $E[J^2]$, is

$$E[J^2] = \frac{1}{r}\int P(A)\left[A - \frac{r-2}{r\mu}\right]^2 dt + \frac{r-2}{r}\int P(A)\frac{1}{r\mu}dt,$$

which implies

$$E[J^2] = \frac{1}{r}E[A^2] - \frac{2}{r}\frac{r-2}{r\mu}E[A] + \left[\frac{r-2}{r\mu}\right]^2 + \frac{r-2}{r^2\mu}.$$

Substituting the first and second moments of A, we obtain

$$\mathrm{E}[J^2] = \frac{2}{r\lambda^2} - \frac{2(r-2)}{r^2\mu\lambda} + \frac{(r-2)(r+\mu-2)}{r^2\mu^2}. \quad (5.30)$$

Therefore, the variance of the request arrival time is

$$\mathrm{Var}[J] = \frac{2}{r\lambda^2} - \frac{2(r-2)}{r^2\mu\lambda} + \frac{(r-2)(r+\mu-2)}{r^2\mu^2} - \left(\frac{1}{r\lambda}\right)^2. \quad (5.31)$$

Unfortunately, new streams often arrive while some requests of other existing streams are being served. Hence, the inter-requests arrival pattern is unknown in general. Instead of pursuing further on the request arrival patterns, we switch to use the transition probabilities for the number of streams at steady state and the group service time to model the performance of the storage system.

5.3.3 State Transitions of Concurrent Streams

We shall describe the state transition probability for the unlimited concurrency condition in part (a) and the limited concurrency condition in part (b).

(a) Unlimited Concurrency

We denote the status of the tertiary storage system by an infinite number of states. The system is said to be in state j if j streams are being served concurrently or waiting in queue, where j varies from 0 to ∞. We focus on a time interval, h, which is small enough that the system can only change by one stream at a time. Letting P_j be the probability that j streams are concurrent, λ_j be the arrival rate of

new streams at j concurrent streams and μ_j be the departure rate at j concurrent streams, we have

Prob (HSS changes from state j to state j+1) = $\lambda_j h P_j$,

Prob (HSS changes from state j to state j-1) = $\mu_j h P_j$, and

Prob (HSS stays in state j) = $(1 - \lambda_j h - \mu_j h) P_j$,

for j = 1, 2, and so on. The transition states are shown in Figure 5.5.

At steady state, the probability of the system changing above any state j should be equal to the probability of changing below state j+1. Therefore, we have

$$\mu_j h P_j = \lambda_{j-1} h P_{j-1}, \qquad j = 1, 2, \ldots,$$

which implies

$$P_j = \frac{\lambda_{j-1}}{\mu_j} P_{j-1}, \qquad j = 1, 2, \ldots . \tag{5.32}$$

When j = 1, we have

$$P_1 = \frac{\lambda_0}{\mu_1} P_0 .$$

Expanding Equation (5.32) repeatedly from j to 1, we have

$$P_j = \frac{\lambda_{j-1} \ldots \lambda_1 \lambda_0}{\mu_j \ldots \mu_2 \mu_1} P_0, \tag{5.33}$$

for j = 1, 2, and so on.

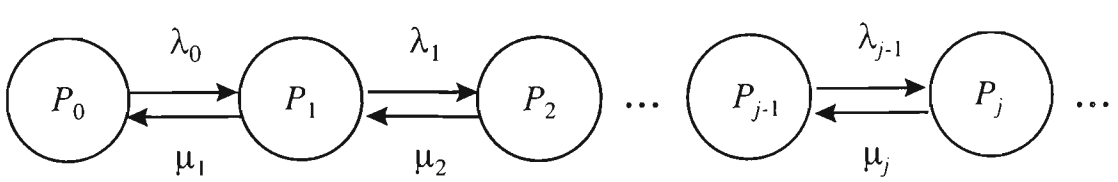


Figure 5.5. State transition diagram of large multimedia storage system

(b) Limited Concurrency

Let $\hat{S}$ be the maximum number of acceptable concurrent streams to the storage system. Each group may serve 0 to $\hat{S}$ concurrent streams. Since the number of streams being served concurrently cannot exceed $\hat{S}$, the departure rate at all states above $\hat{S}$ are equal to the departure rate at $\hat{S}$. Since the arrival rate is independent of the number of concurrent streams, all λ_j are the same. Let λ be the stream arrival rate, we have

$$P_j = \frac{\lambda}{\mu_{\hat{S}}} P_{j-1}, \qquad \text{for } j > \hat{S}. \tag{5.34}$$

Expanding this from $j = \hat{S}+1$, we have

$$P_j = (\frac{\lambda}{\mu_{\hat{S}}})^{j-\hat{S}} P_{\hat{S}}, \qquad \text{for } j \geq \hat{S}. \tag{5.35}$$

If $\frac{\lambda}{\mu_{\hat{S}}} \geq 1$, the system cannot serve streams fast enough and it becomes unsteady when the maximum number of concurrent streams is reached. Hence, we assume that $\frac{\lambda}{\mu_{\hat{S}}} < 1$ for a stable system. By summing all P_j terms for $j \geq \hat{S}$, we have

$$\sum_{j=\hat{S}}^{\infty} P_j = \frac{P_{\hat{S}}}{1 - \frac{\lambda}{\mu_{\hat{S}}}}. \tag{5.36}$$

Since the sum of all P_j terms where $j \geq 0$ must be equal to 1, we have

$$\sum_{j=0}^{\infty} P_j = 1,$$

which implies

$$\sum_{j=0}^{\hat{S}-1} P_j + \sum_{j=\hat{S}}^{\infty} P_j = 1.$$

Substituting $\sum_{j=\hat{S}}^{\infty} P_j$ from Equation (5.36), we have

$$\sum_{j=0}^{\hat{S}-1} P_j + \frac{P_{\hat{S}}}{1-\frac{\lambda}{\mu_{\hat{S}}}} = 1,$$

which implies

$$P_0 + \sum_{j=1}^{\hat{S}-1} \frac{\lambda^j}{\mu_j \ldots \mu_2 \mu_1} P_0 + \frac{1}{(1-\frac{\lambda}{\mu_{\hat{S}}})} \frac{\lambda^{\hat{S}}}{\mu_{\hat{S}} \ldots \mu_2 \mu_1} P_0 = 1.$$

Hence, we obtain

$$P_0 = \frac{1}{1 + \sum_{j=1}^{\hat{S}-1} \prod_{k=1}^{j} \frac{\lambda}{\mu_k} + \frac{\lambda^{\hat{S}}}{(1-\frac{\lambda}{\mu_{\hat{S}}}) \prod_{j=1}^{\hat{S}} \mu_j}}. \tag{5.37}$$

The probability of j present streams can then be found by substituting P_0 into Equation (5.33).

5.3.4 Service Times

The service time of each group of requests is divided into three parts: media exchange time, reposition time, and data transfer time. Other service time components are much smaller and hence not included in our model.

Media units are accessed in a round robin like manner. A media exchange is required with a certain probability. If an exchange is required, it is performed within a range of exchange service time. We describe the media exchange time for each group of requests in part (a).

A reposition is always required for each request. In order to achieve high throughput, the SCAN scheduling rule is used to serve requests. Requests for data on the same media unit are served in the relative order of their data location on the media units. We assumed that the head resides on the starting position of the media unit after exchange. The request for data nearest to the starting position is first served. Then the request for data next nearest to the starting position is served next and so on. In this way, the full reposition time is shared among the requests. We explain the reposition time for each group of requests in part (b).

A data transfer time is always required for every request. Since we use a constant time length approach to store the segments, the segment size varies. We assume that the segment sizes vary randomly within a fixed range. We also assume that the rate of transfer is fixed. The transfer time is, hence, directly proportional to the segment size. We present the data transfer time for each group of requests in part (c).

Lastly, we describe the individual group service time and mean group service time in part (d) and part (e) respectively.

(a) Media Exchange Time

The first component of the group service time is the media exchange time. The media exchange time is composed of the exchange device waiting time and the exchange request service time. We use the exchange request service time to find the exchange device waiting time as follows.

- **Exchange Request Service Time**

When each drive has its own exchange device, the exchange requests may be served immediately, and no waiting is required. We assume that the exchange service time varies uniformly within a range of variation which is around 10% of the exchange time.

Let χ be the time to perform an exchange and 2σ be the range of variation. Then, the exchange times are uniformly distributed between $\bar{\chi}-\sigma$ and $\bar{\chi}+\sigma$ with density function $\varphi(.)$. By definition, $E[\chi^2] = \int_{\bar{\chi}-\sigma}^{\bar{\chi}+\sigma} t^2 \varphi(t)dt$. Since $\varphi(.) = \frac{1}{2\sigma}$ for uniform distribution, we have

$$E[\chi^2] = \frac{1}{2\sigma}\int_{\bar{\chi}-\sigma}^{\bar{\chi}+\sigma} t^2 \mathrm{d}t ,$$

which implies

$$E[\chi^2] = (\bar{\chi})^2 + \frac{\sigma^2}{3}. \tag{5.38}$$

Hence, we have

$$\mathrm{Var}[\chi] = \frac{\sigma^2}{3}. \tag{5.39}$$

- **Exchange Waiting Time**

When the tertiary drives share an exchange device, a new exchange request wait for the completion of all prior exchange requests before it can be served. Using the same notations, we have $\omega = w + \chi$, where ω is the media exchange time and w is the waiting time in queue. We now have a nested queue whose model is shown in Figure 5.6.

We assume that this queueing system is also conservative. We also assume the non-preemptive scheduling condition that incomplete requests are not interrupted. We shall investigate the waiting time and response time for the popular FCFS scheduling. We assume that the exchange requests arrive randomly to the queue. Using the queueing notations in [95], we have a M/G/1 queue of the exchange requests.

The mean request arrival rate to each drive is

$$I = \frac{r\lambda}{D}. \tag{5.40}$$

where r is the number of requests per stream and D is the number of tertiary drives.

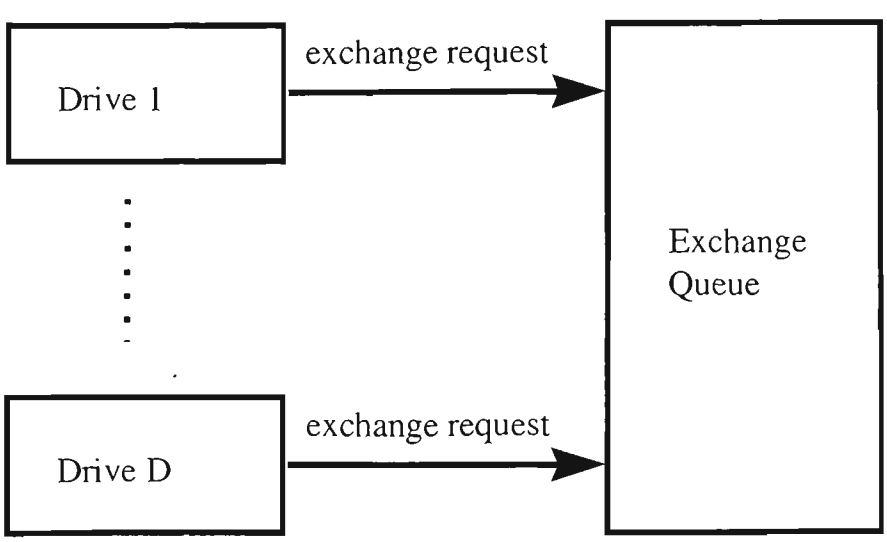


Figure 5.6. Exchange request queue model

Since one of the S requests in the group sends an exchange request to the exchange device, the combined arrival rate to the exchange queue, I_2, is

$$I_2 = D\frac{1}{S}\left(\frac{r\lambda}{D}\right) = \frac{r\lambda}{S}. \tag{5.41}$$

We apply the Pollaczek-Khintchine formula [95] to find the media exchange waiting time at S concurrent streams, $E_S[w]$, as

$$E_S[w] = \frac{\left(\frac{r\lambda\bar{\chi}}{S}\right)\bar{\chi}\left(1+\frac{\sigma^2}{3\bar{\chi}^2}\right)}{2\left(1-\frac{r\lambda\bar{\chi}}{S}\right)},$$

which implies

$$E_S[w] = \frac{r\lambda\left(3\bar{\chi}^2+\sigma^2\right)}{6\left(S-r\lambda\bar{\chi}\right)}. \tag{5.42}$$

Given $\omega = w + \chi$, we obtain the mean media exchange time at S streams, $E_S[\omega]$, as

$$E_S[\omega] = \frac{r\lambda\left(3\bar{\chi}^2+\sigma^2\right)}{6\left(S-r\lambda\bar{\chi}\right)} + \bar{\chi}. \tag{5.43}$$

By definition, $E[\chi^3] = \int_{\bar{\chi}-\sigma}^{\bar{\chi}+\sigma} t^3\varphi(t)\mathrm{d}t$. Since $\varphi(t) = \frac{1}{2\sigma}$ for uniform distribution, we have

$$E[\chi^3] = \bar{\chi}\left(\bar{\chi}^2+\sigma^2\right). \tag{5.44}$$

Using Takács recurrence formula [84], we find the second moment of the waiting time, $E_S[w^2]$, as

$$E_S[w^2] = 2\left(E_S[w]\right)^2 + \frac{I\,E[\chi^3]}{3(1-I\,E[\chi])},$$

which implies

$$E_S[w^2] = 2\left[\frac{r\lambda\left(3\bar{\chi}^2+\sigma^2\right)}{6\left(S-r\lambda\bar{\chi}\right)}\right]^2 + \frac{r\lambda\bar{\chi}\left(\bar{\chi}^2+\sigma^2\right)}{3\left(S-r\lambda\bar{\chi}\right)}. \quad (5.45)$$

The second moment of the media exchange time can hence be found [84] using

$$E_S[\omega^2] = E_S[w^2]+2E_S[w]E[\chi]+E[\chi^2]. \quad (5.46)$$

We assume that each media unit has the same probability, $\frac{1}{K}$, of being accessed by a stream. The media exchange time of the non-striping placement method can be found similarly using the arrival rate of $\lambda\left(1-\frac{1}{K}\right)$. Similarly, the media exchange time of the parallel striping placement method can be found using the arrival rate of $\lambda D\left(1-\frac{1}{K}\right)$.

(b) Reposition Time

The second component of the group service time is the reposition time. We shall determine the reposition time when the SCAN policy is used. We assume that the data positions are uniformly distributed among the length of the media. We need to find the data positions of S requests in the full length, L, of a media unit. The probability that one data position resides within the range y to $y+dy$ in a uniform distribution is

$$\frac{dy}{L}. \quad (5.47)$$

Hence, the probability that the data position falls within a distance y from the beginning of a media unit is

$$\text{Prob}(Y_i \le y) = \frac{y}{L}. \quad (5.48)$$

The probability that all S requests fall within a distance y from the beginning of a media unit is

$$\text{Prob}(Y_1 \le y, Y_2 \le y, \ldots, Y_s \le y) = (\frac{y}{L})^S . \tag{5.49}$$

Let F(y) be the distribution function of the probability that the longest reposition distance of S requests. When the SCAN policy is used for S requests, this is the same as the total reposition distance of S requests. We have

$$F(y) = (\frac{y}{L})^S . \tag{5.50}$$

Let $f(y)$ be the probability of the longest reposition distance from the beginning of the media unit is between y and y+dy, where dy tends to 0. Now, $f(y)$ can be found from the first derivative of the above distribution function. Hence, we have

$$f(y) = \frac{Sy^{S-1}}{L^S} dy . \tag{5.51}$$

Let Y_S be the reposition distance of each request, then SY_S is the total reposition distance in a group of S streams. We have

$$\text{Prob}\ (SY_S \le\ y) = \frac{Sy^{S-1}}{L^S} dy ,$$

which implies

$$E[SY_S] = \int_0^L y \frac{Sy^{S-1}}{L^S} dy .$$

After simplifications, we obtain

$$E[SY_S] = \frac{SL}{S+1} .$$

Also, we have

$$\text{Prob}\,(SY_S \le y) = \frac{Sy^{S-1}}{L^S}dy\,,$$

which implies

$$\text{Var}[SY_S] = \int_0^L y^2 \frac{Sy^{S-1}}{L^S}dy - \left(\frac{SL}{S+1}\right)^2.$$

Therefore, we obtain

$$\text{Var}[SY_S] = \frac{SL^2}{(S+1)^2(S+2)}. \tag{5.52}$$

Let α_S be the reposition time of a request inside a group of S requests. Since the reposition time of magnetic tapes is in proportionate to the reposition distance, the reposition time of a request inside a group of S requests is

$$\alpha_S = a + bY_S,$$

where a, b are constants. Let α be the total reposition time of the group and $E_S[\alpha]$ be the expected total reposition time for S requests, we have

$$\alpha = aS + bSY_S,$$

which implies

$$E_S[\alpha] = \int_0^L (aS + by)\frac{Sy^{S-1}}{L^S}dy\,,$$

giving

$$E_S[\alpha] = aS + bL\frac{S}{S+1}. \tag{5.53}$$

Also, we have

$$E_S[\alpha^2] = \int_0^L (aS + by)^2 \frac{Sy^{S-1}}{L^S}dy\,.$$

This gives

$$E_S[\alpha^2] = a^2S^2 + 2abL\frac{S^2}{S+1} + b^2L^2\frac{S}{S+2}. \quad (5.54)$$

The reposition time of the non-striping placement method and the parallel striping method can be found similarly with the reposition time being shared among r and $\frac{r}{D}$ requests respectively.

(c) Data Transfer Time

The third component of the group service time is the data transfer time. Let Z be the random variable of the segment size. We assume that Z is uniformly distributed between $(Z-\zeta, Z+\zeta)$ and the rate of transfer, ϕ, is fixed. Letting τ be the transfer time of each request, we have

$$E[\tau] = \frac{Z}{\phi} \quad (5.55)$$

and

$$Var[\tau] = \frac{\zeta^2}{3\phi}. \quad (5.56)$$

Letting T be the total time of a group of S requests spent in transferring data, we have

$$T = S\tau,$$

giving

$$E_S[T] = \frac{S\bar{Z}}{\phi}, \quad (5.57)$$

and

$$E_S[T^2] = \left(\frac{\bar{Z}}{\phi}\right)^2 S^2. \quad (5.58)$$

Similarly, the data transfer time for r requests of the non-striping placement method and the data transfer time for $\frac{r}{D}$ requests of the parallel striping placement method can be obtained.

(d) Individual Group Service Time

The requests on the same media unit are served in parallel as a concurrent group of requests. The concurrent group of requests are served using the following operations:

1. Exchange media unit,
2. Reposition to the start of the first request,
3. Transfer data for the first request, and
4. Repeat steps 2 and 3 for all requests in the group.

The group service time is hence the sum of media exchange time, reposition time, and the transfer time. Letting B_S be the service time for a group of S concurrently serving requests, we have

$$B_S = \omega + \alpha + T,$$

Substituting expected values of the media exchange time from Equation (5.43), reposition time from Equation (5.53) and data transfer time Equation (5.57), we can get the mean group service time at S streams, $\overline{B_S}$ or $E_S[B]$, using

$$\overline{B_S} = E_S[\omega] + E_S[\alpha] + E_S[T]. \tag{5.59}$$

Since $B^2 = (\omega + \alpha + T)^2$ and the service times are independent, we have

$$E_S[B^2] = E_S[\omega^2] + E_S[\alpha^2] + E_S[T^2] +$$

$$2E_S[\alpha]E_S[\omega] + 2E_S[\alpha]E_S[T] + 2E_S[\omega]E_S[T]. \quad (5.60)$$

Substituting the first and second moments of the media exchange time, reposition time, and data transfer time, we can obtain the second moment and the variance of group service time at S streams.

(e) Mean Group Service Time

Since a stream finishes when all its r requests are served, each one of D drive serves $\frac{r}{Dj}$ groups of requests to finish a stream, where j is the number of requests per group. The mean stream service time is then equal to $\frac{r\overline{B_j}}{Dj}$. Hence, the departure rate at j streams, μ_j, is

$$\mu_j = \frac{jD}{r\overline{B_j}}. \quad (5.61)$$

Substituting μ_j into Equation (5.33) and Equation (5.37) and we obtain

$$P_0 = \frac{1}{1 + \sum_{j=1}^{\hat{S}-1} (\frac{r\lambda}{D})^j \frac{1}{j!} \prod_{k=1}^{j} \overline{B_k} + \frac{(r\lambda)^{\hat{S}}}{(1 - \frac{r\lambda B_{\hat{S}}}{D\hat{S}\mu_{\hat{S}}}) D^{\hat{S}} \prod_{j=1}^{\hat{S}} \overline{B_j}}},$$

$$P_j = (\frac{r\lambda}{D})^j \frac{P_0}{j!} \prod_{k=1}^{j} \overline{B_k}, \qquad j = 1, 2, \ldots, \hat{S}-1,$$

and

$$P_j = (\frac{r\lambda}{D})^j (\frac{B_{\hat{S}}}{\hat{S}})^{j-\hat{S}} \frac{P_0}{\hat{S}!} \prod_{k=1}^{j} \overline{B_k}, \qquad j \geq \hat{S}.$$

Therefore, the mean number of concurrently serving streams can be found from

$$E[S] = \sum_{j=0}^{\hat{S}-1} jP_j + \sum_{j=\hat{S}}^{\infty} \hat{S}P_j . \quad (5.62)$$

Using $E[B] = \sum_{j=0}^{\hat{S}-1} P_j \overline{B_j} + \sum_{j=\hat{S}}^{\infty} \overline{B_{\hat{S}}} P_j$, we can find the mean service time for concurrent groups of requests using Equation (5.59). Similarly, we can find the second moment of the group service time using

$$E[B^2] = \sum_{j=0}^{\hat{S}-1} P_j E_j[B^2] + \sum_{j=\hat{S}}^{\infty} E_{\hat{S}}[B^2] P_j$$

and Equation (5.60). The variance of the group service time is hence found using

$$\mathrm{Var}[B] = E[B^2] - (E[B])^2.$$

Therefore, the mean stream service time, $E[G]$, is obtained as

$$E[G] = \frac{r}{DS} E[B].$$

The stream service time for r requests of the non-striping placement method and the stream service time for $\frac{r}{D}$ requests of the parallel striping placement method are both found as sum of media exchange time, reposition time, and data transfer time.

Since there is no solution available for this M/G/m queue, we approximate the waiting time with that of a M/G/1 queue. We apply Pollaczek-Khintchine formula again to get the mean stream waiting time.

5.3.5 System Throughput

In order to display the streams without starvation, the storage system must retrieve each segment before it is due for display. That is, a segment can start to display only after the segment is retrieved.

In order to achieve high system throughput, the SCAN scheduling algorithm is used to serve requests on the same media unit. Since the data can be stored anywhere on the media unit, the maximum number of requests between two consecutive requests of the same stream is $2\hat{S}-2$, where $\hat{S}$ is the maximum number of concurrent streams in each group. If we use this finishing time difference between any two consecutive requests as an upper bound on the segment retrieval time, then we have

$$\frac{DX}{\delta} \geq \omega + (2\hat{S}-1)(\alpha_S + \frac{Z}{\phi}),$$

which X is the request service time, δ is the display bandwidth, Z is the segment size, and ϕ is the data transfer rate. This is equivalent to

$$\frac{DX}{\delta} \geq \mathrm{E}_{2\hat{S}-1}[B]. \tag{5.63}$$

Alternatively, an accepted stream may start to display after all the requests on a media unit are served. The deadline to any request is the end of each cycle. Hence, we have the continuous display requirement as

$$\frac{DX}{\delta} \geq \omega + \hat{S}(\alpha_S + \frac{Z}{\phi}),$$

which is equivalent to

$$\frac{DX}{\delta} \geq E_{\hat{S}}[B]. \tag{5.64}$$

When more streams are concurrently served, the group service time is longer. Hence, $E_j[B]$ increases with j. Thus, the upper bound on the number of concurrent streams as given in Equation (5.64) is higher than that in Equation (5.63). We can find the maximum number of streams, $\hat{S}$, such that

$$E_{\hat{S}}[B] \le \frac{DX}{\delta} \le E_{\hat{S}+1}[B]. \tag{5.65}$$

Therefore, the continuous display requirement is guaranteed for any number of concurrent streams that is less than the maximum number of concurrent streams.

In the high concurrency placement, one segment is retrieved for each stream after each media exchange. If D drives are serving S streams, then DS stripes are retrieved every cycle. Therefore, we have for the system throughput

$$\frac{DSZ}{E_S[B]}. \tag{5.66}$$

From Equation (5.66), we can see that the system throughput increases when larger segments are used.

Also, the system throughput is maximized when the maximum number of concurrent streams are served. Therefore, the maximum system throughput is

$$\frac{D\hat{S}Z}{E_{\hat{S}}[B]}. \tag{5.67}$$

Intuitively, the number of requests being retrieved from each media unit is increased when more streams are concurrent. When the segments being retrieved from one media unit are more than the size of each object, the average media exchanges per stream is reduced. However, the number of repositions within a media unit is also increased when more streams are concurrent. Fortunately, the reposition time can be shared among all requests by the SCAN scheduling

method. Hence, it is possible to increase the system throughput by serving more streams concurrently.

The system throughput of the non-striping placement method is found as the number of drives times the object size divided by the stream service time, and the system throughput of the parallel striping placement method is found as the data stripe size divided by the stream service time.

5.3.6 Stream Response Time

We describe the stream response time in three typical conditions. The response time of a single drive and multiple drives with exponential service time are described in part (a) and part (b) respectively. After that, the response time of multiple drives with concurrent group service time is shown in part (c).

(a) Single Drive with Exponential Service Time

We first consider a simple case when there is only one tertiary drive, the service rate is exponentially distributed, and the requests are served in the FIFO order. The model of a single drive queue with feedback requests is shown in Figure 5.7.

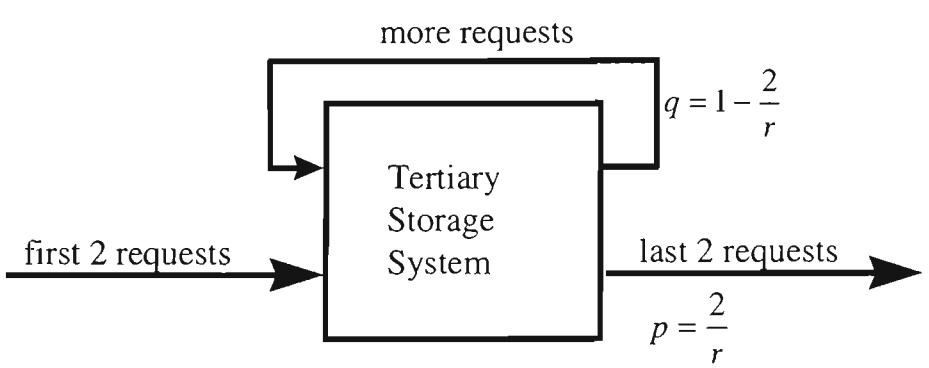


Figure 5.7. Model of single drive with feedback queue

Since the first two requests of each new stream are generated from external input and the last two requests of each new stream will exit, we have external input rate = 2λ. Let I be the combined input rate and we apply the Jackson's theorem [84] to get

$$I = 2\lambda + qI ,$$

which implies

$$2\lambda = \left(1 - \frac{r-2}{r}\right) I ,$$

giving

$$I = r\lambda. \tag{5.68}$$

This is intuitively the same as r requests of each stream arriving at the tertiary storage system. Letting μ be the service rate, we have for the system utilization, ρ, as

$$\rho = \frac{I}{\mu} = \frac{r\lambda}{\mu}. \tag{5.69}$$

Since the service pattern of the requests is assumed to be exponentially distributed, the coefficient of variation equal to unity. We apply the Pollaczek-Khintchine formula to find the mean request waiting time, $\overline{w}$, as

$$\overline{w} = \frac{\frac{r\lambda}{\mu}\frac{1}{\mu}(1+1)}{2\left(1 - \frac{r\lambda}{\mu}\right)},$$

which gives

$$\overline{w} = \frac{r\lambda \overline{X}^2}{\left(1 - r\lambda \overline{X}\right)}. \tag{5.70}$$

(b) Multiple Drives with Exponential Service Time

We extend the above case for multiple tertiary drives with exponential service time. Let D be the number of tertiary drive and $D \geq 1$. We still assume that the service time of requests are exponentially distributed. Each new stream generates two requests to each drive immediately after it is accepted. The external input rate of requests to each drive is then 2λ. At each drive, the first two requests of each stream will arrive externally and the last two requests of each stream will depart the system (Figure 5.8).

As each stream has r requests, the probability, p, of departing from the system is $p = \frac{2D}{r}$. The probability, q, of generating more requests is

$$q = 1 - p = 1 - \frac{2D}{r}. \tag{5.71}$$

Letting I be the combined input rate to each drive, we apply the Jackson's theorem again to find the request arrival rate.

$$I = 2\lambda + qI = 2\lambda + \frac{r - 2D}{r} I,$$

which implies

$$I = \frac{r\lambda}{D}. \tag{5.72}$$

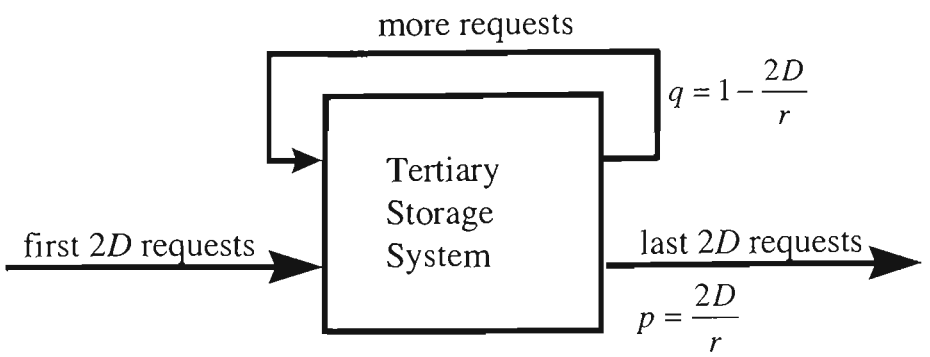


Figure 5.8. Model of multiple drives

Letting μ be the service rate, we obtain the system utilization, ρ, as

$$\rho = \frac{I}{\mu} = \frac{r\lambda}{D\mu}. \tag{5.73}$$

We apply the Pollaczek-Khintchine formula again to find the mean request waiting time, $\overline{w}$, as

$$\overline{w} = \frac{\frac{r\lambda}{D\mu}\frac{1}{\mu}(1+1)}{2\left(1-\frac{r\lambda}{D\mu}\right)},$$

which gives

$$\overline{w} = \frac{Dr\lambda\overline{X}^2}{\left(D-r\lambda\overline{X}\right)}. \tag{5.74}$$

(c) Multiple Drives with Concurrent Group Service Time

A new stream first waits for acceptance to the storage system. On acceptance, it sends two requests to each drive's queue. The first request at each drive then waits for the group that accesses the required media unit being served. The first request is then served within the concurrent group. The details stream response time is illustrated in Figure 5.9.

The concurrent group service time is composed of the media exchange time and the request service time for all the requests in the group. Each media exchange time is composed of waiting time for the exchange device and the time to perform an exchange. The request service time of each request consists of waiting for concurrent requests to be served, the request's reposition time, and the request's data transfer time.

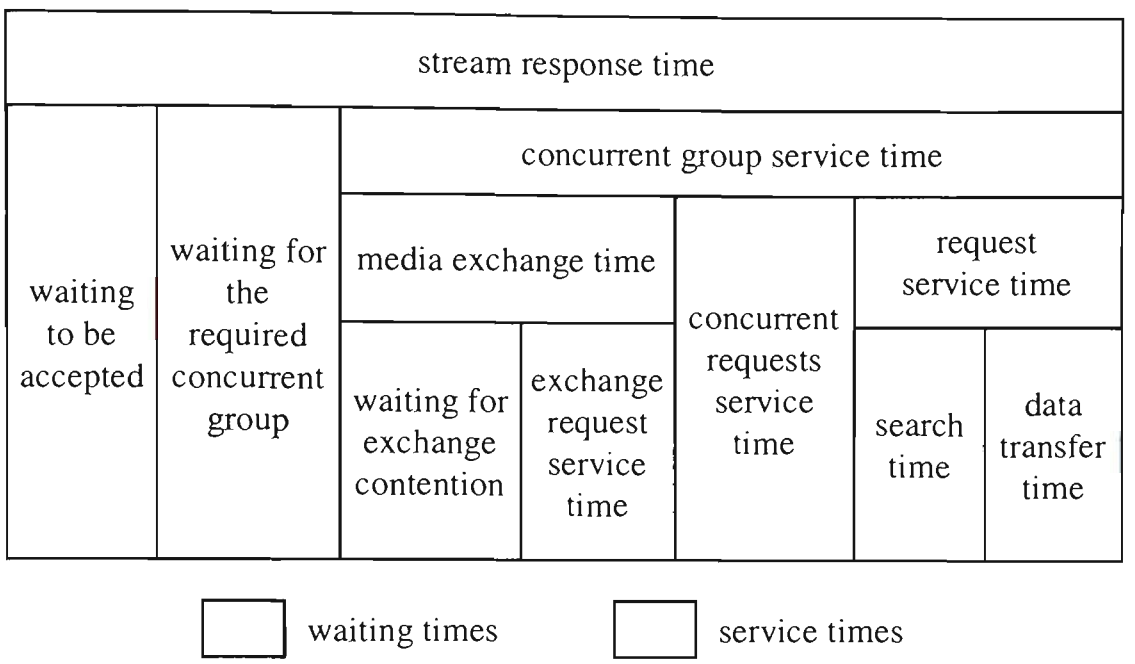


Figure 5.9. Components of stream response time

The waiting time is the time that a new stream spends in the queue before its first request is served. The waiting time is unbounded if the system is busy. The start up latency is the time that an accepted stream waits before it starts to display.

Unfortunately, the high concurrency placement method introduces two new latency times to the stream response time. The two latency times are the waiting time for other concurrent groups and the concurrent requests service time. Fortunately, both latency times are bounded above.

Since the tertiary drives serve requests according to the fixed sequence of media units in the high concurrency placement method, they cannot immediately respond to new streams that need media units different from the current one on the tertiary drives. The waiting time for concurrent group is bounded by the round

time. If new streams arrive randomly, then the expected stream start up latency is half of the round time. Hence, the expected new stream start up latency is $\frac{K}{2}\mathrm{E}[B]$.

Since a number of requests are served concurrently on the same media unit, the tertiary drives serve requests according to data locations on the media units. The concurrent request service time is bounded by the group service time. Hence, the concurrent request service time is found as E[B]. Therefore, we have for the new expected stream response time, E[R_s], as

$$\mathrm{E}[R_s] = W_s + \left(\frac{K}{2} + 1\right)\mathrm{E}[B], \tag{5.75}$$

where W_s is the new stream waiting time.

Since each drive serves all r requests of each stream together in the non-striping method, the new stream response time, E[R_s], is found as

$$\mathrm{E}[R_s] = W_s + E[\omega] + \frac{1}{r}\left(E[\alpha] + E[T]\right). \tag{5.76}$$

Similarly, each drive serves r/D requests of each stream together in the parallel striping method, the new stream response time, E[R_s], is found as

$$\mathrm{E}[R_s] = W_s + E[\omega] + \frac{D}{r}\left(E[\alpha] + E[T]\right). \tag{5.77}$$

5.3.7 Buffer Size

When data are retrieved from tertiary storage, they are first kept in memory buffers. Data can then be consumed from the buffers at the display bandwidth. Since the supply of data may not match exactly with the display bandwidth, some data are then written to staging buffers on disks to release memory temporarily. Hence, the staging buffer size depends on the data retrieval rate and the data consumption rate.

We calculate the staging buffer size to store data that are retrieved from tertiary storage faster than they are consumed. Since the time that the tertiary drives spend in serving each group of concurrent requests is E[B], the mean time to retrieve one segment for a stream is also E[B]. Since the tertiary drives retrieve DZ blocks in one group for each stream, the time to display the data being retrieved from one group for each stream is $\frac{DZ}{\delta}$. The amount of extra time to display the data in buffer is

$$\frac{DZ}{\delta} - \mathrm{E}[B].$$

Since the display bandwidth is δ, the amount of extra data in each cycle is

$$\delta(\frac{DZ}{\delta} - \mathrm{E}[B]).$$

This buffer size becomes the largest when the entire object has just been retrieved. Since there are r requests sent to D drives by each stream, the largest buffer size is

$$\frac{r\delta}{D}(\frac{DZ}{\delta} - \mathrm{E}[B]).$$

Therefore, the staging buffer size per stream for the high concurrency placement method is

$$rZ - \frac{r\delta}{D}\mathrm{E}[B]. \tag{5.78}$$

Similarly, the staging buffer size per stream for the non-striping method and parallel striping method is

$$rZ - \delta(\mathrm{E}[G]), \tag{5.79}$$

where $\mathrm{E}[G]$ is the expected stream service time.

5.4 Segment Based Pipelining

We describe the model to study the segment based pipelining method in this Section. First, we explain our assumptions for this model in Section 5.4.1. If these assumptions can be met, this model can provide closed form solutions to the maximum user latency and the amount of resident disk space. We adopt the notations in Table 5.3 for this segment based pipelining model.

We shall elaborate the continuous display requirements in Section 5.4.2. In Section 5.4.3, we determine the upper bound on the number of data stripes per segment. The upper bound on the logical segment size and the minimum storage space for the start up data stripes are obtained in Section 5.4.4. We then derive the upper bound on the reposition latency and start up latency in Section 5.4.5.

Table 5.3. Notations used in segment based pipelining model

Parameter	Meaning
ω	media exchange time
α	reposition time
ϕ	data transfer rate
δ	display bandwidth
β	disk bandwidth
m	number of segments per object
n	number of data stripes per segment
η	number of data stripes in a segment
O	object size
$\overline{Z}$	mean logical segment size
$\overline{Z_j}$	mean size of the jth data stripe in segments
$Z_{i,j}$	size of jth data stripe of ith segment
M	media block size

5.4.1 Assumptions

With current technology, the secondary bandwidth of a single disk is around 10MB/s, the tertiary bandwidth of low end tertiary storage devices is of the order of 100 KB/s, and the display bandwidth of MPEG-2 compressed video is around 1.5 MB/s. Therefore, it is reasonable to make the following assumptions for our tertiary storage systems:

1. The secondary bandwidth is high enough to support many concurrent displaying streams.
2. The tertiary bandwidth is lower than the display bandwidth. Otherwise, all data stripes on tertiary store could be of one media block in size.
3. The number of concurrent streams directed to the tertiary storage is not more than the number of tertiary drives so that there is not much queue contention for the tertiary drives.
4. There is no contention for exchange device. This is suitable for low end drives with dedicated exchange devices such as automatic CD changers. Contention is also unlikely when the utilization of the drives is low.
5. Each segment is stored contiguously on only one media unit to avoid media exchange or reposition during the pipelining process. Large objects may have its segments stored on a few media units.

5.4.2 Continuous display requirement

In order to provide instant response to user requests, the display time of the start up data stripe should be longer than the media exchange time plus the time to retrieve the second data stripe. Hence, we have

$$\frac{Z_{i,1}}{\delta} \geq \omega + \alpha + \frac{Z_{i,2}}{\phi}, \qquad 1 \leq i \leq m\text{-}1. \tag{5.80}$$

where $Z_{i,j}$ is the size of the jth data stripe in the ith segment, ω is the media exchange time, α is the reposition time, and ϕ is the data transfer rate. In order to display the object continuously, the display time of each data stripe should be at least the retrieval time of the successive data stripe. Therefore, we have

$$\frac{Z_{i,j}}{\delta} \geq \frac{Z_{i,j+1}}{\phi}, \qquad 2 \leq j \leq n\text{-}1,\ 1 \leq i \leq m, \tag{5.81}$$

and

$$\frac{Z_{i,n}}{\delta} \geq \frac{Z_{i+1,1}}{\beta}, \qquad 1 \leq i \leq m\text{-}1. \tag{5.82}$$

Since the segment sizes and data stripe sizes can be very different, we find it useful to use the mean segment size and the mean data stripe size. Variations in data stripe size and segment size can be handled by read-ahead buffers. Hence, we have

$$\overline{Z} = \frac{O}{m}. \tag{5.83}$$

Hence, we have

$$\overline{Z_j} = \frac{1}{m}\sum_{i=1}^{m} Z_{i,j}. \tag{5.84}$$

Applying the mean to Equation (5.80), Equation (5.81), and Equation (5.82), we obtain

$$\frac{\overline{Z_1}}{\delta} \geq \omega + \alpha + \frac{\overline{Z_2}}{\phi}, \tag{5.85}$$

$$\frac{\overline{Z_j}}{\delta} \geq \frac{\overline{Z_{j+1}}}{\phi}, \qquad 2 \leq j \leq n\text{-}1, \tag{5.86}$$

and

$$\frac{\overline{Z_n}}{\delta} \geq \frac{\overline{Z_1}}{\beta}. \tag{5.87}$$

5.4.3 Number of Data Stripes and Size of the Last Data Stripe

The pipelining method is more efficient when more data stripes are created in each segment because the ratio of the first data stripe to segment size is reduced when more overlapping is achieved. However, the continuous display requirements impose limitations on the retrieval times of the data stripes. We evaluate the number of data stripes in each segment below.

Applying Equation (5.86) recursively, we obtain

$$\overline{Z_n} \leq \frac{\phi}{\delta}\overline{Z_{n-1}} \leq \cdots \leq (\frac{\phi}{\delta})^{n-2}\overline{Z_2},$$

which implies

$$\overline{Z_j} \geq \left(\frac{\delta}{\phi}\right)^{n-j}\overline{Z_n}, \qquad 2 \leq j \leq n\text{-}1. \tag{5.88}$$

From Equation (5.85), we have

$$\frac{\overline{Z_1}}{\delta} \geq \omega + \alpha + \frac{\overline{Z_2}}{\phi},$$

giving

$$\overline{Z_1} \geq (\omega + \alpha)\delta + \left(\frac{\delta}{\phi}\right)^{n-1}\overline{Z_n}. \tag{5.89}$$

From Equation (5.87) and Equation (5.89), we obtain lower and upper bounds on the start up data stripe

$$(\omega+\alpha)\delta+\left(\frac{\delta}{\phi}\right)^{n-1}\overline{Z_n}\le\overline{Z_1}\le\frac{\beta}{\delta}\overline{Z_n},$$

which gives

$$\overline{Z_n}\ge\frac{(\omega+\alpha)\delta}{\beta/\delta-(\delta/\phi)^{n-1}}. \tag{5.90}$$

As both $(\omega+\alpha)\delta$ and $\overline{Z_n}$ must be greater than zero, we have

$$\frac{\beta}{\delta}>(\frac{\delta}{\phi})^{n-1}. \tag{5.91}$$

This implies

$$n<\frac{\log(\beta/\delta)}{\log(\delta/\phi)}+1. \tag{5.92}$$

In order to achieve the maximum pipeline efficiency, we may use the maximum number of data stripes. Hence, we have

$$n=\left\lfloor\frac{\log(\beta/\delta)}{\log(\delta/\phi)}\right\rfloor+1. \tag{5.93}$$

We have used the floor function because n is an integer. The actual number of data stripes in short segments can be less.

Unfortunately, the size of the data stripes could become very large when $\frac{\beta}{\delta}-\left(\frac{\delta}{\phi}\right)^{n-1}$ is small. In such a condition, only a few segments are created. The size of the start up data stripe is large, and the user latency is high. Only a few logical segments could be created and hence making the first level data striping

not useful. Therefore, the maximum number of data stripes per segment should not be used when $\frac{\beta}{\delta} - \left(\frac{\delta}{\phi}\right)^{\eta-1}$ is small.

If we have sufficient secondary bandwidth such that $\beta \geq \frac{(\omega+\alpha)\delta^2}{M}$, it is desirable to use slightly less data stripes in each segment. We keep the size of the last data stripe fixed at one media block size. Since the size of each data stripe should be at least one media block, we have $\overline{Z_j} \geq M$. Thus, we have

$$M \geq \frac{(\omega+\alpha)\delta}{\beta/\delta - (\delta/\phi)^{\eta-1}},$$

which implies

$$\eta = \left\lfloor \frac{\log\ (\beta/\delta - (\omega+\alpha)\ \delta/M)}{\log\ (\delta/\phi)} \right\rfloor + 1. \tag{5.94}$$

5.4.4 Number of Segments and Segment Size

If the length of each segment is too short, less overlapping is achieved by pipelining. More disk space is then required to store the start up data stripes. If the length of each segment is too long, the start up data stripes are widely spaced apart. User can start to display from only a few locations. In order to use minimum disk space and to provide sufficient display start up locations, the segment size should be minimized.

We substitute $\overline{Z_j}$ from Equation (5.88), and the smallest mean segment size, $\overline{Z}$, should be

$$\overline{Z} = \sum_{j=1}^{\eta} \overline{Z_j},$$

which implies

$$\overline{Z} \geq (\omega + \alpha)\delta + \sum_{j=1}^{\eta} (\frac{\delta}{\phi})^{j-1} \overline{Z_\eta}.$$

Hence, we obtain

$$\overline{Z} \geq (\omega + \alpha)\delta + \frac{(\delta / \phi)^{\eta} - 1}{(\delta / \phi) - 1} \overline{Z_\eta}. \tag{5.95}$$

This is the lower bound on the segment size when η data stripes are used. Substituting $\overline{Z}$ into Equation (5.83), we obtain the maximum number of logical segments

$$m \leq \frac{O}{(\omega + \alpha)\delta + \frac{(\delta / \phi)^{\eta} - 1}{(\delta / \phi) - 1} \overline{Z_\eta}}. \tag{5.96}$$

Since the start up data stripe of all segments should reside in disks permanently, the total size of start up data stripes of all segments, $m\overline{Z_1}$, is

$$m\overline{Z_1} = m\left[\overline{Z} - \sum_{j=2}^{\eta} \overline{Z_j}\right].$$

Substituting $\overline{Z_j}$ from Equation (5.88), the required disk space is then bounded by

$$m\overline{Z_1} \leq m\left[\frac{O}{m} - \sum_{j=2}^{\eta} \left(\frac{\delta}{\phi}\right)^{j-1} \overline{Z_\eta}\right],$$

which implies

$$m\overline{Z_1} \leq O - m\frac{\left(\delta / \phi\right)^{\eta-1} - 1}{\left(\delta / \phi\right) - 1} \overline{Z_\eta}. \tag{5.97}$$

5.4.5 Stream Start Up Latency

We describe the time spent in loading data segments in part (a) and part (b). Afterward, we describe the user start up latency in the part (c).

(a) Loading Low Temporal Resolution Segments

If the low temporal resolution segments are not downloaded beforehand, time is required to load them on demand. Let Z_0 be the size of the low temporal resolution data segments. Since they are stored contiguously on tertiary storage, the time to download the low temporal resolution data segments is

$$\omega + \alpha + \frac{mZ_0}{\phi}. \tag{5.98}$$

If the low temporal resolution segments are downloaded to disks, the time to load these segments from disks is

$$\frac{mZ_0}{\beta}. \tag{5.99}$$

(b) Loading Start Up Data Stripes

If the start up data stripes are not resident on disks, then time is required to load them on demand. If all start up data stripes are stored together with the low resolution segments, the additional latency to load start up data stripes is

$$\alpha + \frac{m\overline{Z_1}}{\phi}. \tag{5.100}$$

If the start up data stripes are already resident on disks, only the start up data stripe of the first segment needs to be loaded prior to usage. Since the disk bandwidth is

much higher than the display bandwidth, the start up data stripes of other segments can be loaded just prior to its usage. Hence, the start up latency is

$$\frac{\overline{Z_{i,1}}}{\beta}, \tag{5.101}$$

when the ith segment is consumed.

(c) User Start Up Latency

If the start up data stripes and low temporal resolution segments are not resident on disks, the system can only respond after they are downloaded from tertiary storage. Hence, the start up latency of a new stream, R, is

$$R = \omega + 2\alpha + \frac{m(Z_0 + \overline{Z_1})}{\phi}. \tag{5.102}$$

If the low temporal resolution segments are resident on disks but the start up data stripes are not resident on disks, the system can respond after the start up data stripe are downloaded. Hence, the start up latency of a new stream, R, is

$$R = \omega + \alpha + \frac{m\overline{Z_1}}{\phi}. \tag{5.103}$$

If the start up data stripes and low temporal resolution segments are resident on disks, the system responds to the new stream after the start up data stripe of the required segment is retrieved from disks. Hence, the start up latency of new stream, R, is

$$R = \frac{Z_{i,1}}{\beta}. \tag{5.104}$$

In a situation when $\frac{\beta}{\delta} - \left(\frac{\delta}{\phi}\right)^{n-1}$ is small and $\beta \geq \frac{(\omega+\alpha)\delta^2}{M}$, the size of the last data stripe of each segment is set to M. Since the size of the start up data stripe is bounded in Equation (5.87), the start up latency is bounded by

$$R \leq \frac{M}{\delta}. \tag{5.105}$$

Hence, the start up latency in this situation is bounded by the display time of only one media block.

5.5 Analysis of Large Multimedia Storage Systems

We present in this Section our analysis on the behaviour of large multimedia storage servers (LMSS) using our performance model. We assume that each multimedia object is a two-hour video of MPEG frames. Each frame contains 576 x 384 24-bit colour pixels being displayed at 15 frames per second. The videos are MPEG compressed at 30:1 ratio. Therefore, the size of each object is 2278 MB and the display bandwidth is 0.316 MB/sec. The tertiary drive parameters similar to the Ampex tape library are used [39]. Other parameters are listed in Table 5.4.

Table 5.4. List of parameter values in LMSS performance analysis

Parameters	Default Value
stream arrival rate	5 to 60 streams/hour
number of tertiary drives	3
mean media exchange time	55 seconds
reposition start up time	1 second
reposition rate	0.06 sec/inch
maximum reposition length	2000 inches
mean segment length	5 to 20 minutes
media units per drive	4
transfer rate	14.5 MB/sec
secondary bandwidth	20 MB/sec

5.5.1 Group Service Time

When more requests are served in each group, the total reposition time increases slightly and the data transfer time increases proportionately. Hence, the individual group service time increases proportionately with the number of concurrent streams (Figure 5.10).

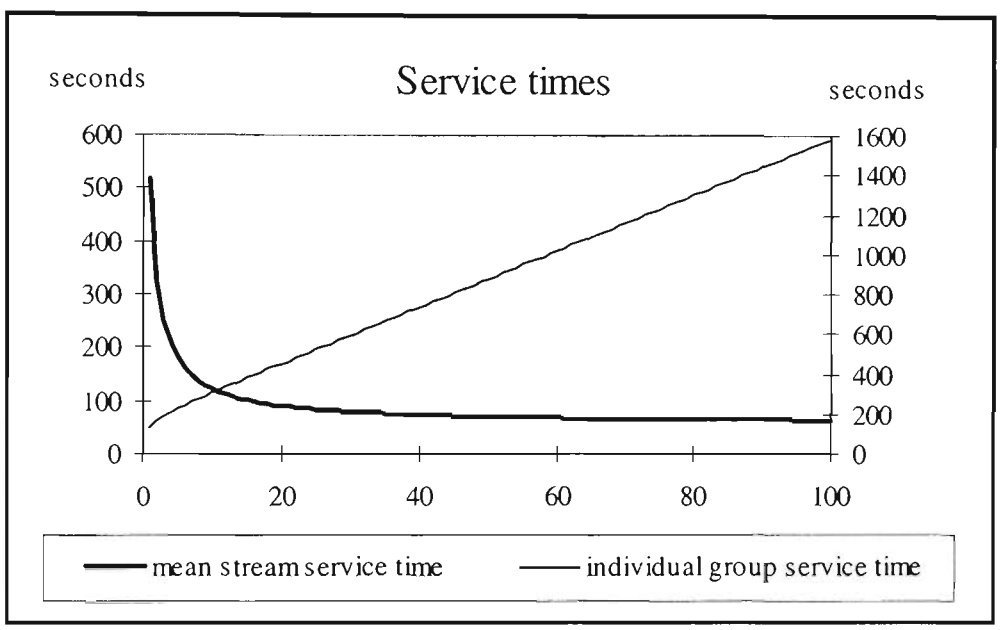


Figure 5.10. Individual group service time and mean stream service time vs number of concurrent streams

When more requests are served in each group, more streams would share the media exchange and reposition overheads. The overhead time per stream is inversely related to the number of concurrent streams. The data transfer time per stream is however unaffected. Therefore, the mean stream service time varies inversely to the number of concurrent streams. It levels down to the data transfer time when the concurrent streams is sufficiently many.

5.5.2 Concurrent Retrieving Streams

If only a few streams are accepted, then the system is under-utilized. If too many streams are accepted, starving would occur. Hence, the maximum number of acceptable streams should maintain below the limit according to the continuous display requirement in order that system is fully utilized without starving.

When longer segments are used, fewer requests are issued to retrieve each object. Less media exchange and reposition overheads are involved; resulting in more concurrent streams can be accepted (Figure 5.11).

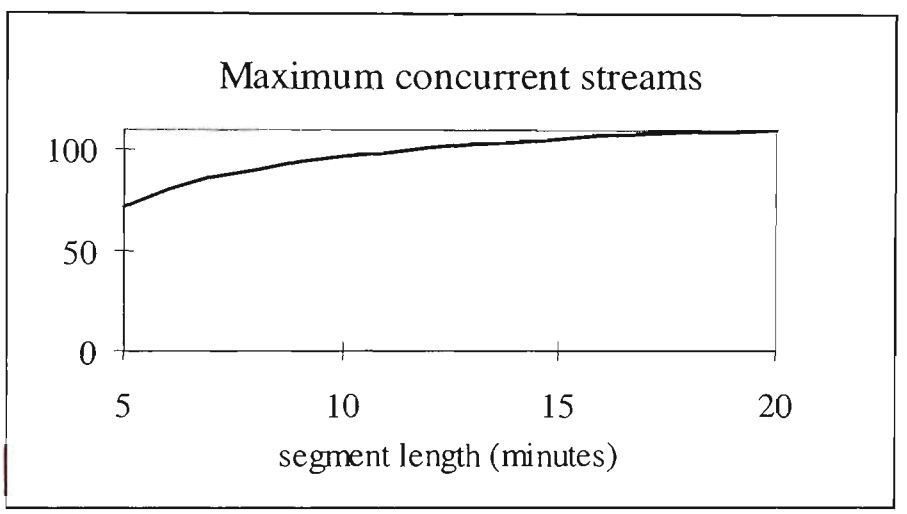


Figure 5.11. Maximum number of concurrent streams vs segment length

5.5.3 Maximum System Throughput

The maximum system throughput shows the ability in clearing requests from waiting queues. We can see from Figure 5.12 that the maximum throughput of the high concurrency placement is always higher than that of the parallel striping method. It is also significantly higher than the non-striping method by over 30% when data segments are longer than 3 minutes.

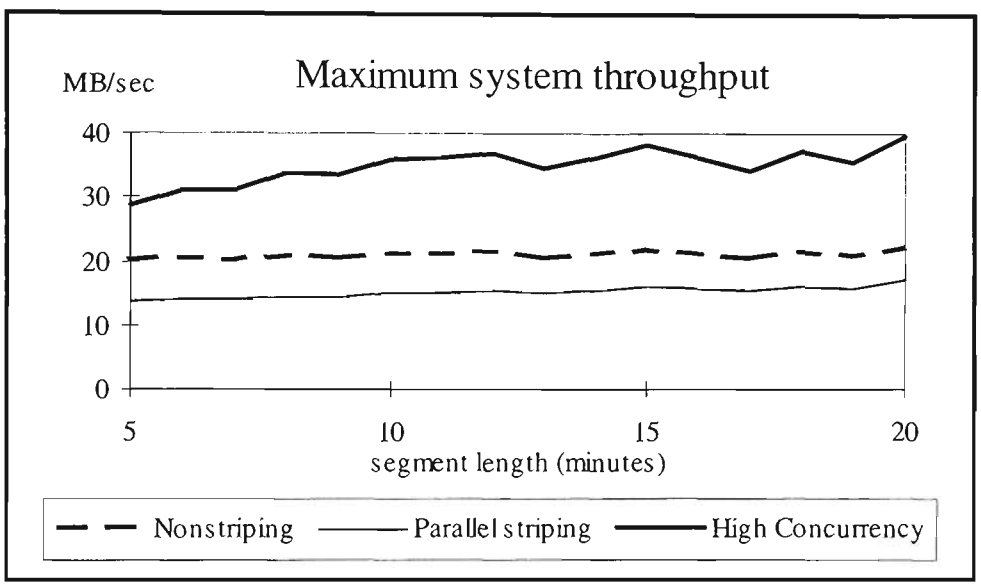


Figure 5.12. Maximum system throughput vs segment length

The system throughput of the methods increase when data segments are longer for two reasons: First, more data are retrieved per media exchange. Each stream performs less media exchanges and reposition. Thus, less overheads are involved. Second, longer data segments are displayed for a longer time, more streams can be served concurrently. More requests are then served per media exchange. Hence, the same media exchange overhead is shared among more requests. In addition, the full length of reposition is also shared in SCAN

scheduling among more requests. Therefore, the system throughput is raised by using longer data segments.

Since our high concurrency placement method achieves the highest system throughput, this makes our new method very suitable for scheduled streams and burst of streams. Under these stream arrival patterns, the system should serve streams quickly to reduce the waiting queues. Hence, system throughput is the major concern and our method is the best choice for systems that exhibits these characteristics.

5.5.4 Mean Stream Response Time

The stream response time indicates the quality of service to users. We model the stream response time as the earliest time that a stream can start to display using a pipelining method. The mean stream response time is shown against the stream arrival rate in Figure 5.13. The stream response time is the sum of the queue waiting time and the start up latency. It is dominated by the start up latency at low stream arrival rate, but it is dominated by the queue waiting time at high stream arrival rate.

For parallel striping method, the response time is short when the arrival rate is very low. Above this arrival rate, the media exchange device becomes highly utilized and each request experiences very long exchange waiting time. This exchange waiting time abruptly raises the stream service time and the system utilization. Hence, the drives suddenly become fully utilized with most time spent on waiting for the exchange device. Therefore, the stream response time increase indefinitely due to a long waiting queue.

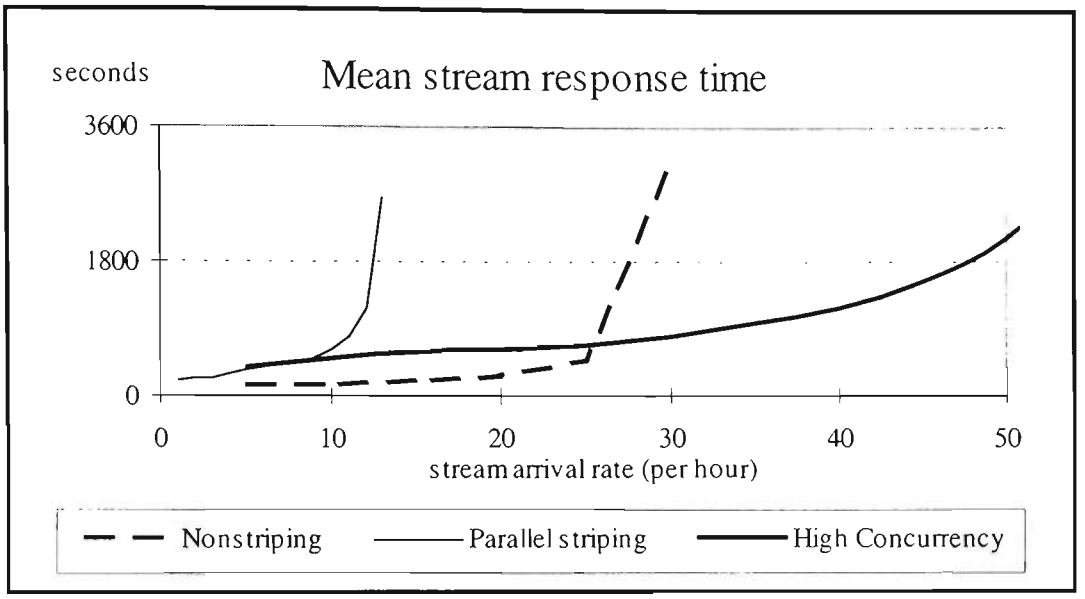


Figure 5.13. Mean stream response time vs stream arrival rate

At arrival rate under 30 streams per hour, the non-striping method responds faster than the high concurrency placement method. This is because our method has a longer start up latency. Hence, it responds slowly under a light load environment.

At higher stream arrival rate, the high concurrency placement responds faster than other methods. As the queue of the non-striping method grows, the response time increases rapidly. Since the high concurrency placement method achieves higher system throughput, it serves streams faster. Therefore, stream response time under heavy load is reduced using this new method.

Since the drives serve media units in rounds in high concurrency placement, the number of media units has an influence on the mean stream response time. This is because new streams would experience a longer response time for more groups per round (Figure 5.14).

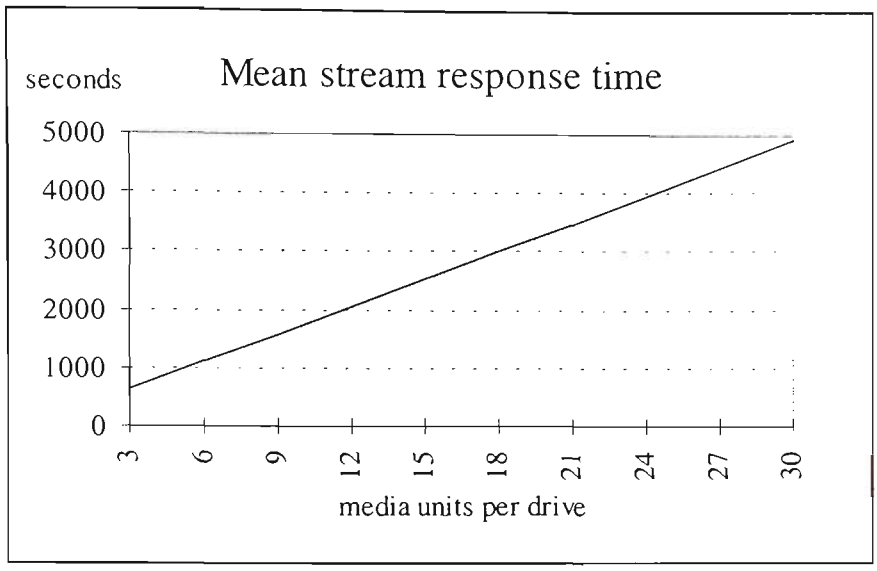


Figure 5.14. Mean stream response time vs media units per drive

5.5.5 Buffer Size

The buffer size provides an indication of the amount of resources required by each method. We can see from Figure 5.15 that all methods use the similar buffer size when the stream arrival rate is very low. Nearly the entire object is kept in staging buffer for display. The buffer sizes of both traditional methods are not affected by the stream arrival rate.

The situation is different for the high concurrency placement method. When more streams are concurrent with faster stream arrival rate, the individual group service time increases (as shown in Figure 5.10), but the same amount of data are retrieved per stream in each group. Hence, data are retrieved slowly for each stream. Even though the system data rate is raised, the data rate per stream is reduced; resulting in smaller buffer size per stream. Since the total buffer size is the sum of buffer size for all concurrent streams, the total buffer space requirement can hence be constrained by a smaller total buffer size. Therefore, our new method requires less total buffer space than other existing methods.

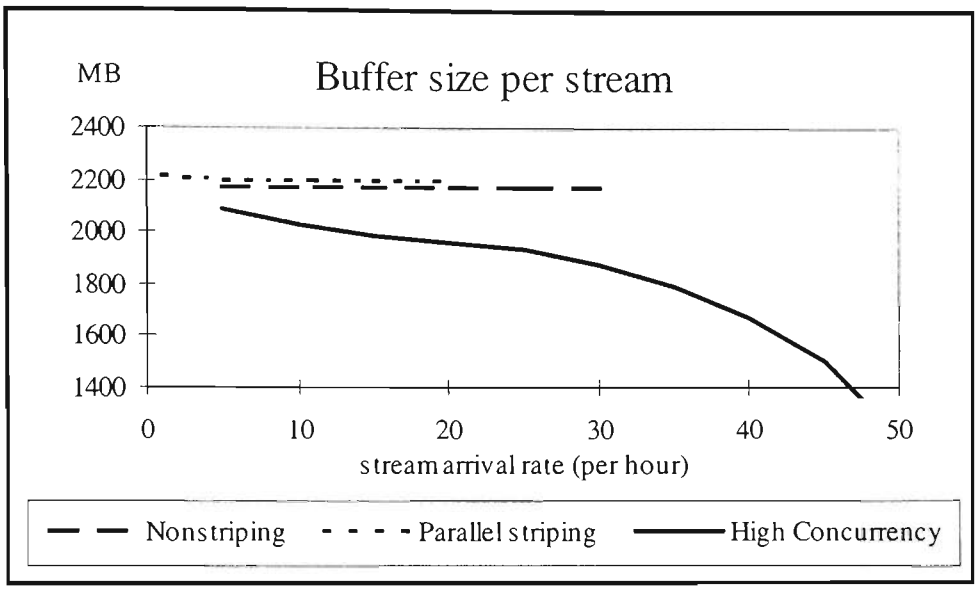


Figure 5.15. Buffer size per stream vs stream arrival rate

5.6 Performance of Data Migration Methods

We analyse the performance of various data migration methods in this Section. The performance of our segment based pipelining method is compared with the performance of simple pipelining methods and Two Phase Service Model (2PSM) which are described in Section 2.6.3. We use the parameters in Table 5.5 to study the performance of data migration methods using the performance model in Section 5.4. We varied only one parameter in each case using values in the range column; other parameters will use the default value unless otherwise stated.

Table 5.5. Parameters values in data migration analysis

Parameter	Range	Default value
Tertiary bandwidth	0.4 to 1.30 MB/s	0.9 MB/s. This is the bandwidth of retrieving one media block consecutively from a 24X CD-ROM drive. The tertiary bandwidth can be very low when repositioning is required in tapes or it may be higher when an array of optical drives is used.
Object size	20 to 200 minutes	two hours of video at 30 frames/second.
Disk bandwidth	20 to 110 MB/s	80 MB/s. This is the aggregate bandwidth of about 8 current disks.
Display bandwidth	1.35 to 2.25 MB/s	1.35 MB/s. This is the data bandwidth of MPEG-2 compressed SVGA video.
Media exchange overheads	5 to 15 seconds	10 seconds. This is the media exchange time of optical jukeboxes. Automatic CD changers can exchange media faster at around 5 seconds.
Media block size	Not varied	30 SVGA frames compressed at the ratio of 50 to 1.

We first analyse the disk space and user latency in Section 5.6.1. The number and size of data stripes and segments are then investigated in Section 5.6.2. After that, the influence of object size on the user latency is described in Section 5.6.3. Lastly, the impact of disk bandwidth and display bandwidth are analysed in Section 5.6.4 and Section 5.6.5 respectively.

5.6.1 Disk Space and User Latency

When low temporal resolution segments and start up data stripes are kept resident in disks, the amount of disk space required in our method is compared with the size of the first slice in the pipeline method (Figure 5.16). In general, the amount of necessary disk space drops linearly when the tertiary bandwidth increases. Our method uses slightly extra disk space to cater for the media exchange time that is not considered by other studies. This media exchange time is necessary when an object spans across several media units.

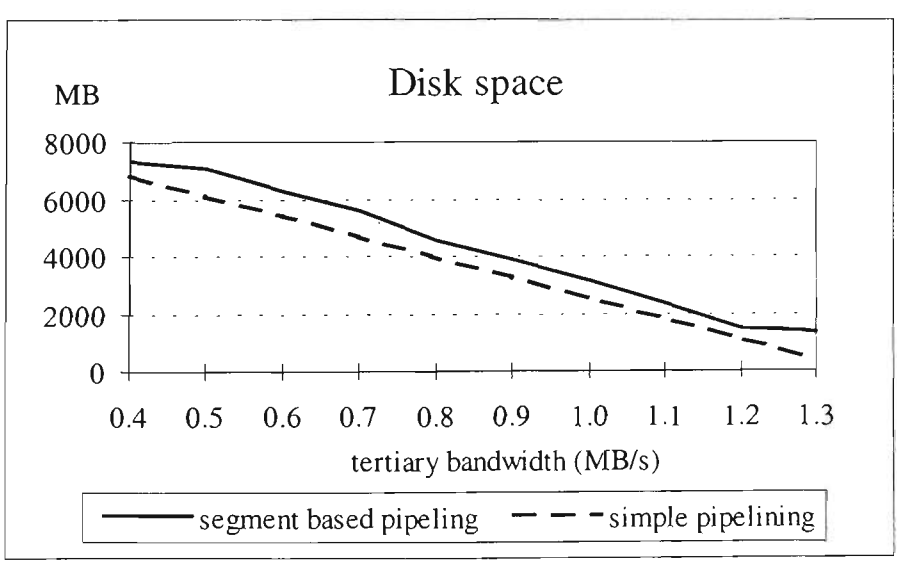


Figure 5.16. Disk space required vs tertiary bandwidth

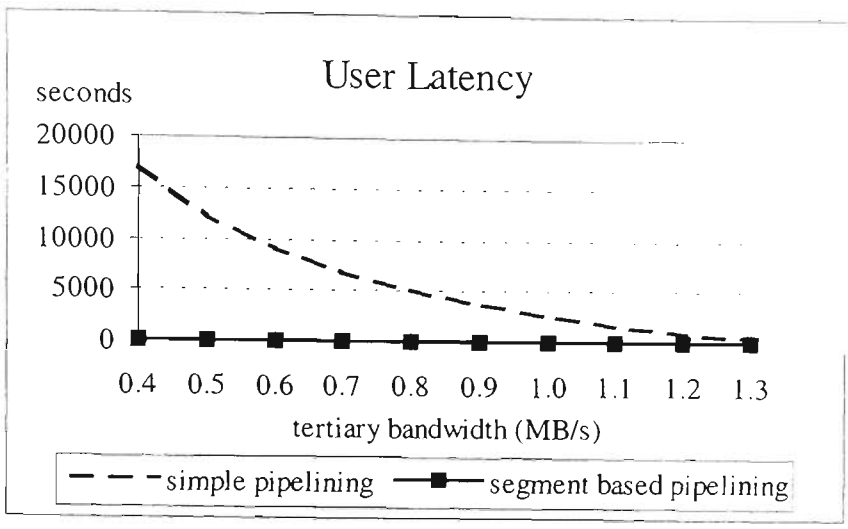


Figure 5.17. User latency vs tertiary bandwidth

When user requests for an object, start up latency is involved in the pipelining methods to load the start up data stripe or the first slice from tertiary storage. This start up latency is minimized in our method by keeping all the start up data stripes resident on disks (Figure 5.17).

Since the first slices in the pipelining method is comparable to the start up data stripes in the segment based pipelining, they can be loaded in roughly the same time using consecutive requests. As start up latency that significantly affects the response time under light loads is a key concern of users, it is a worthwhile tradeoff to reduce user latency with some disk space.

When user begins normal display after previewing, the system may need to retrieve some data before it can resume display. This reposition latency is compared with the fast forward resumption latency of 2PSM in Figure 5.18. It can be seen that our method responds significantly faster. It can usually respond in one second whereas the 2PSM takes much longer to respond. This is important to achieve smooth transition back to normal display. When tertiary bandwidth is almost the same as the display bandwidth, the 2PSM requires over 30 seconds to

resume normal display (not shown). Our method is also more tolerant to changes in tertiary bandwidth.

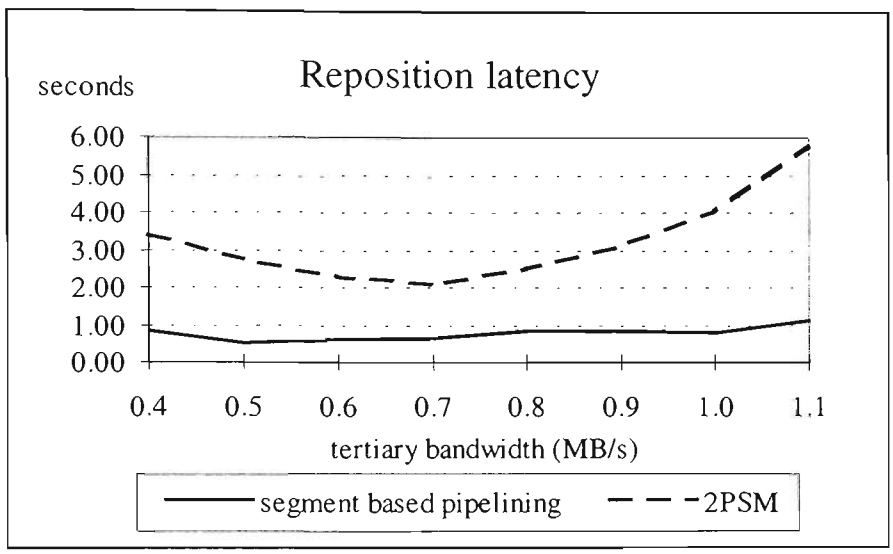


Figure 5.18. Reposition latency vs tertiary bandwidth

5.6.2 Data Stripes and Segments

When the tertiary bandwidth is low, only a few data stripes are created in each segment (Figure 5.19), and the size of the start up data stripe is small. When the tertiary bandwidth is very close to the display bandwidth, many data stripes exist in each segment. The start up data stripe becomes very large and the pipelining method is efficient at the expense of much less preview data and longer reposition latency.

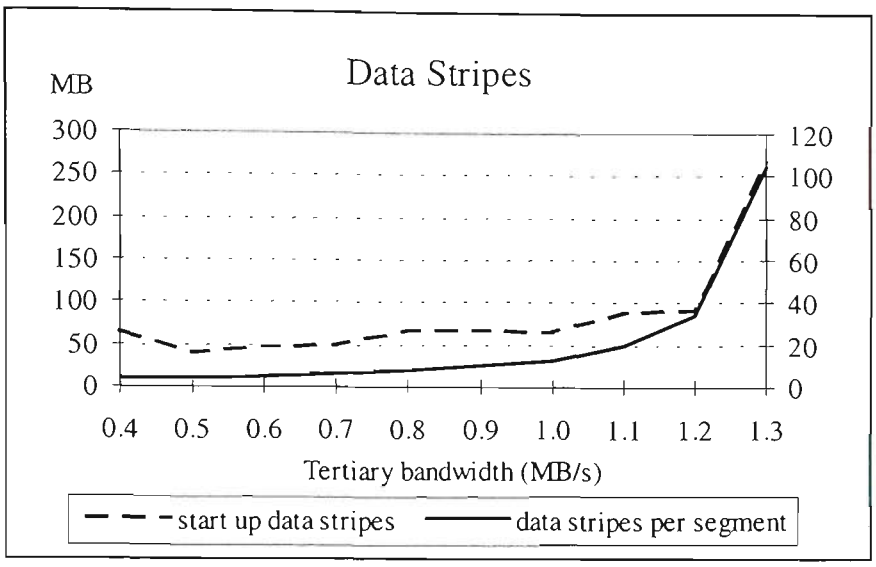


Figure 5.19. Start up data stripe size and data stripes per segment vs tertiary bandwidth

When the tertiary bandwidth is low, a large number of small segments are created (Figure 5.20). When the tertiary bandwidth is high and close to the display bandwidth, the segment size increases rapidly and the number of logical segments hence decreases. The number of segments also affects the amount of preview data during fast forward display.

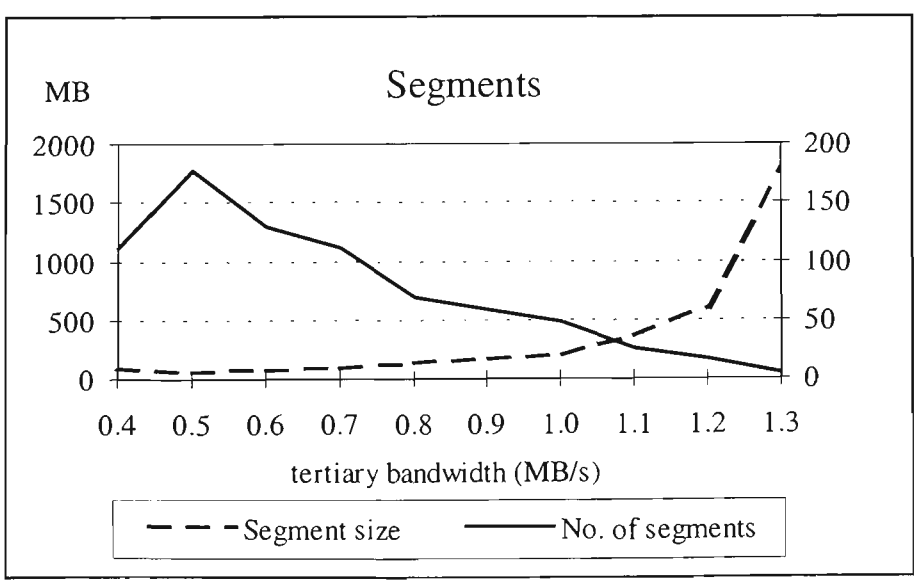


Figure 5.20. Segment size and number of segments vs tertiary bandwidth

5.6.3 Object Size

Large and medium sized multimedia objects may both be stored in the same storage system. User latency of the pipelining method increases linearly but our method responds consistently to requests for objects of various sizes (Figure 5.21). This is because we keep sufficient data on disks to enable normal display to start before any data are actually retrieved on demand from tertiary storage.

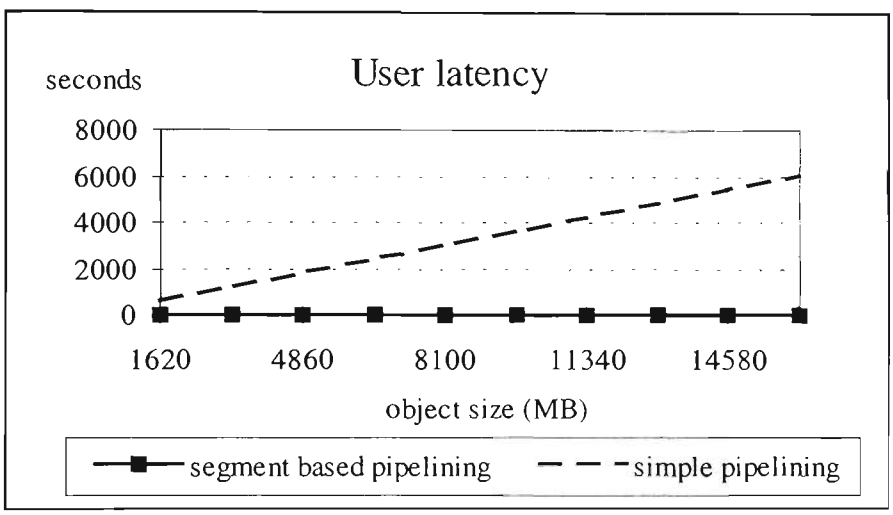


Figure 5.21. User latency vs object size

5.6.4 Disk Bandwidth

When the disk bandwidth is insufficient (Figure 5.22), the retrieval time of the start up data stripes limits the number of data stripes in each segment. Extra disk space is then required to store more start up data stripes. When the disk bandwidth is sufficient, the start up data stripes are smaller in size. It is therefore desirable to use multiple disks to provide sufficient disk bandwidth as well as disk space. The disk bandwidth only affects the reposition latency slightly (Figure 5.23) because we create bigger start up data stripes when the disk bandwidth is higher.

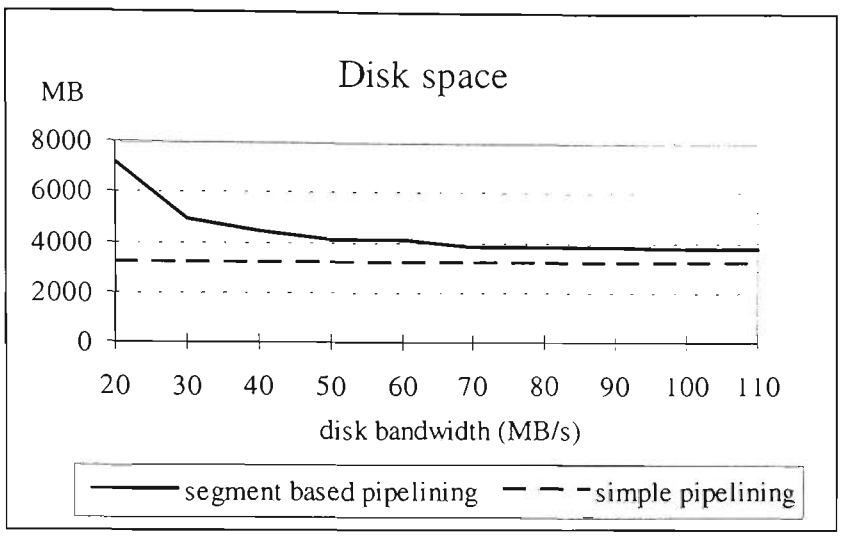


Figure 5.22. Disk space required vs disk bandwidth

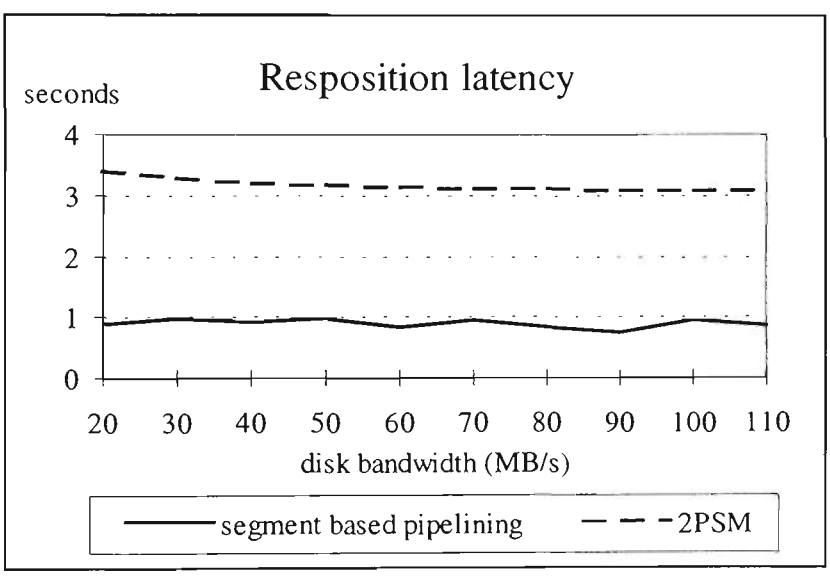


Figure 5.23. Reposition latency vs disk bandwidth

5.6.5 Display Bandwidth

When higher quality multimedia data are retrieved, the display bandwidth increases. Shorter segments should be created and more disk space is required. Although the pipeline efficiency drops, the reposition latency is also reduced (Figure 5.24). Our method also responds consistently faster than the 2PSM.

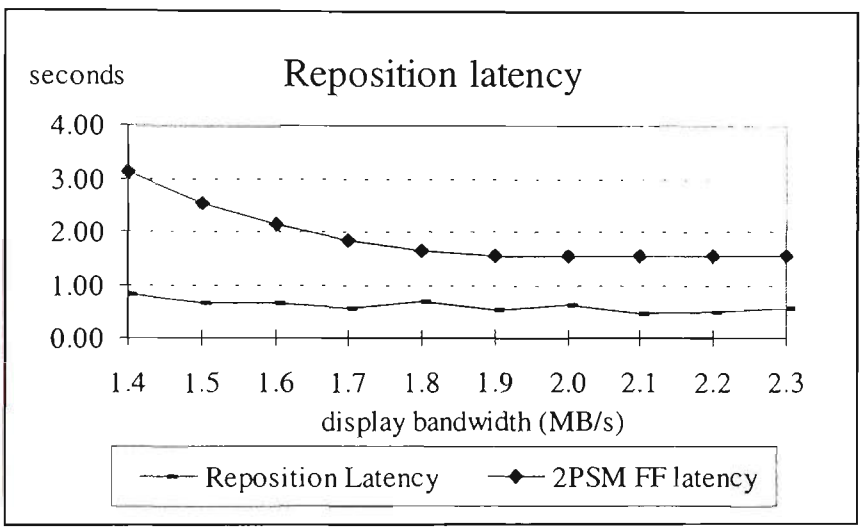


Figure 5.24. Reposition latency vs display bandwidth

5.7 Large Multimedia Storage Server Simulations

A storage server program that measures the performance of multimedia streams of requests is created. Each stream retrieves a multimedia object that is stored using the parallel striping, non-striping, or high concurrency placement method. The simulation program runs on a Pentium personal computer to generate the multimedia data request streams. No other simulation tools are used.

The non-striping method stores each object on only one media unit. Hence, only one media exchange is normally required to retrieve an object. Since each segment may reside at a different location on the media unit, the reposition time is shared according to the SCAN scheduling algorithm.

The parallel striping method stores segments on a few media units. We use the stripe width same as the number of drives. Since the drives serve one stream in parallel, one media exchange is performed on every drive. The reposition times are also shared among requests on the same object.

The simulation program accepts values in specifying the drive parameters. The drive parameters includes the number of drives, the mean and variation of media exchange time, the reposition start up time, the reposition speed, the tertiary transfer rate, the full length of each media unit, and the number of media units per drive. The disk bandwidth can also be specified.

The multimedia objects are specified using the number of horizontal pixels, the number of vertical pixels, the bytes per pixel, the frame rate, the compression ratio, the display duration, the media block size, and segment length. The display bandwidth and object size are calculated accordingly.

A number of streams can be generated randomly at a specified stream rate. New streams are inserted to an event list. The program keeps track of the simulation time. When the simulation time reaches the event time, new stream is inserted to the stream waiting queue.

Streams in the waiting queue are accepted when they can be served without violating the continuous display requirement. Accepted streams are placed in the list of concurrent streams. Each accepted stream then sends two initial requests to each simulated drive to retrieve data.

Each drive keeps track of its current status which is either free, exchanging, repositioning, or transferring data. Each drive keeps requests in two queues. The first queue keeps the requests for data on the current media unit. The second queue keeps requests for data on other media units. After requests for data on the current media unit are served, requests are selected to move from the second queue to the first one. The waiting time is measured as the time in queue before a request is served.

A media exchange request is generated if a request needs a media unit different from the current one. This exchange request is then placed in the exchange request queue. The exchange device serves requests from this exchange request queue. Requests on the drive that has the least number of served requests are chosen to be served prior to other requests. The exchange service time is randomly generated according to a uniform distribution within ±10% deviations from the mean exchange time.

Data locations of requests are generated randomly within the total media length. The reposition time is then calculated as a function of the distance being

travelled. The segment length is generated randomly according to a uniform distribution within ±10% deviations from the mean segment length. The data transfer time is calculated as the segment length divided by the data transfer rate.

After a request is served at a drive, the stream sends the next requests to the same drive. The stream starts to display the first segment of the object after the first cycle of request is served by all the drives. It then waits for the display period of the segment. If the request for the next segment has returned before the display time expires, it then displays the next segment. Otherwise, the stream starves. After all segments are displayed, the stream exits the system.

The program measures all the time components of requests and streams and stores them in a relational database. The number of starving requests in each stream, the mean stream response time, the mean and maximum system throughput, and maximum buffer size are also summarized and stored.

5.7.1 Simulation Parameters

We have performed simulations using the drive characteristics similar to the Ampex robotic tape library in Table 5.6 [39]. We randomly generate the streams at specified stream arrival rate and measure their performance.

Table 5.6. Parameters for multimedia storage server simulations

Parameters	Default value
Number of simulated streams	200
Stream arrival rate	5 to 60 streams/hour
Number of tertiary drives	3
Mean media exchange time	55 seconds
Reposition start up time	1 second
Reposition rate	0.06 sec/inch
Maximum reposition length	2000 inches
Segment length	10 minutes
Media units per drive	4
Transfer rate	14.5 MB/sec
Secondary bandwidth	20 MB/sec

5.7.2 Number of Concurrent Streams

We have measured the number of starving requests by varying the maximum number of concurrent streams as shown in Figure 5.25. The starving requests were measured for one to four drives. If the maximum number of concurrent streams is maintained below the limit being imposed by the continuous display requirement, starving rarely occurs. Otherwise, some streams would starve and the number of starving requests accelerates rapidly. This is in agreement with the maximum number of acceptable concurrent streams predicted in our model.

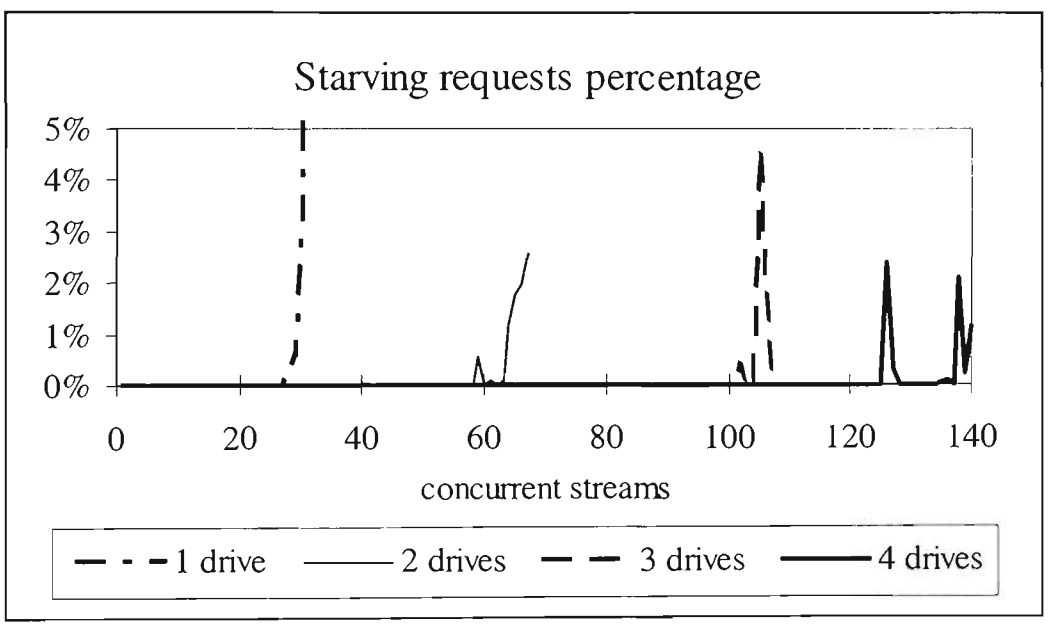


Figure 5.25. Starving requests percentage vs number of accepted concurrent streams for 1 to 4 drives

5.7.3 Maximum System Throughput

We have generated the streams at 100 streams per hour, and we have measured the maximum system throughput being achieved during the simulations. The maximum system throughput being measured is plotted on the curve as shown in Figure 5.26. The measured values are also listed in Table A.8 in Appendix A. We have observed that all the measured values are close to the predicted curves. This verifies that our performance model is sufficiently accurate in predicting the maximum system throughput for all three storage methods.

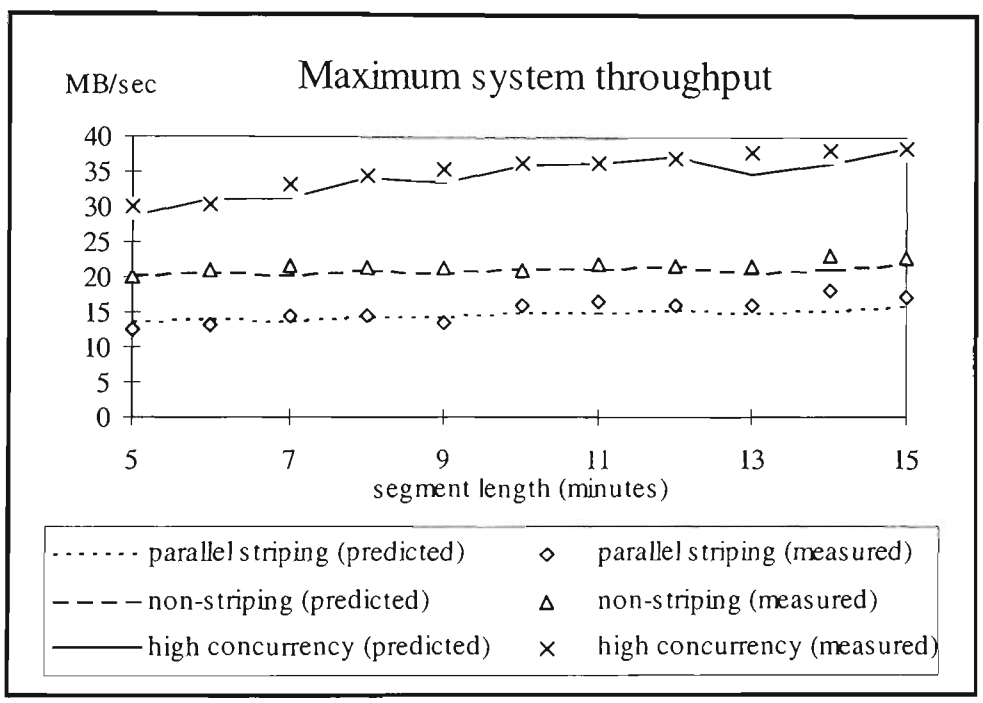


Figure 5.26. Measured maximum system throughput

We have observed that the maximum system throughput of the parallel striping method is lower than that of the non-striping method which is also lower than that of the high concurrency placement method. It is clear that the high concurrency placement method achieves higher maximum system throughput than other existing methods. This confirms that our new method is the best storage organization for multimedia storage system that requires high system throughput.

5.7.4 Mean Stream Response Time

We have generated the streams for 10 minute segments at 5 to 60 streams per hour, and we have measured the mean response time of 2000 streams for the non-striping method and the high concurrency placement method. We have simulated the streams at 1 to 15 streams per hour on the parallel striping method since its maximum system throughput is low. We assume that the simple pipelining method is employed on the two traditional methods to start new streams as early as possible. The mean stream response time being measured are plotted on the predicted curve in Figure 5.27. The measured values are also listed in Table A.9.

We have observed that most of the measured values fall on the predicted curves. Since an unstable queue can amplify to a large variation in waiting time when the utilization value is large, the error rate is expected to be high when the system is highly utilized. We focus on stable queues when the utilization values are up to around 80%. The measurements verify that our model is reasonably accurate in predicting the mean stream response time.

The measured mean stream response time also shows that the high concurrency placement method performs best under heavy load. Intuitively, this is because the high concurrency placement method achieves higher system throughput than other methods especially when large number of concurrent streams are present.

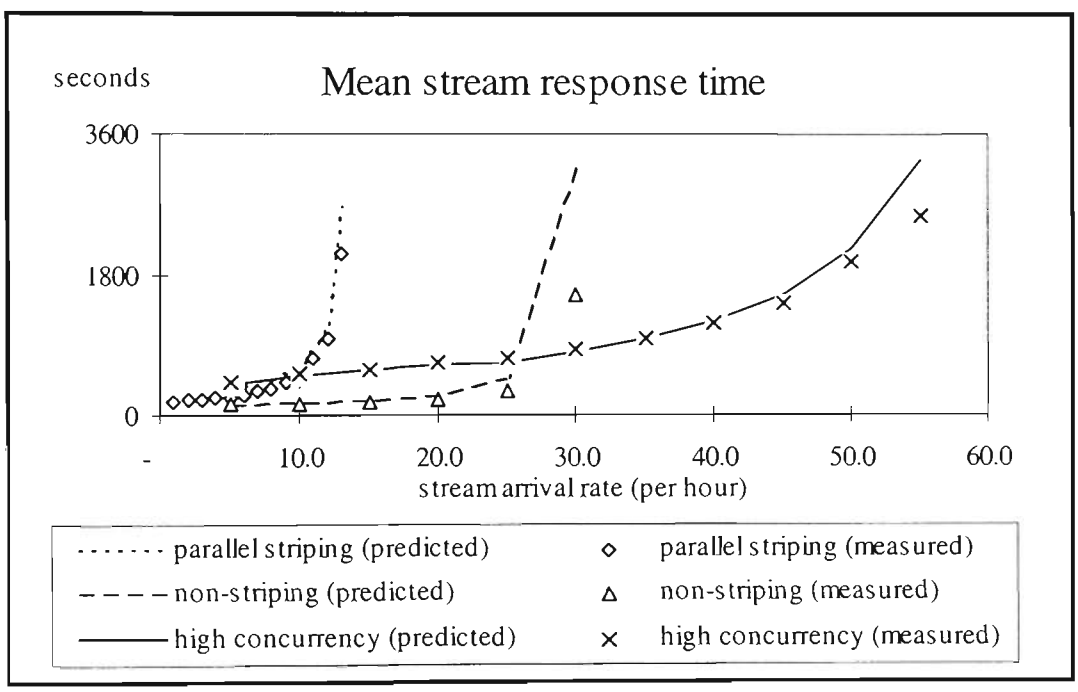


Figure 5.27. Mean stream response time

5.7.5 Buffer Size Per Stream

We have also measured the buffer size on the simulations for 10 minute segments at 5 to 60 streams per hour. During these simulations, we measure the maximum number of segments in each stream that are retrieved but not yet displayed. These data segments must be kept in buffers which is either in memory or on disks. We plot in Figure 5.28 the buffer size being measured in simulations. The predicted and measured values are also listed in Table A.10 in Appendix A.

The measured values of the non-striping method and parallel striping method are consistently within 10% below our predictions. The measured buffer size per stream of the high concurrency placement method is close to the predicted curve when the streams arrival rate is low. When the stream arrival rate is high, the measured buffer size per stream is higher than our prediction. The measurements verify that the buffer sizes per stream being predicted in our model are reasonably accurate for the traditional methods and it is also accurate for the new method when the arrival rate is low.

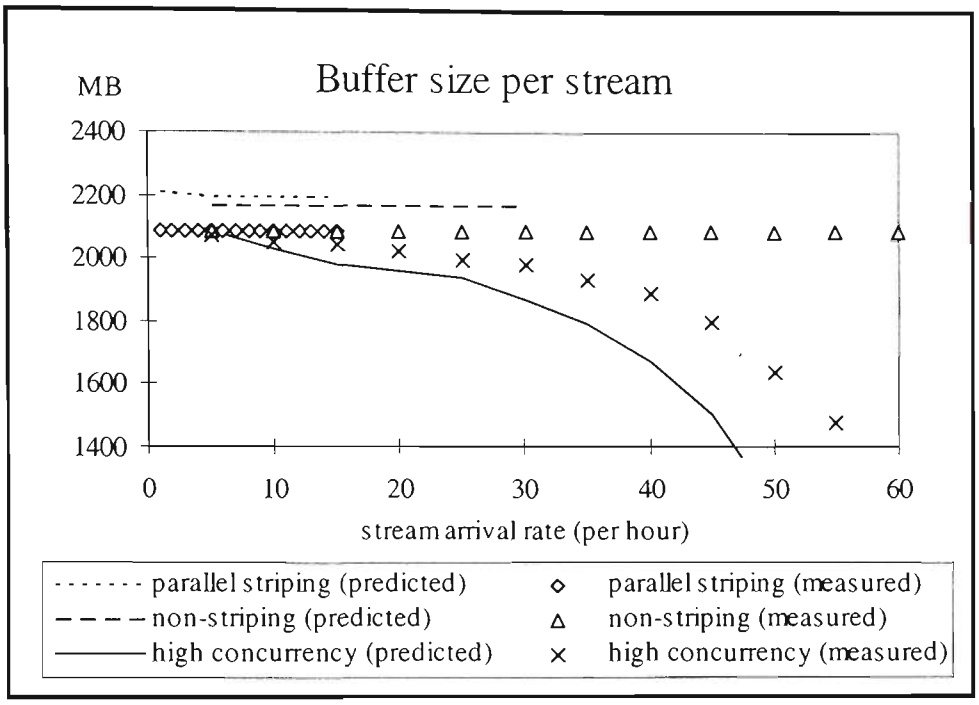


Figure 5.28. Measured buffer size per stream

We have also observed that the measured buffer size per stream of both traditional methods is unaltered by variations in the stream arrival rates. This confirms that the same buffer size is required for traditional methods under various stream arrival rates. Nevertheless, the measured buffer size per stream agrees with our prediction that the buffer size of the high concurrency placement method reduces gradually when the stream arrive rate increases. Hence, the buffer size can significantly be smaller when the stream arrival rate is high. These simulation results confirm that smaller buffer size is required when more streams are concurrently served in the high concurrency placement method than other traditional methods.

5.8 Chapter Summary

We have established the feasibility condition for accepting concurrent multimedia heterogeneous streams on multiple servers, which is applicable to general storage organizations and arbitrary scheduling policies. This feasibility condition is applicable to different storage systems in their optimization of the performance of multimedia streams.

We have developed a mathematical model based on a feedback queueing network to analyze the performance of large multimedia storage system, where request streams are represented as feedback queues to the tertiary drives. Using this model, closed form solutions are obtained for various components of the stream service time. System throughput, mean stream response time, and staging buffer size of various tertiary storage organizations are analysed and compared quantitatively.

We have compared the performance of large multimedia storage servers using various tertiary storage organizations. It is found that the high concurrency placement method is able to achieve the highest system throughput per stream among all methods. The stream response time can also be maintained at a low level under heavy loads. The buffer size per stream reduces gradually and is the smallest among all methods.

We have also presented a performance model for the segment based pipelining method, and have established the conditions for which continuous display can be guaranteed. We have optimized the data stripe sizes to achieve the shortest user latency while maintaining the most preview data. Closed form

solutions are obtained for the maximum number of data stripes per segment, minimum segment size, and the necessary amount of resident disk space.

We have compared the start up latency in different data migration methods, and have found that the segment based pipelining method responds quickly to browsing, fast forward, fast reverse functions faster than other pipelining methods. We have also quantitatively analysed the storage system behaviour against changes in tertiary bandwidth, secondary bandwidth, display bandwidth, and object size.

We have also presented our simulations on large multimedia storage servers. The measurements agree with our calculated limits on the maximum number of concurrent streams. The simulation results have verified our model in predicting the performance behaviour of the high concurrency placement method. The simulation results confirm that our high concurrency placement is superior to other existing methods in terms of the maximum system throughput, response time under heavy loads, and necessary buffer size.

Chapter 6

Conclusion

6.1 Summary of Research Results

We have presented novel and efficient storage and retrieval methods for multimedia objects, and we have used a two level data striping method to partition multimedia object prior to storage. We have designed the high concurrency placement and the segment based pipelining method to store and retrieve multimedia data on tertiary storage. The bandwidth based placement method is designed to efficiently store heterogeneous multimedia data on CDR disks. We have enhanced the pipelining method to migrate multimedia data from tertiary storage. The performance of CDR disks and large multimedia storage systems using these novel storage organizations have been analysed, and simulations have been performed to test and validate our results. The specific research results are summarized as follows.

- **New High Concurrency Placement**

The use of tertiary storage devices for large multimedia databases will be inevitable in future systems. The main concerns in using these systems is their relatively poor response characteristics and large resource consumption in terms of buffer requirements. We address these problems by making use of a novel high concurrency placement method to efficiently serve concurrent multimedia streams on tertiary storage systems.

A new storage organization for the efficient execution of multimedia data requests is designed. This high concurrency placement method has several advantages. The first advantage is that it increases the system throughput for storing data to and retrieving data from tertiary storage. This system throughput can be adjusted by varying the segment length according to the system requirements. This method is hence very suitable for scheduled requests of which system throughput is the primary concern.

The second advantage is that new streams can respond much faster under heavy load conditions which are very often the practical concern in multimedia databases. This new method is hence very suitable for multimedia databases where bursts of streams are expected to respond in the shortest possible time.

The third advantage is that smaller staging buffers are used. When the system is bounded by the staging buffer size, this method can serve more streams than other methods while utilizing the same amount of

buffers. Our new method hence uses less resource than traditional methods, making it suitable for systems that are hungry on staging space.

These advantages make the high concurrency placement method the most efficient storage organization for multimedia data storage on tertiary storage systems.

Since we have enhanced the tertiary storage system performance by the efficient high concurrency placement method, the tertiary storage bandwidth cost is hence lowered. Therefore, large multimedia systems become more economic and practical than before.

We have presented our model for the performance of large multimedia storage system using a feedback queueing network. We then use this model to analyse and compare the system performance using various tertiary storage organizations.

We have analysed the performance of high concurrency placement method, non-striping method, and parallel striping method. We have quantitatively compared their maximum system throughput, mean stream response time, and buffer size per stream. We have shown that the high concurrency placement method achieves the highest system throughput without violating the continuous display requirement. It also achieves the shortest response time under heavy loads while utilizing the smallest buffer space.

- **New Segment Based Pipelining**

We have also presented a novel data migration method to retrieve multimedia data from our hierarchical storage systems. This hierarchical storage organization is able to respond to multimedia data access at disk latency. Although tertiary storage devices are used to store data, this becomes transparent to users in terms of access latency. Hence, multimedia data can be stored on the more economical tertiary storage devices without much performance drawbacks.

We have compared the start up latency in various data migration methods. We have shown that the segment based pipelining method can respond quickly to browsing, fast forward, fast reverse operations faster than other pipelining methods.

Our performance analysis enables us to determine the conditions under which continuous display can be guaranteed. Our model is optimized to achieve the minimum resident disk space, the shortest user latency, and the most amount of preview data. Closed form solutions are obtained for the maximum number of data stripes per segment, the minimum segment size, and the necessary amount of resident disk space.

We have also analysed the hierarchical storage system performance against changes in tertiary bandwidth, secondary bandwidth, display bandwidth, and object size. These results allow valuable insights to be gained on multimedia database operations and permit the tuning and optimization of performance parameters in different situations.

- **New Bandwidth Based Placement**

We have also presented a novel bandwidth based placement method to store heterogeneous multimedia objects on constant density recording disk storage structures. Bandwidth based placement has the advantages of maintaining the balance between the space and bandwidth of disk zones to secure high bandwidth zones for high bandwidth objects. High bandwidth data can enjoy higher throughput of the outer zone groups. Low bandwidth data can still enjoy the same I/O rate in the inner zone groups.

We have also analysed the data transfer time of our bandwidth based placement method. We have then studied the impact on system throughput of various zone groups when our placement method is used. We have shown that the bandwidth based placement performs better than the random placement for multimedia data requests on constant density recording disks. Hence, we have increased the disk throughput for multimedia data requests.

- **Performance Analysis of Constant Density Recording Structure**

We have presented a detailed mathematical model of the performance of CDR disks. In this model, we have obtained closed form solutions to data access time on CDR disk. By comparing quantitatively the performance between CDR disk and VDR disk, we have shown that CDR disks always perform significantly better than VDR disks with similar parameters. Not only can we have more bytes per inch by using constant density recording technique, we also have more bytes per second.

We have used this model to study the impact of various disk parameters on disk performance. We have found that the minimum block size that can deliver necessary throughput should be chosen to reduce memory buffer and start up latency. Data stream size may be reduced through segmentation methods to an extent that maximum disk throughput is maintained.

We have also shown that CDR disks provide more significant performance advantage particularly on large multimedia data access compared to small traditional data access. For individual data access, over 60% of data access time is reduced. For streams of continuous media requests, CDR disk can significantly increase the number of acceptable streams, reduce response time, and reduce queue length.

A key obstacle in the building of large scale multimedia information systems is the relative inefficiency of the storage system in managing and retrieving multimedia data. In this thesis, we have discovered new techniques in building efficient storage systems for managing multimedia information. Our research results show that multimedia storage systems become more efficient in serving multimedia requests by using either high concurrent placement method or segment based pipelining together with bandwidth based placement on constant density recording storage structure.

6.2 Extensions and Future Research

Many theoretical and practical researches are possible in the future. Some of these are listed below.

- *Storage organizations*. Bandwidth based placement can be extended to place interleaving data stripes on constant density recording disks. The stationary and transition probability of objects can also be considered in the placement of data stripes to optimize the overheads in data retrieval. Optimal disk size can be found for CDR disks to achieve the best throughput.
- *Tertiary storage organizations*. The ordering of media units can be optimized to reduce the media exchange service time. The object bandwidth, transition probability, and interleaving gaps may be considered in the placement of objects on the media units. Parallel staggered striping can be investigated for multiple segments on each media unit.

- *Start up latency*. Since new streams are accepted only when the maximum number of concurrent streams is not reached, the first few data stripes are stored contiguously and retrieved consecutively with only one disk seek action to quickly fill the read-ahead buffers. Methods to efficiently utilize the gap time to increase the amount of data retrieved in each round in filling up the read-ahead buffers will likely to bring benefits.
- *Variations in buffer replacement policies*. The choice of buffer replacement policy would affect the performance of VCR type operations, such as fast forward, fast backward, jump to a new position. It would be interesting to investigate the impact of buffer replacement policies on these operations and choose the most suitable buffer replacement policy.
- *Efficiency and reliability in objects modification*. When multimedia objects are being modified, the efficiency in maintaining the hierarchical storage organization. Also, there would be a need for reliable operations. These can be investigated in future researches.

Bibliography

1. C. Aggarwal, J. Wolf, and P. S. Yu, "On Optimal Piggyback Merging Policies for Video-On-Demand Systems", *Proceedings of the ACM SIGMETRICS Conference*, pages 200-209, 1996.

2. D. P. Anderson, Y. Osawa, and R. Govindan, "A File System for Continuous Media", *ACM Transactions on Computer Systems*, volume 10, number 4, pages311-337, November 1992.

3. P. C. Arnett and D. Lam, "Software Channel Approach to Data Binding", *IEEE Transactions on Magnetics*, volume 26, number 5, pages 2324-2326, September 1990.

4. B. Beizer, *Micro-Analysis of Computer System Performance*, Van Nostrand Reinhold Company, 1978.

5. S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju, "Staggered Striping in Multimedia Information Systems", *Proceedings of the ACM SIGMOD Conference*, pages 79-90, May 1994.

6. S. Berson and R. Muntz, "Just-in-time Scheduling for Video-on-Demand Storage Servers", Technical Report, University of California, Los Angeles, 1995.

7. Y. Birk, "Deterministic Load-Balancing Schemes for Disk-Based Video-On-Demand Storage Servers", *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 17-22, 1995.

8. J. Boulos and K. Ono, "VOD Data Storage in Multimedia Environments", *Special Issue on Multimedia Communications in Heterogeneous Network Environments, IEICE Trans. Commun.*, volume E81-B, number 8, pages 1656-1665, August 1998.

9. J. Brown, "The US Army IETM One year Later", *Proceedings of IEEE Systems Readiness Technology Conference AUTOTESTCON'94*, pages 157-165, 1994.

10. D.W. Brubeck and L.A. Rowe, "Hierarchical storage management in a distributed VOD system", *IEEE Multimedia*, volume 3, number 3, pages 37-47, 1996.

11. N. Bui, M. K. Sundareshan, and H. S. Tharp, "Seek Reliability Enhancement in Optical and Magneto-Optical Disk Data Storage Devices", *IEEE Transactions on Magnetics*, volume 29, number 6, pages 3802-3804, November 1993.

12. M. J. Carey, L. M. Haas, and M. Livny, "Tapes Hold Data, Too: Challenges of Tuples on Tertiary Store", *Proceedings of the ACM SIGMOD Conference*, pages 413-417, 1993.

13. R. Chambers and M. Davis, "Petabyte class storage at Jefferson Lab (CEBAF)", *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 77-90, September 1996.

14. K. Y. Chan, K. K. Lee, P. C. Leong, N. Sriskanthan, and S. C. Hui, "ParaServe: A Parallel Mass Storage Server", *Proceedings of 1994 IEEE Region 10's Ninth Annual International Conference*, volume 1, pages 486-490, 1994.

15. S. H. G. Chan and F. A. Togabi, "Hierarchical storage systems for on-demand video servers", *Proceedings of SPIE*, volume 2604, *High Density Data Recording and Retrieval Technologies*, pages 103-120, 1996.

16. S. H. G. Chan and F. A. Togabi, "Hierarchical storage systems for Interactive Video-on-demand", Technical Report CSL-TR-97-723, Stanford University, 1997.

17. E. Chang and A. Zakhor, "Variable Bit Rate MPEG Video Storage on Parallel Disk Arrays", *Proceedings of the First International Workshop on Community Networking Integrated Multimedia Services to the Home*, pages 127-137, 1994.

18. E. Chang and H. G. Molina, "BubbleUp: Low Latency Fast-Scan for Media Servers", *Proceedings of the ACM Multimedia Conference*, pages 87-98, 1997.

19. E. Chang and H. G. Molina, "Reducing Initial Latency in Media Servers", *IEEE Multimedia*, volume 4, number 3, pages 50-61, 1997.

20. Y. T. Chen, R. L. Kashyap, and A. Ghafoor, "Physical Storage Management for Interactive Multimedia Information Systems", *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 1-6, 1992.

21. C. Y. R. Chen, K. C. Nwosu, and P. B. Berra, "Multimedia Object Modelling and Storage Allocation Strategies for Heterogeneous Parallel Access Storage Devices in Real Time Multimedia Computing Systems", *Proceedings of the Seventeenth Annual International Computer Software and Applications Conference*, pages 216-223, 1993.

22. M. S. Chen, D. D. Kandlur, and P. Yu, "Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams", *Proceedings of the ACM Multimedia Conference*, pages 235-241, 1993.

23. P. M. Chen and D. A. Patterson, "Storage Performance-Metrics and Benchmarks", *Proceedings of the IEEE*, volume 81, number 8, pages 1151-1165, August 1993.

24. P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-Performance, Reliable Secondary Storage", *ACM Computing Surveys*, volume 26, number 2, pages 145-185, June 1994.

25. L. T. Chen, D. Rotem, A. Shoshani, B. Drach, M. Keating, and S. Louis, "Optimizing Tertiary Storage Organization and Access for Spatio-Temporal Datasets", *Proceedings of the Fourth NASA Goddard Conference on Mass Storage Systems and Technologies*, *http://esdis-it.gsfc.nasa.gov/MSST/conf1995/B4_1.html*, 1995.

26. H. J. Chen and T. D. C. Little, "Storage Allocation Policies for Time-Dependent Multimedia Data", *IEEE Transactions on Knowledge and Data Engineering*, volume 8, number 5, pages 855-864, October 1996.

27. S. Chen and M. Thapar, "Zone-bit-recording-enhanced video data layout strategies", *Proceedings of the IEEE MASCOTS Conference*, pages 29-35, 1996.

28. A. L. Chervenak, D. A. Patterson, and R. H. Katz, "Storage systems for movies-on-demand video servers", *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 246-256, 1995.

29. T. C. Chiueh, "Performance Optimization for Parallel Tape Arrays", *Proceedings of the 9th ACM International Conference on Supercomputing*, pages 375-384, 1995.

30. S. Christodoulakis, P. Triantafillou, and F.A. Zioga, "Principles of Optimally Placing Data in Tertiary Storage Libraries", *Proceedings of the 23rd VLDB Conference*, pages 236-245, 1997.

31. T. S. Chua, J. Li, B. C. Ooi, and K. L. Tan, "Disk striping strategies for large video-on-demand servers", *Proceedings of the ACM Multimedia Conference*, pages 297-306, 1996.

32. A. Cohen and W. A. Burkhard, "Segmented Information Dispersal (SID) for Efficient Reconstruction in Fault-Tolerant Video Servers", *Proceedings of the ACM Multimedia Conference*, pages 277-286, 1996.

33. R. A. Coyne and H. Hulen, "An Introduction to the Mass Storage System Reference Model, Version 5", *Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems*, pages 47-53, 1993.

34. R. Cummings, "System Architectures Using Fibre Channel", *Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems*, pages 251-256, 1993.

35. A. Dan and D. Sitaram, "An online video placement policy based on bandwidth to space ratio (BSR)", *Proceedings of the ACM SIGMOD Conference*, pages 376-385, 1995.

36. A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram, and R. Tewari, "Buffering and Caching in Large-Scale Video Servers", *Proceedings of IEEE International Computer Conference*, pages 217-224, 1995.

37. Y. N. Doganata and A. N. Tantawi, "A cost/performance study of video servers with hierarchical storage", *Proceedings of the IEEE Conference on Multimedia Computing and Systems*, pages 393-402, 1994.

38. R. Drach, S. W. Hyer, A. Shoshani, D. Rotem, A. Segev, S. Seshadri, H. Samet, and P. Bogdanovich, "Optimizing Mass Storage Organization and Access for Multi-Dimensional Scientific Data", *Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems*, pages 215-219, 1993.

39. A. L. Drapeau and R. H. Katz, "Striped Tape Arrays", *Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems*, pages 257-265, April 1993.

40. A. L. Drapeau and R. H. Katz, "Striping in Large Tape Libraries", *Proceedings of the Conference on Supercomputing'93*, pages 378-387, 1993.

41. C. Drummond, D. Ionescu, and R. Holte, "Intelligent Browsing for Multimedia Applications", *Proceedings of the IEEE Multimedia Conference*, pages 386-389, 1996.

42. C. Federighi and L. A. Rowe, "A Distributed Hierarchical Storage Manager For A Video-on-demand System", *Proceedings of SPIE*, volume 2185, *Conference on Storage and Retrieval for Image and Video Databases II*, pages 185-195, 1994.

43. A. Finkelstein, C. E. Jacobs, and D. H. Salesin, "Multiresolution Video", *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pages 281-290, 1996.

44. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by Image and Video Content: The QBIC System", *IEEE Computer*, volume 28, number 9, pages 23-32, September 1995.

45. D. A. Ford and S. Christodoulakis, "Optimal Placement of High-Probability Randomly Retrieved Blocks on CLV Optical Discs", *ACM Transactions on Information Systems*, Volume 9, Number 1, pages 1-30, 1991.

46. D. A. Ford, R. J. T. Morris, and A. E. Bell, "Redundant Arrays of Independent Libraries (RAIL): A Tertiary Storage System", *Proceedings of COMPCON'96*, pages 280-285, 1996.

47. S. Franchi, M. Imperato, and F. Prampolini, "Multimedia Perspectives for Next Generation PAC Systems", *Proceedings of the Fifth Annual IEEE Symposium on Computer-Based Medical Systems*, pages 156-159, 1992.

48. C. S. Freedman and D. J. DeWitt, "The SPIFFI Scalable Video-on-Demand System", *Proceedings of ACM SIGMOD International Conference on Management of Data,* pages 352-363, 1995.

49. B. Furht, "MultiMedia Systems: An Overview", *IEEE MultiMedia*, volume 1, number 1, pages 47-59, 1994.

50. B. Furht and M. Milenkovic, *A Guided Tour of Multimedia Systems and Applications*, IEEE Computer Society Press, 1995.

51. B. Furht and R. Westwater, "Video Presentation and Compression," in *Handbook of Multimedia Computing*, B. Furht (Ed.), CRC Press, 1999.

52. T. Furukawa, K. Nakane, R. Arai, "High Density Recording Method for Magneto-Optical Disk", *IEEE Transactions on Magnetics*, volume 24, number 6, pages 2536-2538, November 1988.

53. D. J. Gemmell and S. Christodoulakis, "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval", *ACM Transactions on Information Systems*, volume 10, number 1, pages 51-90, 1992.

54. D. J. Gemmell, J. Han, R. J. Beaton, and S. Christodoulakis, "Delay-Sensitive Multimedia on Disks", *IEEE Multimedia*, volume 1, number 3, pages 56-67, 1994.

55. D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe, "Multimedia Storage Servers: A Tutorial", *IEEE Computer*, volume 28, number 5, pages 40-49, May 1995.

56. B. A. Gennart and R. D. Hersch, "Multimedia Performance Behaviour of the GigaView Parallel Image Server", *Proceedings of Thirteenth IEEE Symposium on Mass Storage Systems*, pages 90-98, 1994.

57. S. Ghandeharizadeh and L. Ramos, "Continuous retrieval of multimedia data using parallelism", *IEEE Transactions on Knowledge and Data Engineering*, volume 5, number 4, pages 658-669, 1993.

58. S. Ghandeharizadeh and C. Shahabi, "On Multimedia Repositories, Personal Computers, and Hierarchical Storage Systems", *Proceedings of the ACM Multimedia Conference*, pages 407-416, 1994.

59. S. Ghandeharizadeh, S.H. Kim, and C. Shahabi, "On Configuring a Single Disk Continuous Media Server", *Proceedings of the ACM Multimedia Conference*, pages 37-46, 1995.

60. S. Ghandeharizadeh, S.H. Kim, W. Shi, and R. Zimmermann, "On Minimizing Startup Latency in Scalable Continuous Media Servers", *Proceedings of SPIE*, volume 3020, *Multimedia Computing and Networking Conference*, pages 144-155, 1997.

61. S. Ghandeharizadeh, S. H. Kim, and C. Shahabi, "On Disk Scheduling and Data Placement for Video Servers", Technical Report 97-650, University of Southern California, 1997.

62. G. A. Gibson, R. H. Patterson, and M. Satyanarayanan, "Disk Reads with DRAM Latency", *Proceedings of the Third Workshop on Workstation Operating Systems*, pages 126-131, 1992.

63. G. A. Gibson, *Redundant Disk Arrays Reliable, Parallel Secondary Storage*, ACM 1991 Distinguished Dissertation, The MIT Press, 1992.

64. J. J. Gniewek and S. M. Vogel, "Influence of Technology on Magnetic Tape Storage Device Characteristics", *Proceedings of Fourth NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 237-251, 1995.

65. J. J. Gniewek, "Evolving Requirements for Magnetic Tape Data Storage Systems", *Proceedings of Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 477-492, September 1996.

66. L. Golubchik, J. C. S. Lui, and R. Muntz, "Reducing I/O Demand in Video-On-Demand Storage Servers", *Proceedings of ACM SIGMETRICS Conference*, pages 25-36, 1995.

67. L. Golubchik, R. R. Muntz, and R. W. Watson, "Analysis of Striping Techniques in Robotic Storage Libraries", *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 225-238, 1995.

68. C. Griwodz, M. Bär, and L. C. Wolf, "Long-term Movie Popularity Models in Video-on-Demand Systems", *Proceedings of the ACM Multimedia Conference*, pages 349-357, 1997.

69. R. Grossman, X. Qin, W. Xu, H. Hulen, and T. Tyler, "An Architecture for a Scalable, High-Performance Digital Library", *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 89-98, 1995.

70. A. Guha, A. Pavan, J. C. L. Liu, and B. A. Roberts, "Controlling the Process with Distributed Multimedia", *IEEE Multimedia*, volume 2, number 2, pages 20-29, 1995.

71. W. B. Hanlon, E. F. Fener, and J. W. Downs, "Data Storage and Management Requirements for the Multimedia Computer-Based Patient Medical Record", *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 11-16, 1995.

72. B. K. Hillyer and A. Silberschatz, "Random I/O Scheduling in Online Tertiary Storage Systems", *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 195-204, 1996.

73. B. K. Hillyer and A. Silberschatz, "Storage Technology: Status, Issues, and Opportunities", http://www.bell-labs.com/user/hillyer/papers, 1996.

74. B. K. Hillyer and A. Silberschatz, "On the Modeling and Performance Characteristics of a Serpentine Tape Drive", *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 170-179, 1996.

75. "HP SureStore 330/600 fx Optical Jukebox Product Details", http://www.hp-optical.com/ProductCatalog/330fxDetailsSection.htm.

76. J. Hsieh, M. Lin, J. C. L. Liu, D. H. C. Du, and T. M. Ruwart, "Performance of a mass storage system for video-on-demand", *Proceedings of Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM'95*, volume 2, pages 771-778, 1995.

77. D. Isaac, "Hierarchical Storage Management for Relational Databases", *Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems*, pages 139-144, 1993.

78. M. A. Jabbar, "Some Novel Ideas for Disk Drive Spindle Motors", *Proceedings of IEEE Asia-Pacific Workshop on Advances in Motion Control*, pages 171-176, 1993.

79. T. Johnson, "Queuing Models of Tertiary Storage", *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, volume 2, pages 529-552, September 1996.

80. R. H. Katz, G. A. Gibson, and D. A. Patterson, "Disk System Architectures for High Performance Computing", *Proceedings of the IEEE*, volume 77, number 12, pages 1842-1858, December 1989.

81. M. G. Kienzle, A. Dan, D. Sitaram, and W. Tetzlaff, "Using Tertiary Storage in Video-on-Demand Servers", *Proceedings of IEEE COMPCON*, pages 225-233, March 1995.

82. M. Y. Kim, "A Multimedia Information System for Home Health-Care Support", *IEEE Multimedia*, volume 2, number 4, pages 83-87, 1995.

83. R. Kirk, T. Christianson, and D. Faizullabhoy, "Embedded Intelligence", *Byte*, volume 17, number 3, pages 195-203, March 1992.

84. L. Kleinrock, *Queueing Systems, Volume I: Theory*, John Wiley & Sons, 1975.

85. J. Kohl, M. Stonebraker, and C. Staelin, "HighLight: A File System for Tertiary Storage", *Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems*, pages157-161, 1993.

86. J. Korst, "Random Duplicated Assignment: An Alternative to Striping in Video Servers", *Proceedings of the ACM Multimedia Conference*, pages 219-226, 1997.

87. M. Krunz and H. Hughes, "A Traffic Model for MPEG-Coded VBR Streams", *Proceedings of the ACM SIGMETRICS Conference*, pages 47-55, 1995.

88. T. L. Kunii, Y. Shinagawa, R. M. Paul, M. F. Khan, and A. A. Khokhar, "Issues in storage and retrieval of multimedia data", *Multimedia Systems*, volume 3, pages 298-304, 1995.

89. A. Kuratti and W. H. Sanders, "Performance Analysis of the RAID 5 Disk Array", *Proceedings of IEEE International Computer Performance and Dependability Symposium IPDS'95*, pages 236-245, 1995.

90. L. Kuvayev, C. L. Giles, J. Philbin, and H. Cejtin, "Intelligent Methods for File System Optimization", *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, pages 528-533, 1997.

91. S. W. Lau, J. C. S. Lui, and P. C. Wong, "A Cost-effective Near-line Storage Server for Multimedia System", *Proceedings of IEEE Conference on Data Engineering*, pages 449-456, 1995.

92. M. H. Lee, C. H. Wen, C. Y. Cheng, F. C. Wang, and Y. J. Oyang, "Storage Hierarchy Design in Video-On-Demand Servers", *Proceedings of SPIE*, volume 2670, *Storage and Retrieval for still image and video databases IV*, pages 300-307, 1996.

93. C. H. C. Leung and Q. H. Choo, "The effect of fixed-length record implementation on file system response", *Acta Informatica*, volume 17, pages 399-409, 1982.

94. C. H. C. Leung (Ed.), *Visual Information Systems*, LNCS 1304, Springer-Verlag, 1997.

95. C. H. C. Leung, *Quantitative Analysis of Computer Systems*, John Wiley & Sons, 1988.

96. M. Y. Y. Leung, J. C. S. Lui, and L. Golubchik, "Buffer and I/O Resource Pre-allocation for Implementing Batching and Buffering Techniques for Video-on-Demand Systems", Technical Report CS-TR-96-03, The Chinese University of Hong Kong, April 1996.

97. D. Lignos, "Digital Linear Tape (DLT) Technology and Product Family Overview", *Proceedings of Fourth NASA Goddard Conference on Mass Storage Systems and Technologies*, *http://esdis-it.gsfc.nasa.gov/MSST/conf1995/B1_1.html*, 1995.

98. T. D. C. Little and D. Venkatesh, "Popularity-based assignment of movies to storage devices in a video-on-demand system", *Proceedings of ACM Multimedia Systems*, volume 2, pages 280-287, 1995.

99. P. Lougher and D. Shepherd, "The design of a storage server for continuous media", *The Computer Journal*, volume 36, number 1, pages 32-42, 1993.

100. P. Lougher, D. Shepherd, and D. Pegler, "The Impact of Digital Audio and Video on High-Speed Storage", *Proceedings of the Thirteenth IEEE Symposium on Mass Storage Systems*, pages 84-89, 1994.

101. G. K. Ma, C. S. Wu, M. C. Liu, and B. S. P. Lin, "Efficient Real-time Data Retrieval Through Scalable Multimedia Storage", *Proceedings of ACM International Multimedia Conference*, pages 165-172, November 1997.

102. R. E. Matick, *Computer Storage Systems and Technology*, John Wiley & Sons, Inc., 1977.

103. J. A. McCormick, *The New Optical Storage Technology Including Multimedia, CD-ROM, and Optical Drives*, Second Edition, Irwin Professional Publishing, 1994.

104. T. Mori, H. Suzuki, K. Nishimura, and H. Nakano, "Playback Techniques for a Video-on-Demand System Using an Optical Mass Storage System", *Japanese Journal of Applied Physics*, volume 35, part 1, number 1B, pages 495-499, 1996.

105. J. Myllymaki and M. Livny, "Disk-Tape Joins: Synchronizing Disk and Tape Access", *Proceedings of ACM SIGMETRICS'95*, pages 279-290, 1995.

106. T. Nakagomi, M. Holzbach, R. V. Meter III, and S. Ranade, "Refining the Storage Hierarchy: An Ultra-Fast Magneto-Optical Disk Drive", *Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems*, pages 267-274, 1993.

107. T. Nemoto, M. Kitsuregawa, and M. Takagi, "Design and implementation of scaleable tape archiver", *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 229-238, September 1996.

108. S. W. Ng, "Improving Disk Performance Via Latency Reduction", *IEEE Transactions on Computers*, volume 40, number 1, pages 22-34, January 1991.

109. S. W. Ng, "Latency Reduction for CD-ROM and CLV Disks", *Proceedings of the 25th Hawaii International Conference on System Sciences*, volume 1, pages 100-108, 1991.

110. K. Nishimura, T. Mori, Y. Ishibashi, and N. Sakurai, "System Architecture for Digital Video-on-Demand Services", *Proceedings of the Second IEEE International Conference on Image Processing,* pages 602-606, 1992.

111. K. C. Nwosu, C. Y. R. Chen, and P. B. Berra, "Multimedia Object Modeling and Storage Allocation Strategies", *Journal of Intelligent Information Systems,* volume 3, pages 357-391, 1994.

112. K. C. Nwosu, P. Bobbie, and B. Thuraisingham, "Data Allocation and Spatio-Temporal Implications for Video-On-Demand Systems", *Proceedings of the 1995 IEEE Fourteenth Annual International Phoenix Conference on Computers and Communications*, pages 629-635, 1995.

113. W. S. Oakley, "Progress toward demonstrating a high performance optical tape recording technology", *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 571-582, September 1996.

114. V. E. Ogle and M. Stonebraker, "Chabot: Retrieval from a Relational Database of Images", *IEEE Computer,* volume 28, number 9, pages 40-48, September 1995.

115. Y. J. Oyang, M. H. Lee, C. H. Wen, and C. Y. Cheng, "Design of multimedia storage systems for on-demand playback", *Proceedings of the Eleventh International Conference on Data Engineering*, pages 457-465, 1995.

116. B. Özden, A. Biliris, R. Rastogi, and A. Silberschatz, "A low-cost storage server for movie on demand databases", *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 594-605, September 1994.

117. B. Özden, R. Rastogi, and A. Silberschatz, "On the design of a low-cost video-on-demand storage system", *Multimedia Systems*, volume 4, pages 40-54, 1996.

118. H. H. Pang, "Tertiary storage in multimedia systems: staging or direct access ?", *Multimedia Systems*, volume 5, pages 386-399, 1997.

119. D. A. Patterson and J. L. Hennessy, *Computer Organization and Design The Hardware/Software Interface*, Morgan Kaufmann Publishers, 1994.

120. R. H. Patterson and G. A. Gibson, "Exposing I/O Concurrency with Informed Prefetching", *Proceedings of the Third International Conference on Parallel and Distributed Information Systems*, pages 7-16, 1994.

121. W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Compression Standard*, Van Nostrand Reinhold Publisher, 1993.

122. P. V. Rangan and H. M. Vin, "Efficient storage techniques for digital continuous multimedia", *IEEE Transactions on Knowledge and Data Engineering*, volume 5, number 4, pages 564-573, August 1993.

123. A. L. N. Reddy and J. C. Wyllie, "Disk scheduling in a multimedia I/O system", *Proceedings of First ACM Conference on Multimedia*, pages 225-233, 1993.

124. A. L. N. Reddy and J. C. Wyllie, "I/O Issues in a Multimedia System", *IEEE Computer*, volume 27, number 3, pages 69-74, March 1994.

125. L. A. Rowe, J. S. Boreczky, and D. A. Berger, "A distributed hierarchical video-on-demand system", *Proceedings of IEEE Conference on Image Processing*, pages 334-337, 1995.

126. C. Ruemmler and J. Wilkes, "An introduction to disk drive modeling", *IEEE Computer*, volume 27, number 3, pages 17-28, March 1994.

127. E. Salmon, "Storage and network bandwidth requirements through the year 2000 for the NASA center for computational sciences", *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 273-286, September 1996.

128. S. Sarawagi, "Database Systems for Efficient Access to Tertiary Memory", *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems*, pages 120-126, 1995.

129. S. Sarawagi and M. Stonebraker, "Reordering Query Execution in Tertiary Memory Databases", *Proceedings of the 22nd Very Large Database Conference*, pages 156-167, 1996.

130. D. E. Speliotis, "High Density Recording on Particulate and Thin Film Rigid Disks", *IEEE Transactions on Magnetics*, volume 25, number 5, pages 4048-4050, September 1989.

131. S. G. Stan and J. L. Bakx, "Adaptive-Speed Algorithms for CD-ROM Systems", *IEEE Transactions on Consumer Electronics*, volume 42, number 1, pages 43-51, February 1996.

132. R. Steinmetz, "Analyzing the multimedia operating system", *IEEE Multimedia*, volume 2, number 1, pages 68-84, 1995.

133. H. Suzuki, K. Nishimura, A. Uemori, and H. Sakamoto, "Storage hierarchy for video-on-demand systems", *Proceedings of SPIE*, volume 2185, *Storage and Retrieval for Image and Video Databases II*, pages 198-207, 1994.

134. W. Tavanapong, K. A. Hua, and J. Z. Wang, "A framework for supporting previewing and VCR operations in a low bandwidth environment", *Proceedings of the ACM Multimedia Conference*, pages 303-312, 1997.

135. D. B. Terry, and D. C. Swinehart, "Managing stored voice in the etherphone system", *ACM Transactions on Computer Systems*, volume 6, number 1, pages 3-27, 1988.

136. R. Tewari, R. King, D. Kandlur, and D. M. Dias, "Placement of Multimedia Blocks on Zoned Disks", *Proceedings of SPIE*, volume 2667, *Multimedia Computing and Networking 1996*, pages 360-367, 1996.

137. F. A. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming RAID-a disk array management system for video files", *Proceedings of First ACM Conference on Multimedia*, pages 393-400, 1993.

138. P. Triantafillou, S. Christodoulakis, and C. Georgiadis, "Optimal Data Placement on Disks: A Comprehensive Solution for Different Technologies", Technical Report, Technical University of Crete, 1996.

139. P. Triantafillou and T. Papadakis, "On-Demand Data Elevation in a Hierarchical Multimedia Storage Server", *Proceedings of the 23rd VLDB Conference*, pages 1-10, 1997.

140. C. Vassilakis, M. Paterakis, and P. Triantafillou, "Video Placement and Configuration of Distributed Video-On-Demand Systems", Technical Report, Technical University of Crete, 1997.

141. M. Vernick, C. Venkatramani, and T. C. Chiueh, "Adventures in Building the Stony Brook Video Server", *Proceedings of ACM Multimedia'96*, pages 287-295, 1996.

142. H. M. Vin and P. V. Rangan, "Designing a Multiuser HDTV Storage Server", *IEEE Journal On Selected Areas In Communications*, volume 11, number 1, pages 153-164, January 1993.

143. A. Vina, J. L. Lérida, A. Molano, and D. del Val, "Real-Time Multimedia Systems", *Proceedings of the Thirteenth IEEE Symposium on Mass Storage Systems*, pages 77-83, 1994.

144. VIS96, *Proceedings of the First International Conference on Visual Information Systems*, 1996.

145. A. Vogel, B. Kerherve, G. V. Bochmann, and J. Gecsei, "Distributed Multimedia and QOS: A Survey", *IEEE MultiMedia*, volume 2, number 2, pages 10-19, 1995.

146. J. Z. Wang, K. A. Hua, and H. C. Young, "SEP: a space efficient technique for managing disk buffers in multimedia servers", *Proceedings of IEEE Multimedia Computing and Systems Conference*, pages 598-607, 1996.

147. Y. C. Wang, S. L. Tsao, R. Y. Chang, M. C. Chen, J. M. Ho, and M. T. Ko, "A Fast Data Placement Scheme for Video Server with Zoned-Disks", *Proceedings of SPIE*, volume 3229, *Conference on Multimedia Storage and Archiving Systems*, pages 92-102, 1997.

148. E. W. Williams, *The CD-ROM and Optical Disc Recording Systems*, Oxford University Press, 1994.

149. J. L. Wolf, P. S. Yu, and H. Shachnai, "DASD Dancing: A Disk Load Balancing Optimization Scheme for Video-on-Demand Computer Systems", *Proceedings of ACM SIGMETRICS'95*, pages 157-166, 1995.

150. I. Yamada, M. Saito, A. Watabe, and K. Itao, "Automated Optical Mass Storage System with 3-Beam Magneto-Optical Disk Drives", *IEEE Transactions on Magnetics*, volume 29, number 3, pages 2172-2176, July 1993.

151. C. Yu, W. Sun, D. Bitton, Q. Yang, R. Bruno, and J. Tullis, "Efficient placement of audio data on optical disks for real-time applications", *Communications of ACM*, volume 32, number 7, pages 862-871, 1989.

152. J. B. Yu and D. J. Dewitt, "Processing satellite images on tertiary storage: a study of the impact of tile size on performance", *Proceedings of the Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 460-476, September 1996.

153. P. C. Yue and C. K. Wong, "On the Optimality of the Probability Ranking Scheme in Storage Applications", *Journal of the ACM*, volume 20, number 4, pages 624-633, 1973.

154. R. Zimmermann and S. Ghandeharizadeh, "Continuous Display Using Heterogeneous Disk-Subsystems", *Proceedings of ACM Multimedia 97*, pages 227-238, 1997.

Appendix A

Simulation Measurements

Table A.1. Data access time

10.88 inch disk	VDR disk			CDR disk		
access size	predicted	measured	% error	predicted	measured	% error
1 KB	35.0	33.9	3.3%	32.6	32.5	0.1%
10 KB	37.2	36.3	2.4%	32.9	33.4	1.5%
100KB	58.6	58.9	0.5%	36.5	36.6	0.1%
1000 KB	272.9	272.5	0.1%	72.6	73.2	0.8%
10000 KB	2659.5	2531.7	4.8%	465.9	480.1	3.0%
5.25 inch disk	VDR disk			CDR disk		
access size	predicted	measured	% error	predicted	measured	% error
1 KB	25.7	25.8	0.3%	24.9	24.9	0.3%
10 KB	28.7	28.9	0.4%	25.8	24.8	3.9%
100KB	58.7	57.1	2.7%	35.4	36.5	2.9%
1000 KB	358.7	357.5	0.3%	131.4	130.1	1.0%
10000 KB	3586.6	3505.4	2.3%	1165.4	1155.9	0.8%
3.5 inch disk	VDR disk			CDR disk		
access size	predicted	measured	% error	predicted	measured	% error
1 KB	21.4	20.3	4.8%	21.0	20.7	1.1%
10 KB	23.6	25.3	7.2%	22.0	20.7	5.9%
100KB	46.1	45.7	0.8%	32.0	32.6	1.9%
1000 KB	271.1	271.4	0.1%	132.0	126.6	4.1%
10000 KB	2690.3	2631.6	2.2%	1194.7	1181.3	1.1%
2.5 inch disk	VDR disk			CDR disk		
access size	predicted	measured	% error	predicted	measured	% error
1 KB	21.3	20.0	6.0%	21.0	19.9	5.0%
10 KB	25.2	24.8	1.6%	23.2	21.9	5.8%
100KB	64.3	66.7	3.7%	45.6	47.5	4.1%
1000 KB	476.5	468.0	1.8%	269.2	277.9	3.2%
10000 KB	4640.2	4583.7	1.2%	2650.8	2682.5	1.2%
1.8 inch disk	VDR disk			CDR disk		
access size	predicted	measured	% error	predicted	measured	% error
1 KB	20.2	18.8	6.9%	20.0	19.6	1.9%
10 KB	24.9	24.8	0.6%	23.4	21.4	8.6%
100KB	72.3	67.0	7.3%	57.2	57.3	0.1%
1000 KB	565.6	566.6	0.2%	415.2	407.9	1.8%
10000 KB	5577.7	5533.3	0.8%	3995.1	3873.9	3.0%

Table A.2. Throughput of a CDR disk being partitioned into two zone groups

% of inner	Inner zone group			Outer zone group		
partition	predicted	measured	% error	predicted	measured	% error
0%	2.79	2.86	2%	8.73	7.06	19%
10%	3.39	3.35	1%	9.32	8.20	12%
20%	3.98	4.09	3%	9.91	8.73	12%
30%	4.57	4.51	1%	10.51	8.77	17%
40%	5.17	4.86	6%	11.10	9.34	16%
50%	5.76	5.55	4%	11.69	9.93	15%
60%	6.35	5.85	8%	12.29	10.36	16%
70%	6.95	6.29	9%	12.88	10.76	16%
80%	7.54	6.85	9%	13.47	11.58	14%
90%	8.13	7.22	11%	14.07	12.03	15%
100%	8.73	7.06	19%	14.66	12.01	18%

Table A.3. Disk utilization

Arrival rate	VDR disk			CDR disk		
(requests/min)	predicted	measured	% error	predicted	measured	% error
40	24%	23.89%	0.10%	8.76%	8.71%	0.61%
80	48%	47.82%	0.03%	17.52%	17.30%	1.27%
120	72%	71.51%	0.33%	26.29%	26.26%	0.10%
167	100%	99.48%	0.51%	36.63%	36.75%	0.30%
200				43.81%	43.74%	0.17%
240				52.57%	52.67%	0.18%
280				61.33%	61.51%	0.29%
320				70.10%	70.22%	0.18%
360				78.86%	77.40%	1.85%
400				87.62%	87.24%	0.43%
440				96.38%	95.18%	1.25%

Table A.4. Mean waiting time

Arrival rate	VDR disk			CDR disk		
	predicted	measured	% error	predicted	measured	% error
40	0.056	0.055	2.09%	0.006	0.008	25.11%
80	0.165	0.169	2.59%	0.014	0.015	9.89%
120	0.456	0.402	11.87%	0.023	0.021	9.10%
167	4062	6.428	99.84%	0.038	0.045	18.68%
200				0.051	0.065	27.23%
240				0.073	0.080	9.92%
280				0.105	0.112	7.48%
320				0.154	0.166	7.40%
360				0.246	0.221	9.94%
400				0.466	0.499	6.96%
440				1.755	1.245	29.07%

Table A.5. Mean response time

Arrival rate	VDR disk			CDR disk		
	predicted	measured	% error	predicted	measured	% error
40	0.42	0.41	0.41%	0.14	0.14	0.49%
80	0.52	0.53	0.83%	0.15	0.14	0.36%
120	0.81	0.76	6.77%	0.15	0.15	1.42%
167				0.17	0.18	4.43%
200				0.18	0.20	7.59%
240				0.20	0.21	3.73%
280				0.24	0.24	3.61%
320				0.29	0.30	4.07%
360				0.38	0.35	7.10%
400				0.60	0.63	5.36%

Table A.6. Mean queue length

Arrival rate	VDR disk			CDR disk		
	predicted	measured	% error	predicted	measured	% error
40	0.04	0.04	2.04%	0.00	0.01	25.22%
80	0.22	0.22	2.54%	0.02	0.02	10.09%
120	0.91	0.80	11.91%	0.05	0.04	9.15%
167	11324	17.86	99.84%	0.11	0.13	18.69%
200				0.17	0.22	27.13%
240				0.29	0.32	9.80%
280				0.49	0.52	7.23%
320				0.82	0.88	7.43%
360				1.47	1.33	9.99%
400				3.11	3.32	6.85%
440				12.87	9.00	30.06%

Table A.7. Mean number of requests

Arrival rate	VDR disk			CDR disk		
	predicted	measured	% error	predicted	measured	% error
40	0.28	0.28	0.35%	0.09	0.09	0.58%
80	0.70	0.70	0.78%	0.19	0.19	0.17%
120	1.63	1.52	6.81%	0.31	0.31	1.47%
167				0.47	0.49	4.44%
200				0.61	0.66	7.51%
240				0.82	0.85	3.62%
280				1.10	1.14	3.37%
320				1.52	1.59	4.10%
360				2.26	2.10	7.16%
400				3.99	4.19	5.25%

Table A.8. Maximum system throughput

segment length	non-striping		
	predicted	measured	% error
5	20.25	20.15	0.5%
6	20.55	20.94	1.9%
7	20.23	21.41	5.9%
8	20.97	21.29	1.5%
9	20.57	21.38	3.9%
10	21.28	20.89	1.8%
11	21.31	21.84	2.5%
12	21.53	21.57	0.2%
13	20.68	21.66	4.8%
14	21.14	22.99	8.7%
15	21.83	22.84	4.6%
segment length	parallel striping		
	predicted	measured	% error
5	13.6	12.38	9.2%
6	13.9	13.15	5.7%
7	13.9	14.30	2.9%
8	14.5	14.44	0.2%
9	14.4	13.57	5.6%
10	15.0	15.96	6.7%
11	15.1	16.63	10.0%
12	15.4	16.05	4.3%
13	15.0	15.81	5.7%
14	15.4	17.99	16.9%
15	16.0	17.25	7.8%
segment length	high concurrency placement		
	predicted	measured	% error
5	28.61	30.05	5.0%
6	31.00	30.46	1.7%
7	31.16	33.07	6.1%
8	33.99	34.25	0.8%
9	33.36	35.17	5.4%
10	35.86	36.15	0.8%
11	36.21	36.13	0.2%
12	37.08	36.89	0.5%
13	34.68	37.78	9.0%
14	36.16	38.22	5.7%
15	38.34	38.38	0.1%

Table A.9. Mean stream response time

stream arrival rate	non-striping		
	predicted	measured	% error
5.00	138	128	7.0%
10.00	158	134	15.3%
15.00	193	159	17.6%
20.00	264	202	23.7%
25.00	465	299	35.7%
30.00	3123	1526	51.1%
stream arrival rate	high concurrency placement		
	predicted	measured	% error
1.0	214	173	19.2%
2.0	229	189	17.3%
3.0	250	203	18.6%
4.0	283	215	24.0%
5.0	339	230	32.2%
6.0	372	263	29.5%
7.0	453	301	33.6%
8.0	442	332	24.9%
9.0	508	423	16.9%
10.0	611	445	27.2%
11.0	792	718	9.3%
12.0	1184	990	16.3%
13.0	2661	2057	22.7%
stream arrival rate	high concurrency placement		
	predicted	measured	% error
5.00	385	419	9.0%
10.00	497	517	4.1%
15.00	581	595	2.3%
20.00	628	660	5.1%
25.00	677	732	8.2%
30.00	804	827	3.0%
35.00	966	989	2.3%
40.00	1191	1170	1.8%
45.00	1531	1416	7.5%
50.00	2111	1948	7.7%
55.00	3269	2533	22.5%

Table A.10. Buffer size

stream arrival rate	non-striping		
	predicted	measured	% error
5.00	2172	2088	3.8%
10.00	2171	2088	3.8%
15.00	2170	2088	3.8%
20.00	2170	2088	3.7%
25.00	2169	2088	3.7%
30.00	2168	2088	3.7%
stream arrival rate	parallel striping		
	predicted	measured	% error
1.0	2211.1	2088	5.6%
2.0	2209.2	2088	5.5%
3.0	2206.3	2088	5.4%
4.0	2201.8	2088	5.2%
5.0	2195.1	2088	4.9%
6.0	2195.1	2088	4.9%
7.0	2195.1	2088	4.9%
8.0	2195.1	2088	4.9%
9.0	2195.1	2088	4.9%
10.0	2195.1	2088	4.9%
11.0	2195.1	2088	4.9%
12.0	2195.1	2088	4.9%
13.0	2195.1	2088	4.9%
stream arrival rate	high concurrency placement		
	predicted	measured	% error
5.00	2083	2073	0.5%
10.00	2027	2048	1.1%
15.00	1984	2041	2.9%
20.00	1960	2023	3.2%
25.00	1936	1997	3.2%
30.00	1871	1978	5.7%
35.00	1789	1930	7.9%
40.00	1675	1886	12.6%
45.00	1503	1798	19.6%
50.00	1209	1639	35.5%
55.00	625	1480	136.7%
60.00	#N/A	853	#N/A