Tertiary Storage in Digital Video Archives

Olav Sandstå

Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway

May, 2004

Abstract

In order to efficiently manage the large amounts of video data stored in a digital video archive, computerized management systems must be developed for storing and making the video available to users. In this thesis, we study tertiary storage technologies and storage architectures for storing and retrieving digital video in video archives.

We evaluate serpentine tape as a storage medium for digital video. In order to increase the performance of storage systems using serpentine tape, we present and evaluate a detailed access-time model for serpentine tape and a novel scheduling algorithm for optimizing concurrent accesses to the tape. The scheduling algorithm is used for evaluating serpentine tape for storing images and video sequences. The main conclusion is that by using the access-time model and the proposed scheduling algorithm, it is possible to achieve significant improvements in initial latency, average access time, and the number of requests that can be served by a single tape drive.

Tertiary storage technologies including magnetic tape and DVD are evaluated for use in digital video archives. The evaluation is performed using a simulator of the storage system of a video archive. The simulation model is based on the architecture of the Elvira II video archive server. Different configurations for the storage system are evaluated with regards to performance and cost. In the evaluation different allocation strategies, access distributions, and user loads are studied. The effect of using a cache based on magnetic disks is investigated.

The main conclusion is that the choice of architecture and storage technology for a video archive depends on the user generated load, the size of the requested video sequences, and the access distribution for the stored videos. It also depends on whether throughput, response time, storage cost, or cost per retrieved video is the main evaluation criterion. Furthermore, we show that a video archive based on DVD as the main storage technology outperforms a video archive using magnetic tape, and that including a relatively small disk cache in most cases improves the performance and reduces the total cost of the archive.

The ideas and results presented in this thesis are also useful outside the video archive context. The strategies and results are beneficial for applications that require hierarchical storage management systems for managing large data volumes.

Contents

Preface

This doctoral thesis is submitted to the Norwegian University of Science and Technology in partial fulfillment of the degree "doktor ingeniør." The work has been carried out at the Database Systems Group, Department of Computer and Information Science at the Norwegian University of Science and Technology under the supervision of Professor Roger Midtstraum and co-advisor Professor Kjell Bratbergsengen. The doctoral study was funded by Telenor Research and Development.

Acknowledgements

First of all I want to thank my advisor, Professor Roger Midtstraum, for all the guidance, ideas, and the always constructive feedback, and for reading and providing valuable comments to various drafts of this thesis. I also want to thank co-advisor Professor Kjell Bratbergsengen for always taking the time for interesting discussions and for answering questions I have had.

During the years I have worked on this thesis, I have been working together with and received help from many people. In particular, I would like to thank the members of the VideoSTAR project. Working together with Rune Hjelsvold, Roger Midtstraum and Stein Langørgen on the VideoSTAR project gave a good introduction to the research area and provided the inspiration to start the work on this thesis. I also want to thank Stein Langørgen who implemented the first prototype of the Elvira video server and Øystein Torbjørnsen, then at Telenor R&D, for useful discussions about the design of the Elvira II video archive server.

I want to thank the other members of the Database Systems group for providing a good environment for doctoral students. In particular, I would like to thank Kjetil Nørvåg for insightful discussions, valuable feedback and for taking time to proofread major parts of this thesis, and Ståle Smedseng for valuable comments and proofreading. I am also grateful to the always friendly and helpful technical and administrative staff of the CS department. I would also thank Professor Thomas Plagemann for many detailed and valuable comments on the thesis and Professor Betty Salzberg for general comments on the thesis. I am indebted to Clustra, now Sun Microsystems, for making it possible to finish the work on the thesis and to my colleagues at Clustra and Sun Microsystems for all the support I have received during the last part of my doctoral study.

Finally, I wish to express my gratitude to Aliona, who has encouraged and inspired me and given me all the support and love I needed to finish this thesis. I also want to thank my parents and my sister and my brothers for all the help and support I have received.

Chapter 1

Introduction

It is already possible by ingenious optical contrivances to throw stereoscopic photographs of people on screens in full view of an audience. Add the talking phonograph to counterfeit their voices, and it would be difficult to carry the illusion of real presence much further.

The December 22, 1877, issue of Scientific American reporting
of Thomas Alva Edison's tin foil phonograph.

When William Kennedy-Laurie Dickson in 1889 showed the first moving images synchronized with recorded sound using the *kinetophonegraph* it was an important point in the history of equipment for producing and replaying *film*. The quality of the "film" produced by the kinetophonegraph was poor compared to the film quality we expect today. The kinetophonegraph was only able to show tiny flickering images with barely synchronized sound. Since then, the technical quality of film has been greatly improved. Today, we have film and video that both in image and sound quality are close to fulfill the *illusion of real presence* predicted in the December 1877 edition of Scientific American.

From the first black and white motion picture productions till today's high quality digital HDTV productions, much research and great inventions have been done within multiple research areas. During the first hundred years after Dickson played his first film, the main research was on improving the technology used for capturing and storing the light and sound waves on a medium in order to be able to replay it as a film. The research has involved scientists from many different areas. Physics, chemistry, and material technology have all played an important role in improving the technical quality of the film medium. Twenty years ago, computer systems had become powerful enough to handle the amount of data motion picture requires. Since then, digital video has become an important research area for mathematicians and computer scientists. Within computer science, the research involves multiple disciplines, ranging from low-level coding techniques for digital video to technologies for storage and delivery of digital video and systems capable of interpreting the content of the video.

Today, millions of hours of film and video are stored in different kinds of archives around the world, and the amount of film and video is growing every day. A lot of important documentation about our modern history exists in form of film and video. To avoid losing valuable information, it is important to have systems that take care of these films and videos, and make them easily available for the users.

Most of the film and video is still stored on a variety of analog formats. The main problem with the media used for storing the analog film and video is that as time goes by, they will be subject to wear and tear. This results in a degradation of the quality of the film or video. Making copies of the film or video, introduces even more noise in both the images and the audio. By converting the analog film or video to a digital format, it is possible to make new copies that are identical to the original. In this way, the film and video can be preserved without degradation and loss of quality.

A major advantage of having the film and video on a digital format is related to easy access for the end-user. By developing systems that stores and manages the digital film and video, the end-user no longer needs to get a physical copy of the film or video in order to watch it. The digital video can be delivered directly to the user's computer or television screen, with a higher quality than could have been achieved if the video was delivered using an analog medium.

Due to these important advantages of digital video, much of the analog film and video will be converted to digital video formats and new films and videos will be created using digital formats. The amount of digital film and video will increase rapidly, leading to large amounts of digital data that have to be managed. In this thesis, we address some of these challenges and present the results from our work on storage systems that can manage the huge amount of data that will be stored in digital film and video archives.

In this chapter, we give an introduction to the work that is presented in this thesis. The first section introduces the main challenges within management of large amounts of digital video. The following sections provide an overview of the work and the main contributions of this thesis. The research methodology used to perform the work is presented, and in the final section we outline the structure of the rest of the thesis.

1.1 Managing Large Amounts of Digital Video

Computer technology plays an important role in storing and making information available for users. Technologies for computerized management of numeric and textual data have been available for many years. For instance, database management systems have been available from the 1960s allowing simple data to be integrated and shared in safe and productive ways. Compared to simple data like numbers and text, video is a very complex data type. For computer systems

to manage video, they must be extended to handle the video data type and the large data volume. They must also support delivery and playback of the video. In order to manage large amounts of digital video, it is paramount to develop systems that provide safe storage, promote sharing and ensure the integrity and consistency of the video data.

Video data contains both *media data*, which is the digitized version of the images and the audio, and *meta-data* that describes properties about the video. Meta-data may include everything from physical descriptions of the video like frame rate and storage location to high level features like composition and name of persons that are present in the film. To better support management of video data, computer systems that handle the media and meta-data as an *integrated* data type must be made available. Systems that support storage of meta-data have been available for years, but due to the high storage and processing requirements, film and video have mostly been recorded, edited, and stored using analog formats. With the increased performance of processors, storage systems, and networks, it has become possible to process and store video in a digital format.

In this thesis, we study some of the challenges of computerized management of large amounts of digital video. The focus is on storage of digital video in video archives. The remainder of this section gives a short presentation of digital video archives and an overview of the main challenges we have addressed.

Digital Video Archives

A digital video archive is a computer system that is responsible for management of digital video and video meta-data. In addition to provide safe storage for the videos, the system should support searching and browsing the meta-data and playback of the videos. The most obvious group of organizations that will have a video archive is broadcasting corporations and film and video producers. Also other organizations like hospitals storing ultrasound scans and oil companies storing seismic scans of an oil field might have video archives.

The Norwegian Broadcasting Corporation has a very large television archive containing approximately 250,000 hours of film and video documents on more than 200,000 tapes (2002). This archive consists of television programs from a period of almost 40 years. The film and video are recorded using about ten different storage technologies, both analog and digital tapes. None of these tape media are under direct control of a computer system. A database stores meta-data about the television programs with references to the corresponding tapes. The meta-data can be searched and used for identifying the tape where a particular program is stored, but to get access to the program, a manual operation must be performed by one of the employees at the television archive. By digitizing the film and video and place it under control of a computer based video archive system, the video could be made much more accessible to all the employees of the organization.

The premier advantage of converting to a digital video archive is the much

better and more practical access to the video in the archive. A user can access the video archive from her own workstation, even if she is working in a remote office. With the video on analog format, she had to request a physical copy of the video being sent to her. Thus, going digital can reduce the access time for the user from days to minutes or seconds. Integrating the video archive system with meta-data and data about the content of the videos makes it easier to find and retrieve the relevant video sequences. An additional benefit is the safer and more reliable storage. Videos do not get worn out when played back, and the storage system can automatically monitor the quality of the storage media at given intervals and if necessary make a new copy of the video data, without loss of image or audio quality. Due to the higher storage density of digital storage media compared to analog tape and the higher degree of automation a computer based system can offer, a digital video archive should require less space and less human intervention than a traditional video archive.

Storage and Delivery of Digital Film and Video

Within the area of management of digital video, a major challenge is how to design and implement storage systems that are capable of storing and delivering the large amounts of data that is required for applications like digital video archives. In this thesis we study technologies and architectures for use in storage systems for digital video. The two challenges we address in this thesis are related to:

1. **Storage of digital film and video.**

   Compared to other data types, digital video requires large amounts of data storage. A two hour film digitized at high quality may require more than one terabyte of storage. By the use of compression technologies, it is possible to reduce the data volume by a factor of 10 to 100. Still, the compressed version of the film will require more than ten gigabytes of data storage. These large data volumes make it costly to store large amounts of digital video, and although current trends show that the cost of data storage is rapidly decreasing, for a large scale video archive the storage cost will represent a large fraction of the total cost of the archive.

2. **Retrieval of digital film and video.**

   Compared to retrieving numeric and textual data types, delivering film and video to an end-user imposes extra challenges on the storage system. High quality digital video may have a compressed bit rate of 20 Mbit/s (ATSC, 2001b). Playback of digital film and video requires high bandwidth both for the storage system and the network.

What makes this even more challenging, is that the storage systems managing large amounts of digital video, must be able to handle both the high bandwidth required for retrieving the video and the large amounts of storage required for

storing the video. This combination makes it much more complex to design and implement storage systems. To reduce the cost of storing large amounts of video, the price of the storage system is important. Unfortunately, the storage technology with the lowest price normally also has the lowest bandwidth. To retrieve video, the storage system must have a high bandwidth. To get the necessary bandwidth, more expensive storage devices might be needed. Thus, to get the optimal configuration of the storage system, the storage requirement and the bandwidth requirement can not be handled independently. In addition to requirements regarding the cost of the storage system, different applications have different requirements for throughput and response times, making this an even more challenging problem to solve.

These problems make it important to consider different storage technologies and storage architectures when implementing a large video archive. In order to select the best storage technology to use in a digital video archive, it is important to have knowledge about the properties of the different storage technologies. It is important to find out how the different storage technologies and storage architectures handle different load scenarios and to determine how to optimize the performance of the storage system. In this thesis, we address some of these issues by investigating serpentine tape for use as video storage and evaluating several tertiary storage technologies for use in a digital video archive. The technologies are evaluated based on performance and cost criteria.

1.2 Overview of the Work and the Contributions

The work done during this research project has been divided into four major phases. The first phase was on the VideoSTAR project. This was followed by the second phase, which focused on developing a prototype for a video archive server. In the third phase, we studied serpentine tape as a storage medium for digital video, while we in the fourth phase studied storage technologies and strategies for use in digital video archives. The main emphasis of this study is on the last two phases. In this section we describe each of the phases and provide an overview of the work and contributions.

VideoSTAR — Database Support for Video Information

In order to get into the research area, the first part of the work was done as part of the then ongoing VideoSTAR project. The VideoSTAR project focused on providing database support for video information management and support sharing of video information between applications. The project developed a prototype of a video database system. The work was performed in cooperation with the other members of the project team, which consisted of Rune Hjelsvold, Roger Midtstraum and Stein Langørgen. The VideoSTAR project was initiated and done as Rune Hjelsvold's dr.ing. work (Hjelsvold, 1995). The results from the project are

also presented in the following papers: "Searching and browsing a shared video database" (Hjelsvold, Midtstraum and Sandstå, 1995a), "A temporal foundation of video databases" (Hjelsvold, Midtstraum and Sandstå, 1995b), and "Integrated video archive tools" (Hjelsvold, Langørgen, Midtstraum and Sandstå, 1995).

The VideoSTAR prototype only supported sharing of video meta-data. The media data was stored using ordinary files. In order to be able to provide access to and share large amounts of video, the research on storage and delivery of video was initiated.

Video Server for Digital Video Archives

The first part of the work on storage and delivery of digital video, was to design and implement a video server. The first prototype, named Elvira, was implemented as part of a master thesis (Langørgen, 1994). As part of this doctoral study, the architecture and performance of this video server was evaluated. The results from this evaluation are presented in the paper "Video server on an ATM connected cluster of workstations" (Sandstå, Langørgen and Midtstraum, 1997).

Based on the experiences from the first video server, it was decided to design and implement a second prototype of a video server. The goal was to make a video server that was more efficient, had better scalability, and that supported storage of large amounts of digital video. The Elvira II Video Archive server has been designed and implemented. The architecture is presented in Appendix A. The research issues studied further in this thesis were motivated and selected based on the work on the Elvira II video archive server.

Serpentine Tape as Storage Medium for Digital Video

Compared to most other data types, digital video requires much storage space. When storing large amounts of digital video, the cost of the storage system is one of the main factors for deciding which storage technology to use. Today, magnetic tape is the storage technology which has the lowest storage cost. The main drawback of magnetic tape is the high access times.

The research performed in this part of the project is on optimizing the use of *serpentine* tape as a storage medium for digital video. The focus has been on reducing the access time for video sequences stored on tape. The result of the work is an access-time model for serpentine tape and a new scheduling algorithm for optimizing concurrent accesses to the tape. The access-time model and scheduling algorithm have been evaluated using Tandberg MLR1 and Quantum DLT 2000 drives. The main contributions from this part of the work are:

1. An *Access-Time Model* for serpentine tape that are more generic and has a lower cost for use in real systems than other proposed models. The access-time model has been presented in the paper "Low-cost access time model for a serpentine tape drive" (Sandstå and Midtstraum, 1999b).

2. A new scheduling algorithm for optimizing the utilization of serpentine tape. The scheduling algorithm is evaluated using simulations and practical experiments. The algorithm has been presented in the paper "Improving the access time performance of serpentine tape drives" (Sandstå and Midtstraum, 1999a).

3. An evaluation of use of serpentine tape for storage of video clips and images. The results from this evaluation has been presented in the paper "Analysis of retrieval of multimedia data stored on magnetic tape" (Sandstå and Midtstraum, 1998).

The access-time model and the scheduling algorithm are used in the last part of this research, which study use of different storage technologies for storing large amounts of digital video.

Large Scale Storage of Digital Video

To store large amounts of digital video requires a large scale storage system. During the last part of the work on this thesis, we study different storage technologies and systems for a digital video archive. The architecture for the video archive system is based on the architecture for the Elvira II Video Archive server. The experiments are performed using simulations of a video archive system.

The main focus is on how different storage technologies can be used for building a video archive. The storage technologies that are studied are magnetic tape, DVD, and magnetic disks. Different storage system configurations based on using these storage technologies are compared under different load scenarios. The main criteria for comparing different storage systems are performance and cost. For evaluating the performance of the different systems, both response time and throughput are studied. When comparing the cost of systems, both the storage cost and the cost of bandwidth, i.e., the cost of accessing a video are studied.

Two main configurations are investigated. In the first part of the work, a video archive consisting entirely of tertiary storage is studied. In the second part of the work, the video archive is extended to contain a disk cache for caching the most used videos. For each of these two main configurations, the effect of the number of drives, access distributions and allocation strategies are studied. The main contributions from this part of the work are:

1. **Video archive simulator.** An architecture and simulator for a digital video archive system based on tertiary storage and a disk-based video cache.

2. **Evaluation of tertiary storage technologies.** Three tertiary storage technologies are evaluated with regards to performance and cost for use in a video storage system. The evaluation includes a study of how improvements of the storage technologies will effect the overall performance of the storage system when used for storing and delivering video.

3. **Access distributions**. An evaluation of the effect different access distributions have on the performance of a tertiary storage system used for storing digital video.

4. **Allocation strategies.** Several strategies for allocating video sequences to storage media and library units are presented and evaluated for different storage system configurations and users loads.

5. **Caching video sequences on magnetic disk.** An evaluation of the effect of using a disk cache for improving the performance and reducing the cost of the video archive is performed. This study includes an investigation of the trade-off between the cache size and the bandwidth of the tertiary storage system.

1.3 Methodology

There exist many methods for doing research within computer science. In this section we give an overview of the scientific methods used for the research presented in this thesis. Most of the work is done as performance evaluations. Jain (1991) states that the three main techniques for performance evaluation are analytical modeling, simulations and measurements. Each of these methods has been used for doing various part of the research. In addition to these methods, parts of the work have been done by implementing working prototypes.

The work on the Elvira video servers was done by designing and implementing a working prototype for the video archive server. This prototype was used for performing experiments and measurements. In addition to performing the experiments, this gave valuable hands-on experience from working with a real system that stored and delivered video and provided input and ideas for the subsequent work performed in this thesis.

The access time model for serpentine tape was determined by first doing extensive measurements of how a Tandberg MLR1 drive performs when retrieving data from tape. Based on these measurements, the analytical model of access times was established. To establish the accuracy of the model, access times estimated by the model were compared against measured access times. A presentation of the validation process is included in Appendix B. The scheduling algorithms were evaluated both by use of simulations and practical experiments. The simulations used the access time model for estimating the access time of a schedule. The practical experiments estimated the total access time by using the scheduling algorithm and compared it against the measured time for executing the accesses on the tape. Both the access time model and the scheduling algorithms are validated by repeating the main experiments using a Quantum DLT 2000 drive.

Storage capacity		Network and
CS standard	non-CS standard	video bandwidth
KB = 1024 bytes	kbytes = 1000 bytes	kbit/s = 1000 bit/s
MB = 1024^2 bytes	Mbytes = 1000^2 bytes	Mbit/s = 1000^2 bit/s
GB = 1024^3 bytes	Gbytes = 1000^3 bytes	Gbit/s = 1000^3 bit/s
TB = 1024^4 bytes	Tbytes = 1000^4 bytes	Tbit/s = 1000^4 bit/s

Table 1.1 Data units used in the thesis.

The study of storage technologies for large scale storage of digital video is performed using simulations. The statistical validity of the simulation results are ensured by using confidence intervals for determining when to end each simulation run. Just as ordinary software, a simulator might contain errors that lead to wrong results. To validate that the results from the simulations are correct, several of the simulation results have been studied carefully. The overall performance results given by the simulator has been compared against the performance and utilization of each of the basic components in the simulation model. We have also validated some of the simulation results against known analytical models (see for instance Section 14.2.4).

1.3.1 Some Remarks on Terminology

In this thesis we study storage and delivery of video data. Although it is the same data that is stored on disk or tape and delivered using a network, normally two different units are used for describing the main property of storage media and network bandwidth. Storage capacity is normally given in bytes while network capacity is given in bit/s. Intuitively, with a byte consisting of eight bits, it should be very easy to convert between these two units. However, due to the use of the binary number system in computer science, this is not so straightforward, especially when we start to use prefix like *M* and *G* to the units.

The question is *"How many bytes are there in a MB (megabyte)?"*. The answer depends on whether you are a disk manufacturer or a database researcher. Most disk manufacturers claim that one MB is 1,000,000 bytes, while many database researchers will say it should be 1,048,576 bytes. In this thesis we use the units given in the first column of Table 1.1 when discussing the size of storage media and bandwidth for storage devices. Only when referring to specifications given by storage manufacturers, we will use the storage units given in the second column of Table 1.1. To clearly distinguish between these cases, we will respectively use the *B* and *bytes* as the basic unit.

When referring to bandwidth of networks or videos, we use the data units in the last column of Table 1.1. For network bandwidth, both producers of network equipment and the computer scientists have agreed on using the decimal

numbering system. To illustrate how this complicates the conversion between network bandwidth and storages space, lets consider one second of MPEG-2 compressed video having a bit rate of 8 Mbit/s. This video clip contains 8,000,000 bits or 1,000,000 bytes. In order to store this you will need 1,000,000 bytes or 0,95 MB of disk space. In this thesis, this conversion is done every time we have to convert between network/video bandwidth and storage capacity/bandwidth.

1.4 Outline of the Thesis

The first part of this thesis gives an introduction to digital video, storage technologies for digital video, and server technologies for storage and delivery of digital video:

- Chapter 2 introduces digital video, the technologies for creation and compression of digital video, and the main standards for digital video.
- Chapter 3 presents the main storage technologies that are relevant for storage and delivery of digital video in a video archive.
- Chapter 4 gives an introduction to video servers, storage systems for digital video, and the main research areas within storage and delivery of digital video.
- Chapter 5 gives an introduction to the problems related to storing large amounts of digital video in a video archive. The architecture for the video archive system studied in this thesis is presented together with an overview of the main issues we focus on in this thesis.

The second part is a performance evaluation of using serpentine tape as storage medium for images and video:

- Chapter 6 presents an access-time model for serpentine tape drives. The model is evaluated by experiments using a Tandberg MLR1 drive and a Quantum DLT 2000 drive.
- Chapter 7 presents a new scheduling algorithm for random I/O accesses to serpentine tape. The algorithm is evaluated using simulations and measurement against tape drives. The algorithm is compared to other known scheduling algorithm for serpentine tape.
- Chapter 8 contains an evaluation of using serpentine tape for storage and retrieval of multimedia objects.

The third part presents the work done on evaluating tertiary storage for use in large scale video archives. The work is based on using simulations of the video archive system. Two main architectures are considered. The first is a video archive based on tertiary storage only. The second architecture includes a disk-based cache in order to improve the performance of the video archive:

- Chapter 9 presents the architecture and design of a simulator for the storage system of a video archive. Detailed models for the tertiary storage technologies used in the study are presented.
- Chapter 10 presents the simulations and the results from studying the performance of a single library containing storage devices based on different tertiary storage technologies.
- Chapter 11 evaluates the performance improvement that can be achieved by improvements in the tertiary storage technologies.
- Chapter 12 contains an investigation of how different access distributions influence the performance of a large scale video archive. The video archive consists of multiple library units and is entirely based on using tertiary storage. The effect of the access distributions on the performance is evaluated for different user loads and tertiary storage technologies.
- Chapter 13 evaluates strategies for allocating video sequences to storage media and library units in a large scale video archive. The study investigate how the allocation strategies influence the performance of a video archive when using different tertiary storage technologies.
- Chapter 14 presents results from extending the video archive to include a disk-based cache. Both performance and cost of the archive are studied. An evaluation of how a disk cache can be designed and the resources used by a disk cache is included.
- Chapter 15 contains a study of how the storage technologies used in this thesis have developed during the last five years. The effects the improvement in storage technology have on the results presented in this thesis are discussed.

The fourth part contains the main conclusions and presents an overview of the main contributions from this work:

- Chapter 16 concludes the thesis, presents the contributions and outlines directions for further research.

The fifth part contains appendices that go into more details on some of the parts discussed earlier in the thesis:

- Appendix A gives an overview of the Elvira II video archive server that was designed and implemented as part of the work on this thesis.
- Appendix B contains a detailed analysis of the error rates for the access-time model presented and evaluated in Chapter 6.
- Appendix C contains an evaluation of the MPScan* scheduling algorithm using a Quantum DLT 2000 drive. This work is included for evaluating the MPScan* algorithm using an alternative tape drive.

Chapter 2

Digital Video Data

MPEG-2 is a complex (binary) file format for storing a complex data structure representing a complex compression format based on complex mathematical and physical properties on the three natural phenomena sound, live images, and time.

Anonymous

In this thesis, we study storage of digital video. In order to manage digital video, it is necessary to know the basic properties of digital video. The purpose of this chapter is to give an introduction to digital video. Readers who are familiar with digital video may skip this chapter.

We start the chapter with a presentation of video as a data type for use in computer systems. Following this, is a presentation of digital video, the advantages of having video on a digital format, and standards for digital video. In the last part of this chapter we give an overview of video compression and standards for video compression.

2.1 The Motion Picture Data Type

Today, there exists three main technologies for what is normally referred to as film or video. These technologies are *traditional film*, *analogue video*, and *digital video*. Common for all three is that they consist of *moving pictures* accompanied by *synchronized sound*. From a computer science view, these technologies are often referred to as variants of the *Motion Picture Data Type*. In this thesis we use the term *video* when referring both to the *motion picture* data type and to the different film and video technologies.

The motion picture data type differs from most other data types by two main properties. The first is the size of a video. Compared to other data types like numbers, characters and graphics, the amount of storage required to store a video is several magnitudes larger. A two hour movie compressed using MPEG-2 compression may require more than 4 GB of digital storage. The second property is

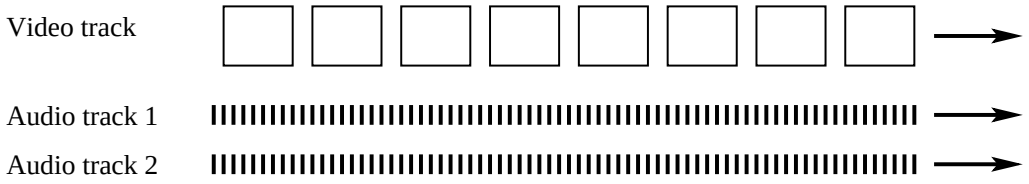


Figure 2.1 Illustration of how a video consists of an isochronous stream of images synchronized with two audio tracks.

that a video has a temporal dimension. When a video is played back, it is normally done frame by frame at a given speed. As illustrated in Figure 2.1, video has an inherently temporal dimension just by the way the storage medium allows for normal playback. In addition to the images, the video may also contain one or more audio tracks, which must be synchronized with the images during playback.

From a technical point of view, video is an *isochronous stream* of images that are presented to the user at a fixed rate (the *frame rate*). As an example, the European television standard PAL has a frame rate of 25 images per second. A two hour movie on this format contains 180,000 images. During playback, a new image has to be presented to the viewer every 40 ms. This fixed temporal relationship between the frames in a video imposes strict timing and synchronization requirements for systems that present video to users. Unlike traditional data types that should be delivered to a requesting user as *quickly as possible*, a system replaying video must adhere strictly to the given frame rate. If the system is not able to deliver the images on the correct time, the user will observe this as *jitter* (Ferrari, 1990).

For most films and videos, the images are accompanied by audio. The audio also represents an isochronous stream of data. While the images are presented with a frequency of 24–30 images per second independent of it being analogue or digital video, the audio has much higher frequencies. For digital audio the sample rate is typically in the area between 20 to 48 kHz. The audio and video streams have to be *synchronized* to avoid that users observe glitches between what happens in the video and when they hear the corresponding sound. Audio has gone from being mono, via different versions of analog and digital stereo till today's digital surround sound with five channels plus one separate channel for bass (5.1-channel sound).

2.1.1 Video Meta-Data

In addition to the physical information stored on the film or video medium that is required in order to play back each of the images in the video and to recreate

	Frame rate	Resolution	Technology
Film	24		analogue
NTSC	29.97	525 horizontal lines	analogue, interlaced
PAL[a]	25	625 horizontal lines	analogue, interlaced
Secam	25	625 horizontal lines	analogue, interlaced
HDTV	24, 25, 30	up to 1920 x 1080	digital, interlaced or progressive

[a]These data is for PAL-1. PAL-M used in Brazil has 525 horizontal lines and a frame rate of 30 fps.

Table 2.1 Motion Picture Formats.

the sound, there will normally be associated some extra information about the content of the video. This information is normally referred to as meta-data for the video. The main purpose of associating meta-data to videos (and multimedia documents in general) is to be able to search and browse collections of videos, or as defined by Jain and Hampapur *"video meta-data refers to data which is used in organizing video to facilitate content based retrieval"* (Jain and Hampapur, 1994). In order to efficiently support storage and retrieval of videos (and general multimedia documents), both the meta-data and the videos should be under control of a computerized system.

2.1.2 Motion Picture Formats

During the development of new technologies for film, video, and television, multiple standards have emerged. The main characteristics of the major film, video, and television standards are presented in Table 2.1.

For analog television, the main formats are NTSC, PAL, and SECAM. Most of the analog video formats are *interlaced*. Each frame is made of two *fields* that contain alternating lines. During each frame time, these two fields are shown on the screen. In addition to the formats that have been used for television broadcasts, numerous analog video formats have been developed for video recording. For the consumer market, VHS has been the dominating format.

Today, the digital video technology is out-performing the analog video technology, and digital video is gradually replacing analog video in most areas due to the higher quality that can be achieved. In several countries, digital television broadcast has started. Compared to analog broadcast both quality and resolution are improved. Most digital television broadcast systems will support *high definition television* (HDTV) with resolutions up to 1920 by 1080 pixels (Bhatt, Birks and Hermreck, 1997). In the consumer market, the DV (Digital Video) format is replacing Super-VHS and High-8 for capturing of video. For distribution of film, DVD is replacing the traditional VHS cassette, and with the introduction of recordable DVDs, it is likely to replace VHS as a recording medium. We provide more infor-

mation about DVD as a storage medium for digital video in Section 3.3.1.

Movies presented in cinemas world-wide are recorded using *traditional film*. The film is recorded using 24 frames per second. Unlike analog and digital video, where the resolution is limited by the number of lines or pixels recorded, the resolution of traditional film is only limited to the physical properties of the film media. Analog video has never been able to compete with traditional film when regarding quality, and until recently, traditional film has been the only format with a quality good enough for playback in a large cinema auditorium. With the introduction of high definition digital video, it is possible to achieve a quality that is equal or better than what can be achieved using traditional film.

2.2 Digital Video

With the introduction of the video disc in the late 1970s, it was possible to make computer systems handling large amounts of video (Fox, 1991). On the video disc, both the video and the audio are stored in an analog format. The random access capability of the interactive video disc made it possible to use computers for *controlling* the playback of the video. In most systems, the video was transferred directly from the video disc to the monitor. The only interaction from the computer on the video content was to provide textual or graphical overlays.

For a computer to be able to store and to perform operations on a video, the video has to be on a *digital* format. In the physical world, both video and audio are analogue in nature. To create digital video, the analog signals that the video and audio consist of must be transformed into corresponding digital signals. The process for doing this conversion is called *digitalization*. Digital video is thus analog video and audio signals that have been *digitized*.

2.2.1 Advantages of Digital Video

Traditional film and analog video have been a great success for many years. During the last years, digital video has increased its popularity and is starting to replace analog video in some areas and has created new areas where video can be used. The reason for this is that digital video has some important advantages compared to traditional film and analog video. The main advantages are:

- **Durability.** Analog video and film are not robust against tear and wear. Each time an analog video or a film is played back, the physical media is subject to physical forces that degrade the quality of the video signal or images. When the video is stored on a digital format it is possible to control the aging by reading the media at regular intervals. Errors that occur during reading can be handled by including error correction codes as part of the data or by having *identical* copies on backup.

	Video resolution (v × h)	Uncompressed data rate	Compressed data rate
CIF	352 × 288	37 Mbit/s	384 kbit/s
SDTV (ITU-R BT.601)	720 × 480/576	165 Mbit/s	4–8 MBit/s
HDTV (ITU-R BT.709)	1920 × 1080	1485 Mbit/s[a]	20–40 Mbit/s
Two channel audio		1.4 Mbit/s[b]	
Five channel audio		3.5 Mbit/s[b]	

[a] This is one of several possible data rates defined by the standard.
[b] The audio data rates are for 16 bits PCM coded audio sampled at 44.1 kHz.

Table 2.2 Standards for digital video and audio.

- **Quality.** One of the main advantages of digital video over analog video is the higher quality that can be achieved and maintained during both editing and copying. When the video is digital it is possible to make copies that are identical to the original.

- **Indexing and archiving.** When having the video on a digital format it is possible to include both indexing information and the video itself in *one* integrated system.

- **Accessibility.** Having the video stored in a computer system makes it is possible to have *direct* and *easy access* both to the video, parts of the video, and to the individual frames of the video.

  This makes *editing* of videos and reuse of video footage much easier compared to working with analogue video. Having the videos available to computer applications opens up for a whole world of new possibilities in how video can be used in different kind of applications. Storing the video on a computer system also makes it possible to have *remote access* to waste amounts of video footage across a network.

- **Distribution.** Film and analog video are today distributed to the viewers either by using a physical media or through broadcasting, either terrestrial, satellite or cable network. All these processes incur degradation of the quality either during copying of the media before distribution or of the broadcasted signal. If the video is digital, the distribution can be made both easier and without loss of quality.

2.2.2 Standards for Digital Video

To exchange digital video between different applications and to present the video using products from different vendors require standards for the format of the

video. In this section we present some existing standards for digital video formats. These video standards specify the format of uncompressed video. Digital video on these formats is often used as input for video compression. Video compression and standards for digital video compression are presented in Section 2.3.

An overview of some representative video standards is given in Table 2.2. The table gives the resolution of the video and the corresponding data rate for uncompressed and compressed video. The table also contains representative data rates for stereo and multichannel surround audio. In this introduction we only cover the most common standards for digital video for use in video conferencing and digital television.

Video Telephony and Video Conferencing

The Common Intermediate Format (CIF) was defined by CCITT (International Consultative Committee for Telephone and Telegraph) to be a standard for exchanging digital video. It has a resolution of 352×288 pixels and the uncompressed data rate is approximately 37 Mbit/s (Tekalp, 1995). It was mainly developed to support video telephony and video conferencing. By using H.261 compression (see Section 2.3.1) the data rate can be reduced to $p \times 64$ kbit/s, which are the ISDN rates (Liou, 1991). Typically, video conferencing using the CIF format requires 384 kbit/s ($p = 6$). The CIF video format is also the basis format used for MPEG-1 compression.

Standard Definition Television

Standard Definition Television (SDTV) refers to digital television with specifications similar to today's analog television systems (Tekalp, 1995). The ITU-R (International Telecommunication Union – Radiocommunication) Recommendation BT.601 (ITU-R, 1995) defines the video to be interlaced with a standard 4:3 aspect ratio or a wide-screen 19:6 aspect ratio. Resolutions defined by ITU-R BT.601 are given in Table 2.2. The raw data rate for video in the ITU-R BT.601 format is 165 Mbit/s. By the use of e.g., MPEG-2 compression, the data rate of ITU-R BT.601 video can be reduced to between 4 and 8 Mbit/s.

High Definition Television

High Definition Television (HDTV) refers to television systems with much higher resolution and quality than today's television systems. HDTV supports resolutions that are doubled both in the horizontal and vertical dimensions compared to conventional television. ITU-R Recommendation BT.709 (ITU-R, 2002) specifies HDTV to use the High-Definition Common Image Format (HD-CIF) format, which has a resolution of 1920×1080. The high resolution of the video results in very high data rates for the uncompressed video. For example, interlaced video

with 25 frames per second will have a data rate of 829 Mbit/s. By using compression, this can be reduced to 20 to 40 Mbit/s without noticeable loss of quality.

HDTV as well as SDTV are supported by the new digital television systems being developed. Digital television is likely to become one of the main ways for users to request film and video to be delivered from video servers and digital video archives (Milenkovic, 1998). In addition to being a medium for television broadcast and delivery of digital video, digital television will also provide other interactive services. In the US, the Advanced Television System Committee standard (ATSC) will be used for implementing digital television (ATSC, 2001b). Digital Video Broadcast (DVB) is a corresponding standard developed for use in Europe (ETSI, 2000). DVB is using ITU-R BT.709 as the standard format for HDTV[1]. Both systems uses MPEG-2 for the video compression. For the audio compression, MPEG or Digital Audio Compression (AC–3) (ATSC, 2001a) are used. The encoded and compressed video and audio are packaged using MPEG-2 transport streams.

2.3 Digital Video Compression

The data rates following from digitizing a high resolution video are in the order of 100 – 1500 Mbit/s. These data rates are hard to handle for both the computer system that should store and process the video and for the communication network that must deliver the video. With today's technology, only use of compression makes it possible to store and transmit large amounts of digital video. Fortunately, video data contains redundancies which make it possible to compress the digital video to only a fraction of the original size.

The main properties that make it possible to efficiently compress the digital video are (Gonzalez and Woods, 1992):

1. **Coding redundancies.** After the digitalization process, the video and audio are coded using a set of scalar values. Each of the scalar values represents a given signal value. Some of the scalar values are more frequently present in the digital video or audio. To reduce the amount of data, the most frequently occurring sample values could be stored using a less number of bits, while the lesser frequently sample values are coded using a higher number of bits. This is commonly referred to as *variable-length coding* (Gonzalez and Woods, 1992). The best known and most used algorithm for variable-length coding is Huffman coding.

2. **Inter-pixel redundancies (spatial redundancies).** Within one video frame, it is common that the value of neighboring pixels are strongly correlated. This can be used for compressing the frame. To achieve a highest possible effect of the inter-pixel redundancies within a video frame, the video

[1] ATSC is based on the corresponding standard named SMPTE 274M (SMPTE, 1998).

frame is usually transformed into an alternative representation as part of the compression process. The two most used transformations are the Discrete Cosine Transform and the Wavelet Transform (Hilton, Jawerth and Sengupta, 1994).

3. **Psycho-visual redundancies.** Some of the information that is contained in video or audio signal is more or less ignored by the human brain. This can be used to do compression based on human perceptive features. For example, in the parts of an audio signal that contain low frequencies at a high volume, higher frequencies will not be heard by the human ear. Similar effects exists for images and video. Since these parts of the signal are ignored by the human brain, there is no reason for including these parts of the original signal in the compressed signal.

4. **Temporal redundancies.** Between adjacent frames in a video sequence there is normally very little change. This can be used for compression. Instead of storing each of the frames, only the difference between the two frames needs to be stored. The information contained in the difference is normally much less than the corresponding frames and is therefor much better suited for compression. Also techniques like *motion estimation* are used for compression of neighboring frames in a video (Haskell, Puri and Netravali, 1997, Chapter 6).

Based on these four properties of video, different compression algorithms have been developed. The most sophisticated algorithms use all of the properties to achieve a highest possible compression rate. Coding strategies that only uses the first three properties are called *intraframe* compression. Coding strategies that also uses temporal redundancies between frames are refereed to as *interframe* compression.

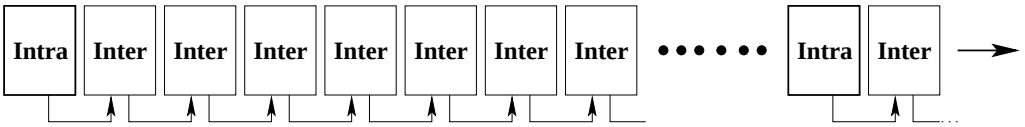
Compression strategies fall into one of two categories, *lossless* and *lossy compression*. In lossless compression, the decompressed video will be identical to the original video. In lossy compression, some of the information is lost during the compression and the decompressed video will only be an approximation of the original video. To achieve the compression rates that is necessary for digital video, most compression strategies for video are *lossy*.

2.3.1 Standards for Digital Video Compression

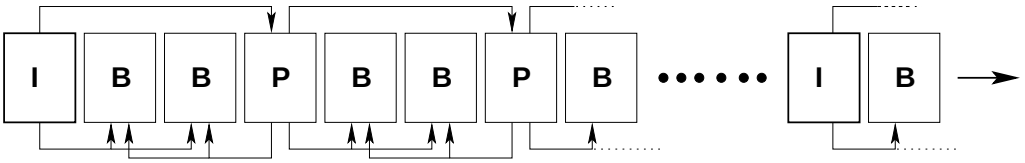
There exists multiple standards for coding and compression of digital video. In this section we present some of the most important standards. For each of the standards, we give a short introduction of the main features.

Motion-JPEG JPEG is an ISO standard for compression of still images (Wallace, 1991). Is is also used for compression of digital video. When used for digital video, each frame within the video is compressed individually. This

(a) JPEG



(b) H.261 and H.263



(c) MPEG video compression

Figure 2.2 Use of intraframe and interframe compression for some video compression algorithms.

is shown in Figure 2.2(a). Since interframe compression is not used, it is easy to get access to the individual frames. Because of this, Motion-JPEG compressed video is well suited for video editing. The drawback of not using interframe compression is a relatively low compression rate, typically around 15:1 (Furht, 1994). Another drawback of Motion-JPEG is that only the compression of the individually frames is standardized. There exists no standard for a Motion-JPEG file format.

H.261 and H.263 The ITU-T H.261 and H.263 video compression standards were developed to implement video conferencing over ISDN, regular phone lines and LAN. H.261 supports the CIF and QCIF (Quarter-CIF) video formats while H.263 supports resolutions up to 16CIF (1408 $\times$ 1152 pixels). These standards use interframe coding with motion compensation (Liou, 1991). Figure 2.2(b) contains an illustration of the interframe coding used by these standards.

MPEG-1 The MPEG-1 (ISO/IEC 11172) standard (MPEG-1, 1992) was the first standard for compression of digital video from the Motion Picture Expert Group. MPEG-1 was targeting compressed video with a data rate that could

be delivered by a CD-ROM drive (Le Gall, 1991). With a video resolution of 352 $\times$ 288 and datarate of 1.2 Mbit/s for video and 256 kbit/s for audio, the video quality is comparable to analog VHS. MPEG-1 uses a complex interframe coding scheme as shown in Figure 2.2(c).

MPEG-2 The MPEG-2 (ISO/IEC 13818) standard (MPEG-2, 2000) is the successor for the MPEG-1 standard. It supports compression of high quality video and is aiming to support broadcast quality video for Standard Definition TV (SDTV) and High Definition TV (HDTV). In addition to support video delivered from a local storage device, it also supports delivery over computer networks, cable television networks, ground based and satellite based broadcast. For supporting applications to access stored videos and control the video playback, MPEG-2 includes a protocol of generic commands called *Digital Storage Media – Command and Control* (DSM-CC) (Balabanian, Casey, Greene and Adams, 1996).

MPEG-4 The MPEG-4 (ISO/IEC 14496) standard is aiming for delivery of multimedia streams to a wide range of devices, from video at very low bit rates delivered to mobile devices to high quality video for broadband internet services. In addition to video and audio, the standard also supports coding and compression of more general multimedia content (Koenen, 1999). Multimedia content is delivered as streams of multimedia objects where each object has an independent coding. The multimedia objects can be both synthetic and natural objects. Compared to earlier MPEG standards, the MPEG-4 standard gives the viewer much better possibilities for interacting with the multimedia objects.

In addition to these standards defined by international standardization organizations, there exists numerous industry standards. For the consumer video recording market, the DV (Digital Video) is the most widely used format. DV is an industry standard for recording digital video. DV uses only intraframe compression and is able to achieve a 1:5 compression rate. The resulting video stream is approximately 25 Mbit/s.

For the personal computer and the internet, there exist multiple formats for downloading and playing video. The main formats are Quicktime from Apple, the Real format from Real Networks, DivX from DivXNetworks and Windows Media from Microsoft. All of these, with the exception of DivX, have their own proprietary compression standard and video format, but also support playback of most of the other formats. DivX uses the MPEG-4 standard for video compression.

Chapter 3

Storage Technologies for Digital Video

A greater quantity of information made available at a lower price does not necessarily result in improved knowledge.

Martin Fransman
in *Information Regarding the Information Superhighway and Interpretive Ambiguity*
(Fransman, 1996)

In this chapter we give an introduction to the storage technologies that are most likely to be used by systems that store and deliver digital video. When deciding which storage technology to use in an application, it is important to know the characteristics and properties of the storage technologies. Table 3.1 contains an overview of the most important properties for different storage technologies. These properties can be used when comparing different storage technologies. In addition to comparing storage technologies by using these properties, it is necessary to consider the requirements of the application. For example, a video archive where the main goal is to provide safe storage for archiving large amounts of video will likely rate the relative importance of the properties differently than a video server where the main goal is to deliver as many concurrent video streams as possible.

Digital video has two important properties that influence how well different storage technologies are suited for use in systems that manage digital video. The first property is the size of digital video, which is much higher than for most other types of digital data. A two hour MPEG-2 compressed video can easily require more than 4 GB of storage. With such storage requirements, one terabyte of storage is only able to store 500 hours of digital video. The second property is that the video must be delivered to the user as an isochronous stream of data. This data stream has stringent timing requirement, every second a fixed numbers of frames have to arrive at the client. Most current storage devices are not optimized

Property	Description
Access time	The amount of time from the application issues a request for data stored on a device until the data is available for the application.
Transfer capacity	The average amount of data the storage device is able to read or write. This is often referred to as the *bandwidth* or *transfer rate* of the device. The transfer capacity is normally given in either MB/s or Mbit/s.
Storage capacity	This is used both for referring to the amount of data that can be stored on one storage device, but also how much data it is possible to store per volume unit.
Cost per MB	The cost of storing one MB of data on a device.
Durability	The amount of time that a storage device is expected to reliably store the data.
Mobility	How easy it is to move a storage medium (without loosing the data stored on it), e.g., for use in distribution of data, or for moving the device to a remote backup location.

Table 3.1 The most important properties for storage technologies. This table is based on a presentation found in (Bratbergsengen, 1997).

for delivering data streams of a given rate, but instead optimized to deliver the data as fast as possible. One of the main challenges in video servers is to optimize the use of the storage media's bandwidth. We discuss this further in Chapter 4.

Storage technologies are usually classified into *primary*, *secondary*, or *tertiary* storage. This classification is normally based on the performance characteristics of the storage device. Primary storage is memory that is directly addressable for the processor. The most common form is the semiconductor memory (RAM) used as "main memory" of the computer. The access times of primary memory is typically measured in nanoseconds. The most common form for secondary storage is magnetic disks that are permanently connected to the computer. Compared to primary memory, secondary memory has approximately 100 times higher storage capacity, 100,000 times higher access time, and costs approximately one percentage of what primary memory costs. Tertiary storage refers to storage that exceeds secondary storage by at least one order of magnitude for both access time, storage capacity per drive, and cost per MB (Hillyer and Silberschatz, 1996c). Today, mainly *removable* media like optical disks and tape falls into this category.

In Table 3.2 we present some representative performance and cost characteristics of four different storage technologies that belong to each of the three storage

Technology	Access time	Storage capacity	Bandwidth	Cost per GB
SDRAM	4–12 ns	32–1024 MB	800–4400 MB/s	200
Magnetic disk	5–10 ms	10–200 GB	10–66 MB/s	5
DVD-R	100–500 ms	4.5–9.0 GB	1.3–8 MB/s	0.5
Magnetic tape	10 s – 2 min	10–300 GB	2–36 MB/s	0.5

Table 3.2 Some performance characteristics for different storage technologies in 2004. The cost numbers are given in US dollars per GB.

classifications. The first is SDRAM, which is typically used as main memory for computers. The second is magnetic disk, which is the most important secondary storage technology today. The last two, DVD and magnetic tape, are examples of tertiary storage. We present more detailed examples of the difference in performance and cost between different storage technologies in Section 9.3 and 9.4.

Systems that store and deliver digital video may use multiple storage technologies for storing the video. Which storage technology to use depends on requirements like access time and storage cost. In the remainder of this chapter, we present the main storage technologies that are likely to be used in a digital video archive.

3.1 Semiconductor Memory

Semiconductor memory is primary storage used as main memory, caches, and registers in computers. The most used semiconductor memory is *random access memory* (RAM). The main advantage of RAM is speed, the drawbacks are that the storage is non-permanent (as soon as the electricity is turned off, the data is lost) and rather costly. RAM comes in two main versions, static (SRAM) and dynamic (DRAM). SRAM is able to keep the data, while data stored in DRAM has to be periodically refreshed, typically every few milliseconds. Due to the lower complexity and lower cost, DRAM is the version used as main memory in most computers. In order to reduce the negative effect of the data refresh and to improve the access time and transfer rate, numerous improved versions of DRAM have been developed. Among these are SDRAM (synchronous DRAM), DDR SDRAM (Double Date Rate SDRAM) and RDRAM (Rambus DRAM).

The price of DRAM has been greatly reduced during the last years and the relative price ratio between RAM and magnetic disk is decreasing. Still, RAM is yet not cheap enough to be used for storing large amounts of digital video. With a price of approximately 0.2 dollar per MB, to store a two hour video with an average bit rate of 5 Mbit/s in RAM would cost more than $800. In today's video servers, the main memory is mainly used as a buffer for caching parts of

videos that are currently being delivered. Optimal use of the buffer can reduce the amount of video data that has to be read from secondary or tertiary storage and thus improve the performance of the video server. In Section 4.3, we discuss the use of main memory as a buffer cache for delivering video.

3.2 Magnetic Disks

Magnetic disks are the *work horses* of most of today's video servers. For most video servers, the video is stored mainly on magnetic disks. Each time a user requests playback of a video, it is usually a disk-based storage system that is responsible for delivery of the video to the user (via the video server's main memory buffer).

A magnetic disk consists typically of three to nine round disks mounted on a disk spindle. Both sides of the disks are coated with a magnetic recording material. For each disk surface there is a disk head that is mounted on a disk arm, which is used for reading and writing data. Data is written to the disk in circular *tracks*. Each track is divided into *sectors* which is the basic storage unit for a disk. A sector typically holds 512 bytes of user data. Magnetic disks rotate with a constant number of rotations per minute (rpm), typically from 3600 rpm to 15,000 rpm. Disks with a fixed number of rotations per minute are called *constant angular velocity* (CAV) speed disks.

Internally, data on a disk can be addressed by using an address format consisting of the following numbers: `(cylinder, track, sector)`. The *cylinder* number specifies the set of tracks that has the same distance (in number of tracks) from the edge of the disks. The *track* number specifies the disk surface where the data is stored. The *sector* number specifies a sector within the track. On modern disks, this address format is only used by the disk controller. By using standardized interfaces for I/O devices like SCSI, the disk is presented to the host computer and operating system as a linear array of data blocks.

Traditionally, magnetic disks have had a fixed number of sectors per track, making the physical length of a sector close to the edge of the disk longer than a sector close to the center of the disk. As a consequence, the storage density (bits per area unit) is higher close to the center of the disk. To better utilize the storage capacity of the disk surface, newer disks are often divided into *zones*. A zone is a collection of cylinders on the disk. To increase the storage capacity of the disk, the number of sectors per track increases when we changes from one zone on the disk to a zone that is further from the center of the disk. In addition to increasing the storage capacity of the disk, the use of zones also changes the behavior of the disk. A disk with a fixed number of sectors will have a transfer rate that is independent of which cylinder it reads from or write to. A multi-zone disk will have a higher transfer rate when reading data on a cylinder close to the edge of the disk than when reading data on a cylinder close to the center of the

disk. Van Meter (1997) has developed a disk model for multi-zone disks. The use of multiple zones on a disk complicates optimization of the disk bandwidth, but it can also be used for placing data that requires higher transfer rate close to the edge of the disk.

Several analytical models for the performance of magnetic disks have been developed. One of the first detailed models of the performance was made by Ruemmler and Wilkes (1994). A more general analytical performance model for disk devices is presented in (Triantafillou, Christodoulakis and Georgiadis, 2002). This model supports both traditional disk layout as well as zoned disks. It also supports both CAV (constant angular velocity) and CLV (constant linear velocity) disks. During the years, the disk controller has been extended to include software for optimizing accesses to the disk and memory for caching disk blocks for both read and write operations (Worthington, Ganger, Patt and Wilkes, 1995). A more detailed disk model that takes these features of the disk controller into account is presented in (Shriver, 1997; Shriver, Merchant and Wilkes, 1998). An evaluation of how disk scheduling algorithms are influenced by the optimizations done by the disk controller is found in (Worthington, Ganger and Patt, 1994).

Since disk bandwidth is one of the most important resources for delivering video from a video server, much work has been done on optimizing the use of disks. We present some of this work in Section 4.4.1.

3.3 Optical Storage Media

Optical storage media are named so because they are read and written using laser light. The intensity of the reflected laser light is used for determining the bit pattern of the data stored on the medium. For most of the optical media technologies, the media and drives are sold separately. In order to read data from the medium, it first has to be inserted into a media drive. Due to this, most optical media types are regarded as tertiary storage. Optical media are available as both disks and tapes.

Based on the support for writing and updating of data stored on the medium, optical storage technologies can be divided into one of the following categories:

1. **Read Only Medium** (ROM). This is media where the data has been written during the production process. This is a suitable medium for distribution of software and multimedia content like audio and video.

2. **Write Once Read Many** (WORM). Optical media in this category are normally written by using a laser with higher effect than the laser used for reading the medium. By heating the storage medium, it is possible to change the reflection properties of laser light for the medium (Steinmetz and Nahrstedt, 1996). This is used for writing data to the medium.

DVD-ROM	Read only DVD, including a file system.
DVD-R	Writable DVD, can only be written once.
DVD-RW	Re-writable DVD medium, tailored for sequential access.
DVD-RAM	Re-writable DVD medium, optimized for random access.
DVD-Audio	For distribution of digital audio.
DVD-Video	For distribution of digital video and audio.

Table 3.3 Overview of the main DVD formats.

3. **Multiple Writes.** Optical media in this category can be re-written many times. Most of the media types use magneto-optical methods for writing the data. The storage medium consists of a magnetic material where it is possible to change the direction of individual dipoles (Steinmetz and Nahrstedt, 1996). During reading, the direction of the magnetic field within a dipole determines how much of the laser light that is reflected. Writing of data is done by heating the medium and using a stronger magnetic field to change the direction of the individual dipoles.

There exists many different technologies for optical storage media. So far, the most successful has been the *Compact Disk*, which is available for distribution of audio (CD-DA), distribution of software and data (CD-ROM), and in several writable versions. In the remaining of this section, we give an introduction to the DVD technology. The main reason for focusing on this technology is that the DVD was primarily developed for storage of digital video. In the investigation of tertiary storage technologies for use in digital video archives, DVD is one of the technologies that is studied.

3.3.1 DVD

DVD, which stands for *Digital Video Disc*, *Digital Versatile Disc*, or just *DVD*, is the successor of the highly successful Compact Disc (CD). It has the same physical format as the CD and is intended for storing video, audio, and data.

Compared to the CD, the storage capacity has been increased. The storage capacity has been increased by improving the storage density, by using both sides of the disc and by adding a second storage layer to each side of the disc. Thus, DVDs support storage sizes from 4.38 GB (single-sided, single layer) to 15.9 GB (double sided, dual layer) per medium. Compared to the standard CD-ROM, this is a 25 times increase of the storage capacity.

The standard transfer rate of a DVD drive is 11.08 Mbit/s (1.32 MB/s). This is sufficient for most video and audio applications. For computer applications, drives are available that are capable of reading the data at a much higher speed. Unlike hard disks which use a CAV speed, DVD drives originally read the disk using a *constant linear velocity* (CLV) speed. This means that the relative speed

between the disc medium and the laser is constant. Thus, the number of rotations per minute increases as the drive approaches the center of the disc. In order to make the DVD more suitable for use in computer applications that perform random accesses for data stored on a DVD disk, drives operating as CAV drives have been introduced. By operating as a CAV drive, the transfer rate can be improved and the seek time reduced, particularly close to the outer edge of the disk. To improve the transfer rate, alternative data layouts have been proposed. P-CAV (partial constant angular velocity) is a combination of CLV and CAV. Close to the center of the disk, it is read using CAV, but the drive switches to CLV operation when it get closer to the edge of the disk. ZCLV (zone constant linear velocity) divides the disk into several zones. Within each zone the disk is read with CLV speed. The different zones have a different speed (Sadashige, 2000).

For supporting applications with different storage requirements, a number of standards have been defined for the DVD format. An overview of the main DVD formats is given in Table 3.3. Four of these standards define the physical properties for data storage on the DVD medium (Bell, 1999). For mass distribution of software and multimedia content, the DVD-ROM is the main format. For applications needing to store data, write-once DVD-R and re-writable DVD-RW media are available. For applications that require frequent updates of the data on the disk, DVD-RAM has been developed (Sadashige, 2000). All of the media formats support a common file system named the Universal Disc Format (UDF) (Bell, 1999).

Based on the UDF specification, application formats are defined. Currently the DVD-Video and DVD-Audio formats have been defined for mass distribution of video and audio (Taylor, 1999). The DVD-Video standard specifies the video content to be MPEG-2 encoded. For the audio, up to eight tracks of multichannel PCM, Dolby Digital, MPEG-2, or DTS (Digital Theater Systems) are supported (Taylor, 1999). A single-sided, single layer DVD is able to store more than two hours of high-quality compressed MPEG-2 video with an average data rate of 5 Mbit/s.

3.4 Digital Tape Technologies

Traditionally, tape has been the medium for storage of huge amounts of data and data archiving. A number of different technologies are used for tape storage, based on both magnetic and optical storage technologies. In this section we present the main storage technologies for magnetic tape.

Tape offers the highest storage densities of current storage media. Unfortunately, tape is a sequential access device with access times up to a few minutes. In addition to having sequential access, if the tape is not already mounted in a tape drive it has to be transported to the drive and mounted in the drive, which might add a few more minutes to the access time. Because of this, tape has in

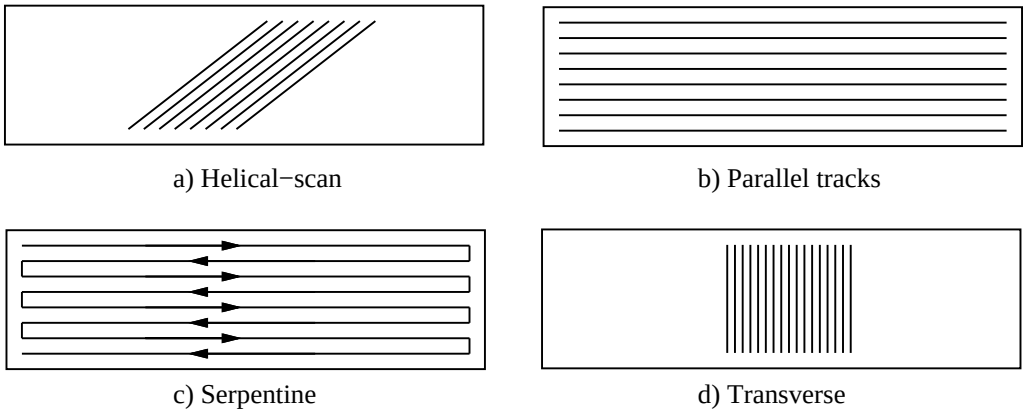


Figure 3.1 The main tape technologies.

the last years mainly been used as an archival medium. But with the increasing storage requirements of new applications like digital libraries and video archives, tape might once again be a storage alternative for providing enough storage at an affordable price.

Tapes are available in two forms, cartridge and cassette. A cartridge contains only one reel. When a cartridge is loaded in a tape drive, the tape is mounted onto a second reel which is part of the drive. A cassette has both reels inside the tape chassis, making the loading and unloading process less complex (Prabhakar, Agrawal, Abbadi and Singh, 1996).

Most tape drives offer compression of the data as the data is written to the tape. For many applications, this can double the amount of data that can be stored on each tape. Based on this, most tape vendors give performance data for their drives where they have expected that the drive's compression is able to double both the storage capacity and the bandwidth of the drive. Since digital video normally already is compressed, applications storing digital video on tape will not get any effect from the tape drive's built-in compression.

Tapes are categorized by how the tape is written. There are three main categories of digital tape technology. These are called *helical-scan*, *linear*, and *transverse*.

Helical-scan Tape

A helical-scan tape drive writes vertical or diagonal tracks on the tape. Figure 3.1(a) shows an example of how diagonal tracks are placed on a helical-scan tape. Several standards use helical-scan technology. The most well-known are 4mm (DAT), 8mm (video), and 19mm (Hillyer and Silberschatz, 1996c). The popular analog VHS video cassette also uses helical-scan technology. Tape drives using this technology usually obtain high data densities and high transfer rates. The reason is that the head is mounted on a rotating cylinder which scans the tape

as it passes by.

It has traditionally been believed that tape based on helical-scan technology has problems when used by applications that require random I/O operations on a tape, due to wear-out caused by the rotation of the head. This might be a myth, since there have been studies showing that tape drives based on helical-scan technology (Exabyte 8505) did not have problems with tapes that had been subject to more than 10.000 passes (Schuett, Katz and Chervenak, 1997).

Linear Tape

Common for tape drives that are based on *linear tape technology* is that the data is written in parallel tracks along the tape. Two main technologies exist that use parallel tracks on the tape. The classical tapes used by mainframes are called *parallel* because all the tracks along the tape are written and read in parallel (see Figure 3.1(b)). A newer technology for linear tapes is the *serpentine* technology. A serpentine tape drive writes one track (or group of tracks) down the tape (forward direction), then writes the next track (or group of tracks) in the opposite direction (reverse direction) as seen in Figure 3.1(c).

Today, there exists three main technologies for serpentine drives:

QIC – Quarter Inch Cassette – initially a standard for making inexpensive tape storage with modest capacity and bandwidth, but during the last years both the storage capacity and transfer bandwidth have been increased (QIC Development Standard, 1994). As the name implies the width of the tape is a quarter of an inch. QIC tapes have both reels inside the cassette. QIC provides standard tape storage formats covering the range from 60 MB to 140 GB (native) and have transfer rates up to 6 MB/s.

DLT – Digital Linear Tape – is a technology developed by DEC (Digital Equipment Corporation) (Lignos, 1995). It uses a half inch wide tape which is stored in a cartridge having only one reel. The second reel is a part of the tape drive. When inserting a DLT tape into the drive, the tape first has to be mounted onto this reel. DLT tapes exists in the range from 10 GB up to 300 GB native capacity. DLT drives support transfer rates up to 36 MB/s. The newest tapes and drives in the DLT series are called Super DLTtape.

LTO Linear Tape - Open Technology is a tape technology developed by Hewlett-Packard, IBM and Seagate (Linear Tape-Open Technology, 1998). The LTO standard specifies two different technologies, Accelis and Ultrium. Accelis is intended for transaction-processing applications, while Ultrium is mainly meant for backup systems. At the moment (2004) only drives and tapes based on the Ultrium standard are available. Ultrium tapes are available with native capacity of 200 GB (native) and drives with transfer rates up to 35 MB/s. The LTO cartridge contains an electronic chip (LTO-CM) that

maintains information about location of files on the cartridge. By using this information, loading/unloading the tape and locating files on the tape can be done more quickly. This chip can also be used by applications to store application dependent information.

Hillyer and Silberschatz have developed detailed analytical performance models for two serpentine tape drives. The first performance model is for the DLT 4000 drive (Hillyer and Silberschatz, 1996a). The second model is for an IBM 3570 Magstar MP drive (Hillyer and Silberschatz, 1998), which is a tape drive optimized for random accesses.

Transverse Tape

A transverse tape drive writes the data in tracks that are *transverse* to the length of the tape. This is shown in Figure 3.1(d). When data is written, the tape is stopped, and it is the head that moves sideways writing one track. As one track is finished, the tape *steps* forward before writing the next track. Transverse tape systems are mainly used in systems that perform long-duration recording of low data rate sensor information (Hillyer and Silberschatz, 1996c).

3.5 Storage Architectures

In the last section of this chapter we give a short introduction to some strategies for organizing multiple storage devices in order to improve some of the main properties of the storage system. The first technology we consider is storage device arrays, where the goal is to increase the bandwidth and the reliability of the storage system. The second case is hierarchical storage management systems, which utilize different storage technologies to reduce the storage cost while still providing fast access to the most frequently accessed data.

3.5.1 Arrays of Storage Devices

According to Moore's Law, processing capacity is doubled every 18 months. Compared to the processing capacity, transfer rate of external storage devices increases at a much lower rate (Gray and Shenoy, 2000). This has lead to an increasing mismatch between internal bandwidth between the processor and the main memory and the bandwidth of external storage devices.

To improve the I/O bandwidth of storage devices, several strategies have been proposed. Most of these group multiple storage devices to form an array of storage devices. The goal is that for the application using it, this array of storage devices will provide a bandwidth close to the aggregate bandwidth of all the devices. The two basic strategies for data layout are *mirroring* and *striping* of the

data. Mirroring increases the transfer rate for reading by storing the data on multiple disks (Bitton and Gray, 1988). Striping increases the transfer rate for both reading and writing by distributing the data on all the disks (Salem and Garcia-Molina, 1986). Unfortunately, these two strategies have some major drawbacks. Mirroring doubles the cost of the storage system, while striping makes the system vulnerable to disk crashes. If one disk fails, the data on all disks becomes unavailable.

To reduce the problems of basic mirroring and striping, a set of disk array architectures referred to as RAID (Redundant Arrays of Inexpensive Disks[1]) has been proposed. Originally, this set consisted of five disk layout strategies named RAID level 1 through RAID level 5 (Patterson, Gibson and Katz, 1988). This has later been extended to seven RAID levels (Chen, Lee, Gibson, Katz and Patterson, 1994). With the exception of RAID level 1, which is similar to mirroring, RAID uses data striping to increase the bandwidth, and parity data to increase the reliability of the storage system.

The different RAID levels have different performance characteristics and storage costs. There exists systems which combine multiple RAID levels for improving the performance and reducing the storage cost. On example is the HP AutoRAID (Wilkes, Golding, Staelin and Sullivan, 1995). This is a two-level storage hierarchy implemented inside a single storage array. In the upper level, mirroring of data is used for improving the performance. In the lower level, RAID level 5 is used for providing protection of the data at a lower cost than by mirroring.

For applications like database management systems, the number of accesses the storage systems is able to serve is more important than the transfer rate. With the exception of RAID level 1, reading data from the RAID system requires potentially more than one disk to participate in the read operation. For use in applications that perform a large number of disk operations, alternatives to RAID have been proposed. One proposed alternative is to avoid striping the data files, and only stripe the parity data across the disks (Gray, Horst and Walker, 1990).

For improvement of the transfer rate of tertiary storage, similar strategies as used for disk arrays have been proposed. Striping of data in large tape libraries has been studied (Drapeau and Katz, 1993a; Drapeau and Katz, 1993b; Golubchik, Muntz and Watson, 1995). The main conclusion is that for applications which require sequential access to large amount of data, striping can improve the performance of the storage system. For applications that have a more random access pattern, striping results in poor performance (Chervenak, 1994). To increase the reliability of tertiary libraries, a similar data layout scheme as RAID level 5 has been proposed for creating Redundant Arrays of Independent Libraries (Ford, Morris and Bell, 1996). This has been evaluated using libraries containing optical disks and for use in a log-structured file system (Ford and Myllymaki, 1996).

[1]RAID is also commonly said to stand for Redundant Array of Independent Disks.

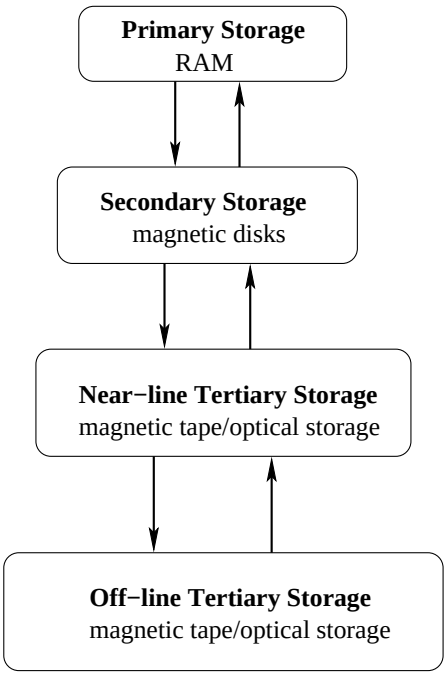


Figure 3.2 An example of a hierarchical storage system.

3.5.2 Hierarchical Storage Management

A Hierarchical Storage Management (HSM) system is a storage system that consists of storage devices based on different technologies organized in several levels. The storage devices on the different levels in the storage hierarchy have different performance, capacity and cost. The goal is to combine primary, secondary and tertiary storage in order to achieve a performance that is close to the performance of primary storage at a cost that is close to the cost of tertiary storage.

An example of a HSM system is shown in Figure 3.2. At the highest level in the hierarchy, RAM is used. RAM is fast, but expensive and non-permanent. The second level contains magnetic disks. The third level contains near-line storage which consists tertiary storage in form of optical disks or magnetic tape stored in library units. The fourth level is referred to as off-line and includes tertiary storage media that require manual intervention in order to be made available for the computer system.

The software controlling the HSM system *elevates* files between the different levels based on the usage pattern of each file. When a file is accessed, the HSM system moves the file to the highest level in the storage hierarchy. The file will remain at this level as long as there is enough space or the user is accessing the file. When there is need for more storage space at one of the levels, files are moved down to the next level in the storage hierarchy by using an LRU strategy.

Chapter 4

Server Technologies for Storage and Delivery of Digital Video

During the last twenty years, a lot of research has been performed on using computers for storage and delivery of digital video. Much of this research has been related to video servers. Computer based systems that deliver digital video to users, normally consist of three main components that are involved in the delivery of the video. The first component is the video server, which is responsible for storing the video data and delivering the video to the network. The second component is the network, which is responsible for transporting the video from the server to the client. The last component is the client, which is responsible for presenting the video to the user. The purpose of this chapter is to give an overview of some of the main problem areas within this field. As this thesis is about storage of video in digital video archives, our focus is on technologies for building high performance video servers that can handle the large amounts of digital video stored in a digital video archive and the many concurrent users accessing the videos.

The chapter is organized as follows: The first section gives an overview of how digital video can be delivered from a video server to the clients, and presents the main challenges for storing and delivering digital video. Then we continue with an introduction to admission control strategies, which are the algorithms that handle the clients and client interactions, and that have the overall responsibility for the resource utilization in the video server. In the following sections, we present strategies for how to utilize the main memory and the disk based storage system, in order to maximize the number of clients the video server can handle. In the last part of the chapter, we give an overview of research done on using tertiary storage for digital video.

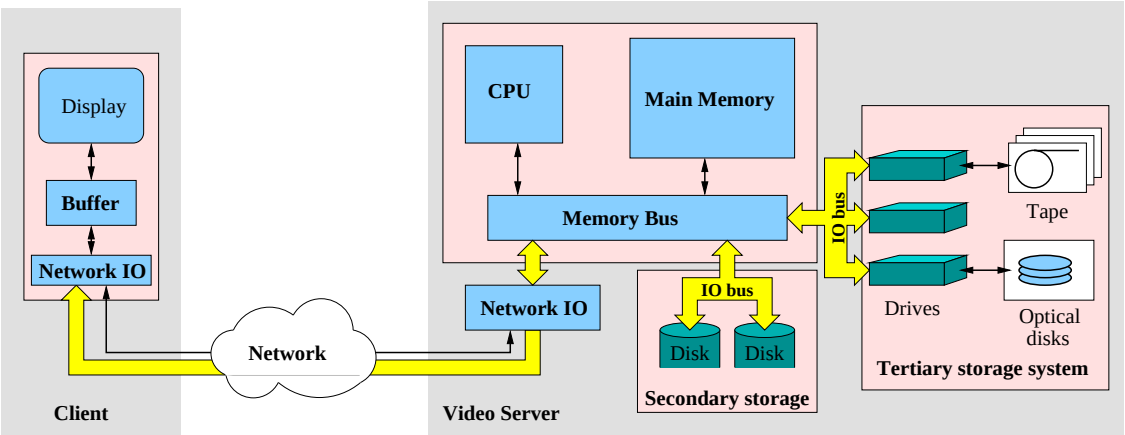


Figure 4.1 The main components involved during video delivery.

4.1 Delivering Digital Video

We start this chapter with an example showing how a user requests playback of a video stored in a video archive, and how this video is delivered from the video archive server to the user. This example shows one possible architecture for how a video archive server using tertiary storage might be implemented. The architecture consists of three main components, as illustrated in Figure 4.1:

- **The video client.** The video client is responsible for initiating the playback of the video and for controlling the playback if the user wants to pause or change the playback speed. Protocols like the MPEG-2 DSM-CC (Balabanian et al., 1996) and the Real Time Streaming Protocol (RTSP) (Schulzrinne, Rao and Lanphier, 1998) might be used for controlling the video playback. The client may also support more complex operations like browsing and searching of meta-data for the videos stored in video archive (Hjelsvold, Midtstraum and Sandstå, 1996).

  The client is also responsible for receiving the digital video from the server and displaying it to the user. This involves decompressing and decoding the compressed video. The client may also contain a buffer for buffering a shorter or longer part of the video (Chen and Kandlur, 1996; Pappas and Christodoulakis, 2000).

- **The network.** The network is used by the client for sending commands to the video server and by the video server for sending the digital video to the client. Compared to most other data types, video delivery is more sensible to variations in network delays. Video is also challenging for the network due to the high data rates over a long period of time.

  Considerable research has been performed on extending network technologies to better support digital video. The Internet protocol version 6 has

been extended with better support for streaming data (Stallings, 1996). New protocols for real-time transport (RTP) (Schulzrinne, Casner, Frederick and Jacobson, 2003) and for reserving network resources (bandwidth) (RSVP) (Zhang, Deering, Estrin, Shenker and Zappala, 1993; Braden, Zhang, Berson, Herzog and Jamin, 1997) have been added to the IP protocols. The ATM standard supports video data in general communication networks (McDysan and Spohn, 1995). In order to support video on existing phone lines, new digital network standards like ADSL and SDSL have been implemented. Similar protocols for delivering digital video on cable have been defined and implemented. To get an overview of some of the research on network support for digital video, we refer to the proceeding of the NOSSDAV conferences. Further discussion of network related issues is outside the scope of this thesis.

- **The video server.** The video server is responsible for storing the digital video and delivering the requested videos to the clients. The main challenge when developing a video server system is to be able to deliver as many video streams as possible without compromising negotiated QoS requirements.

When a user issues a request for a given video, a number of operations takes place in order to deliver the video to the client. Assuming a server architecture as shown in Figure 4.1, the following main operations have to be performed to deliver a video that is currently stored on a tertiary storage medium:

1. **Admission Control.** When the client sends the request for playback of a given video to the server, the server has to decide whether this request can be handled immediately or must be delayed or rejected. The server has to assure that it has the necessary resources for delivering the video *before* it starts the delivery. It is the responsibility of the *Admission Control* system to accept new clients and for ensuring that there are enough free resources for delivering the video. If the admission control decides that there are indeed enough resources, the video delivery can be initiated. See Section 4.2 for an overview of some admission control strategies.

2. **Tertiary Storage System.** Since we in this example assume the video is stored on tertiary storage, the tertiary storage system is instructed to start reading the video file(s). There are several possible strategies for delivering a video from tertiary storage. The video may be delivered "directly" from the tertiary storage system via a small main memory buffer to the network, the video can be read in larger chunks from tertiary storage into main memory and then be delivered from the main memory to the network, or the video might be read from the tertiary storage and written to disk, and then be delivered from disk to the client. In order to utilize the bandwidth of the

tertiary storage system, the last of these alternatives is the most common solution and will be used in this example. See Section 4.5 for an overview of use of tertiary storage in video servers.

3. **Disk Based Storage System.** As soon as the first part of the video is available on disk (or in main memory), the delivery of video can start[1]. The video is normally read as large blocks from disk into a main memory buffer. In order to ensure that the streaming of the video is not interrupted, the disk system must deliver a new block of video to the main memory buffer at regular intervals. See Section 4.4 for an overview of the main issues related to delivering video from a disk based storage system.

4. **Main Memory Buffer.** The main memory is used for pre-fetching and buffering of small parts of the currently delivered videos. The video files are read from the disk system as large data blocks. From the main memory, the video is divided into packages suitable for transportation on a network. These packages are written to the network interface for transportation to the client. When sending video data, the packages must be sent within a specified time interval to avoid under-flow or over-flow of the client buffer. The main memory might also be used for caching larger parts of the videos. See Section 4.3 for an overview of some of the strategies used for managing the main memory buffer in a video server.

5. **Network Transportation.** The network is responsible for transporting the data packages containing the video from the server to the client. Since video is sensible to jitter it is important that the data delivery is performed with only small variations in the network delay (Ferrari, 1990). If a package is lost, or delivered after a specified time-limit, it is normally no reason for retransmitting it.

6. **Video Presentation.** The client is responsible for receiving the video from the network and displaying the video on the user's video terminal. This includes assembling network packages, handling lost or delayed video data, and decompressing and decoding the compressed video. The client will normally have a buffer that stores a small part of the video. The size of this buffer determines how much jitter that can be accepted in the video stream delivered by the server without the user noticing it.

In this example we have shown the main operations that are performed during delivery of a video to the user. With the exception of the use of tertiary storage, most video servers perform these operations during video delivery.

[1]We assume here that the tertiary storage system is able to deliver the video faster than it is consumed by the client.

4.1.1 Workload Characteristics of Video Delivery

Most server based systems like traditional DBMSs and Web servers perform relatively short-lived tasks where the goal is to deliver the answers to client requests as quickly as possible. For this type of systems, the challenge is to handle as many concurrent requests as possible with a lowest possible response time. High throughput and low response time are also important requirements for a video server. However, compared to most server based systems, there are a number of factors that make it more challenging to achieve a high utilization of the computing resources in systems that serve digital video:

- **High data rates.** Digital video requires data rates in the interval from approximately 100 kbit/s for low quality video to more than 20 Mbit/s for high quality HDTV programs. This makes delivery of video a data intensive task. For example, a video server that should be able to deliver 1000 concurrent high quality video streams each having an average data rate of 5 Mbit/s, has to be capable of delivering a data rate of 5 Gbit/s to the network. In order to do this, all parts of the video server, from the storage subsystem through main memory and to the network interfaces and the network itself must be able to handle these kinds of data rates. Complicating this further is that many video compression algorithms produce video with a variable bit rate.

- **Tight timing requirements.** As soon as a user has started playback of a video, a new video frame has to be presented on the viewers screen every 40 ms (assuming 25 fps PAL video). Thus, the video server must be able to deliver a new video frame to the user at fixed intervals. If the variation in time between each frame being delivered becomes too large, it will be noticed as jitter by the user (Ferrari, 1990). In order to deliver the video as an isochronous data stream with tight timing requirements, the video server must be implemented as a system with (soft) real-time properties.

  For audio, this requirement is even more important. The user might not notice if a frame in the video is lost, but if parts of the audio are lost or delayed, it is easily noticeable. The timing requirements can be relaxed by using buffering of video and audio in the client, at the cost of increased response times (Pappas and Christodoulakis, 2000; Allen, 2001).

- **Long-lived requests.** Depending on the characteristics of the application the video server is used for, the duration of a request is typically from tens of seconds for delivery of short video clips up to a few hours for delivery of movies. This is in contrast with most other server based systems where a typical request last from a few milliseconds to a few seconds. In addition to the higher duration of a video request compared to "traditional" systems, there is another important difference between a video request and a "traditional" request. A system serving a "traditional" request should normally

do it as fast as possible. Most users of a video server would not like the video to be delivered as "fast as possible", but instead at the video's correct speed. Thus, to deliver a two hour video should take two hours, not one hour and fifty minutes or two hours and ten minutes.

There is also a difference between traditional server based systems and most video servers in the way the data is delivered. In a traditional server, it is the client that sends a new request when it requires more data. This is referred to as a *pull* strategy. Most video servers use a *push* strategy (Shenoy, Goyal and Vin, 1995). After the delivery of a video has started, the client does not need to send requests for more video data. The server will be responsible for *pushing* the data to the client as an isochronous data stream.

- **Large data volumes.** As shown in Section 2.2, even when the video is compressed it requires large amounts of storage space. For example, a video server that stores 1000 video titles, each having an average length of two hours, must manage a storage volume of more than four terabytes when the videos are compressed as 5 Mbit/s MPEG-2 video.

- **Read-only requests.** With the exception of the initial storage of a video in the server, most of the requests for videos are read-only. Very few systems allow for manipulation of the stored video data. The only changes to the system are when new videos are added to the system and when existing videos are removed.

The strict timing requirements and the long lived requests make it more challenging to achieve optimal use of the available computing resources in a video server than in a more traditional server. For the video server to be able to *guarantee* handling requests with this long duration requires considerable planning and control of the use of the computing resources available within the server.

Fortunately, video and audio also have one property that makes it possible to increase the utilization of the computing resources. For most video applications, to loose one video frame is not catastrophic. It might not even be noticeable for the viewer. This property can be used for allowing more concurrent video streams to be delivered than what the server actually can guarantee to deliver without loss of data.

4.1.2 Resource Management

The main challenge when implementing a video server is to make optimal use of the available resources in order to deliver as many concurrent video streams as possible while still guaranteeing that the specified QoS requirements are fulfilled. Given plenty of computing resources, delivery of video is a relatively easy task. But as for all other server based systems, it is important to be able to get a highest

possible performance out of the available resources. The challenges come when the server becomes loaded and there are little free resources.

In a video server, delivery of video consists mainly of transporting data from the storage system through main memory and onto the network. Figure 4.1 contains an overview of the main resources that are used by a typical video server when delivering video. The main resources that must be managed by the video server are:

- **Storage space.** Most video servers use magnetic disks for storing the video. Video servers used in large video archives might also use tertiary storage in addition to the magnetic disks. In order to be able to achieve a high utilization the storage devices, it might be necessary to have control of the physical data layout on the storage devices.
- **Storage bandwidth.** For retrieving video from storage devices and into main memory, both the bandwidth of the storage devices and the storage interface (e.g., the SCSI bus) are important.
- **Main memory.** Most video servers use main memory either for caching (parts of) videos or as temporary buffer space when reading large blocks of video from disks and sending the video as smaller packages to the network.
- The **CPU** is used for running the video server software. The main responsibilities of the video server software are to handle client requests and to administrate how the other resources in the system are used for delivering the video. Compared to other server based systems, data transfer to and from main memory constitute a larger fraction of the total work for a video server. This work is done by the CPU and by using DMA transfer of data.
- **Server internal communication.** Delivering digital video requires much internal transfer of data within the servers. The video has to be transported from disk and tertiary storage devices into main memory. It has also to be transported from the main memory to the network devices. This data transport is performed using the I/O buses (e.g., SCSI and Fiber Channel) (LoBue, 2002), network adapters and the main memory bus. All of these can become bottlenecks during video delivery.
- **Network bandwidth.** This includes both bandwidth for copying data from main memory to the network interface and bandwidth of the network.

All of these resources are needed for delivering one video stream. Thus, if there is shortage of one resource, it will hamper one or more of the video streams being delivered. It does not help to have plenty of one resource, e.g., bandwidth to the storage system, if another resource is over-utilized. For a video server to make optimal use of these resources, it is necessary that the operating system provides support for efficiently use of the resources (Steinmetz, 1995; Plagemann, Goebel, Halvorsen and Anshus, 2000).

4.1.3 Video Server Architectures

The video server illustrated in Figure 4.1 consists of a single machine that is delivering the video. This is just one of several possible architectures used for building video servers. Most video server architectures can be divided into either being centralized or distributed.

Centralized Video Servers

A centralized video server stores all the videos in a centralized location. The server delivers the video streams directly to the user by use of unicast, multicast, or broadcast networks. Centralized video servers can further be grouped by whether they consist of a single machine or multiple machines.

- **Single machine.** As the name indicates, the video server consists of a single machine that is responsible for storing and delivering all the videos. One example of a single machine video server is the Fellini multimedia storage server (Martin, Narayanan, Özden, Rastogi and Silberschatz, 1996).
- **Multiple machines.** A video servers consisting of multiple machines is often referred to as a parallel video server. In a video server containing multiple machines, there are two main alternatives for how the video is delivered. In the simplest configuration, one of the machines stores the entire video and is responsible for delivering it to the client. An alternative configuration is to distribute the video on all (or a subset) of the machines. All machines takes part in the delivery of the video. A major challenge in a parallel video server is to achieve a good load balancing. We discuss this further in Section 4.4. An example of a parallel video server is the Tiger video file server (Bolosky, Barrera III, Draves, Fitzgerald, Gibson, Jones, Levi, Myhrvold and Rashid, 1996). The Elvira II video archive server presented in Appendix A is another parallel video server built on a cluster of independent workstations.

Distributed Video Servers

A distributed video server consists of multiple servers that are distributed across a network. The main purpose of building a distributed video server is to reduce the amount of network transport, and to reduce the cost of the network infrastructure. Most distributed video server systems are *hierarchical*. The original of each video is stored in one or a few *root servers*. The other servers are cashing copies of a subset of the videos (Nussbaumer, Patel, Schaffa and Sterbenz, 1995) or parts of the videos (Chan and Tobagi, 2001). The cost savings emerge from being able to store the most frequently used videos closer to the clients. Many of the proposed strategies for video delivery use multicast for distributing videos to cache servers or directly to the clients (Aggarwal, Wolf and Yu, 1996b). To reduce

response time, several strategies have been proposed for caching the initial part of each video in proxy servers while using multicasting for delivering the main part of the videos (Sen, Rexford and Towsley, 1999; Griwodz, Zink, Liepert, On and Steinmetz, 2000; Bradshaw, Wang, Sen, Gao, Kurose, Shenoy and Towsley, 2001). An alternative topology to using an hierarchical structure is peer-to-peer systems. The main drawback of peer-to-peer systems is that it is difficult to achieve centralized control of both content and resources, and to maintain digital rights of the content.

The focus of this chapter is on technologies for efficiently storing and delivering video in a centralized video server. Still, most of the strategies presented in this chapter are suitable for use in distributed video server systems since each server in a distributed system also has similar performance requirements with regards to storing and delivering videos.

4.1.4 Classification of Video Servers

There are multiple ways to classify video servers. One strategy is to classify video servers based on the amount of control the user has on the delivery of the video she wants to watch (Gelman, Kobrinski, Smoot and Weinstein, 1991; Little and Venkatesh, 1995). Using this as the criterion, most video servers can be divided into either *Broadcast*, *Near-Video-On-Demand*, or *True-Video-On-Demand* servers. Some video servers may cover multiple of these classifications.

Broadcast Servers

A video server using broadcast for delivering the videos is typically used when there is a limited number of videos that must be delivered according to a scheduled program. Examples of applications that may utilize broadcast for delivering the video are educational institutions that send a lecture at a given time or hotels that have set of films that are started at regular intervals throughout the day.

The broadcast can be limited to a local area network or sent to a larger part of the Internet by using protocols like IP Multicast or IP Simulcast (Furht, Westwater and Ice, 1998). The main advantage of using a broadcast is that a video server with limited computing resources can deliver video to a high number of users. The cost per delivered video can be kept very low. The drawback is that the user has no or limited control of the video playback. One example of a multicast strategy for video servers is presented in (Xu, 2001). By reducing the amount of time between each broadcast of a video and including a buffer in the client, it is possible to provide limited user control of the playback (Tantaoui, Hua and Sheu, 2002).

Near-Video-On-Demand Servers

Near-Video-On-Demand (N-VOD) servers offer some functionality for the users to control the video playback. Most N-VOD servers limit this functionality to allow the users to request which movie to watch, to pause and to resume the playback from the same or a different position within the video. Each time the user requests a new video or wants to resume the playback of the current video, she has to expect to wait some period before the playback of the video starts.

The reason for offering this limited functionality is to be able to increase the number of viewers served by the video server. This is achieved by serving multiple viewers with a single video stream. The support for resume playback after a pause or from a different position in the video is achieved by having the video server delivering multiple video streams containing the same video where each is delayed by a fixed or variable amount of time. Each time a user requests playback to resume, she will be served by the first following video stream. In Section 4.2.1 we present strategies that can be used by N-VOD video servers. These strategies group multiple viewers in order to reduce the resource consumption per viewer and to increase the total number of users that can be handled by the video server.

True-Video-On-Demand Servers

True-Video-On-Demand (T-VOD) servers offer the users full control of the video playback. The server supports the functionality offered by a standard VCR or DVD player. The user can pause and resume the playback from any position within the video, and issue fast forward and fast rewind commands at different speeds.

In a T-VOD server, each user is served independently. This means that compared to a broadcast or N-VOD server, the resource consumption per user will be higher. The much better support for control of the video playback increases the challenges on the video server to make optimal use of the resources.

4.2 Admission Control and Client Scheduling

One of the main challenges in a video server is to administrate client requests. These requests are either a request for starting playback of a new video or to perform a VCR operation on a currently played video stream. When a request for starting playback of a new video arrives, the server has to decide *if* this request can be granted and *when* the playback should be started. When the server makes this decision, the goal is to be able to handle as many clients as possible, but granting access to a new client should not reduce the quality of the video streams already delivered. The strategies used for making these decisions are referred to *Admission Control* and *Video Client Scheduling*.

The main resources used by a video server for delivering video to a client were presented in Section 4.1.2. If any of these resources are over-utilized, the user may observe this as irregularities in the delivered video. Thus, in order to achieve continuously delivery of the video streams without interruptions, the following must be done (Sitaram and Dan, 2000, Chapter 4):

1. Reservation of the necessary resources on all server and network components that take part in the delivery of a video stream.

2. Each component must have an appropriate scheduling strategy for its resources in order to achieve the timing requirements of the video streams.

The set of resources that must be reserved for guaranteeing continuously delivery of a video stream is often referred to as a *logical channel* (Dan, Sitaram and Shahabuddin, 1994). To allocate logical channels for new requests is non-trivial, since this involves multiple components in the server and the network that have very different behavior. In addition, different requests for a video stream may have different QoS requirements, both including physical properties of the video like average and peak bandwidth, and user specified requirements like acceptable delay on startup, response time on VCR commands and tolerable loss of data.

Factors that make it complicated to estimate how much resources that must be reserved for a given video stream are:

- **VCR operations.** When the user issues an operation like fast-forward, the video server must deliver the video at a higher speed. The simple solution to this is just to send the video data at a higher speed to the user. Unfortunately, this requires more resources than this stream has allocated. To avoid that VCR operations like fast forward and fast rewind require more resources than normal playback, different strategies for reducing the bandwidth have been proposed. Examples of such strategies are dropping frames or changing to a more highly compressed version of the video.

- **Variable Bit Rate (VBR) video.** Depending on the compression strategy for the video, the compressed video requires either a variable bit rate (VBR) or a constant bit rate (CBR). The challenge with VBR video is that the peak bit rate may be considerable higher than the average bit rate. To satisfy the QoS requirements, the logical channel must have enough allocated resources to handle the peak bandwidth (Vogt, 1995).

- **Variations in storage system performance.** When the storage system consists of secondary storage (magnetic disks) and possible tertiary storage (tape, optical disks) the performance of the storage system may vary depending on the physical placement of the video on the storage device. Thus,

it may not be possible to use the average response time and average bandwidth when allocating resources to video streams if the server should guarantee a given performance also during periods where the performance of the storage devices are below average.

Deterministic strategies for admission control must take all these factors into account when deciding whether a new request can be handled or not. To guarantee that all currently delivered streams get the necessary resources, resources must be allocated to handle the *worst case* scenarios where all video streams have the highest consumption of resources during a lowest possible performance of the storage system. Using such strategies lead to low average utilization of the resources.

To increase the number of concurrent video streams and the utilization of the computing resources, more optimistic admission control strategies have been proposed. The *observation-based admission control* algorithm (Vin, Goyal, Goyal and Goyal, 1994a) uses the current status for the resources on the server to determine if the server has enough free resources to admit a request for a new video stream.

The deterministic algorithms provide strict performance guarantees while the observation-based algorithm gives no guarantees about the quality of delivered video. To achieve a much higher utilization than deterministic admission control algorithms while still being able to give *statistical* guarantees about the service quality, *statistical admission control strategies* have been proposed (Vin, Goyal, Goyal and Goyal, 1994b). These algorithms take into account both bit rate variations of the video streams and variation in the access time for retrieval of data from the storage system. By using statistical estimates for the total resource consumption by the currently delivered video instead of worst case estimates, statistical admission control algorithms are able to achieve a higher utilization of the video server than deterministic algorithms.

4.2.1 Resource Sharing and Merging of Clients

So far we have assumed that each client is allocated resources that are used solely for this client's video stream. The admission control strategies presented this far will limit the number of clients that can be served to the number of video streams (or logical channels) that the server can deliver. To increase the number of clients the server can handle, multiple clients must be grouped together and served by one logical channel. In this subsection, we present three strategies for how this can be achieved.

Batching

A method for increasing the number of clients beyond the number of streams the server can deliver is to use *batching* of clients. By batching multiple client requests for the same video, a larger number of clients can be served. In order to

batch clients, it is necessary that multiple clients request the same video within an acceptable amount of time. The main drawback with batching is that in order to increase the probability for being able to batch two or more clients, it is necessary to delay the start of the video playback.

Multiple batching strategies have been proposed. The simplest model for batching is to use a FCFS (*first-come-first-served*) strategy when selecting which client that should be allowed to start playback. If there are other clients waiting for the same video, they are also served by the same logical channel (Dan et al., 1994). In order to increase the number of clients served, the *Maximum Queue Length* (MQL) strategy selects the video with the highest number of waiting clients when there are available resources for starting a new video (Dan, Sitaram and Shahabuddin, 1996). The MQL strategy priorities delivery of popular videos. Clients requesting less popular videos will experience higher response times and might be delayed infinite. In order to increase the fairness of the MQL strategy, Aggarwal, Wolf and Yu (1996a) have proposed the *Maximum Factored Queue Length* (MFQ) strategy. This strategy selects the video that has the highest *factored queue length* as the next video. The factored queue length is computed by giving each video a *weight* that is inversely proportional with the popularity of the video. Thus, less popular videos will have a higher weight than more popular videos.

In a video server, videos will have different popularities. Some batching algorithms partition the videos into different groups based on the videos' popularity. The FCFS-n algorithm pre-allocates a set of channels for the n most popular videos (Dan et al., 1996). The number of pre-allocated channels is usually higher than n. The pre-allocated channels are used for starting playback of the most popular videos at regular intervals. The channels not pre-allocated for the most popular videos are used for serving requests for the less popular videos. These requests are scheduled using a FCFS strategy. The *Group-Guaranteed Server Capacity* (GGSC) strategy divides the videos into multiple groups, where each group consists of videos that have nearly equal expected batch size (Tsiolis and Vernon, 1997). Each group is then assigned a number of logical channels. Within a group, FCFS is used for scheduling requests. This algorithm makes it possible to give estimates for the maximum time a client requesting a given video has to wait before the playback begins. Being able to give an estimate on the maximum waiting time reduces the probability for a waiting client to leave before playback starts.

One difficulty with batching multiple clients occurs when one of the clients issues a VCR command. The client must then be removed from the logical channel. If the VCR command is to resume after a pause or a fast-forward or fast-rewind, the user expects this to happen relatively immediately. Unfortunately, all resources that were allocated to this video are still in use by the other clients in the batch. Dan, Shahabuddin, Sitaram and Towsley (1995c) propose to reserve a set of logical channels to handle clients that issue VCR commands. This set is referred to as *contingency channels*. When a client wants playback to resume, a

channel is allocated from the set of contingency channels.

Caching

The main problem with batching is that all clients that are to be served as one batch have to wait until they get a free channel. Further, when the video has started, the clients have to watch the video continuously if they should be served by the same channel. By caching parts of the video either in the server or in the client, some of the drawbacks of batching can be reduced.

Techniques that use server caching, store parts of the most popular videos or entire videos in main memory. The parts that are stored in main memory can then be used for serving clients without having to use storage bandwidth. Interval caching is one example of a buffering strategy (Dan, Dias, Mukherjee, Sitaram and Tewari, 1995a). The idea is to cache enough video in main memory to cover the size of a batch interval. By doing this, it is possible to allow new clients that arrive just after a batch has started to join this batch. It is also possible to use this for handling VCR operations like small pauses and reposition as long as the position to resume from is within a batch interval that is buffered in main memory. This form for caching in the server reduces the disk bandwidth requirement to the storage system. The cost of these strategies is that the required amount of main memory is increased. These strategies do not reduce the need for network bandwidth.

Buffering parts of the video in the client can be used for handling VCR operations and thus reduce the load on the server when a client issues a VCR operation. The obvious solution is to store the already played video in main memory (or on a disk) and to continue to receive the video stream if the user issues a pause. This relatively simple strategy can support both fast rewind, reposition to an earlier position in the video and resume after pause without the server even having to notice it. A more complex strategy is the Pyramid scheme proposed by Viswanathan and Imielinski (1996). This requires that multiple clients want to watch the same video and that the videos are sent by using a network supporting broadcast. The Pyramid algorithm partitions the video into contiguous segments of geometrically increasing sizes. Each of the video segments are sent on a separate channel. The size of the first segment is relatively short. Since this segment is sent repeatedly on one channel, the maximum time a client has to wait is the length of this segment. As soon as the client has finished playback of the first segment it switches to the next channel where the second segment is streamed. The main advantage of this scheme is that a high number of clients can be served with a limited number of channels and still achieve a relatively low startup latency. The drawback is that the clients have to be able to receive video on multiple concurrent streams and buffer a substantial part of the video. An improved version of the Pyramid scheme that reduces both the storage requirement in the client and the startup latency is presented in (Aggarwal et al., 1996b).

Rate Adaptation

Rate adaption is a strategy for merging clients while avoiding the high startup delay of the batching strategies or the high memory requirements of the caching strategies. Rate adaption is a technique that tries to merge video streams by adjusting the display rates (Golubchik, Lui and Muntz, 1996). This is based on the observation that it is possible to change the rate of a video (and the corresponding audio) by 2–3 percent without the viewer noticing it. Rate adaption is also referred to as *adaptive piggy-backing* (Golubchik et al., 1996).

The basic strategy of rate adaption is that when the first request for a video is received, the delivery is started immediately. If the next request for the same video is sufficient close to the first, it is started immediately, but at a slightly higher speed. When the second video has caught up with the first video, they are merged and from then on served as a single stream. Thus, before the two videos are merged, they are served by two channels, after the merging they are served by one channel. Golubchik et al. (1996) present several strategies for how to merge streams by changing the speed of the videos. VCR operations can be handled by giving the client a new channel for the duration of the VCR operation. After the client resumes to normal playback, this video can again be merged with another video stream by either increasing or reducing the speed of it for some time. A similar strategy is used by the Split and Merge (SAM) protocol (Liao and Li, 1997).

In order to deliver the same videos in multiple speeds, it is necessary to either have extra software or hardware that do on-line rate adaption of the video and audio as it is sent to the user, or to have the video and audio stored on multiple versions with slightly different display rates. This increases the cost of the video server compared to a server that uses pure batching of clients. Krishnan and Little (1997) have proposed a solution, which uses a special data placement and disk scheduling strategy that supports two different rates based on a single storage format.

4.3 Buffer and Memory Management

Most video servers use main memory as buffer for the storage system. Delivering the video directly from the disks to the network would result in low utilization of the disk bandwidth. In order to efficiently utilize the disks, the video is read from the storage system into main memory using relatively large blocks. From main memory, the video is then at regular intervals delivered to the network in smaller data units suitable for transport on the network. The main memory buffer is also used for ensuring that the video is delivered as an isochronous data stream. By having the currently delivered part of the video in memory, this can be used for avoiding jitter and for smoothing out bit rate variations in VBR video and disk variations.

Using memory as a buffer for the storage system is not new. Particularly for use in database management systems (Effelsberg and Haerder, 1984), file systems and operating systems, main memory has been used for caching parts of the disk-based data. Compared to secondary storage, main memory is expensive. As a consequence, the main memory buffer will have limited space for storing video data. The main advantages of main memory are the very short access times and the high bandwidth, making it capable of delivering data to many concurrent video streams. In order to utilize the main memory, the primary objective for a buffer management strategy is to serve as many requests as possible from the main memory buffer.

Most of the buffer management strategies can be divided into one of two main categories based on how the buffer space is divided between currently delivered videos (Dan et al., 1995a):

1. *Buffering.* Each video stream has allocated a given amount of buffer space that is only used by this video stream. This buffer space is used for pre-fetching and *buffering* a small number of disk segments. As soon as the video in one buffer slot is delivered to the network, the buffer slot is made available for reading in the next disk segment.

2. *Caching.* The main memory is organized as a global buffer cache that is shared between all videos. In addition to be used as buffer for pre-fetching blocks from disk, the main memory is used for caching parts of the videos. The purpose of caching parts of videos in memory is to make it possible to deliver the video to multiple users without having to read it from disk. If the video server is able to deliver videos from the main memory cache, this may reduce the cost of the storage system.

We present each of these categories in more detail in the rest of this section.

4.3.1 Buffering Strategies

Buffering is used for increasing the disk utilization and thus reduce the cost of the storage system[2]. The basic idea of buffering is that each currently delivered video stream has allocated a buffer in main memory where data from the storage system is written, and where data is read from and sent on the network to the client. The buffer used by each stream can either consists of a fixed or a variable number of buffer blocks. Typically, the buffer consists of at least two blocks per video stream. These are used for *double buffering*. The video is delivered from one of the buffer blocks while the storage system writes to the other buffer block. The size of each buffer block is the same as the amount of disk data read in one disk operation. Since the video is delivered as an isochronous stream to the client, it is important to avoid underflow in the buffer.

[2]Buffering is also commonly referred to as read-ahead buffering or pre-fetching.

There is a close relationship between the utilization of the disk bandwidth, the size of the buffer per stream, the cost per delivered stream, and the probability for jitter:

- Increasing the utilization of the disks *reduces* the cost per delivered video stream, but increases the probability of not having the video data in buffer memory when it should be sent to the user. The user will observe this as jitter (loss of frames or disturbances in the video and audio).
- The probability of jitter can be reduced by increasing the size of the buffer, but this *increases* the cost of main memory per video stream.

Thus, by increasing the amount of read-ahead buffer allocated for each stream, it is possible to increase the utilization of the disks (Chang and Garcia-Molina, 1997a). For a system with a specified QoS level regarding jitter, it is possible to determine the optimal configuration for disk utilization and buffer size that gives the lowest cost (Dan et al., 1995a). To reduce the amount of buffer space required, Garcia-Martinez, Fernandez-Conde and Vina (2000) propose to allocate buffer space per storage device instead of per client. This requires a round-based disk scheduling strategy and that the client is able to receive all data retrieved during one round as one bursty transfer.

The amount of buffer per stream also influences the startup cost when starting delivery of a new video stream and the response time when doing VCR operations. Each time a new video delivery starts or there is a reposition of the playback, the buffer has to be at least partially filled up before the playback can start. Thus, a large buffer per stream will normally give a higher startup cost and longer response times.

4.3.2 Caching Strategies

Traditional caching strategies aim at keeping the hot set of data objects in the cache. In order to determine what is the hot set, these strategies are based on how applications access the data or hints from the applications. Examples of traditional caching strategies are LRU, CLOCK, and DBMIN (Chou and DeWitt, 1985).

Compared to the data cached by applications like databases and operating systems, videos are much larger. Using traditional caching strategies for caching the hottest videos in main memory would either require a very large cache or result in a very limited number of videos in the cache. One possible solution to this problem is to divide the videos into smaller units, e.g., blocks and use these as basis for caching. Unfortunately, due to the sequential access pattern of videos, traditional caching strategies perform bad also when the caching unit is of this size. As soon as the content of a block has been sent to the viewer, it is unlikely that this block will be accessed soon.

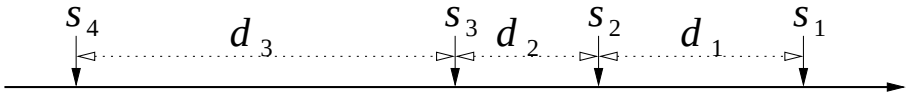


Figure 4.2 Illustration of how to compute the *distance* of video streams as defined by the DISTANCE algorithm (Özden et al., 1996a).

For caching of videos, new caching strategies have been proposed that include and take advantage of the sequential access pattern of videos. Most of these strategies are based on using knowledge about the relationship between concurrently delivered streams of the same video. Just as for traditional caching strategies, caching strategies for video must take into account both when to *place* a part of a video in the cache and when to *replace* a part of a video already stored in the cache. The caching strategies must also be integrated or interact with the overall *admission control* used by the server (see Section 4.2). The admission control strategy must know whether a video can be delivered from the cache or not in order to know the resource consumption for the video. It must also be able to handle that a video currently delivered from the cache no longer finds its data in the cache.

In this section we give a short introduction to three caching strategies that have been proposed for use in video servers:

- **BASIC** is the first of two caching strategies proposed by Özden, Rastogi and Silberschatz (1996a). BASIC uses a very simple approach. When a new buffer slot is needed for a video, BASIC selects the buffer slot containing video that will not be accessed for the longest period of time by any of the currently delivered video streams. This strategy performs very close to an optimal caching strategy, but the computational cost of the algorithm is very high since for each new buffer allocation, all buffer slots must be compared against all currently delivered video streams.

- **DISTANCE** (Özden et al., 1996a) uses, as the name indicates, the *distance* between concurrent video streams delivering the same video as criteria for deciding which buffer slot in the cache to reuse. The distance, d_i, of video stream v_i is measured as the number of buffer slots between this stream and the first following video stream, v_{i+1}. This is illustrated in Figure 4.2.

  The DISTANCE algorithm is round based. Used buffers are returned to the free pool on the end of each round. When freeing used buffers, the video streams are sorted on increasing distance value. Buffers from the video with the shortest distance value are freed first and buffers from the video with the highest distance value are freed last. Allocation of buffers occurs in the opposite order. When a stream needs a free buffer, the buffer which was freed most recently is allocated. In this way, buffers freed by a video stream

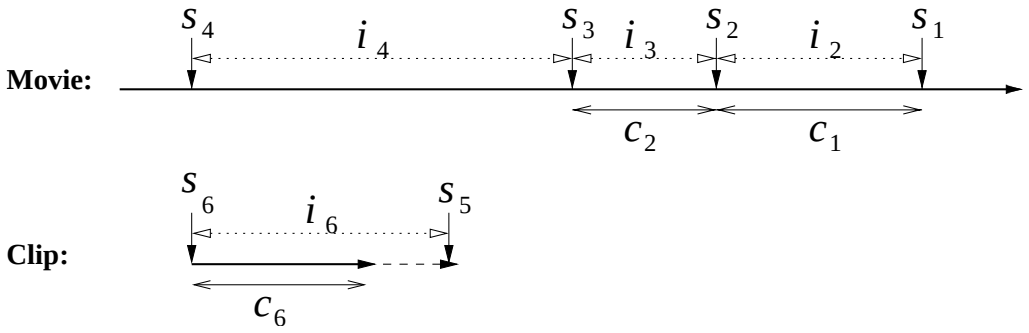


Figure 4.3 Illustration of how the *General Interval Caching* strategy works. The figure is based on figure found in (Dan and Sitaram, 1997).

that is followed closely by another video stream have less probability for being re-allocated than buffers freed by a video stream with no following video stream or followed by a video stream at a greater distance.

Compared to the BASIC algorithm, the DISTANCE algorithm has a much lower computational cost. The *distance* value for a video stream only has to be recomputed when there is a change in the playback of this video stream (e.g., a pause or a reposition) or when the playback is done with a speed different from normal playback speed.

Caching strategies like BASIC and DISTANCE work best for long videos where multiple clients access the same video. They require that at least two clients are playing the same video in order to have parts of the video cached. For a video server serving small video clips or a mix of long and short videos, it is less likely that there are multiple clients accessing the short video clip. Thus, these caching strategies will either perform badly or disfavor short video clips.

- **General Interval Caching (GIC)** (Dan and Sitaram, 1997) is a caching strategy that supports a wider workload than BASIC and DISTANCE. It is based on the same principle as DISTANCE. What the DISTANCE algorithm refers to as the *distance* between two video streams (see Figure 4.2), the GIC algorithms refers to as an *interval*. Just as the DISTANCE algorithm attempts to cache the video between video streams that are following closely in distance, the GIC algorithm caches the shortest intervals.

  The main extension of GIC compared to DISTANCE is the way it includes short videos. This is illustrated in Figure 4.3. For each video, the GIC algorithm remembers when the last client finished. When a new client requests playback of a video that no other clients are accessing, an *anticipated* interval (Dan and Sitaram, 1997) is created between the new client and where the last client would have been if the video had been longer (see video stream s_5

and s_6 in Figure 4.3). If this anticipated interval is short enough to deserve a place in the cache, the entire video is stored in the cache. An important difference between normal intervals and anticipated intervals is that the normal interval moves along the video as the playback progresses, while for an anticipated interval, the entire video will be in the cache as long as the client is playing the video.

These three buffer management strategies presented here assume sequential playback of the videos with little or no user interaction.

For supporting video delivery in more interactive applications, buffer management strategies that takes the user interaction into account have been proposed. One of these strategies is the L/MRP (*l*east/*m*ost *r*elevant for *p*resentation) buffer management strategy (Moser, Kraiß and Klas, 1995). This strategy assigns a relevance value to each presentation unit[3]. Typically, the frames that are about to be shown to users will have a high relevance value while frames which have been shown to the user will gradually get a lower value. The relevance value is the basis for pre-fetching and replacing data in the buffer. Presentation units with a high relevance value that are not already in memory will be pre-fetched into main memory, while the presentation units with the lowest relevance value will be replaced. Several extensions to L/MRP have been developed. Q–L/MRP is an extension to L/MRP that supports multiple concurrent users and dynamically adaption to the disk and network I/O (Halvorsen, Goebel and Plagemann, 1998). Another extension is MPEG–L/MRP (Boll, Heinlein, Klas and Wandel, 2000) that aims at reducing the response time of user interactions during video playback.

In this section, we have presented some examples of caching strategies. Common for all of the caching strategies is that they try to optimize the amount of sharing of the video data that is stored in the cache in order to increase the number of videos that can be served by the video server and reduce the cost per delivered video. How much data to pre-fetch for each client and the choice of caching strategy depends on the application's access pattern and the amount of sharing of data between clients.

4.4 Storage System Management

A video server's storage subsystem is responsible for storage of video data and for retrieval of video data from the storage devices. The main issues when designing the storage system are *organization of the storage devices*, *data placement*, *resource reservation*, and *data delivery*. The storage system should also be able to handle and recover from hardware failures without making the videos unavailable. The overall goal is to have a storage subsystem that makes optimal use of the avail-

[3]In the paper this is referred to as a Continuous Object Presentation Unit (COPU) (Moser et al., 1995). A COPU is typically one video frame.

able resources, both storage space and bandwidth, while still assuring that the QoS criteria are fulfilled.

From the storage subsystem's point of view, delivering video data has some unique characteristics compared to delivering more traditional data:

1. **Real-time data access.** When the video server requests the storage system to deliver some video data, it normally needs to have these data available in main memory within a given time limit in order to avoid interrupts in the video stream sent to the user. If the storage system is not able to retrieve the data within the time limit, the request can in many cases just as well be dropped.

2. **Sequential data access.** Most viewers watch the videos with little user interaction. Because of this, the access pattern to video files will be mainly sequential.

   One important implication is that unless there are multiple viewers for the same video that are close enough in time, the video data read from the storage system has very little probability of being reused.

3. **Periodic data access.** Most video servers read the video data from a main memory buffer and deliver it to the user as an isochronous stream. At regular time intervals, it is necessary to fill the main memory buffer by reading the next part of the video from the storage system to the main memory buffer. This results in a periodic access pattern for the video files.

These differences in how video data and more traditional data are accessed have made it necessary to develop new storage architectures and algorithms for accessing video data in order to optimize the utilization of the storage system. There are two main issues when implementing a storage system for digital video. The first is how the storage system should be organized and how video data should be placed on the different storage devices. The second issue is strategies for retrieval of video data from the storage system. We cover both of these issues in the remainder of this section. For a more detailed survey we refer to (Halvorsen, Griwodz, Lund, Goebel and Plagemann, 2003a).

Resource Reservation and Scheduling of Video Retrievals

In Section 4.2, we gave an introduction to how the admission control and client scheduling algorithms were responsible for deciding whether a new request could be granted or not, and when the delivery should start. As part of the admission control, the necessary resources should be reserved in the storage subsystem for delivering the video.

When designing the storage system of a video server, it is important to take into account how the admission control strategy allocates and reserves resources

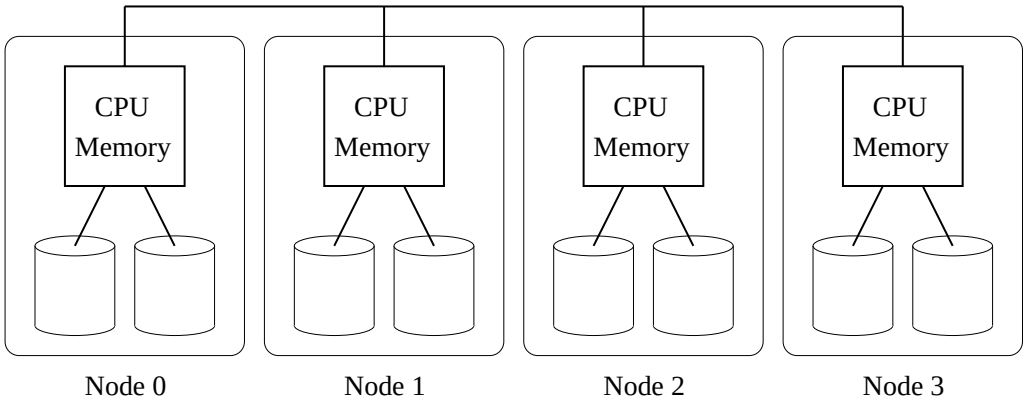


Figure 4.4 Video server containing multiple nodes and multiple storage devices.

for the logical channels from both the storage system, the buffer management system and the network system. After the admission control has allowed the playback to start for a given video, it is the responsibility of the storage system to guarantee that the video data is retrieved from the storage devices at the required rate and within the time limits of when the data has to be present in main memory.

Storage Organization and Data Placement

For a video server, the strategy for optimal data placement may involve several levels. In Figure 4.4 we show an example of a video server that contains multiple nodes where each node has multiple storage devices. A storage strategy for this server must consider all of the following levels:

- **Server.** For a parallel or distributed video server, which consists of multiple nodes, the data placement strategy must decide where each video should be stored. This can either be to store it on one node, a subset of the nodes, or all of the nodes. A node is normally one computer.
- **Node.** Each node or computer within a video server has typically multiple storage devices. The storage strategy must decide whether a video should be stored on one or multiple of the storage devices.
- **Storage Device.** The storage strategy must decide how a video (or video fragment) should be stored within one storage device.

We discuss strategies for each of these areas in the following. Since the strategies for deciding which machine(s) within a server and which storage device within a machine are based on the same algorithms, these are presented together.

Most video servers use magnetic disks as their main storage. Because of this, we focus on the use of magnetic disks in this presentation.

4.4.1 Magnetic Disks in Video Servers

In Chapter 3 we presented the storage technologies that are most likely to be used by a video server. In order to optimize the use of the storage devices, it is necessary that the storage subsystem utilizes information about the performance characteristics for the different storage devices. In this subsection we present strategies for optimizing the use of magnetic disks in systems that store and deliver digital video.

Disk Layout

When storing video on disk, it has to be broken into *logical disk blocks*. A logical disk block is the amount of video data that will be read during one data transfer from disk to main memory. To increase the number of video streams that can be served from a disk, the server should try to minimize the relative overhead due to seek time and rotational latency by using relatively large block sizes. For video servers, the size of a logical disk block is typically in the order from 64 KB to 1 MB.

From the storage system's point of view, there exists two different types of videos. These are *Constant Bit Rate* video (CBR) and *Variable Bit Rate* video (VBR). For the storage system, CBR video is much easier to handle, since the amount of resources each video requires is constant and known in advance. VBR video is more difficult, since the amount of video the storage system must deliver per time unit is variable. VBR video has also an extra challenge when deciding the physical layout on disk. A VBR video can be stored in either *Constant Data Length* (CDL) blocks or *Constant Time Length* (CTL) blocks (Vin, Rao and Goyal, 1995). If CDL blocks are used, the video is stored in blocks of constant size. If CTL blocks are used, the logical disk blocks will have variable size. Each block will contain a fixed length portion of the video, e.g., 0.5 seconds.

Disk Layout for Multi-zone Disks

Most of the early research on layout of video data assumed that disks either had a constant transfer rate, or used the average transfer rate of the disks in the disk models. With the introduction of multi-zone disks (see Section 3.2), this is no longer a valid assumptions. For example, for a Seagate ST31200 disk, the transfer rate of the outermost zone is approximately 80 percent higher than the transfer rater of the innermost zone (Ghandeharizadeh, Kim, Shahabi and Zimmermann, 1996). In order to be able to take advantage of the increased transfer rate of the outer zones on a disk, the data placement strategy should take into account that the transfer rate is different for different zones.

Ghandeharizadeh et al. (1996) propose two strategies called FIXB and VARB for increasing the utilization of multi-zone disks. The basis for both of these strategies is to divide the video into respectively fixed size (FIXB) and variable sized (VARB) blocks. The blocks are then allocated to the disk by storing the first block in the first disk zone, the following block in the next zone and so on in until reaching the last disk zone. The process is then repeated by starting on the outermost zone. The disk is read in the same way. During a scan from the outermost to the innermost zone, in each zone exactly one block for each currently delivered video is read. For a Seagate ST31200 disk, the number of concurrent videos that can be delivered is increased by 40 percent when utilizing that the disk has zones compared to using a disk model that assumes the bandwidth of the innermost zone (Ghandeharizadeh et al., 1996). The drawbacks of this allocation strategy is that it only supports constant bit rate video, it wastes relatively much of the storage space and has high startup latencies.

An alternative strategy for utilizing multi-zone disks is *Track-Pairing* (Birk, 1995). This strategy supports both constant and variable bit rate videos and utilizes the storage space better compared to FIXB and VARB. Track-Pairing divides the disk into two parts, where the first part contains the outermost tracks and the second part contains the innermost tracks. For a disk containing totally N_t tracks, tracks are paired by grouping track i from the outermost part and track $N_t - i$ from the innermost part. When allocating a video to the disk, the video is allocated to *track pairs* by writing the first block to one or more tracks from the outermost part and the next block to the corresponding tracks form the innermost part of the disk. This process is repeated until the entire video is allocated to disk.

The Microsoft Tiger video server (Bolosky et al., 1996) is an example of a video server that utilize the multi-zone disk in how video data is stored. Each disk is partitioned into two partitions. The outer part of the disks, where the transfer rate is highest, is used for storing the primary version of the videos while the inner part of the disks is used for storing backup copies of the videos.

Disk Scheduling

Traditional disk scheduling algorithms like SSTF (Shortest Seek Time First) and SCAN do not work well for video data since they do not take into account the deadlines of when the requests have to be finished (Reddy and Wyllie, 1994). Real-time scheduling algorithms like the *deadline driven scheduling algorithm* (also known as the Earliest-Deadline-First (EDF) algorithm) (Liu and Layland, 1973) support requests with deadlines, but do not take into account the physical layout of a hard disk. As a result, by using traditional real-time scheduling algorithms, the disk performance would be poor. In order to have algorithms that take both the geometry of the disk and the real-time properties into account, algorithms like SCAN-EDF (Reddy and Wyllie, 1993) have been proposed for use in video systems.

Within video server research, *round based* disk scheduling algorithms have been the most popular (Gemmell, Vin, Kandlur, Rangan and Rowe, 1995; Gemmell, 1996). Round based algorithms require that the videos are stored as *Constant Time Length* (CTL) blocks on the disk. The playback time of each block is called the *round length*. The disk scheduler groups all requests into a round. During one round, the disk has to service *one* request for each video that is delivered to avoid interrupts in the playback. Thus, the round length is an upper limit on the time the storage system can use for deliver at least one block for each of the videos. In round based algorithms, the time it takes to retrieve all data that is needed for a round, is referred to as the service time.

Within one round, different scheduling algorithms can be used. The different videos can for example be served using a round-robin strategy or a SCAN algorithm. The round-robin strategy gives lower start-up latencies each time the user issues a VCR operation and requires less main memory buffer due to the more regular time interval between two succeeding read operations within a video. SCAN provides better disk utilization due to lower seek overhead at the cost of higher start-up latencies and more main memory buffer. The Group Sweeping Scheduling (GSS) algorithm (Yu, Chen and Kandlur, 1993) is a compromise between round-robin algorithms and SCAN. This algorithm assigns the currently delivered videos into groups. The different groups are scheduled using a round-robin strategy. With each group, SCAN is used for scheduling the requests. Common for all round based algorithms is that they work well for constant bit rate (CBR) videos. For variable bit rate (VBR) video it is more difficult to achieve efficient utilization of the disks.

Some multimedia servers might also store other media types in the server that have different timing requirements. Rompogiannakis, Nerjes, Muth, Paterakis, Triantafillou and Weikum (1998) and Wijayaratne and Reddy (2000) have proposed disk scheduling strategies that in addition to support periodic, soft real-time requests for video data, also support best effort scheduling for traditional data. Cello is a two-level scheduling strategy (Shenoy and Vin, 2002), which consists of a class-independent scheduler that allocates disk bandwidth to the applications and a class-specific scheduler that do the actual scheduling of the requests to the disks. APEX is a mixed-media scheduler that uses deadlines, while at the same time is round-based (Lund and Goebel, 2003). By batching requests to the disk, the disk utilization is increased.

4.4.2 Storage Organization

In this section we present some of the main strategies used for allocating videos to storage devices. A storage system usually consists of multiple storage devices. Each of the storage devices has a given storage capacity and a maximum data transfer rate. The main objective for a storage strategy is to be able to maximize the number of video streams that can be delivered.

The allocation strategies fall into two main groups. The first group of allocation strategies stores a complete copy of the video on the storage device. The second group distributes the videos across multiple storage devices. Most of the strategies presented in this section can be used both for allocating videos to a set of storage devices or to a set of servers.

Video Allocation Strategies

When a set of videos are to be stored on a set of disks (or other storage devices), it is important to carefully allocate the videos to the disks in order to achieve a good load balance when accessing the disks. If this is not achieved, during periods with high load some disks will be over-utilized, while other disks will be underutilized. If one or more disks are about to become over-utilized, the admission control strategy has to reject new requests for videos stored on these disks, even if the system still have free I/O resources on most of the other disks.

One proposed strategy uses the *popularity* of each video for assigning the videos to storage devices (Little and Venkatesh, 1995). Based on access data for each video during a previous time period, the probability of access (popularity) for each video can be computed. These access probabilities are then used for assigning videos to disks. In order to minimize the probability of having to reject user requests, the videos are assigned to disk such that each disk has approximately the same probability of being accessed. Videos with high probability for being accessed might be replicated on multiple disks.

If the popularity of the videos changes, the access probabilities have to be recomputed and the videos have to be reallocated to the disks. This means that some videos must be moved from one disk to another disk, some videos will have one of its copies deleted and some videos might get one more copy. This reallocation can be done during periods with low load, e.g., during the night (Little and Venkatesh, 1995).

The popularity-based assignment strategy proposed by Little and Venkatesh (1995) does not take into account that videos might have different storage requirement due to varying length and that the video server might consist of different types of disks. Dan and Sitaram (1995) have proposed an allocation strategy called *Bandwidth-to-space ratio* (BSR) that takes these factors into account. Each storage device is limited by the number of concurrent video streams it can deliver (*bandwidth*) and the amount of video it can store (*space*). This is referred to as the *bandwidth to space ratio* of the storage device. Similarly, with an estimated probability for being accessed, each video object can also be characterized by the expected bandwidth and storage space it will require. Based on BSR numbers for both storage devices and the video objects, the BSR policy gives the number of replicas needed for each video and the storage device each video should be stored on. The BSR strategy can both be used for initial placement of videos and for changing the placement when the expected usage for a video changes. If the

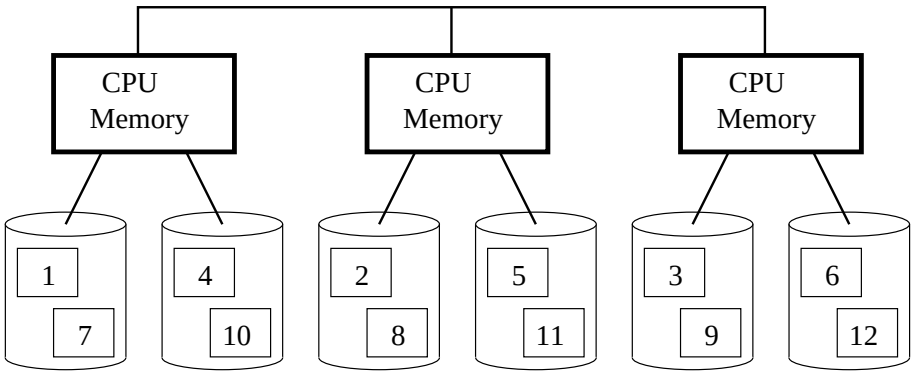


Figure 4.5 Striping of the first twelve blocks of video across all disks in a parallel video server.

expected demand for a video is changed, the policy can be used for dynamically changing the number of replicas of the video or to reallocate the video to better utilize the bandwidth and storage space of the storage devices.

The two strategies presented this far do *static* allocation of videos to storage devices (disks) based on expected access probabilities. If the access probabilities change, the strategies can be used to recompute which videos that must be replicated, deleted or moved to another disk. There are also strategies that do *dynamic* reallocation of videos. These strategies are better at handling short term changes in the load against different storage devices or videos.

One example of a strategy that do dynamic reallocation of videos is *Dynamic Policy of Segment Replication* (DPSR) (Dan, Kienzle and Sitaram, 1995b). DPSR divides each video into a small number of *segments*. DPSR monitors the *load* on each segment, i.e., the number of requests for the segment. If the load on a segment is high, the video this segment belongs to is candidate for being replicated on one extra disk in order to achieve better load balance. To make the replication process as cheap as possible it is done as part of the video delivery to the user. The first time the *following* segments of the video are read as part of a delivery, the video data is written to a disk with free resources (both storage space and bandwidth). By dividing the video into segments and by dynamically monitoring the load on each segment, the DPSR policy can quickly adjust to changes in the load on different videos and achieve better load balancing.

Striping

An alternative strategy to store a complete copy of the video on one storage device, is to distribute the video evenly across multiple disks. This is referred to as *striping* of the video (Özden, Rastogi and Silberschatz, 1996b). The videos are di-

vided into contiguous data units. These data units are distributed among the disks using a round-robin strategy. The size of the contiguous data unit that is stored on a single disk is referred to as the *striping unit* (Salem and Garcia-Molina, 1986).

The goal of using striping is to distribute the workload uniformly across the disks in order to achieve high utilization of the disks. This can help in reducing the total cost of the server. Other advantages are that this strategy does not rely on knowledge of access probabilities. Each video can be delivered to many users without having to replicate the video on multiple disks.

For a video server consisting of only one computer, striping is normally done by striping all videos on all disks. In a video server containing multiple machines, multiple strategies for striping exists. A video might be striped on the disks of only one of the machines, it might be striped across all disks of all machines, or it might be striped across all the machines. In the last configuration, the stripe units can be stored on one of the disks on the machine, or be further divided into smaller units and striped across the multiple disks on the machine. An example showing how a video is striped across all disks in a video server is shown in Figure 4.5.

One of the important decisions to make when deciding on a striping strategy is to determine the size of the *stripe unit*. There exist two major strategies for selecting the size of the stripe unit (Özden et al., 1996b):

1. *Fine-grained striping* uses a striping unit that is normally a bit, a byte or a sector. With such small striping units, all disks must work in parallel to deliver video data for each stream. This is similar to the allocation strategies used by RAID level 2 and level 3 (Chen et al., 1994).

2. *Course-grained striping* uses striping units that are much larger. Typical sizes could be from 64 KB to 1 MB. With striping units of this size, only one disk at the time will participate in the delivery of a video stream.

Course-grained striping retrieves larger amount of data per disk access, which reduces the overhead for positioning of the disk head and increases the utilization of the disk bandwidth. Thus, the performance of course-grained striping outperforms fine-grained striping (Özden et al., 1996b).

When using course-grained striping, the optimal amount of data to be retrieved during one disk access must be determined. During one disk access, one stripe unit will normally be read from the disk and to main memory. Using large stripe units increase the utilization of the disks. Unfortunately, having a large stripe unit also have some drawbacks. For each video that is delivered, it must be room for storing at least one stripe unit in main memory. Thus, increasing the size of the stripe unit also increases the need for more main memory. Larger stripe units will also increase the average response time when a user requests a

video to be deliverer and when she issues VCR operations like fast forward and backward.

One of the most referenced striping strategies is *Staggered striping* (Berson, Ghandeharizadeh, Muntz and Ju, 1994). Staggered striping is a striping strategy that supports videos with different data rates and videos having higher data rates than a single disk can support. In this strategy, the video is segmented into parts, where each part is stored on one disk or striped over a few disks if a single disk can not handle the bandwidth. This strategy is mainly useful for videos with a very high bandwidth, e.g., uncompressed video.

Despite the ability to achieve high disk utilization by using striping, striping also has some problems:

- Most striping strategies assume that all videos have the same data rate. Since striping requires that each disk servers the currently delivered video streams for a given amount of time, mixing videos with different data rates normally lead to a lower utilization of the disk bandwidth.

- Most striping strategies do not support VBR video well. To handle VBR videos, either the disk utilization will be lower or a much higher amount of main memory buffer must be reserved in order to handle peaks in the data rate for the videos.

- The complexity of supporting multiple types of disks is high. Striping normally requires that all disks have the same performance data. If the videos are striped across all disks:

  - With disks with different bandwidth, the *slowest* disk limits the total number of videos that can be delivered.
  - With disks with different size, the *smallest* disk limits the amount of data that can be stored on all disks.

  Several solutions to this problem have been proposed. The disks can be divided into multiple *striping groups*, where the disks within one striping group have similar specifications (Dan et al., 1995b). *Weighted striping* allocates more data to the fastest disks (Wang and Du, 1997). *Disk merging*, *staggered grouping*, and *disk grouping* assign the *physical* disks into *logical* disks where the logical disks have similar performance (Zimmermann and Ghandeharizadeh, 1997). Common for most of these strategies is that they are only able to utilize the bandwidth or the storage space of the disks. *Resource-based striping* is a striping strategy which is able to utilize both the bandwidth and storage space (Ding and Huang, 2003). It uses information about access probability of the different videos to stripe the videos on all or a sub-set of the disks.

- Reconfiguration of the server is complex. Adding new disks or server nodes might require all videos to be reallocated. This problem can also be reduces by using multiple striping groups (Dan et al., 1995b).

- The consequence of a disk failure can be very dramatic. If all videos are striped across all disks, the failure of one disk results in unavailability of all videos (Berson, Golubchik and Muntz, 1995). This problem can be reduced by having multiple striping groups or eliminated by storing parity data or replicating data on multiple disks. The Tiger video server is an example of a striping video server that is replicating the video data on multiple disks to increase the availability (Bolosky et al., 1996).

Some of these problems with striping are caused by the uniform layout of video and that all the disks read in synchronized cycles, where the length of the cycle is determined by using worst case I/O times (Santos, Muntz and Ribeiro-Neto, 2000). Thus in a video server using striping, most of the disks are likely to be idle during the last part of a cycle. Due to this problem with striping, alternative allocation strategies commonly referred to as *Random Data Allocation* have been proposed.

Random Data Allocation

As an alternative to striping, several papers suggest to use *random data allocation* in order to improve the performance and overcome some of the limitations of striping (Tewari, Mukherjee, Dias and Vin, 1996; Korst, 1997; Birk, 1997; Santos et al., 2000). Compared to striping, random data allocation applies a very simple strategy for allocating blocks to the disks. Each new block of video to be stored in the server is written to a *random* free position on a *random* disk. Strategies for reading video data can be done by using any of the traditional disk scheduling algorithms like SCAN or FIFO (Santos et al., 2000).

Since each video stream does not have its own slot in a striping cycle, but possibly have to compete with all other streams about a given disk, and since there is no deterministic load balancing for the disk accesses, systems using random data allocation can only provide statistical guarantees for delivering the video data in time. To reduce the probability of missing a deadline, some of the systems use replication of data (Korst, 1997; Birk, 1997; Santos et al., 2000). For example, the strategy presented by Santos et al. (2000) replicates a small fraction of the data blocks on multiple disks. When a replicated block is requested, the request is directed to the least-loaded disk that has a copy of the block.

One implemented video server that uses random data allocation is the RIO multimedia storage server (Santos and Muntz, 1998). Experiments using RIO have shown that random data allocation outperforms striping when delivering VBR video and performs comparable or better when delivering CBR video (Santos et al., 2000).

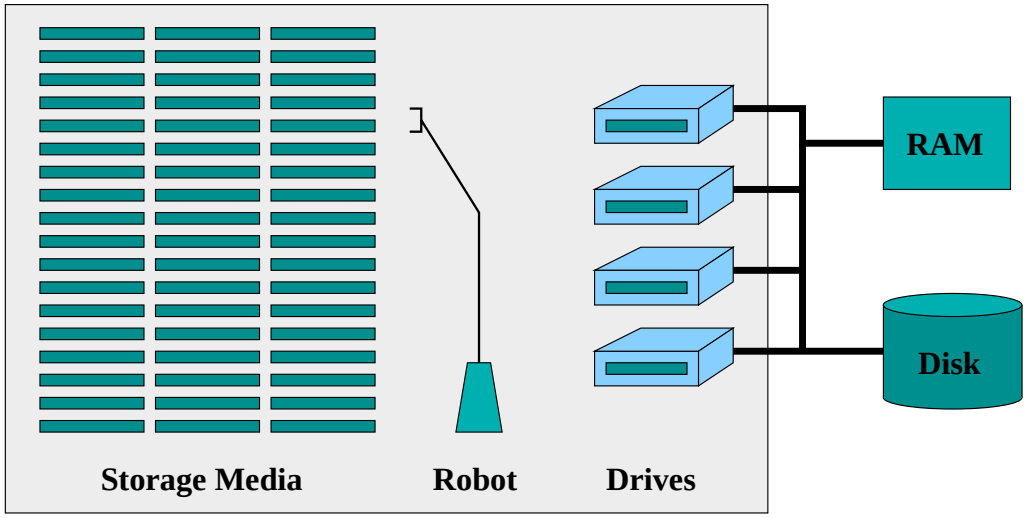


Figure 4.6 Architecture of a tertiary library unit containing four tertiary media drives.

4.5 Using Tertiary Storage for Digital Video

Chapter 3 gave an overview of the main tertiary storage technologies. In this section we focus on issues related to how to efficiently use tertiary storage for storage and retrieval of digital video. A tertiary storage system normally consists of one or more tertiary library units. Each library unit contains tertiary drives, a robot mechanism, and storage media. A generic architecture containing the main components of a library unit is given in Figure 4.6. The main resources of tertiary storage systems are:

- **Tertiary Storage Media.**[4] These are used for storing the digital video. When they are not mounted in a drive, they are stored in the library unit's *shelf*.
- **Tertiary Media Drives.** The tertiary media drives are responsible for reading (and writing) the storage media.
- **Robots.** The robot mechanism is responsible for transporting storage media between the *shelf* and the tertiary drives.
- **I/O buses.** Video data read from the storage media is transported to a host computer's main memory or a disk based storage system using a storage bus. Multiple standards exist for storage interfaces and buses (LoBue, 2002).

[4]These are often referred to as Removable Storage Media (RSM).

When there is a request for a video stored on one of the storage media, and the medium is not already present in one of the drives, it has to be transported from the shelf to an available drive. The drive will *mount* the storage medium and read the video data from the medium. As the drive reads the video, the data is transported to a host computer for further delivery to the client. After the video is read, the medium is normally *unmounted* and transported back to the shelf.

Most of the proposed solutions for improving the performance of the tertiary storage system utilize one or more of the following strategies:

- **Scheduling.** When there are multiple concurrent requests for video sequences, the utilization of the tertiary drives and the robots can be increased by intelligent scheduling of the requests.
- **Data placement.** By carefully choosing how the video sequences are allocated to storage media and to library units, the cost of retrieving them can be reduced by minimizing the access time for the video sequence on the storage medium, by reducing the number of load/unload operations, and by achieving better load balancing. In order to do clever data placement, it is necessary to know the access probability for the different video sequences. We study several allocation strategies in Chapter 13.
- **Data replication.** Replicating video sequences or parts of video sequences provides more than one access path to a given video sequences. This can be used for reducing the consequences of media that become hot spots in the storage system, and for improving the load balancing across both tertiary drives and library units.
- **Caching.** By using a cache based on secondary storage, both the load on the tertiary storage system and the response time of the most frequently accessed video sequences can be reduced.

Most of the studies of using tertiary storage for video have used an architecture similar to the one shown in Figure 4.6 where the number of storage media is much higher than the number of media drives. There are also some projects that have proposed to use as many media drives as there are storage media (Shastri, Rangan and Sampath-Kumar, 1997; Chervenak, 1998; Tsao, 2001). This simplifies the storage system since there is no longer need for having robots for transporting media to and from drives, at the cost of a much higher number of drives. One video server that stores the media permanently in the drives is proposed by Shastri et al. (1997). Due to the low cost of mass produced DVD drives they argue that using one DVD drive per DVD medium is of comparable cost to magnetic disks.

We start this section by discussing how accesses to a single medium can be optimized by use of scheduling. We continue with a presentation of techniques for optimizing concurrent requests to storage media that must compete for a limited number of drives and presents strategies for allocating video data to storage

media. In the final part of this section, we present projects that have evaluated use of tertiary storage for storing digital video.

4.5.1 Modeling and Scheduling of a Single Tertiary Medium

While a lot of work has been done on modeling and scheduling of random accesses on magnetic disks, less research has been performed for optimizing accesses to tertiary storage media. This subsection gives an overview of some of the proposed models and scheduling algorithms for tertiary storage media. The focus is on two of the most used tertiary storage media, optical disks and magnetic tapes.

Optical Disks

In order to improve the performance of hard disk based storage systems, modeling and scheduling have been studied extensively. Section 4.4 gave an overview of some of the main strategies for optimizing video delivery from hard disks. Unlike hard disks, many optical disk drives are *constant linear velocity* (CLV) drives. Thus, the disk models and scheduling algorithms developed for hard disks do not necessarily work well for optical disks.

An overview of optical storage media and DVD in particular was given in Section 3.3. For storage of video data, CD and DVD drives are the most widely used. Initially, both CD and DVD were CLV drives, but with the increased use of these in computers and particularly in applications requiring random access to data, there are also CD and DVD drives that are performing as CAV drives. To improve the transfer rate alternative data layout formats like P-CAV and ZCLV have been implemented (Sadashige, 2000).

General optical disks. In order to estimate the cost of retrieving data and for scheduling the requests, it is necessary to have a model of the performance of the disk drives. An early performance model of CLV optical drives is proposed by Christodoulakis and Ford (1988). Both an exact model for CLV drives and a model based on approximating the CLV drive as a CAV drive are proposed. A more general analytical model for disks is presented in (Triantafillou et al., 2002). By instrumenting the general model, a wide variety of disk technologies can be modeled. The model supports both CAV and CLV disks, as well as zoned disks, and disks that operate as both CLV and CAV depending on the position on the disk.

Lijding (2003) proposes an analytical model for optical disks. This model approximates the data layout on the disk by using concentric circles instead of a spiral. The model uses physical data about the geometry of the disk to provide a mapping from logical byte addresses to physical positions on the disk. To estimate the time needed to access data on a disk, an analytical model of the behavior

of drives are presented. This model supports both CLV and CAV drives. For each drive type, the model must be instrumented with data about the drive like rotation speed, time to insert and mount the disk, time to accelerate the disk, and time to stop and eject a disk from the drive. The model has been validated by using two different CD-ROM drive models, one CLV and one CAV drive (Lijding, 2003).

This optical drive model is used for estimating the cost of retrieving multimedia data and for scheduling multiple requests for multimedia data on a single medium. A scheduling strategy called Latest Deadline Last (LDL) is proposed (Lijding, 2003). All requests have an assigned deadline for when they have to be retrieved from the disk. This deadline is used for sorting the requests. If multiple requests have the same deadline, they are ordered based on a SCAN algorithm. Unlike most other disk scheduling algorithms for retrieval of video data, the entire video object is read in one operation. There is no multiplexing between different video streams. The LDL algorithm is similar to the EDF-SCAN algorithm (Reddy and Wyllie, 1993).

CD-ROM. Shastri, Rajaraman, Jamadagni, Rangan and Sampath-Kumar (1996a) present a performance model for CD-ROM drives. This model provides estimates for the seek time of requests to data stored on a CD-ROM. In (Shastri, Rangan, Rajaraman, Pittet and Kumar, 1996b), this model is used for evaluating the C-SCAN (Circular Scan) algorithm in a video-on-demand server using CD-ROMs as the main video storage. The performance and buffer requirements of using the C-SCAN algorithm for serving multiple video streams from a single CD-ROM is evaluated. The CD-ROMs are permanently stored in the CD-ROM drive.

A similar approach is used by Tsao, Huang, Lin, Liou and Huang (1997) for delivering video from CD-ROM. Their target is to use CD-ROMs as the main storage for a near video-on-demand server. In order to increase the number of streams that can be delivered from a CD-ROM drive, they investigate data placement strategies for reducing the amount of time the drive spends for seeking. Their solution is to re-order the blocks of the video streams so that the video blocks that are to be delivered during the same time period (service round) for all channels are grouped together and can be read continuously. The proposed data layout only supports CBR video and is only useful in a near video-on-demand server. In (Tsao, 2001) the data layout strategy has been extended to support VBR video.

DVD. Due to the much higher storage capacity and transfer rate provided by DVD, it is better suited for being used in a video storage system. A performance model for seek times of a DVD is presented in (Shastri et al., 1997). This model is used for evaluating use of DVD-ROM in a video server. The video server consists of multiple DVD drives each containing one DVD disk. The video is streamed directly from the DVD drive to the user, and the purpose is to serve multiple concurrent streams from each DVD drive. In our study of tertiary storage technolo-

gies for use in video archives, we use the performance model presented in (Shastri et al., 1997) for estimating seek times for DVDs.

Magnetic Tape

An overview of the main digital tape technologies was given in Section 3.4. Due to the large differences in the data layout on tapes based on different tape technologies, it is difficult to make access-time models for data stored on tape that cover more than one of the technologies. Most of the proposed models are tailored for only one drive or drives based on the same technology.

Hillyer and Silberschatz present a detailed model for seek times[5] of a Quantum DLT 4000 drive (Hillyer and Silberschatz, 1996a). The DLT 4000 drive is based on serpentine technology and uses *key points* along each track for locating a given block. Each track contains 13 key points. The proposed model uses the addresses of the key points for deciding how the drive will navigate on the tape to get to a given position. The model consists of a piecewise-linear function for *sections* between the key points. This function consists of 8 major cases and 9 additional subcases for estimating the seek time between two locations on a tape. The proposed model is able to estimate the seek time between any two locations with an error of less than 2 seconds for more than 99 percent of all seeks. The major drawback with this model is that each tape must be characterized by finding the location of all the key points on the tape. This characterization requires tremendous amounts of analysis. To characterize a single tape takes approximately 12 hours of accesses to the tape. In Chapter 6, we extend the work done by Hillyer and Silberschatz and propose a more general model for serpentine tape drives that requires substantially less time for charactering the individual tapes.

The seek time model in (Hillyer and Silberschatz, 1996a) is used for evaluating scheduling algorithms for concurrent accesses to a tape. In (Hillyer and Silberschatz, 1996b), several scheduling algorithms are presented and evaluated. Since we use these algorithms later in this thesis, we provide a short overview of these algorithms here:

- **READ (read the entire tape).** The entire tape is read from the beginning to the end.
- **FIFO (first in, first out).** Read the data from the tape in the order presented to it by the application, i.e., without doing any scheduling of the requests.
- **OPT (the optimal order).** By using the seek-time function, the cost of all possible permutations of the requests are evaluated.
- **SORT (sort on segment number).** The requests are sorted by the *logical* address they have on the tape.

[5] In the paper, the seek time is referred to as the *locate time* of a request.

- **SLTF (shortest locate time first).** The request having the shortest seek time is served first. This process is repeated until all requests are included in the schedule.

- **SCAN (elevator algorithm).** The requests are sorted based on physical position on the tape. The tape is scanned twice. On the forward scan, all requests on the forward tracks are read. On the reverse scan, all requests on the reverse tracks are read. The reading is done one section at a time.

- **WEAVE.** The scheduling is based on following a predefined relative ordering of the sections on the tape. The ordering of the sections is an approximation to the path SLTF would produce.

- **LOSS.** LOSS is a greedy algorithm for providing solutions to the asymmetric traveling salesman problem (Cruyssen and Rijckaert, 1978).

Based on the performed experiments, the conclusions is that OPT is the best algorithm for up to about ten requests. For larger number of requests, OPT is too costly to compute, and LOSS should be used. If there are more than 1500 requests, the fastest way to read all of them, is to read the entire tape. In Chapter 7, we propose a new scheduling algorithm, which we compare against the algorithms suggested by Hillyer and Silberschatz.

Hillyer and Silberschatz (1998) extend the study of modeling and scheduling of requests to serpentine tape to use an IBM Magstar MP tape drive. This drive uses a modified serpentine layout designed for fast random access. Due to a much simpler seek function for this drive, the SCAN and SLTF algorithms perform close to an optimal scheduler. In our video archive simulator, we use the seek time model presented in (Hillyer and Silberschatz, 1998) for simulating the IBM Magstar MP tape drive. In Section 9.3.1, we present more detailed information about this drive and the performance model.

An alternative strategy to scheduling of requests is to do intelligent data placement on the tape. For serpentine tape, Dashti and Shahabi (2000) have suggested an allocation strategy called Wrap ARound Placement (WARP). This strategy utilize the serpentine layout pattern to place data objects onto tracks on the tape, so that each object starts on the beginning of a forward track and ends on the end of a reverse track. Each object occupies at least two tracks on the tape. As a consequence of this, the strategy works best for medium to large sized data objects. This strategy minimizes the initial seek time for an object after loading a tape and reduces the rewind time after having read the object. The main disadvantage of WARP is that it wastes storage space. Using an IBM 3590 tape drive, up to 5 times improvement in access time as compared to other data placement techniques have be observed (Dashti and Shahabi, 2000).

Tape drives are implemented using a wide array of technologies. In (Johnson and Miller, 1998b), the performance of six different tape drives are evaluated.

Both low and high performance drives, serpentine and helical scan drives, and cartridge and cassette tapes are included in the evaluation. For each of these drives, mount time, seek times from both beginning and the middle of the tape, transfer rate and unmount time are measured and presented. For some of the operations and drives, analytical models are suggested for the performance. A simple piece-wise linear regression model is suggested for estimating seek times on serpentine tape drives. For evaluating the performance of new tape drives, they propose a benchmark model. An overview of this benchmark is presented in (Johnson and Miller, 1998a). For linear tape, Li and Orji (1996) have studied efficient scheduling of multiple requests to a single tape.

4.5.2 Scheduling of Tertiary Media

In a library unit there are many storage media, multiple drives and one robot mechanism. The drives, the robot, and even the storage media might become the bottleneck that limits the performance of the storage system. To make optimal use of the resources, it is necessary to schedule the requests for data objects stored on the storage media. In the previous subsection we presented some scheduling strategies for scheduling concurrent requests to *one* storage media. In this subsection we present work on scheduling algorithms that schedule concurrent requests for data objects to multiple media. Such a scheduling strategy must determine the order for when to load and unload the storage media in a drive and which of the potentially many requests that should be served during this load cycle.

There are several criteria that can be used for optimization. The scheduler can for instance optimize on average wait time or on throughput. Most scheduling strategies can be classified as either *real-time* or *non real-time*. A real-time scheduler supports guarantees for executing requests within a negotiated time limit, while a non real-time scheduler will only focus on optimizing for instance the throughput of the storage system.

Schedulers for requests to video data stored in a tertiary storage system can be categorized as either periodic or aperiodic. Most of the scheduling strategies for retrieving video data from hard disk presented in Section 4.4 are periodic. A periodic scheduler will serve one video for a given amount of time and then switch to the next video, whereas an aperiodic scheduler will read the entire video or a large part of it as one operation.

Periodic Schedulers

Lau and Lui (1997) propose two scheduling algorithms, *round-robin* and *least-slack* for scheduling of retrieval of video from tape. These algorithms break the requests into many tasks, where each task is served separately. Each task is assigned a time slice period, which includes time for switching tape in a drive, read the video data and rewind the tape. The main advantages of using these

strategies are that the average response time can be reduced since each tape drive serves multiple videos instead of just one at the time, and if a request is canceled after it has been started, a smaller amount of video data is read from the tape. It also leads to better utilization of the disk cache and the tape drives. If the disk cache is full, the tape drives have to wait for a free buffer. Using these strategies, each video requires less space in the disk cache. The main disadvantages of these strategies are a much higher number of tape switches, which might lead to the robot become the bottleneck, and more wasted time by the drives for seeking and rewinding.

A similar scheduling strategy called *rounds* is proposed by Golubchik and Rajendran (1998). The videos are stored as pages of constant size. The pages are distributed randomly over the media, i.e., to retrieve one page requires a tape switch. The main improvement over the scheduling strategies proposed in (Lau and Lui, 1997) is that video streams with different bandwidth and sizes are supported.

Lijding, Hanssen and Jansen (2002a) propose a scheduling strategy called JEQS (jukebox early quantum scheduler). Using *early quantum task* scheduling (Jansen, Hanssen and Lijding, 2003), the initial filling of the main memory buffer can be scheduled at an earliest possible time, while the following tasks are scheduled in a normal periodic way. The purpose is to achieve a lower initial response time. The authors claim that this is the first periodic scheduler that take resource-contention problems of the robot and media into account, and thus the only period scheduler that can guarantee that the deadlines are met. One of the most interesting findings in their evaluation is a comparison of JEQS against an aperiodic scheduler. The authors claim that due to the many costly media switches and lower resource utilization, using any periodic scheduler leads to higher response times than using an aperiodic scheduler.

Aperiodic Schedulers for Tertiary Drives

An aperiodic scheduler does not switch between multiple videos. It completes the current job before it switches to the next task. This reduces the number of media switches, but may lead to higher start up times for arriving requests since currently scheduled requests must complete before a new request can be started. We start with an overview of some scheduling strategies that only consider the media drives when performing the scheduling.

Prabhakar, Agrawal, Abbadi and Singh (1997) have studied the problem of scheduling I/O requests for robotic libraries containing tape drives by finding the optimal order of inserting the tapes into the drives. The scheduling strategy is static and assumes that all requests are issued before the scheduling starts. The goal is to reduce the total time used by the library to execute all requests. Their main finding is that it is beneficial to read all requests on a medium before switching to the next medium, since this minimizes the number of media switches. They

also show that it is beneficial to schedule concurrent requests to the same tape. They do not take fairness into account.

In (Prabhakar, Agrawal and Abbadi, 2003), the study of optimal scheduling of I/O requests is continued. In order to improve the average waiting time, they extend the minimum switching scheduler proposed in (Prabhakar et al., 1997) to include an heuristic on the order of loading the storage media into drives. By sorting the media on the number of requests, and starting with the medium with the highest number of requests, they show that this scheduling strategy performs close to an optimal scheduling strategy. They validate that the minimum switching strategy is optimal by showing that introducing extra media switches increases the average waiting time. For optical disks, this was always the case. For tape, they found that in a very few cases, the average waiting time could be reduced by introducing an extra tape switch.

Moon and Kang (2001) has done a similar study as in (Prabhakar et al., 2003). They evaluate several heuristics for scheduling the order of tertiary media to drives. Compared to the work in (Prabhakar et al., 1997; Prabhakar et al., 2003), they perform dynamic scheduling by allowing new requests to arrive after the initial scheduling has been performed. Each time all requests for one medium has been executed, re-scheduling is performed. The two main strategies studied are MNQ (Maximum Number of Queries) and MPT (Maximum Processing Time). Each time a drive is free, the medium with respectively the highest number of requests or highest estimated processing time is selected. They show that MNQ results in the lowest average response time and the highest throughput.

Triantafillou and Georgiadis (1999) propose a hierarchical scheduling algorithm called CLUST consisting of two levels. Their goal is to have an efficient scheduling strategy that do not suffer from starvation. The upper level algorithm selects which medium to next be inserted in an available drive. A modified version of a round-robin strategy is used where it is possible for a storage medium with many requests to be scheduled multiple times during a round. Each time a tape is selected by the upper level scheduling algorithm, the first N requests are extracted and put in a queue for this drive. The algorithm used for the lower level is OPT (the optimal scheduler), which is responsible for scheduling the requests that are placed in a drive's queue. Compared to the previous presented scheduling strategies, CLUST only schedules the N first requests for each media in order to avoid starvation.

Aperiodic Schedulers for Drives and Robots

The aperiodic schedulers presented this far have only included the tertiary media drives in the scheduling model. They do not include scheduling of the robot mechanism. This may lead to conflicts when multiple media should be moved between the shelf and drives at the same time. Potentially, this may lead to lower throughput and not being able to retrieve the video data within the estimated

time. In the remainder of this subsection we present some scheduling strategies that include the robot mechanism when doing the scheduling.

Lau and Lui (1996) study scheduling and cache replacement policies in a multimedia server. Two alternative periodic scheduling strategies are evaluated: *aggressive* and *conservative*. The aggressive scheduling strategy schedules each job as early as possible while the conservative strategy schedules and dispatches each job as late as possible. The main application of this work is video-on-demand applications. The video files are divided into large pages. When a user has requested a video file, the pages are retrieved from tertiary storage and stored in a disk cache. Experiments in (Lau and Lui, 1996) show that the aggressive scheduling strategy makes better use of the tertiary storage resources, but leads to a higher page miss ratio. The conservative strategy leads to low utilization of the tertiary storage resources and leads to high response times when the load increases.

The *Relief* algorithm is proposed by (Georgiadis, Triantafillou and Faloutsos, 2002). This algorithm aims at improving the performance by minimizing the average start up time. It uses an *aging* mechanism to ensure fairness and to avoid starvation of requests. For each request, the *relief ratio* is computed as the amount of time the requests have been waiting divided by the amount of time it will take to serve the request[6]. The request with the highest relief ratio is selected for the next free drive. Thus, requests that have been waiting for long time or have a short service time, will be selected. The algorithm incorporates support for multicasting. If there are multiple requests for the same video, the sum of the wait time is used for computing the relief ratio. Experiments show that Relief has a low reject ratio and achieve a higher throughput than *Bypass* scheduling (Christodoulakis, Triantafillou and Zioga, 1997) and *Maximum Queue Length* scheduling (Dan et al., 1996).

More and Choudhary (2000) study scheduling of queries for data objects stored on multiple tapes. The goal is to minimize the response time of the queries. The scheduling strategy models the scheduling problem as a two-machine flow-shop. They study two cases. The first is when the size of the requested data is small. In this case they assume the robot will be the bottle neck due to frequent tape switches. The proposed scheduling strategy for this case is to schedule the jobs in decreasing order of their execution time. The second case is when the size of the requested data is large. In this case, the robot is not a bottleneck. For this case, they propose a heuristic based on scheduling the shortest jobs first.

Lijding (2003) proposes a real-time scheduler called *Promote-IT*. This scheduler supports *complex requests* for data stored on multiple tertiary storage media. For each such request, the scheduler gives a guarantee for which time the data will be ready in a disk cache. To be able to give time guarantees, the scheduler depends on having a detailed model of the performance of the hardware, both the robot mechanism and the drives. As most of the other aperiodic schedulers pre-

[6]It is assumed that each tape only have one video.

sented here, Promote-IT is based on a minimum switching model. The scheduler separates *scheduling* and *dispatching*. The scheduler determines the time limits for when the requests must be finished. The dispatcher may execute the requests earlier than scheduled as long as this does not break the deadline of any other request. In this way, the resource utilization and performance can be improved.

4.5.3 Allocation of Data to Storage Media

An alternative approach for improving the performance of a tertiary storage system is to carefully allocate the data objects to storage media. Christodoulakis et al. (1997) propose an analytical model for assigning data objects to disks and tapes in tertiary storage systems. For disk based systems, they show that in a system containing only one media drive, the optimal placement of data objects is to order the objects by access probability and then allocate the objects with the highest access probability to the first disk and so on. This optimizes the probability of being able to serve multiple requests each time a disk is loaded into a drive, and thus reduces the average access time. In the paper, they show how this strategy can be extended to systems with multiple disk drives under the assumption that the *least popular disk* is selected for replacement each time a new disk needs to be loaded into a free drive. This strategy minimizes the number of disk exchanges, but it is not likely to make optimal use of the disk drives. In Chapter 13, we show that the drive containing the most popular videos may become the resource that limits the performance when using this strategy.

For allocation of data objects to tape, Christodoulakis et al. (1997) only consider the seek and rewind times of the tape in the evaluation. The time consumed by the robot mechanism and for mounting the tape in the drive is not taken into account when computing the access cost. The model further assumes that the seek time on a tape is proportional to the distance between any two positions on the tape. Thus, this model is not useful for optimally placing data objects on serpentine tape. Optimal placement for two cases are studied. For tape drives that always rewind the tape to the beginning of the tape after being used, the optimal placement is to distribute the most popular data objects on the beginning of the tapes. For tape drives that support multiple unload zones, the optimal placement strategy is to distribute the most popular data objects on the middle of the tapes and then use an organ-pipe placement strategy based on access probability for the remaining data objects.

One problem when delivering video from tape is the mismatch between the transfer rate of the tape drive and the consume rate for the video. The straightforward solution to this mismatch is to use either main memory or magnetic disk as buffer. However, both main memory and disk bandwidth may be scarce resources in a video storage server. Triantafillou and Papadakis (1997; 2001) propose a data layout scheme for storing video on tape that reduces the need for main memory buffer and temporary disk buffer and at the same time manages to

utilize the bandwidth of the tertiary drives. The main problem with this strategy, is that it makes it hard to support VCR operations and to optimize the use of the system in cases where multiple users are requesting the same video.

Hillyer, Rastogi and Silberschatz (1999) study placement strategies for allocating data objects to tapes in a tertiary library containing one drive. Both strategies for non-replicated and replicated data is studied. A scheduling algorithm referred to as the *envelop-extension* algorithm is proposed. This algorithm takes into account that some of the data objects is replicated on multiple tapes. For the non-replicated case, their study shows that the hot data should be stored on one or as few tapes as possible. They show that by replicating some of the hottest data objects, the throughput can be improved. In the case of replicated data, they show that the replicated data should be placed at the *end* of the tape.

Nemoto and Kitsuregawa (1999) have implemented and evaluated two strategies for handling uneven load on the tertiary storage system due to a skewed access pattern. These strategies are able to handle that the access distribution changes. The first, *hot declustering*, is a strategy that migrates storage media between library units when the load on the different library units are uneven. The second strategy, *hot replication*, is using idle drive resources for replicating the most popular data objects. During the initial write of a tape, the tape is not completely filled. The storage space on the end of the tape is used for storing replicas of hot data objects. This increases the probability of being able to retrieve multiple objects during a single tape load. It also gives the system alternative media to use when a replicated data object is requested. As the popularity of the replicated objects changes, they can be overwritten with other more popular data objects.

4.5.4 Use of Magnetic Disk in Tertiary Storage Systems

Due to high latency of the tertiary storage system, the high cost of tertiary bandwidth, and the mismatch between transfer rate of the tertiary drives and the bandwidth of the video, many systems use a cache consisting of magnetic disks for streaming the video to the users or for caching the most frequently accessed video sequences. Tertiary storage systems for video differ in how the video is delivered to the user. Most of the systems can be divided into either *direct-streaming* where the video data is delivered directly from the tertiary storage device to the user, *fully staging* where the data is completely stored in secondary storage before the streaming to the user is started, or they use *pipelining* (Ghandeharizadeh, Dashti and Shahabi, 1995) or *stage-streaming* (Chan and Tobagi, 2003) where the tertiary storage system stages the data to secondary storage while at the same time the data is streamed to the user from primary or secondary storage.

Kienzle, Dan, Sitaram and Tetzlaff (1995) evaluates three strategies for how videos in a video server can be stored and delivered. The first strategy, *disk play*, stores the video on hard disk. The second strategy, *direct mode*, stores the video on tertiary storage and delivers the video directly form the tertiary drive to the

user. The last strategy, *staging mode*, stores the video on tertiary storage, but uses hard disks for staging the video data and streaming the video to the user. The model uses the sum of the storage cost and the delivery cost to determine which of these strategies should be used for each video. Their study show that data placement decisions depend on the access probability of the videos. Frequently accesses videos should be stored on hard disk, while less popular videos should be stored using tertiary storage. To deliver the video directly from a tertiary drive or to use staging depends on the the ratio between the transfer rate of the tertiary drive and the data rate of the video. If the transfer rate of the tertiary drive is much higher than the data rate of the video, the videos should be staged to hard disks. Otherwise, the video should be streamed directly from the tertiary drive. A similar model for determining where in the storage hierarchy a video should be place is presented in (Doğanata and Tantawi, 1994; Doğanata and Tantawi, 1996). An algorithm that dynamically chooses between staging and direct delivery from the tertiary storage devices is proposed by Pang (1997).

Chan and Tobagi (1999) present a model of a hierarchical storage system that supports video-on-demand. The system consists of a video cache based on magnetic disks and a tertiary storage system. In order to support user interaction of the playback, the video has to be completely transferred from the tertiary storage system to magnetic disks before playback can be started. This increases the initial start-up delay observed by the users. By use of analysis and simulations, they develop an analytical model for the performance of the storage system. This model can be used for determining the amount of storage and bandwidth required for the secondary and tertiary level given requirements for start-up delay and user load. Compared to the model of the tertiary storage system we present in this thesis, the model in (Chan and Tobagi, 1999) is very simple. The tertiary storage system is characterized only by the aggregate bandwidth of all the tertiary drives. It is assumed that the bandwidth can be used efficiently, e.g., all drives can be used for retrieving one video. The requests to both the secondary level and tertiary level are executed in a FCFS order, i.e., no scheduling of the requests are performed.

In (Chan and Tobagi, 2003) alternatives to completely stage the video in secondary storage before delivering the video to the user is studied. The same model as in (Chan and Tobagi, 1999) is used for evaluating the strategies. The first staging strategy is *stage-streaming*, which is similar to the *pipelining* strategy presented in (Ghandeharizadeh et al., 1995). Videos retrieved from tertiary storage are streamed to the user at the same time as they are written to the secondary storage. The advantages of using stage-streaming is lower initial start-up delay observed by the user and reduced tertiary bandwidth requirements. The reason for this is that for the fully staged case, all tertiary drives should be used in parallel, while the stage-streaming strategy performs best when independent drives are used for the delivery. The seconds strategy is *stage-streaming with trail-deletion.* The video data is deleted from secondary storage as soon as it has been streamed

to the user. The authors shows that this reduces the secondary storage requirements substantially with some cost on increased tertiary bandwidth due to no reuse of video data from the secondary storage.

In (Chan and Tobagi, 1999; Chan and Tobagi, 2003), LRU is used as the replacement policy for the disk cache. Several other replacement strategies have been proposed. (Ghandeharizadeh and Shahabi, 1994) propose the *PIRATE* replacement policy for management of videos stored on disk. The goal is to minimize the average response time when a user requests a video. This is ensured by having sufficient data from the first part of the videos on disk.

Lau and Lui (1996) evaluate two replacement strategies for a disk cache. Both strategies use the access probability of the videos. The first strategy, *Least Frequently Used (LFU)*, replaces the last page of the video sequence having the least access probability. The second strategy, *Longest Expected Access Time (LEAT)*, uses the distribution of the requests for the individual videos to compute the *expected access time* of the pages. The page with the largest expected access time is selected as victim. The two replacement strategies are evaluated and shown to perform better than using a LRU strategy.

Cha, Lee, Oh and Ha (2001) evaluates a caching strategy that includes multiple criteria for selecting which pages to replace. The videos are partitioned into three groups. First, pages belonging to videos that are currently in use are not replaced. Second, completely cached videos are not replaced. Third, pages that belong to videos that are only partially cached, are considered for replacement. For selecting which block to replace first, the *cache distance* is computed as the time it will take to play back the video. The last block of the video with the largest *cache distance* is selected as victim. If no victim is found, one of the completely cached videos are selected as victim by using a LRU strategy.

4.5.5 Evaluation of Tertiary Storage for Video Servers

Chervenak (1994) evaluates the performance and cost of using secondary storage (magnetic disks) and tertiary storage (tape and optical disks) in a movie-on-demand service. A video server based on disk farms with videos stored on a single disk or striped across multiple disks are simulated and compared against a video server where most of the videos are stored on tertiary storage. The tertiary storage system consists of a library containing tape drives or a jukebox containing optical disk drives. The load used in the simulations is requests for full-length movies, where the access distribution to the movies is based on Zipf's Law. For the disk based configurations, replication of the videos are also studied. In the evaluation, the main criteria for evaluating the system is the number of video streams the system is able to deliver and the cost per stream.

The main conclusions from studying disk-based video servers is that a video server using striping of the videos outperforms a video server where each movie is stored on a single disk (Chervenak, Patterson and Katz, 1995). When evaluat-

ing the use of tertiary storage for delivering movies, the main conclusion is that current tertiary technologies are not able to deliver the required number of video streams for the movie-on-demand service that is considered in the study. Further, due to the low number of streams, the cost per stream becomes very high compared to using hard disks for delivering the videos. Based on this, the author concludes that with the load and access distribution used in the study, it is better to use magnetic disks for storing all the videos instead of using tertiary storage for part of the videos. The author also shows that with a more skewed access distribution than the standard Zipf distribution, it becomes feasible to use tertiary storage for a large parts of the videos (Chervenak, 1994).

In this thesis, we evaluate use of magnetic tape and optical disks for storing video. Unlike the comparison done by (Chervenak, 1994), we use both cost per stream and storage cost given throughput or response time requirements when comparing the different storage technologies. Our focus is more on a general video archive than a movie-on-demand service. The cost of storing the video data may be more important than the cost of retrieving the video sequences. The workload will in general be more varied for video archives compared to a movie-on-demand service.

In (Chervenak, 1998), the evaluation of using tertiary storage is extended to include DVD. Unlike the evaluation of optical disks in (Chervenak, 1994), there is no use of a robot mechanism. Each DVD disk has its own DVD drive. This is the same strategy as used in (Shastri et al., 1997). Two different storage strategies are simulated, one-movie-per-disk and striping. Also replication of the videos are evaluated by using the access distribution to determine the number of copies of each video. The conclusions are the same as when evaluating magnetic disks (Chervenak et al., 1995). Striping out-performs having a single movie on each disk. For the one-movie-per-disk strategy, replication based on the access distribution improves the performance, while for striping replication has little effect.

4.5.6 Video Servers using Tertiary Storage

In this section we presents some examples of video servers that use tertiary storage as their main video storage. One of the first implemented video servers using tertiary storage was the The Berkeley Distributed Video-On-Demand system (Federighi and Rowe, 1994; Brubeck and Rowe, 1996). It consists of archive servers that have the original copy of the videos and video file servers that are located closer to the clients. The archive servers utilize a hierarchical storage system for storing the videos. The video file servers is used for caching videos and streaming the video to the users. The users connect to the closest video file server. If this contains a copy of the video, playback can start immediately. Otherwise, the video file server will request a copy of it from one of the archive servers.

Shastri et al. (1997) presents a design and performance analysis of a video-on-demand server that uses RAM, magnetic disks, and DVD disks for storing

the videos. The videos are allocated to one of the three storage types based on access probability. There is no use of caching. Each DVD is stored permanently in a DVD drive, thus there are as many drives as there are disks. The video is streamed directly from the DVD drive to the user. A similar architecture for a near video-on-demand server based on using CD-ROM drives as the main storage is proposed in (Tsao et al., 1997). The CD-ROM media are fixed in the drives and the video is delivered directly from the drives without staging it on hard disks. The focus of their research is on data placement strategies for the video on the drives in order to increase the number of clients that can be server from one drive. The proposed data layout strategy do only support CBR video. In (Tsao, 2001) the data placement strategy is extended to also support VBR video.

Cha, Lee and Oh (2002) study a video server that is based on a magneto-optical disk jukebox as the main video store. A disk cache is used for storing the most popular videos and part of currently delivered videos. All video delivery is performed from the disk cache. When delivering a video stored on tertiary storage, the first part is read from the magneto-optical disk into the disk cache before streaming is started. During the streaming of the video from the disk, at periodic interval more of the video is read from the tertiary storage system and written to the cache. For doing the reading from tertiary storage, a periodic scheduler is used. The disk cache is page based. The replacement strategy for the disk cache uses the *cache distance* as criterion for replacing a page the. The cache distance is defined as the amount of time until the page possibly will be reused by any stream (Cha et al., 2001). Only pages belonging to videos which are currently not is use are possible victims. The video server has been implemented, and (Cha et al., 2002) contains an evaluation of the performance of the video server, the scheduling strategy and the cache replacement strategy.

Chapter 5

Use of Tertiary Storage in Digital Video Archives

Avoid Quantitative Experiments:

- *If you've got good intuition, who needs experiments?*
- *Why give grist for critics' mill?*
- *Takes too long to measure*

David Patterson
in *How to Have a Bad Career in Research/Academia*
(Patterson, 1997)

The main focus of research in the area of storage and delivery of digital video has been on supporting video-on-demand like applications. These applications are characterized by a large number of users accessing a limited number of videos. The videos are usually played back with little user interaction. The main agenda of this research has been to support as many users as possible at a smallest possible cost per delivered video.

In digital video archives the research agenda is different. The most important goal is to safely store and make available huge amounts of digital video at a reasonable storage cost. A typical owner of a video archive can be a broadcasting corporation storing their own production of video footage. A video archive like this will contain much more video data than a video-on-demand service, typically tens of thousands hours of video. The number of users may be limited compared to a video-on-demand service. A typical user may be a journalist or a researcher seeking for video sequences related to some topic of interest. The archive may also be available to a wider audience, for instance by making it accessible on the Internet. The usage pattern will likely be very different from the typical video-on-demand user. The users of the video archive will be more active, both in the use of VCR operations, and in the number of requests issued

to the video archive (Kozuch, Wolf and Wolfe, 2000). As opposed to the two hours of mostly uninterrupted video playback requested by a typical video-on-demand user, a user of a video archive will typically query a meta database after interesting topics, e.g., all news stories related to the Nobel Peace Price, and then request the archive to deliver the corresponding video sequences (Hjelsvold et al., 1996). This request can be for multiple video sequences, where each of the requested video sequences typically will be much shorter than a full-length movie. For many video archives, most of these video sequences will be very seldomly accessed. The access distribution for the video sequences is likely to be very skewed. There might be a set of video sequences that are frequently accessed, but the majority of the video sequences may be more or less randomly accessed.

Storing high quality digital video is storage space intensive. Thus, the cost of storing the video will be important. Due to the large amounts of video that typically will be stored in a digital video archive, keeping all the videos online on magnetic disks will be too expensive for many applications. To reduce the cost of the archive, alternative less costly storage media has to be considered. Tertiary storage media is the main alternative. By using tertiary storage, the cost can be reduced to a fraction of the cost of magnetic disks. Tertiary storage media also have a higher storage density, which reduces the physical space necessary for the video archive. Further, when the tertiary media are not in use, they do not consume electric power or produce heat. Unfortunately, tertiary storage has one major drawback compared to magnetic disks. The access time for data stored on the medium can be several magnitudes higher. For an interactively used video archive, this might be a major problem.

Having decided to use tertiary storage in order to reduce the cost of the video archive, the challenge is to ensure optimal use of the storage system in order to increase the performance, while still offer an acceptable service to the users of the archive. As presented in Section 4.5, a typical tertiary storage system consists of library units containing multiple storage media, a robot and a limited number of tertiary drives. When a user requests a video sequence stored on a medium that is not mounted in a drive, the user has to wait until a drive becomes available, the current medium is unloaded from the drive, and the requested medium is transported to the drive and mounted in the drive. This operation may take tens of seconds. If magnetic tape is chosen as the storage technology, the time needed to locate the video sequence on the tape is typically in the range from 10 to 100 seconds. To improve the performance of a tertiary storage system, one can improve the efficiency of bringing storage media to and from the drives, improve the efficiency for locating the requested video data on the medium, or improve the transfer of the video data from the tertiary medium to the user.

In this thesis, we focus on two main issues related to the performance of a tertiary storage system used for storing and retrieving video data. The first issue is how the *access time* for video sequences stored on magnetic tape can be reduced. The second issue is the use of different *tertiary storage technologies* in a digital video

archive. Both random access (DVD) and sequential (magnetic tape) storage technologies are studied. In the study, we evaluate the *performance* and *cost* of the different technologies and investigate how *access distributions* and *allocation strategies* influence on the performance. In the last part of the study, we evaluate the effect of using *magnetic disks for caching* the most frequently used video sequences.

In the remainder of this chapter, we present the architecture for the video archive system and the tertiary storage technologies that are studied, and give an overview of the problems we address in this thesis.

5.1 Architecture for a Digital Video Archive System

Managing large amounts of digital video requires a storage system that is capable of storing the huge amounts of video data and that has a sufficiently high transfer capacity to deliver the requested video sequences. In this section, we present the architecture for the video archive system studied in this thesis, as shown in Figure 5.1. The architecture of the video archive system builds on the architecture of the Elvira II video archive server presented in Appendix A. A more detailed description of the video archive system can be found in Chapter 9 where we present a simulator for the video archive system.

The video archive system consists of two main parts, a video cache and a tertiary storage system. The video cache consists of magnetic disks. It is responsible for storing the most frequently accessed video sequences. The video cache should be under control of a video server management system responsible for responding to requests (VCR commands) issued by the user and for streaming the video to the user. The tertiary storage system consists of a set of library units. Each library unit contains a number of media drives, a robot mechanism with one or a few robot arms, and a media store (shelf) capable of storing a given number of storage media. The robots execute requests for transporting media between the media store and the drives.

The requested videos are streamed from the video cache to the user. No videos are delivered directly from the tertiary storage system to the user. The main reason for this design decision is to be able to optimize the use of the tertiary media drives. The tertiary media drives typically have a much higher transfer rate than the delivery rate of the video. Delivering the video from the tertiary media drive directly to the user would occupy the drive for a longer time than necessary (Ghandeharizadeh et al., 1995). The drawback of delivering all videos from the video cache is that videos retrieved from tertiary storage first have to be written to the video cache, and then read from the cache and streamed to the user. This requires extra disk bandwidth since the video has to be both written to and read from disk. This also leads to extra copying of video data on the internal memory and I/O buses. For the user, this extra delay should not be noticeable since the video cache can start delivering the video as soon as the first block con-

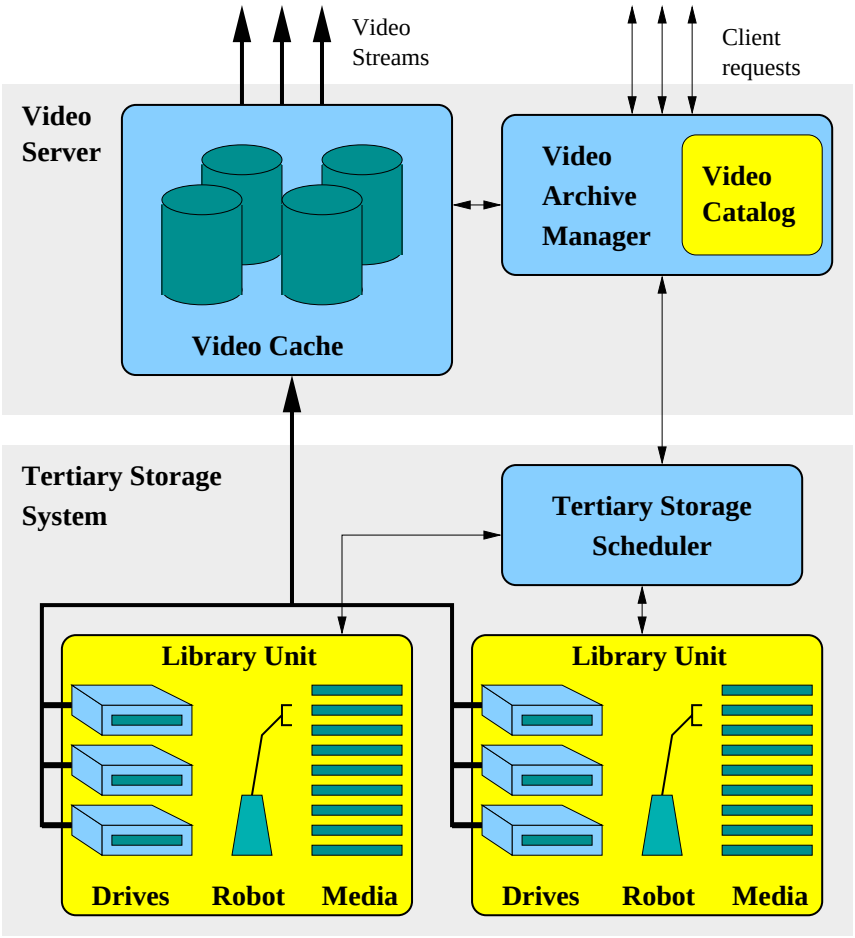


Figure 5.1 Architecture of a video archive system consisting of a disk based video cache and a tertiary storage system.

taining video data is available in main memory or written to disks. The video content in the video cache is replaced using an LRU strategy.

The users of the video archive issue requests for one or several video sequences. If the requested video sequence is available in the video cache, it is immediately ready for delivery. If not, the user has to wait for the video to be fetched from tertiary storage. A scheduler is responsible for optimizing the use of the tertiary storage system. This scheduler manages a request queue for each of the storage media in the tertiary storage system that has pending requests. As soon as the robot has loaded the medium into a drive, the scheduler starts executing the requests. For DVDs the seek time is small compared to the transfer time of a video request, and the requests are executed in the order they arrive. For tapes, the seek time is much higher. To optimize the usage of the tape drives,

the requests are rescheduled in order to reduce the total time used for seeking. For optimizing the use of tape drives, a new scheduling algorithm is developed as part of this work. As soon as a drive starts to deliver the video to the video cache, the user is notified that the video is ready for playback.

The focus of this research is on using tertiary storage in a video archive system. Still, the video cache will be an important part of such a video archive. As presented in Chapter 4, much research has been performed on technologies and strategies for making highly efficient disk based storage systems for digital video. In most of the work we assume that the video cache can handle both the video delivery to the users and the writing of video data transferred from the tertiary storage system. In Section 14.5, we present a more detailed model of a disk-based video cache and discuss the resources needed for delivering the videos to users.

5.2 Tertiary Storage Technologies

The tertiary storage media we use in our experiments are magnetic tape and recordable DVD. Compared to magnetic disks, the main advantages of magnetic tape and DVD are *cost* and *storage density*. For magnetic disk, the storage cost is about 0.004 dollar per MB[1] (2004), while for magnetic tape and recordable DVD the cost of the medium is about 0.0005 dollar per MB[1] (2004). A high-end disk drive stores about 100–150 GB (2004). A magnetic tape stores the same amount of data at a fraction of the price and storage volume. For video archives that require vast amounts of data storage, these two factors can be of sufficient importance to make tertiary storage become the best solution for the storage system.

Magnetic tape is today the most used storage medium in tertiary storage systems. As discussed in Section 3.4, several technologies exist, differentiating on how the data is stored on the tape. Common for all tapes is the sequential access to the data, leading to rather long seek times for accessing the data. Depending on the type of tape, the average seek time varies from about ten seconds up to more than a minute. The transfer rate of high-end tape drives is comparable to magnetic disks.

When using magnetic tape in a video archive, the long seek times of the tapes will be very costly with respect to both throughput and response time. In this thesis, we study how to optimize the performance when using magnetic tape for storing video sequences (and general data objects). The goal of the first part of this study is to minimize the average access time for video sequences stored on magnetic tape. The work is performed by using the Tandberg MLR1 tape drive, which is based on *serpentine* tape technology. Our contributions from this work are a detailed access-time model for serpentine tape drives and a new scheduling algorithm for optimization of concurrent accesses to the tape.

[1]The corresponding numbers in 1998 were 0.05 dollar per MB for magnetic disk and 0.005 dollar per MB for tape and recordable DVD.

The other main storage technology used in tertiary libraries is optical disks. The main advantage of optical disks compared to tape, is the much better performance of random accesses, leading to seek times of a few hundred milliseconds. With an exception for CD-ROM, the main drawback of optical disks compared to magnetic tape, has been the higher storage cost. With the introduction of recordable DVDs, the cost of using optical disks for mass storage in tertiary libraries has been reduced. As shown in Section 3.3, DVDs are available in several standards and storage sizes supporting different storage requirements.

5.3 Research Topics

In this section, we give an overview of the main problems we investigate in this thesis. All of these are related to use of tertiary storage in systems storing digital video. The issues we address are use of serpentine tape for storage and retrieval of multimedia data, evaluation of tertiary storage technologies for storing and retrieval of digital video, the effect access distributions and allocation strategies have on the performance of a video archive, and the effect of using a disk cache for caching the most frequently used videos. Each of these problems are introduced in this section and studied further in the following chapters.

5.3.1 Retrieval of Multimedia Data from Serpentine Tape

Retrieving data items like images and video sequences from magnetic tape is time consuming due to the sequential access of the tape, and can be time critical particularly in interactive applications where the user has to wait for the requested data to be retrieved. The problem we investigate is how to efficiently execute requests for data objects stored on serpentine tape. The problem can be divided into two parts as illustrated in Figure 5.2. The first problem that must be solved is to develop a model of the time to retrieve data objects stored on the tape. This access-time model can then be used for scheduling concurrent accesses to the tape.

Model for Access Times for Serpentine Tape

The data layout used by serpentine tape drives results in a complex relationship between the logical address used by applications and the physical position on the tape. This makes it challenging to develop a model for access times of requests for data objects on a serpentine tape. As illustrated in the upper part of Figure 5.2, the first problem that must be solved is to be able to map the logical address of objects to the physical position on the tape. The second part of this problem is to determine cost functions for the seek time for seeks between all positions on the tape and for the time it takes to read the requested object. The access-time model is presented in Chapter 6.

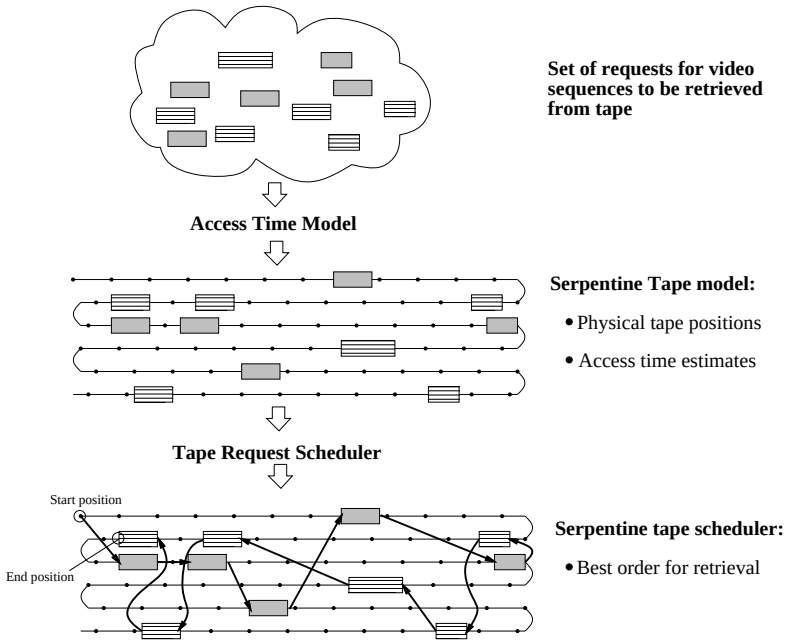


Figure 5.2 Overview of the two main issues needed to be solved in order to efficiently retrieve video sequences from a serpentine tape.

Scheduling of Concurrent Accesses to Serpentine Tape

When multiple users are accessing a tape simultaneously, or when one user wants to retrieve multiple objects from the tape, careful scheduling of the requests can reduce the total time to retrieve the requested video sequences. We study how retrieval of concurrent requests for data objects stored on a single serpentine tape can be optimized by the use of a scheduler. An example of the order for retrieving multiple data objects is given in the lower part of Figure 5.2. The goal is to reduce the initial latency, the average access time and total time for retrieving the requested objects. The proposed scheduling strategy is evaluated for retrieval of images and video sequences stored on tape. The new scheduler for serpentine tape is presented in Chapter 7, while Chapter 8 contains the evaluation of using serpentine tape for storage and retrieval of multimedia data.

5.3.2 Tertiary Storage Technologies for Storing and Retrieving Digital Video

Different tertiary storage technologies have different properties and performance characteristics. In this part of the study, we evaluate three different tertiary storage technologies for use in library units used for storing and retrieving video

data. The goal is to investigate properties of the different storage technologies, to study the performance that can be achieved by using the different storage technologies, and to determine the factors that limit the performance under different load scenarios. The technologies are compared using throughput, response time, and cost as the main criteria. In addition to evaluate the performance that can be achieved using the three tertiary storage technologies, we evaluate how improvement of the main operations performed by the tertiary storage devices influence the performance.

The tertiary storage technologies used in the evaluation are Tandberg MLR1, which is a traditional tape drive based on serpentine tape technology, IBM 3570 Magstar MP, which is a serpentine tape drive optimized for random I/O requests, and a 2X DVD drive. To perform the study, a simulator for a video archive system based on the architecture presented in this chapter is designed and implemented. The design of the simulator is presented in Chapter 9, while the results from the evaluation of the tertiary storage technologies are presented in Chapter 10 and 11.

5.3.3 Access Distributions and Allocation Strategies

In a video archive, the video sequences are likely to have different access probabilities. For instance, in a news archive, the news items from the last week are likely to be much more accessed than the news items from the same week a year ago. As a consequence, the tertiary storage system will contain both frequently accessed video sequences and video sequences that are hardly ever accessed. We study two issues related to having different access probabilities for the video sequences:

1. **Access distributions.** In a video archive where some video sequences are more more popular than other, the storage media containing these video sequences will be used much more frequently than storage media containing only less popular video sequences. We evaluate the effect different access distributions for the video sequences have on the performance of a tertiary storage system (Chapter 12).

2. **Allocation strategies.** When retrieving video sequences from tertiary storage, the two main costs are to move the storage medium from the shelf to the drive and to perform the seek operation. We study if knowledge about access distribution can be used to cleverly allocate the most popular video sequences to storage media and library units in such a way that the number of media exchanges or the average seek time can be reduced (Chapter 13).

Both studies are performed using magnetic tape and DVD storage and for different configurations and user loads. The video archive simulator is used for performing the experiments.

5.3.4 Caching in a Video Archive based on Tertiary Storage

A storage system for video based on tertiary storage only will have relatively high response times and a limited transfer rate. In order to reduce the average response time and to be able to serve more requests for video sequences without increasing the cost substantially, one solution is to cache the most popular video sequences using a disk cache.

We study the effects on the performance and cost of a video archive of using a disk cache. Introducing a disk cache may increase the cost of the storage system, but it may also reduce the need for costly tertiary bandwidth. We study how the optimal size of the disk cache can be found given specified performance criteria and evaluate the relationship between the size of the disk cache and the number of tertiary drives. The effect of different access distributions and allocation strategies are also studied in the context of using a disk cache. The results are presented in Chapter 14.

Chapter 6

Access-Time Model for Serpentine Tape Drives

First, we must understand the tertiary devices we are using... Second, we must determine how to maximize the performance of individual operations.

Michael J. Carey, Laura Haas and Miron Livny
in *Tapes Hold Data, Too: Challenges of Tuples on Tertiary Store*
ACM SIGMOD 1993 (Carey, Haas and Livny, 1993)

In modern computing, magnetic tape has mainly been used by applications that access data on the tape sequentially. Today however, there is a growing interest in building computer applications that store vast amounts of digital data, while still wanting relatively fast random access to the stored data. Due to its high storage density and low cost, magnetic tape can be a relevant storage technology to consider for such systems. The main limitation of magnetic tape is the very long access time, which can easily reach several minutes in unfavorable situations.

Hillyer and Silberschatz (1996b) have shown that the access times of serpentine tape drives can often be substantially reduced by use of a scheduler, which reorganizes the retrieval order of the tape requests. In order to do intelligent reorganizing, such a scheduler must be able to compute fairly accurate estimates of access times. Because of the complex data layout on a serpentine tape, this is not a trivial task. Hillyer and Silberschatz have solved this problem for the Quantum DLT 4000 drive (Hillyer and Silberschatz, 1996a) and the IBM 3570 Magstar drive (Hillyer and Silberschatz, 1998), by use of two complex, tailor-made models, which require tremendous amounts of analysis for each individual tape cartridge. A much simpler model using a piece-wise linear regression model to estimate seek times is suggested by (Johnson and Miller, 1998b).

In this chapter, we present a general access-time model for a serpentine tape drive. This model consists of three main parts. First, we establish a way to estimate the physical position on a tape, given a logical block address. Second, we partition the seek space into eight disjoint seek classes, with regard to the work

that is incurred on the tape drive. For each seek class, we provide analytic cost functions to compute the seek time. Third, we provide a way to compute an estimate for the transfer time of a given data request. The access-time model is designed to balance the need of accuracy against the need of fast characterization of tapes. We provide several algorithms that achieve such characterization at a fairly low cost. The accuracy of the proposed access-time model is validated by measurements on the Tandberg MLR1 and the Quantum DLT 2000 tape drives. In the next chapter we show that the achieved accuracy is sufficient to facilitate efficient scheduling of random retrievals from tape. All experiments presented in this and the following chapters were performed using two Tandberg MLR1 tape drives and one Quantum DLT 2000 tape drive connected to a Fast SCSI-2 bus on a SparcStation 20 workstation.

The content of this chapter has been published in the paper "Low-cost access time model for serpentine tape drives" presented at the 16th IEEE Symposium on Mass Storage Systems/7th NASA Goddard Conference on Mass Storage Systems and Technologies, held in San Diego, California, USA, March 1999 (Sandstå and Midtstraum, 1999b).

6.1 Performance Characteristics of a Serpentine Tape Drive

The main technologies for digital tape were introduced in Section 3.4. Contrary to helical scan and parallel tape drives, serpentine drives do not provide a direct relationship between logical block addresses and physical positions on the tape, making it much harder to estimate the access times. To gain understanding of the behavior of a serpentine tape drive, we have used the Tandberg MLR1 tape drive (Tandberg Data, 1996). This drive uses serpentine data layout and is based on the 13 GB QIC standard (QIC Development Standard, 1994), making it possible to store 13 GB per tape (without compression). The drive can deliver (read/write) a maximum sustained data rate of 1.5 MB/s to/from the host computer. Each tape has 72 logical tracks, 36 in the forward direction and 36 in the reverse direction.

To determine the characteristics for the Tandberg MLR1, we ran several experiments on the tape drive. First, the tapes were written with fixed length logical data blocks of 32 KB. The number of blocks on each tape varied from 398000 to 400100 blocks. The average write time for a block was 22 milliseconds. To write a full tape takes approximately 2.5 hours. By performing seeks on the tape, we found the access time for one block to vary between 1 and 126 seconds. For seeks starting on the beginning of the tape, the average seek time is 65 seconds. For seeks between two random positions on the tape, the average seek time is 45 seconds.

Figure 6.1 shows the seek and rewind times for the first four tracks of a tape. The x-axis contains the logical address of data blocks on the tape, and the y-axis

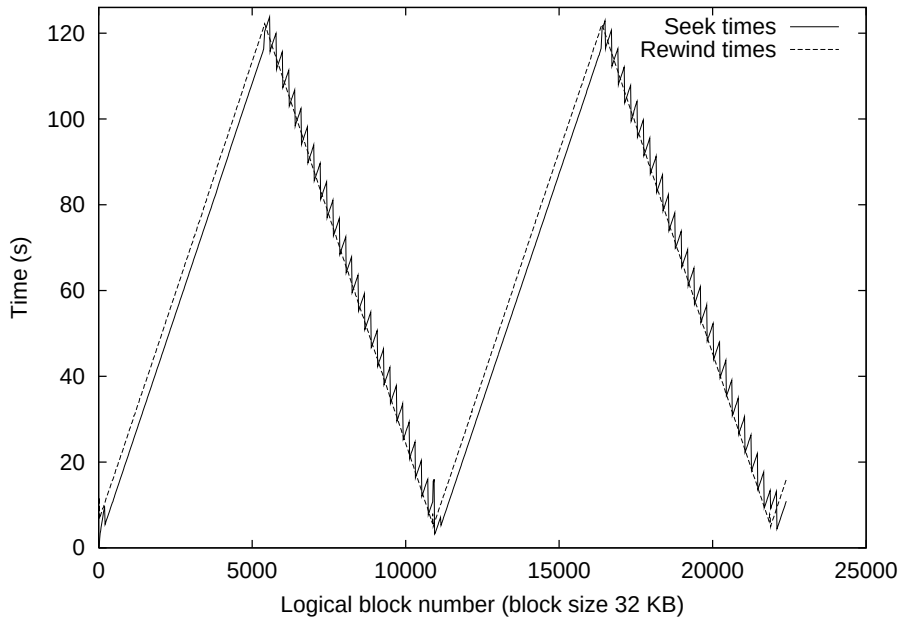


Figure 6.1 Seek and rewind times for the first tracks on a MLR1 tape.

gives the number of milliseconds it takes to seek from the start of the tape to each of the data blocks. For every sixteenth logical block address, we measured the time needed to seek from the beginning of the tape to the block, and the time to rewind back to the beginning of the tape. From the figure, we see that for *forward tracks*, the curves for seek and rewind times overlap and are both straight. For *reverse tracks* the curve for seek times has a sawtooth pattern, while the curve for rewind times is straight.

To explain the sawtooth pattern, we note that each 'tooth' is a straight line covering about 200 logical blocks. The reason for this pattern on the reverse tracks is that these tracks have to be read in the opposite direction of the forward tracks. When the tape drive tries to locate a position on a reverse track, starting from the beginning of the tape, it first has to seek past the sought block, and then start reading in the read direction until it has found the sought block. Figure 6.1 indicates that the tape drive uses a set of predetermined points to decide where to stop the seek in forward direction, and start seeking in the opposite direction. These points correspond to the first block in each sawtooth. Hillyer and Silberschatz (1996a) experienced similar sawtooth patterns for the Quantum DLT 4000 drive. They defined the points where the seek time has a large dip from one sawtooth to the next as the *key points* of the tape. Figure 6.2 shows the serpentine layout of the first tracks on a tape with the key points included. But opposite to what we found, they also experienced sawtooth patterns along the forward tracks. The reason is that the DLT 4000 uses one speed (seek speed) for locating the key point, and a slower speed (read speed) for locating the sought block between two key

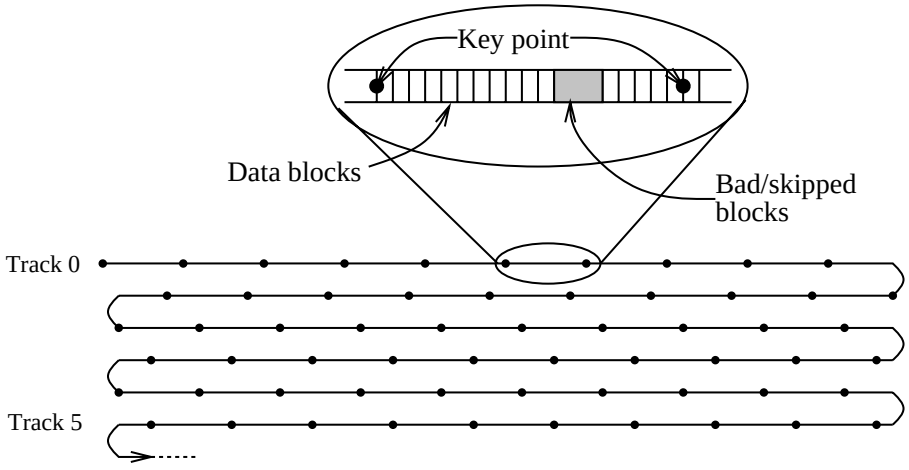


Figure 6.2 The serpentine layout of the first tracks on a tape with key points.

points. The Tandberg MLR1 drive uses the same speed for both seeking and reading. This suggests that there will be key points along the forward tracks too, and by plotting seek times for seek operations starting on a different position than the beginning of the tape, we find the sawtooth pattern on the forward tracks.

6.2 Access-Time Model

The *access time* is the time it takes from the point when a memory device starts execution of an operation, until the data is available to the entity requesting the data, i.e., the sum of the *seek time* and the *transfer time* for the data. For tape operations, there is not much that can be done with the transfer time. As soon as the drive starts reading data from the tape, it will continue reading with a constant transfer rate until it reaches the end of the requested data region. Thus, the transfer time will be proportional to the size of the requested data. Contrary, seek time is essentially wasted time, and should be reduced as much as possible. As a consequence, the main focus of our access-time model is on how to model seek times, since this part of the access time is non-trivial to model, and provides opportunities for substantial optimization of the total access time.

In the presentation of the model, we assume that the tape contains fixed sized blocks. Further, we assume the tape is mounted in the tape drive and positioned at logical block address L_0 when an I/O request arrives. Such an I/O request consists of the logical block address of the first block requested, L_1, and the number of consecutive blocks to be read, N. The purpose of the access-time model is to estimate the time the tape drive will use to re-position the tape from the current logical position L_0, to the logical start position L_1, plus the transfer time for the N

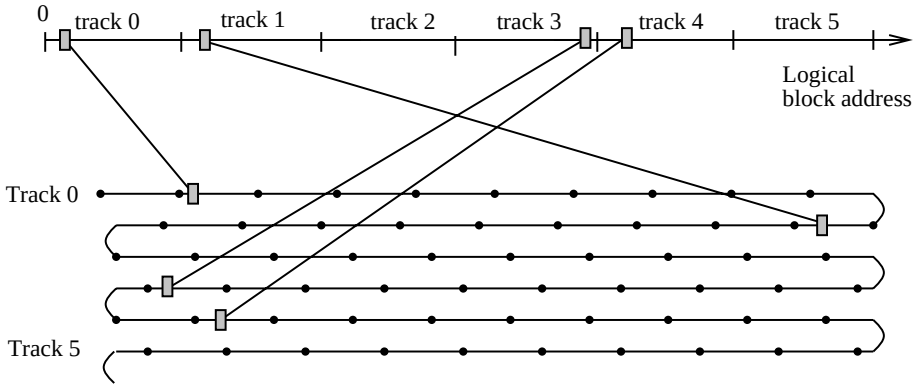


Figure 6.3 Mapping from logical block addresses to physical positions on the tape. The bullets along the physical tape are the key points of the tape.

blocks:

$$accessTime(L_0, L_1, N) = seekTime(L_0, L_1) + transferTime(L_1, N)$$

Hillyer and Silberschatz (1996a) have proposed an access-time model for the Quantum DLT 4000 drive, which relies on locating the address of each key point on the entire tape. This gives a very accurate model, but requires about twelve hours of processing for each tape. To avoid such problems, we propose a model, which does not depend on knowledge of the exact location of each key point. Our model is based on the following strategy:

1. We estimate the physical position of each logical block on the tape, by using the logical address of the first block of each track.
2. We estimate the seek time between two physical tape positions by partitioning the possible seeks into disjunct seek classes. For each seek class, we provide a cost function to estimate the cost of the seeks in the class.
3. We estimate the transfer time as the time it takes for the drive to transport the read area of the tape past the drive's read head, plus the time it takes to make the necessary track changes.

6.2.1 Estimating Physical Tape Positions for Logical Addresses

Applications access data stored on tapes by using logical block addresses. To be able to establish a cost model for seek and transfer times, we have to find the

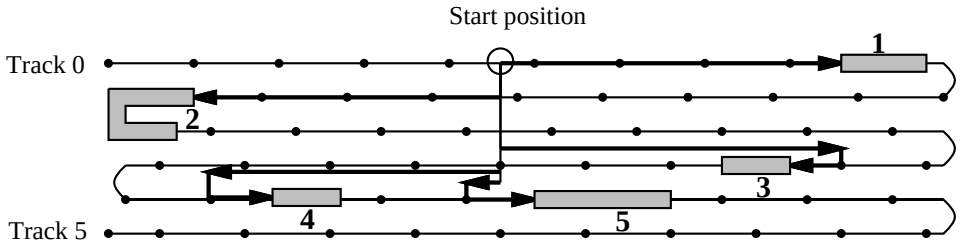


Figure 6.4 Example of a serpentine tape with the key points marked on the tracks, and possible seek patterns for five data requests.

physical tape positions for the logical block addresses. A physical position on a serpentine tape is given by the *track number* and the *physical distance* from the beginning of the tape, $(trackno, tapepos)$. Figure 6.3 shows some examples of how the logical blocks on the first tracks on a tape are mapped to the physical tape.

To establish the mapping from logical block addresses (L) to physical tape positions (p), we use the logical block address of the first block on each track. In this discussion, we assume we have these track addresses available. We will later explain how these addresses can be found. Given these track addresses, it is easy to make a function $track(L)$ which returns the *track number* for any given logical address. Further, assuming the track addresses are stored in the array $trStart[]$, we find the physical distance from the start of the tape as:

$$tapepos(L) = \begin{cases} \frac{L - trStart[track(L)]}{trStart[track(L)+1] - trStart[track(L)]} & \text{if } track(L) \text{ is even} \\ 1 - \frac{L - trStart[track(L)]}{trStart[track(L)+1] - trStart[track(L)]} & \text{if } track(L) \text{ is odd} \end{cases} \tag{6.1}$$

This function returns the tape position as a number between 0 and 1. The reason for dividing by the length of the track is, as we will show later, that the length of the tracks vary within a tape.

6.2.2 Estimating Seek Times

Figure 6.4 shows five examples of possible seeks. When a seek starts, the tape drive is positioned at a forward track. For seek number 1, we have to change neither track nor winding direction. Seek number 2 is an example of a seek where we have to change both track and winding direction. For seek number 3, 4, and 5, the tape drive has to seek beyond the start of the requested data area to locate the closest key point. This results in a longer winding distance than the physical distance.

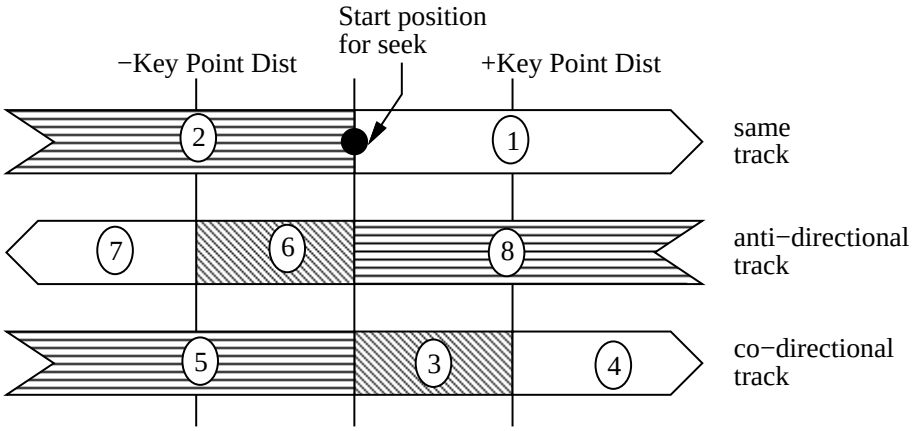


Figure 6.5 Model used to partition seeks into eight seek classes. It is important to note that this figure is seen from the position the tape drive's read/write head has on the tape when the seek starts.

Given the current physical position of the tape drive and the physical position of the start of the requested data item, the model must estimate the time needed by the drive to wind to this position. There are four variables which influence the seek time:

1. the physical distance between the two tape positions,
2. time to change track,
3. time to change winding direction,
4. time to locate the closest preceding key point of the requested data block.

In the remainder of this subsection, we establish an analytical model for how these four cost variables influence the seek time. Every possible seek will be partitioned into one of eight disjunct seek classes based on how the cost variables influences that particular seek. Figure 6.5 shows how the seeks are partitioned into one of the seek classes based on the relative location (seek distance, track changes and winding direction) of the sought block compared to the physical start position of the seek. Table 6.1 contains an overview of which cost variables influence each of the seek classes.

Physical Tape Distance

As seen in the previous section, the seek time between two logical block addresses is dominated by the time to wind the tape from the physical start position to the

Seek class	Dist-ance	Track change	Winding direction	Locating key point	
				Sometimes	Always
1	X				
2	X		X		X
3	X	X		X	
4	X	X			
5	X	X	X		X
6	X	X	X	X	
7	X	X	X		
8	X	X			X

Table 6.1 The different cost variables which influences each of the eight seek classes of the cost model.

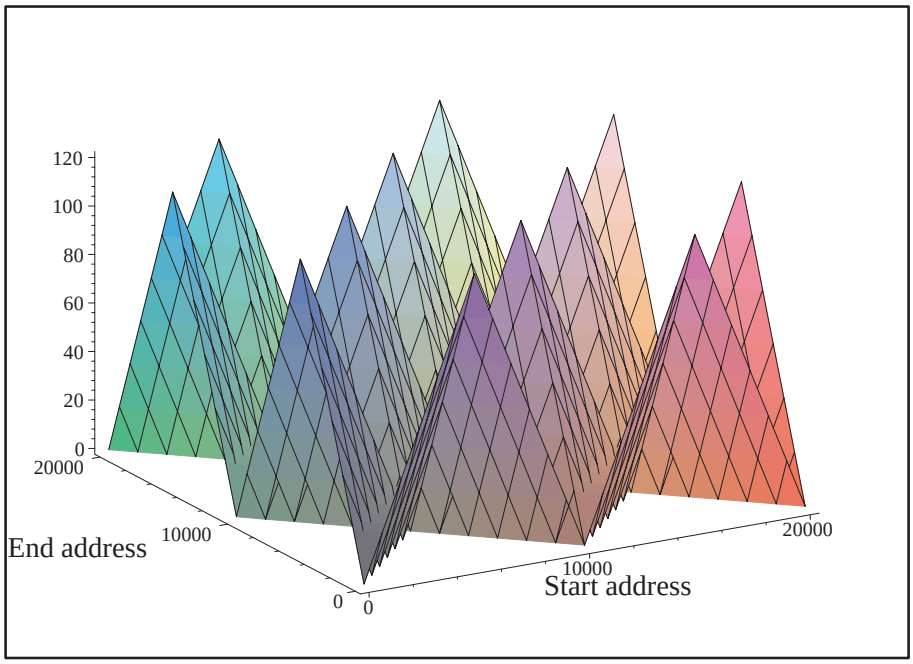


Figure 6.6 Plot of seek times (in seconds) due to tape winding between positions on the first four tracks on a tape. The seek times are computed using Equation 6.2.

Case	Description	Cost
a.	Seeks forward on the same or a co-directional track (e.g., seek 1 in Figure 6.4)	$0 \cdot t_{turn}$
b.	Changes to an anti-directional track (e.g., seek 2 and 3 in Figure 6.4)	$1 \cdot t_{turn}$
c.	Seeks backwards on the same or a co-directional track (e.g., seek 4 in Figure 6.4)	$2 \cdot t_{turn}$

Table 6.2 Overview of the number of times the drive has to change winding direction during a seek operation.

requested position on the tape. As can be seen in Figure 6.1, the time usage is mostly proportional to the physical distance. Thus, in the model we estimate the seek time due to tape winding between two physical tape positions as:

$$t_{seek}(p_{start}, p_{stop}) = t_{wind} \mid p_{stop} - p_{start} \mid \tag{6.2}$$

where the physical positions p_{start} and p_{stop} are found using Equation 6.1 and t_{wind} is the time the drive uses to wind the tape from the beginning of the tape to the end of the tape. Figure 6.6 contains a plot of how the seek time due to winding of the tape varies for seeks between logical addresses on the first tracks of a tape.

When the physical distance between the start position and the requested position is large, this function gives a good approximation of the total seek time. For shorter seek distances, the other cost variables have to be included in the model.

Change of Track and Winding Direction

To improve the model, we include the cost of track changes and change of winding direction. Each time the drive has to change from one track to another, we add the track change cost t_{tc}. There are two reasons for approximating this cost with the constant t_{tc}. First, the cost of a track change is mainly a result of having to reposition the tape head and adjust it to a new servo track, not from the physical distance the head has to be moved. Second, the drive changes between *logical* tracks which do not necessarily correspond to the physical movement of the drive's tape head.

Similarly, we add the cost t_{turn} every time the drive has to change winding direction. Assuming the drive just has finished reading a block (i.e., it is winding in one direction), when it receives a new seek command, the drive has to change winding direction zero, one or two times depending on the relative location of the requested block compared to the current physical tape location. An overview of

Seek class	Probability	Cost of locating key point
1, 4, 7	0	0
2, 5, 8	1	$l_{key}t_{wind}$
3	$1 - \frac{d}{l_{key}}$	$(l_{key} - d)t_{wind} + 2t_{turn}$
6	$1 - \frac{d}{l_{key}}$	$(l_{key} - d)t_{wind}$

Table 6.3 Cost of locating the key point for the different seek classes.

when the drive has to change winding direction and the associated cost is given in Table 6.2.

This far we have included in the seek time the costs that would incur if the drive was able to seek directly from one position to another without having to go through a key point. Unfortunately, in some cases, locating the closest key point incurs extra seek time.

Locating Key Points

Each time the drive has to seek beyond the start of the requested data area to locate the key point, as in seek number 3, 4, and 5 in Figure 6.4, this results in a longer winding distance than the physical distance between the start position and the requested block. This extra seek distance depends on the distance between the requested block and the closest key point. The most accurate method for estimating this distance will be to use the key points as done by Hillyer and Silberschatz (1996a). Unfortunately, the required instrumentation is too costly for most applications. In our approach, we include the *average* cost of locating the closest key point. Since we do not locate the key points, an important thing to note is that there will be seeks where we do not know in advance whether the seek to the key point will incur extra seek distance or not. Fortunately, this will only occur for seeks which are shorter than the distance between two key points. An example is seek 5 in Figure 6.4. If the closest key point for seek 5 is between the start position and block 5, the drive can wind directly to the block, if not it has to rewind until it gets to the key point, then change winding direction and read until it has reached block 5.

As mentioned earlier, all seeks can be partitioned into eight seek classes as shown in Figure 6.5. Table 6.3 shows how the seek times in each seek class will be influenced by locating the key point. For seeks in seek class 2, 5, and 8 (see for example seek 3 and 4 in Figure 6.4), the average extra cost for locating the key point will be the cost of seeking the length of the distance between two key points, l_{key} (half the distance between two key points to locate the key point, and the same distance to get back to the requested data block). For seeks in seek class 3 and 6 (see for example seek 5 in Figure 6.4), the formula for the cost will be more complicated since there only is a certain probability that the seek time will

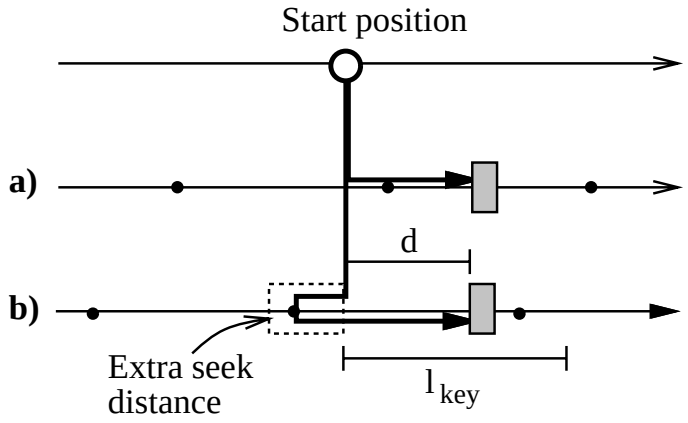


Figure 6.7 The two possible seek patterns for seeks in seek class 3. **a)** There is a key point between the start position and the sought block, and no extra seek distance is needed for locating the key point. **b)** There is no key point between the start position and the sought block, and the tape drive has to rewind to locate the key point. The extra seek distance needed to locate the key point is marked on the figure.

be influenced by having to locate the key point. This is illustrated in Figure 6.7 for seeks in seek class 3. If there exists a key point between the start position for the seek and the requested data block (case a) in the figure), no extra cost will occur. If there is no key point between the start position and the requested data block, the tape drive has to rewind to the closest key point preceding the block as shown in case b) in Figure 6.7. The situation is similar for seeks in seek class 6. The probability of having to seek extra distance to locate the preceding key point depends on the physical distance between the current position and the requested data block, $P(extra\ cost) = 1 - \frac{d}{l_{key}}$. The extra distance the drive will have to seek is $l_{key} - d$. For seeks in seek class 3, the drive will also have to change winding direction twice.

The complete cost functions for all seek classes are given in Table 6.4. These are found by adding the cost for each of the cost variables that influence each seek class (see Table 6.1). In each of the cost functions we have included a constant, $\hat{t}_i$, to account for extra delays due to for example startup delays of the mechanical operations in the drive.

6.2.3 Estimating Transfer Times

Estimating the transfer time of a tape access is much easier than estimating the seek time, because the drive reads the tape at a constant data rate. Only when the drive has to change track during the reading of the data segment (as in seek 2

Class	Seek time cost function
1	$t_{wind}d + \hat{t}_1$
2	$(d + l_{key})t_{wind} + 2t_{turn} + \hat{t}_2$
3	$\frac{d^2 - l_{key}d + l_{key}^2}{l_{key}} t_{wind} + 2(1 - \frac{d}{l_{key}})t_{turn} + t_{tc} + \hat{t}_3$
4	$t_{wind}d + t_{tc} + \hat{t}_4$
5	$(d + l_{key})t_{wind} + 2t_{turn} + t_{tc} + \hat{t}_5$
6	$\frac{d^2 - l_{key}d + l_{key}^2}{l_{key}} t_{wind} + t_{turn} + t_{tc} + \hat{t}_6$
7	$t_{wind}d + t_{turn} + t_{tc} + \hat{t}_7$
8	$(d + l_{key})t_{wind} + t_{tc} + \hat{t}_8$

Table 6.4 Cost functions for the eight seek classes in the model. In the formulas the seek distance is given as $d = \mid p_{start} - p_{stop} \mid$. l_{key} is the physical distance between two key points given as a fraction of the total tape length. t_{turn} and t_{tc} is the amount of time it takes to change winding direction and change tracks. t_{wind} is the total winding time for a track.

in Figure 6.4), the model has to include the cost of a track change in the transfer time. For a request for N blocks starting at logical block address L_1, the transfer time is given by:

$$transferTime(L_1, N) = N \frac{t_{wind}}{trStart[track(L_1) + 1] - trStart[track(L_1)]} + (track(L_1 + N) - track(L_1))\ t_{tc\_read}$$

It is worth noting, that the constant t_{tc_read} is different from the constant t_{tc} used in the seek time functions. The track change during a read operation always occurs on the end of a track, it always changes to the next track and the drive has to determine the start of the data area on the next track.

6.2.4 Instrumenting the Model to be Used with the Tandberg MLR1 Drive

To use the model for a given serpentine drive type, we have to determine values for the constants used by the model. The seek time functions given in Table 6.4 depend only on the physical seek distance. For all seek classes, except for class 3 and 6, the variable part of the functions is proportional to the physical seek distance between the start and end positions. Thus, for these classes the seek time function will be of the form $\alpha + \beta(\mid p_{start} - p_{stop} \mid)t_{wind}$. So instead of determining

Seek class	Cost function	Values for α	Values for β
1	$\alpha_1 + \beta_1(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_1 = 0.814$	$\beta_1 = 0.984$
2	$\alpha_2 + \beta_2(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_2 = 8.805$	$\beta_2 = 0.983$
3	$\alpha_3 + \beta_3(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_3 = 8.285$	$\beta_3 = -0.573$
4	$\alpha_4 + \beta_4(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_4 = 1.036$	$\beta_4 = 0.975$
5	$\alpha_5 + \beta_5(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_5 = 8.636$	$\beta_5 = 0.979$
6	$\alpha_6 + \beta_6(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_6 = 7.633$	$\beta_6 = 0.307$
7	$\alpha_7 + \beta_7(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_7 = 2.068$	$\beta_7 = 0.975$
8	$\alpha_8 + \beta_8(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_8 = 7.760$	$\beta_8 = 0.979$

Table 6.5 Cost functions for the eight seek classes, with corresponding constants determined for the Tandberg MLR1 drive. These functions return the estimated seek time for a given seek. t_{wind} is the total winding time for a track. For a Tandberg MLR1, this takes 120 seconds.

values for the constants t_{turn} and t_{tc}, which would be hard to get exact values for, we determine the constants α and β for each seek class. For seek class 3 and 6, the seek time is not a linear function of the physical seek distance. Still, since these functions are for very short seeks, we can approximate these with a linear function without much loss of accuracy. By doing this, the seek time functions in Table 6.4 can be written as shown in the second column of Table 6.5.

In order to use the model with the Tandberg MLR1 drive we have established values for the constants by practical use of the drive. The constants were found by performing 2000 seeks on three different tapes. The seek positions were selected such that the number of seeks of each seek class was approximately the same. We measured the seek time for each seek, and determined the constants for the seek time functions in each seek class by using linear regression. The resulting constants are given in the third and fourth column of Table 6.5.

To estimate transfer times for the MLR1 we have to determine the constants t_{wind} and t_{tc_read}. t_{wind} is the time the tape drive needs to wind from the start of the tape to the end of the tape. For the Tandberg MLR1, the manufacturer states that the maximum rewind time is 120 seconds. This is consistent with our experiences, as we have measured maximum rewind times between 119.9 and 121.2 seconds.

To estimate the time used to change from one track to the next during continuous reading, we measured the time used by the drive to read 32 MB data segments from the three tapes. By computing the difference in transfer time between those data segments which included a track change during the read operation, and those which did not, we found the average value for t_{tc_read} to be 2.9 seconds.

6.3 Characterizing Individual Tapes

In the previous section, we explained how to estimate the *physical position* of each *logical block address*. This mapping requires knowledge of the logical address of the first block on each track. In this section we present four strategies for estimating/finding these track addresses. It should be obvious that the better the estimate of the track addresses is, the more exact will the estimated access times be.

The first of the strategies is generic, and can be used for all MLR1 tapes. The three other strategies improve the accuracy of the estimated track addresses by characterizing each individual tape.

Average Tape-Length

The first strategy assumes that each tape has the same number of data blocks, and that the data blocks are evenly distributed on all the tracks. Unfortunately, the number of blocks per tape varies rather much. For the tapes we have used, the number of blocks written has been between 398082 and 400055 blocks per tape. The average number of blocks per tape has been 398637, giving an average value of 5537 blocks per track. We use this as the first approximation of the track addresses. Since it is based on an average tape, we call the strategy *Average Tape-Length*.

The problem with the average tape-length strategy is that the estimates for physical positions get worse as we get farther out on the tape. The reason is that we do not know the exact number of blocks per track, and the error in each track length is added as we increase the track number. To make a model without this drifting problem, we need to characterize each individual tape. The straightforward way to characterize a tape completely would be to perform a seek from the start of the tape to each block on the tape. Unfortunately, this is not feasible, since it would take more than a year to perform this for a single tape. Another way to improve the accuracy of the model is to find better estimates for the number of blocks per track on each tape.

Exact Tape-Length

A first approximation of the number of blocks per track can be found by dividing the exact number of blocks written to the tape by the number of tracks on the tape. This can only be done if the entire tape is filled up by fixed sized blocks. We call this strategy *Exact tape-length*.

To further improve the model, we can try to identify the address of the first block on each track. These addresses will vary from tape to tape, due to varying numbers of bad blocks and blocks skipped during writing of the tape. We have tested two different strategies for estimating the start address of each track. The

first strategy is based on the write times of the tape, while the second strategy finds the end of the tracks by performing read operations on the tape.

Write-Turn

If we have control of the writing of the tape, and the tape is written block by block, we can measure the writing time for each block. Writing a 32 KB block to the tape takes on average 22 milliseconds, but every time the tape reaches the end of a track, the tape drive has to stop the tape motion before it can start writing in the opposite direction. By studying the writing times, we have found this change of direction to take about three seconds for the Tandberg MLR1 drive. We use this to get a rather accurate estimate for the start address of each track. Since most tape drives use a write buffer, the addresses found during analysis of the write times have to be adjusted to compensate for this buffer. The reason is that the write times will stay at the average write time after we have reached the end of a track until the write buffer is full. We call this strategy *Write-Turn*.

Read-Turn

If the tape is already written by someone else, or by an application which does not let us have access to the write times for each data block, we can locate the end of the tracks by performing read operations on the tape. One way to do this is to position the tape head on a block close to the end of a track and then start reading contiguous blocks while measuring the read time of each block. As long as the drive reads blocks from the current track, the time for reading one block should be about 20 milliseconds. When the drive reaches the end of the track, it has to change read direction. This change of direction takes about five seconds, and is easily detectable by measuring the time to read each of the blocks. We can use this to detect the block address of the first block on a track.

To reduce the total time it takes to find the end of the tracks, we do this only for the 36 reverse tracks. This saves us from a complete wind/rewind of the tape and from the work of locating the end of the 36 forward tracks.

6.4 Validation of the Model

In this section we validate the model by comparing access times estimated by the model to measurements of access times on tape drives. We also compare the accuracy of the model that can be achieved using the four different strategies for characterizing the tapes presented in the previous section. The access times were obtained by measuring the time used from the moment that the computer sent a request for a 32 KB block to the tape drive, until the block was available in main memory. On the completion of one request, a new request was executed without any pause.

	Average error [*seconds*]			
Strategy	All	Tape 1	Tape 2	Tape 3
Average tape-length	10.0 s	11.5 s	11.1 s	7.49 s
Exact tape-length	6.21 s	12.2 s	4.50 s	1.92 s
Write-turn	1.69 s	1.74 s	1.85 s	1.49 s
Read-turn	1.71 s	1.74 s	1.86 s	1.54 s

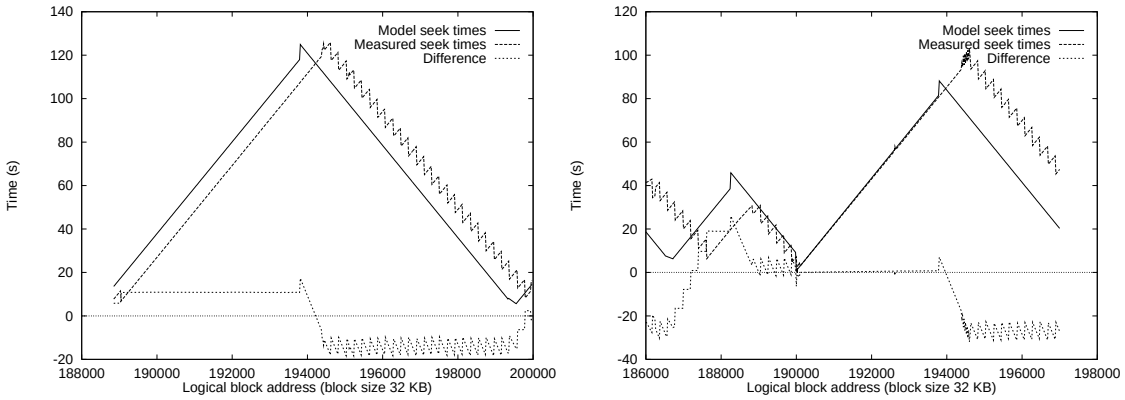
Table 6.6 Results from testing the model on a Tandberg MLR1 drive using the Average Tape-length, Exact tape-length, Write-Turn, and Read-Turn algorithms for instrumenting the model. The table contains the average difference between measured and estimated access times for 2000 random block accesses.

6.4.1 Validation using Tandberg MLR1

Three tapes, which were not used during instrumentation of the model constants, were used in the validation of the model. These were filled with 32 KB data blocks. During the writing of the tapes, we logged the write time for each block. From the log of write times, we got the exact number of blocks on each tape, and by analyzing the write times with the *Write-Turn* strategy we found the start address of each track. We also ran the *Read-Turn* algorithm on each of the tapes to find the start address of each forward track.

To compare access times estimated by the model with measured access times using the Tandberg MLR1 drive, we performed 2000 random block accesses on each of the three tapes. For each access, we measured the access time and compared it to the corresponding access time estimated by the model. Table 6.6 contains the average difference between the measured and estimated access times for each of the four strategies for characterizing the tapes.

Before we comment on these numbers, it is worth noting that without a tape model all that can be said about the access times is that they are in the interval from 1 to 126 seconds with an average of 45 seconds. By studying the table, we see that the model performs worst when we use the *Average Tape-Length* strategy, with an average difference between estimated and measured access times of 10 seconds. This is as expected, since the varying tape sizes lead to bad estimates for the start address of each track. As a result the estimated seek times will drift away from the measured seek times as we get further out on the tape. An example can be seen in Figure 6.8. Figure 6.8(a) shows a plot of measured and estimated seek times for seeks starting at the beginning of the tape to every 20th block on two tracks on the tape, together with the difference. Figure 6.8(b) contains a similar plot for the same two tracks when we use a fixed position about 1/3 out on the tape as the start position for the seeks. These two figures show



(a) Seeks start at beginning of tape

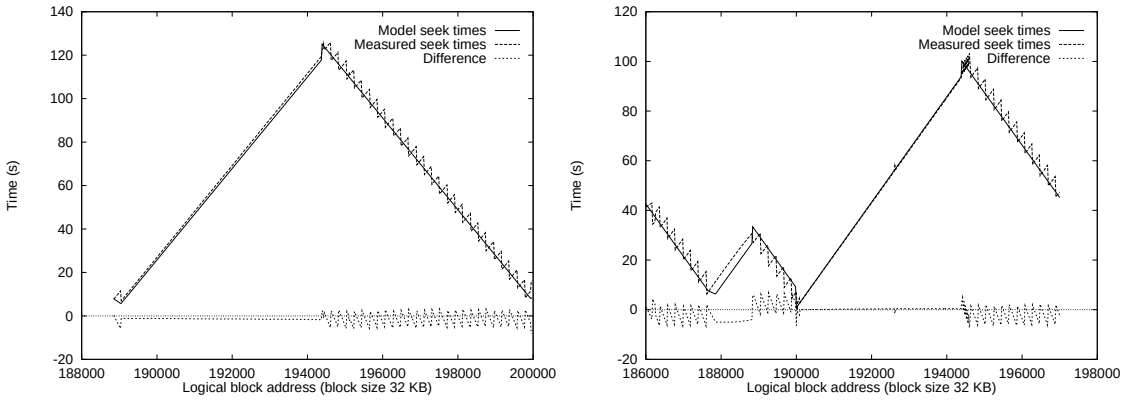
(b) Seeks start at block address 190000

Figure 6.8 Measured and estimated seek times for accesses to data blocks on two tracks using the tape model instrumented by the *Average Tape-Length*.

that the estimated access times do not model the measured seek times very well. The reason is the use of fixed, average track length in the model.

As Table 6.6 shows, the results are much better when we use one of the three strategies which characterize each tape. We also note that the two strategies which estimate the length of the individual tracks perform better than the strategy where we use a constant track length based on the total length of the tape. The reason is that even though the *Exact Tape-Length* strategy gives a correct estimate for the average track length, the track lengths can vary within a tape. As a result, the average difference between estimated and measured access times can vary rather much from tape to tape when using the *Exact Tape-Length*, e.g., compare the results for tape 1 and tape 3 in Table 6.6.

If we compare the two strategies for detecting the ends of the tracks, we see that they perform almost identically, with the *Write-Turn* strategy performing slightly better. In our experiments, the average difference between estimated and measured access times for random accesses was 1.7 seconds when using the *Write-Turn* strategy for finding the track addresses. There are two reasons why the results when using the *Write-Turn* strategy differ from the results when using the *Read-Turn* strategy. First, when using *Read-Turn*, we only localize half of the track addresses. Second, the strategies may not make the exact same decision about what is the first block on each track, due to the use of a buffer during the writing of the tape. In Figure 6.9(a), we have plotted the measured and estimated seek times for seeks starting at the beginning of the tape together with the difference for the same two tracks as shown in Figure 6.8(a) using the *Write-Turn* strategy. This time we observe that the two curves overlap much better, leading to better estimates. Figure 6.9(b) shows the corresponding curves for seeks start-



(a) Seeks start at beginning of tape

(b) Seeks start at block address 190000

Figure 6.9 Measured and estimated seek times for two tracks using the tape model instrumented by the *Write-Turn* algorithm.

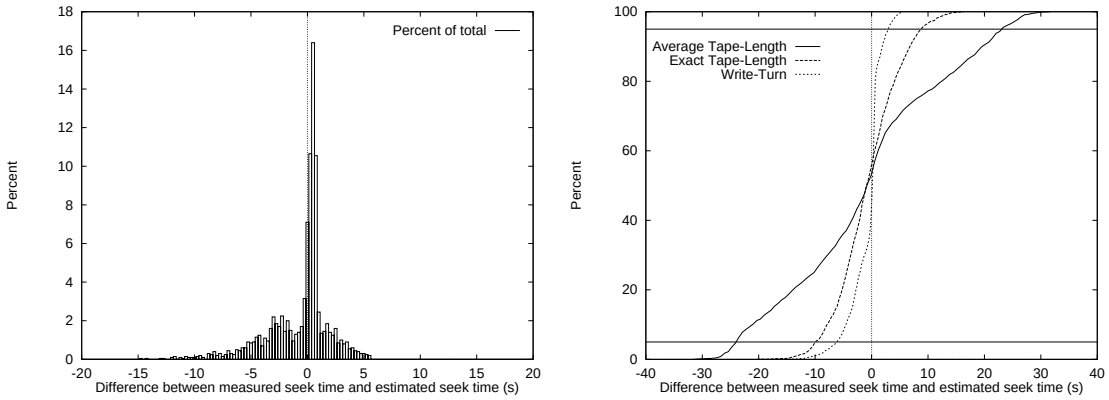
ing at a fixed position 1/3 out on the tape. This figure should be compared to Figure 6.8(b).

In Figure 6.10(a) we have plotted the distribution of the difference between estimated and measured accesses times for random accesses when using the *Write-Turn* strategy. This figure shows that most of the estimated access times are within 5 seconds from the measured access times. Figure 6.10(b) compares the distribution of the difference between estimated and measured times for three of the strategies. For *Write-Turn*, 90 percent of the measured access times are within 5 seconds from the estimated access times, while for *Average Tape-Length* this has increased to almost 25 seconds. A more detailed example illustrating the difference between estimated and measured seek times for each of the seek classes are included in Appendix B.

6.4.2 Validation using Quantum DLT 2000

The model was developed using the Tandberg MLR1 drive. To evaluate how the model performs for a different tape drive, we tested it using a Quantum DLT 2000 drive (Digital, 1992). This is one of the earliest of the high capacity drives of the DLT series. One cartridge is able to store 10 GB of data without the use of compression. The physical data layout on the tape is similar to the Tandberg MLR1 drive. Both drives use tapes that are 366 meters long. The width of the tape used by the DLT drive is twice the width of a MLR1 tape (a half inch compared to a quarter inch). The number of physical data tracks is 128, where two physical tracks are grouped into one logical track, giving 64 logical data tracks.

In Figure 6.11, we have plotted the seek and rewind times for seeks from the



(a) (b)

Figure 6.10 a) Distribution of the difference between measured and estimated access times for random accesses by using the tape model instrumented by the *Write-Turn* algorithm. **b)** Cumulative distribution of the differences between measured and estimated times for random accesses for the *Average Tape-Length*, *Exact Tape-Length* and *Write-turn* strategy.

start of the tape to every twentieth position along the first four tracks on one tape. Comparing this figure to Figure 6.1, the main pattern is the same with some important differences. First, for the DLT 2000 drive we see the sawtooth pattern on the forward tracks too. The reason is that the drive operates with two different speeds for transporting the tape past the read/write head. It uses a high speed to seek to the nearest key point, and then reads the tape with a lower speed. Second, there are fewer key points along each track on a DLT 2000 tape than on a MLR1 tape. The DLT 2000 has about nine key points on each track, compared to about 26 for the MLR1. Third, in the figure we see that the seek times to positions on the reverse tracks are much higher than the corresponding rewind times. The reason might be that the drive actually seeks further down the tape than the first key point past the sought position before it turns the winding direction. Similar behavior is found for the DLT 4000 drive as explained in (Hillyer and Silberschatz, 1996a).

Just as for the MLR1, we found the constants for the cost functions in Table 6.7 by performing seek operations on three tapes, and using linear regression on the seek times within each of the seek classes to determine the constants. For the Quantum DLT 2000 drive, it takes 94 seconds to wind the tape from one end to the other, and thus we use this as the value for t_{wind} in the model. To estimate transfer times, we also need the time to change tracks during read operations. We found this to be $t_{tc\_read} = 3.5$ seconds.

To evaluate the access-time model for the DLT 2000 drive, we characterized

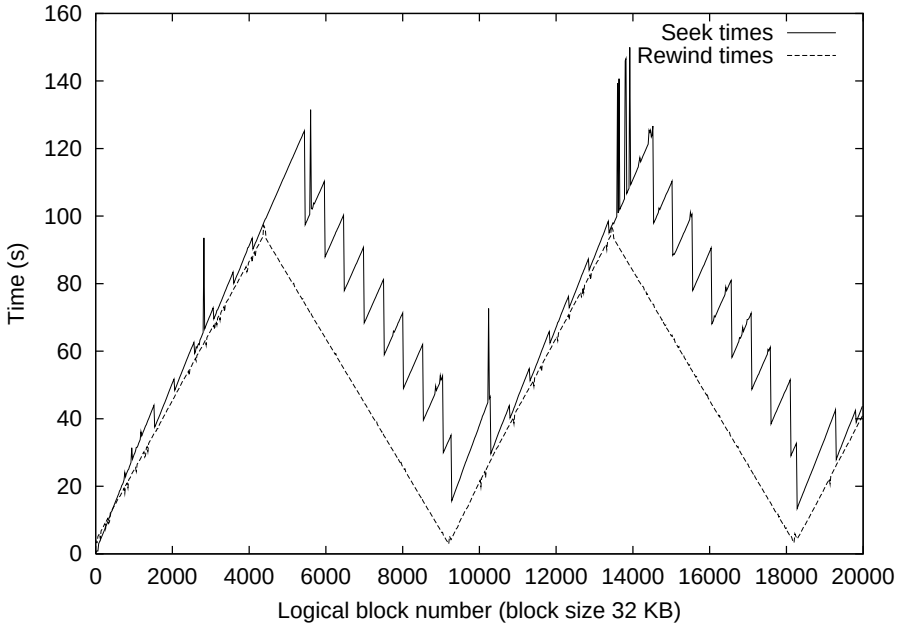


Figure 6.11 Seek and rewind times for the first four tracks on a DLT 2000 tape.

three other tapes by using each of the strategies for characterizing tapes presented in the previous section. By running 2000 random block accesses on these three tapes, and comparing the measured access times with the access times estimated by the model, we got the average difference as given in Table 6.8. These results should be compared to the results given in Table 6.6 for the MLR1 drive.

As can be seen from the table, the average error is about four times higher for the DLT 2000 drive than for the MLR1 drive when using the *Write-Turn* and the *Read-Turn* algorithms for characterizing the tapes. There are three main reasons for this. First, the model was developed for the Tandberg MLR1. While they are both serpentine tape drives, one of the conclusions from studying Figure 6.11 was that the DLT 2000 behaves somewhat differently when locating tape positions. Second, fewer key points on each track result in a longer distance between the key points. This will add to the average error for seeks where the cost functions include extra costs to locate the closest key point (i.e., all seek classes except 1, 4 and 7 in Table 6.4). Third, the use of one speed for seeking, and a lower speed for reading, increases the average error for all seeks.

Of the three reasons for the larger errors mentioned above, it should be easy to improve the model to handle different seek and read speeds while still maintaining a generic model. The other two points are more difficult to improve. To include the difference in behavior when locating tape positions between MLR1

Seek class	Cost function	Values for constants α	β
1	$\alpha_1 + \beta_1(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_1 = 4.601$	$\beta_1 = 1.051$
2	$\alpha_2 + \beta_2(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_2 = 26.877$	$\beta_2 = 1.089$
3	$\alpha_3 + \beta_3(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_3 = 32.101$	$\beta_3 = -0.139$
4	$\alpha_4 + \beta_4(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_4 = 13.126$	$\beta_4 = 0.894$
5	$\alpha_5 + \beta_5(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_5 = 27.069$	$\beta_5 = 1.044$
6	$\alpha_6 + \beta_6(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_6 = 31.070$	$\beta_6 = 0.0210$
7	$\alpha_7 + \beta_7(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_7 = 14.456$	$\beta_7 = 0.859$
8	$\alpha_8 + \beta_8(\mid p_{start} - p_{stop} \mid)t_{wind}$	$\alpha_8 = 29.652$	$\beta_8 = 1.018$

Table 6.7 Cost functions for the eight seek classes, with corresponding constants determined for the Quantum DLT 2000 drive. These functions returns the estimated seek time for a given seek. t_{wind} is the total winding time for a track. For a Quantum DLT 2000 drive this takes 94 seconds.

and DLT 2000 into the model, would make it more complex and less generic. To reduce the effect of the longer distance between key points would require us to include the positions of key points into the model, and worse, it would make the characterization process much more time consuming.

The average access time for a DLT 2000 is about 60 seconds. Compared to not using an access-time model, being able to estimate access times with an average error of about 7 seconds is still a large improvement. Thus although the model was developed using a MLR1 drive, it is also useful for other serpentine tape drives.

6.4.3 Cost of Establishing the Model

It is important to be aware of the cost of achieving the better results by using the model. The cost of establishing the serpentine tape model is low. The cost functions for each of the eight seek classes in Table 6.4, can be established once for each tape drive type. Finding the start addresses of the tracks has to be done once for each tape because these vary from tape to tape. The *Exact tape-length* strategy only requires that we get the total length of the tape when it is written. The *Write-Turn* strategy requires that we are able to measure the writing times of the blocks on the tape. If these writing times are available, the cost of finding the track addresses is virtually zero. The *Read-Turn* algorithm requires that each tape is run through the process of finding the end of the tracks before the model can be used. On average, we have measured the time usage for the algorithm to be about 13 minutes per tape. This is still worth the extra cost because of the much

	Average error [*seconds*]			
Strategy	All	Tape 1	Tape 2	Tape 3
Average tape-length	24.0 s	24.9 s	24.0 s	23.0 s
Exact tape-length	13.8 s	8.69 s	7.27 s	25.5 s
Write-turn	6.84 s	6.39 s	6.59 s	7.54 s
Read-turn	6.89 s	6.64 s	6.83 s	7.20 s

Table 6.8 Results from testing the model on a Quantum DLT 2000 drive using the Average Tape-length, Exact Tape-length, Write-Turn, and Read-Turn algorithms for instrumenting the model. The table contains the average difference between measured and estimated access times for 2000 random block accesses.

better estimates provided by the model.

The implementation of the model consists of about 400 lines of C++ code. For each characterized tape, we must store the track addresses, i.e., one integer per track when using the *Write-Turn* and *Read-Turn* strategies, or the total length of the tape when using the *Exact tape-length* strategy.

6.5 Conclusions

In this chapter we have presented an access-time model for serpentine tape drives. By studying the behavior of a Tandberg MLR1 tape drive, we have partitioned every possible seek into one of eight distinct seek classes. This partition is based on which operations the drive has to perform in order to go from the current position it has on the tape, to the physical position of the requested data block. For each of these cases, we have established cost functions. These cost functions use physical tape positions for estimating access times. To map from logical block addresses used by applications, to physical positions, the model uses estimates for the logical address of the first data block on each track.

Experiments show that the length of each track varies between tapes and within a single tape. As a result, to improve the accuracy of the estimated access times, it is necessary to characterize each tape by estimating the address of the first block on each track. The chapter presents several algorithms with varying costs to perform this characterization. By using the best characterization algorithm, *Write-Turn*, the model is able to estimate access times with an average difference between estimated and measured times of 1.7 seconds for the Tandberg MLR1 drive.

The proposed model balances the need of accuracy with the time needed to characterize each individual tape. One of the strengths of the model is the low

cost of characterizing individual tapes. If we have control of the writing of the tape, the cost of performing the characterization of the tape is virtually zero by using the *Write-Turn* strategy. If we are not able to log the writing of the tape, the address of the first block on each track can be found by using the *Read-Turn* strategy. The *Read-Turn* algorithm uses 13 minutes compared to twelve hours for the algorithms suggested by Hillyer and Silberschatz (1996a).

Although the model is made using a specific tape drive, it is generic enough to be easily adjusted to other serpentine tape drives. This is shown by testing and evaluating the model using a Quantum DLT 2000 drive. In the next chapter, we use this model as basis for scheduling of random accesses against a tape.

To improve the model, the key points have to be included in the model. As shown by Hillyer and Silberschatz (1996a), it is too time consuming to locate these for each tape. It would take even more time to do this on the Tandberg MLR1, since this drive has about twice as many key points per track as the Quantum DLT 4000. If we were to include the key points in the model, information about the location of the key points has to be made available to applications by the producer of the tape drive.

Chapter 7

Scheduling of Random I/O Accesses to Serpentine Tape

A few decades ago, magnetic tape was actively used in data processing. Input (program and data) as well as the results from the processing were stored on tape. The use of the tape was highly optimized as most of the processing was done as batch jobs, which read and wrote the tape sequentially. Since then, magnetic disks have taken over as the premier storage medium for program and data. The main advantage of disks over tape is the shorter random access latency, a few milliseconds rather than tens of seconds. The main advantages of magnetic tape are lower storage cost and higher storage density. For emerging applications, like scientific databases, digital image databases and video archives, which require vast amounts of data storage, these two factors can be of sufficient importance to make magnetic tape a possible choice.

When applications have a non-sequential access pattern to data stored on tapes, it is of foremost importance to minimize the random access delay and thereby maximize the utilization of the tape drives. When more than one data item is requested on a single tape, this can be achieved by careful *scheduling* of the concurrent I/O requests against the tape drive. While a lot of work has been done on scheduling of random accesses on magnetic disks, little research have been performed for serpentine tape. A very notable exception is the work by Hillyer and Silberschatz(1996b; 1998), which evaluates several algorithms for scheduling of random I/O requests on serpentine tape. The scheduling algorithms are evaluated using Quantum DLT 4000 and IBM 3570 Magstar MP tape drives.

In this chapter we study the problem of scheduling random I/O requests for serpentine tape drives. Scheduling of I/O requests requires an access-time model for the storage device. The access-time model presented in the previous chapter is evaluated by scheduling I/O requests with a number of known scheduling algorithms. We propose a new scheduling algorithm, Multi-Pass Scan Star (MP-Scan*), which makes efficient utilization of the streaming capability of serpentine tape, while avoiding the pitfalls of naive multi-pass scan algorithms and greedy

algorithms like Shortest Locate Time First. Using the access time model, the performance of MPScan* and several other scheduling algorithms are simulated, and the results are validated by measurements using Tandberg MLR1 and Quantum DLT 2000 tape drives.

The content of this chapter has been published in the paper "Improving the access time performance of serpentine tape drives" presented at the 15th International Conference on Data Engineering, held in Sydney, Australia, March 1999 (Sandstå and Midtstraum, 1999a).

7.1 Scheduling Algorithms

When a tape cartridge is mounted into a tape drive, several users may have issued any number of I/O requests for data on this single tape. The problem of scheduling such random access I/O requests for a serpentine tape drive can be stated as follows:

> *Given a list of I/O requests and an initial tape position, the goal is to produce a possibly reorganized list, containing the same requests, which will result in a minimum total access time when the requests are executed in this new order.*

In this section, we first describe a number of known scheduling algorithms and then propose a novel algorithm, Multi-Pass Scan Star (MPScan*), which makes clever utilization of the streaming capability of the tape drives. To clarify the discussion, we have classified the algorithms based on the degree to which they take the physical geometry of the tape into consideration.

7.1.1 Zero-dimensional Algorithms

These algorithms perform the "scheduling" of the I/O requests without any consideration of the physical properties of the tape.

READ. The tape is read sequentially until all requests are served. On a Tandberg MLR1 tape drive it takes about two and a half hour to read an entire tape.

FIFO. The requests are read in the order in which they are found in the initial schedule, without any attempts to optimize the execution order. This is the obvious method to schedule I/O requests for storage devices in cases where one does not have any knowledge of the properties of the device.

SORT. The requests are re-organized such that they are executed in order of increasing *logical* addresses.

7.1.2 One-dimensional Algorithms

In this class we find the algorithms that take into account the physical positions (longitudinal dimension) of the requests on the tape, but neglect to take into consideration that requests to close positions, but on different tracks, might incur relatively high seek times.

SCAN. All requests are executed in a single scan back and forth the entire tape – this algorithm is similar to the well known *elevator* algorithm used for disks. The requests are partitioned into two sets based on the read direction of the corresponding track. The requests in each set are sorted on *physical* tape position. If we assume that the tape drive is positioned at the beginning of the tape, SCAN reads all the requests on forward tracks as it *scans* through the tape, and then all the requests on reverse tracks as it scans back to the start of the tape. The complexity of this algorithm is $O(n \log n)$, where n is the number of requests in the schedule.

7.1.3 Two-dimensional Algorithms

The algorithms in this class take into account both the physical tape distance between two tape blocks (longitudinal dimension) and the cost of changing between tracks (latitudinal dimension).

OPT. This is the *optimal* scheduler which always (if the access-time model is exact) gives the shortest possible total execution time. The problem with this algorithm is that it reduces the scheduling problem to the familiar *Traveling Salesman Problem*, which is known to have an exponential time complexity for computing the optimal solution. Because of this, OPT is hardly useful for schedules containing more than ten requests.

LOSS. LOSS is an heuristic for solving the *Asymmetric Traveling Salesman Problem* (Cruyssen and Rijckaert, 1978). It solves the same problem as the OPT scheduler, but in linear time. The reason for including this scheduling algorithm, is to compare our scheduling strategies with the work by Hillyer and Silberschatz (1996b).

SLTF. Shortest Locate Time First (Hillyer and Silberschatz, 1996b), starting on the start position of the drive, it first selects the request with the least seek cost (locate time) as the next tape operation to be performed. It then selects the remaining request with the least seek cost from the new current position. This step is repeated until all requests are scheduled. The cost of this algorithm is $O(n^2)$.

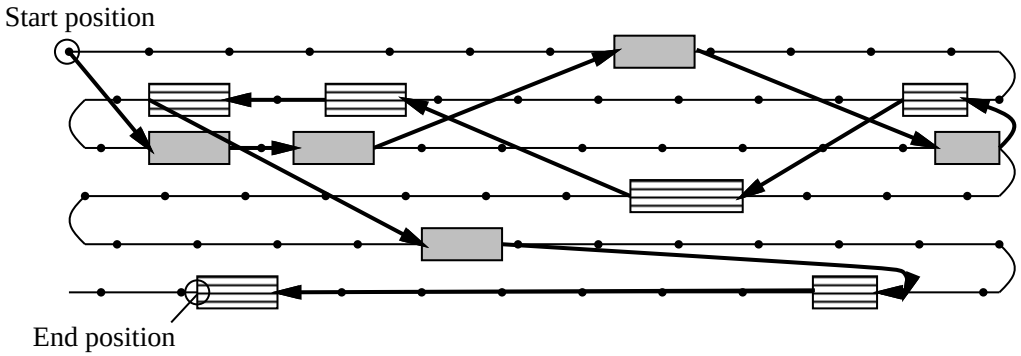


Figure 7.1 A MPScan schedule resulting in two full scans of the tape.

MPScan. Multi-Pass Scan, just like the SCAN algorithm, is an elevator algorithm, but this algorithm allows multiple passes of the tape. The main problem with the SCAN algorithm is that it does not take into account the cost of track changes. As a result, the tape drive might have to rewind to find the closest key point, when the next request in the schedule is physically close to the current tape position. MPScan avoids such interruptions of the streaming by careful selection of the next request to be included in the schedule. At each step in the production of the schedule, the algorithm considers only the requests which are further down the current track and the requests on co-directional tracks which are more than the key point distance further down these tracks. These are the requests in tape regions 1 and 4 as shown in Figure 6.5, relative to the current position. The physically closest of these requests is chosen as the next request to be included into the schedule. When no more requests are fulfilling the requirements for being included in a scan, a scan in the opposite direction is started. This strategy guarantees that the tape drive does not have to rewind when seeking to the next request in the schedule, at the cost of possibly having to make multiple passes through the tape. The complexity of the MPScan algorithm is $O(n^2)$.

An example, which shows how MPScan schedules 11 requests in two full scans of the tape, is given in Figure 7.1.

7.1.4 Multi-Pass Scan Star

Multi-Pass Scan Star, or just MPScan*, is an improved algorithm based on the MPScan algorithm. In schedules produced by MPScan, the number of requests per scan tends to decrease in the last scans of the schedule. The MPScan algorithm is too greedy, and selects the next request to be included in the schedule only by minimizing the cost that this single request will incur on the final schedule. As more requests are scheduled, fewer request are candidates for being scheduled next, and the seek times for each request will increase, leading to rather long seeks in the last part of the schedule. The MPScan* algorithm avoids these long

```
Schedule = MPScan(Request_list)   // produce initial MPScan schedule
N = number of scans in Schedule
MinCost = cost_of(Schedule)
Final_Schedule = Schedule

while ( N > 1 ) {
  remove last scan of requests from Schedule.

  while (some request r in last scan) {
    compute insertion cost for all possible positions to
       insert r into the Schedule.
    insert r where the insertion cost is least.
    remove r from last scan.
  }

  if (cost_of(Schedule) < MinCost) {
    MinCost = cost_of(Schedule)
    Final_Schedule = Schedule     // best schedule so far
  }
  N = N - 1;
}
```

Figure 7.2 Pseudo code for the MPScan* algorithm. The `cost_of` function estimates the execution cost of the schedule on a tape drive.

seeks at the end of a MPScan schedule by reducing the number of scans in the schedule and inserting the affected requests into the remaining scans.

The pseudo code for MPScan* is given in Figure 7.2. The algorithm starts by creating an initial schedule, using the MPScan algorithm. It then proceeds by reducing the number of scans that the tape drive has to perform, by repeatedly removing the last scan from the schedule. For each of the requests in the removed scan, an *insertion cost* is computed for all positions where the request can be inserted into the remaining schedule. The *insertion cost* is the extra seek time that the drive will use if a request, r_z, is inserted into the schedule between two other requests, r_x and r_y. Figure 7.3 illustrates how the *insertion cost* is computed. Each of the requests is inserted into the remaining schedule at the position with the least insertion cost. This process is repeated by removing the next last scan from the schedule, until the schedule consists of only one scan. For each scan removed, the total seek time cost of the new schedule is computed. Finally, the schedule with the least total seek time cost is chosen as the result of the algorithm. The worst case complexity of the MPScan* algorithm is $O(n^3)$.

Figure 7.4 shows the result of applying MPScan* to the same I/O requests as used for MPScan in Figure 7.1. As shown, the number of full scans of the tape is reduced from two to one by inserting the requests of the last scan into the first

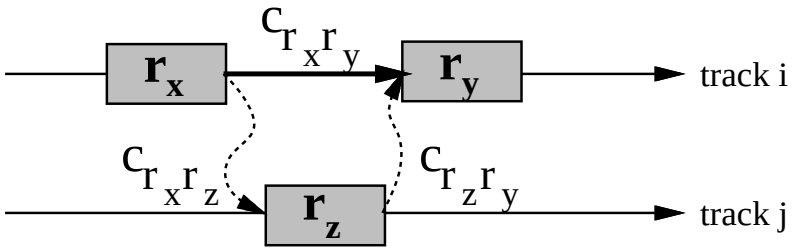


Figure 7.3 The *insertion cost* for inserting request r_z between request r_x and request r_y is computed as $c_{r_x r_z} + c_{r_z r_y} - c_{r_x r_y}$.

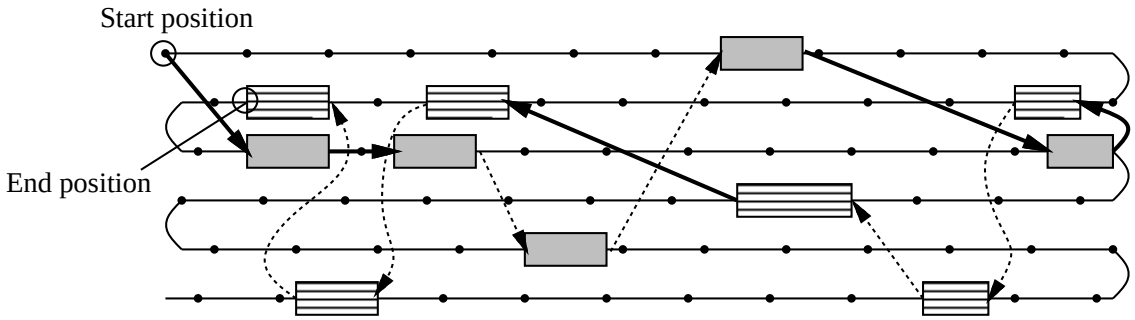


Figure 7.4 The MPScan schedule in Figure 7.1 reduced to one scan by using the MPScan* algorithm. New seeks introduced in the schedule are marked with dotted arrows.

scan, possibly leading to a shorter total execution time.

7.2 Simulations

The algorithms presented in the previous section have been implemented. This section presents results from simulation of the algorithms. The reason for simulating the algorithms is to be able to compare the properties of the different algorithms using the access-time model presented in the previous chapter. In the next section the simulation results will be validated by comparing them to results from execution of schedules on real tape drives.

To instrument the access-time model, we have used the Tandberg MLR1 drive as the target drive. All simulations were performed on a set of request lists containing from one to 2048 requests. Each request was for one 32 KB block on the tape. The block addresses were drawn from a uniform distribution in the inter-

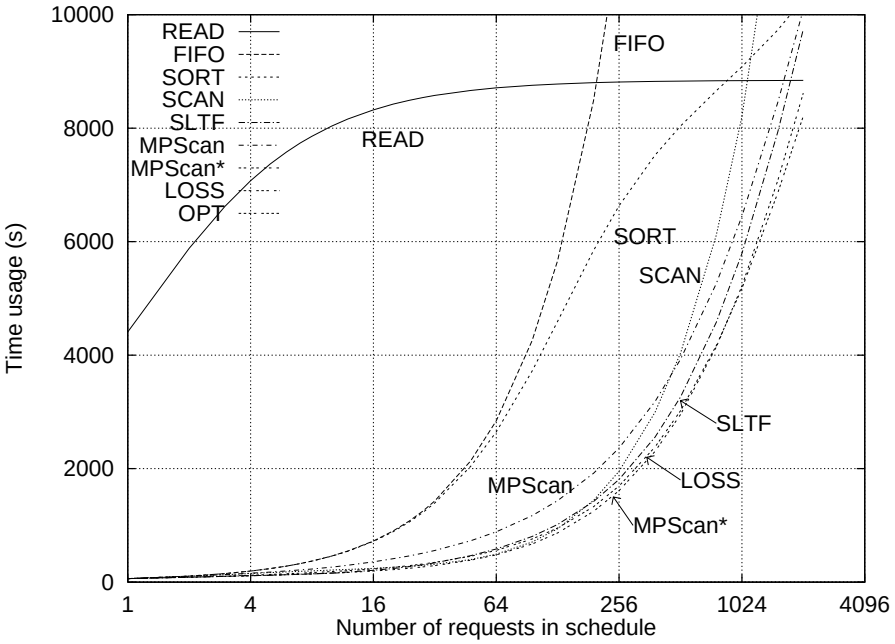


Figure 7.5 Total time for performing a request list for different request list sizes.

val $[0..MaxBlocksOnTape\rangle$. For schedules of length from 1 to 192 requests, we used 100,000 different request lists, for longer schedules we gradually reduced the number of list from 25,000 request lists for 256 requests per schedule, to 500 request lists for schedules of 2048 requests. For the OPT algorithm, the largest request lists contained 12 requests, and was run only 100 times due to the high CPU usage. All simulations started with the tape drive positioned at the beginning of the tape.

Figure 7.5 shows estimated execution times for the different schedule lengths using the different scheduling algorithms. The corresponding average access times per request are presented in Figure 7.6. As can be seen from Figure 7.6, with only one request there is nothing that can be done to improve the performance, and in average the access time for one tape request will be 63 seconds. If we do no scheduling of the requests (i.e., using the FIFO strategy) the average seek time stabilizes on 45 seconds. This can be greatly reduced by using one of the better scheduling algorithms. For schedules with less than 12 requests, the curves for OPT, SLTF and MPScan* overlap and any of them can be used. For longer schedules, it is not feasible to use the OPT algorithm. As long as the length of the schedule is less than 1000 requests, MPScan* produces the best schedules. For schedule lengths from 1000 requests to 2100 requests, LOSS is marginally better than MPScan*. For even longer schedules, the READ strategy gives the shortest total execution time for performing the schedule.

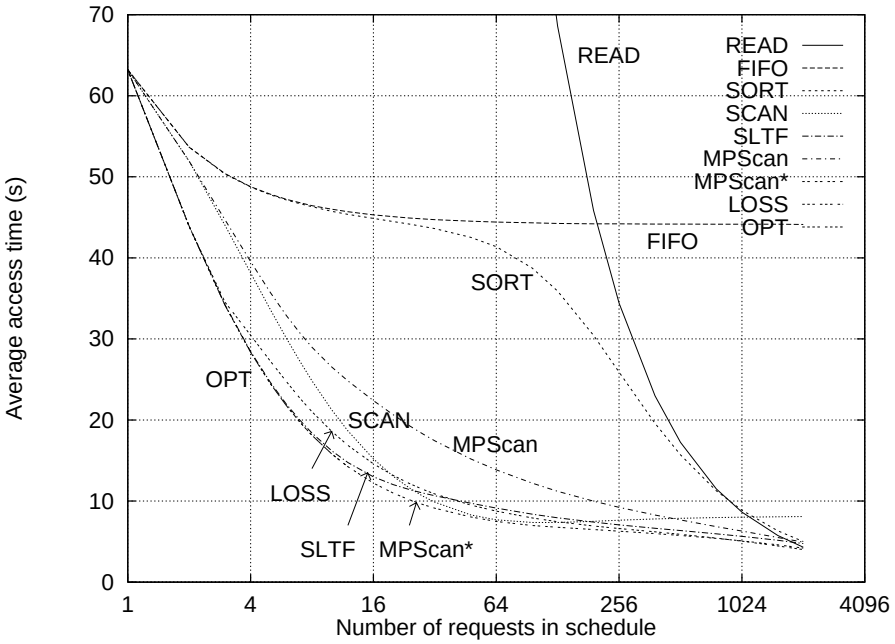


Figure 7.6 Average access time for each I/O request using different scheduling algorithms and for different problem sizes.

7.2.1 CPU Cost

The simulations were run on a SparcStation 20 workstation with a 125 MHz HyperSparc processor. Each run of the algorithms was timed, and the average times used to compute the different schedules are presented in Figure 7.7. If we disregard the OPT scheduler, LOSS and MPScan* are the only schedulers which use more than five seconds for computing the long schedules. Still, the higher CPU time usage of MPScan* and LOSS is much less than the reduction in total execution time achieved by use of these schedulers. Unfortunately, due to this higher CPU time usage, the initial latency for retrieving the first data object from the tape will be longer. This can be a problem for some applications. For MPScan*, this problem is readily solved by starting the execution of the first few requests in the first scan, as soon as the initial MPScan step of the algorithm has finished.

7.3 Experiments and Discussion

To validate the results from the simulations, we have run schedules of varying problem sizes produced by the different schedulers on a Tandberg MLR1 tape drive. As for the simulations, the tape drive was positioned at the beginning of the tape before the first tape operation was started, and each request was for one

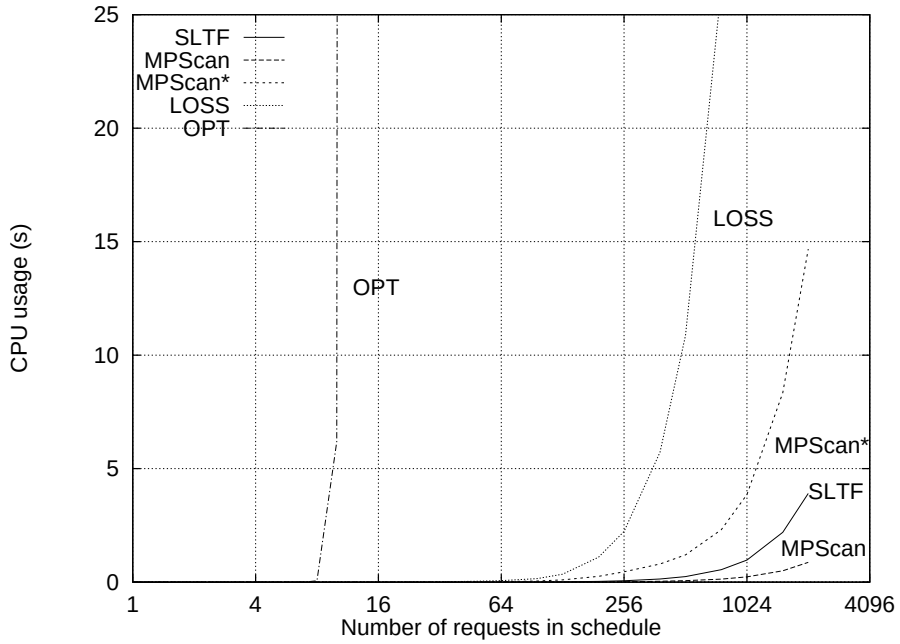


Figure 7.7 CPU usage for scheduling of request lists of different sizes. The cost for scheduling of 12 request using OPT is 912 seconds. To schedule 1024, 1536 and 2048 request using LOSS takes 63, 119 and 227 seconds.

32 KB block from a random position on the tape. For each schedule, the total execution time was measured.

The measured access times per request, for the different scheduling algorithms, are presented in Figure 7.8. By comparison of these results with the corresponding simulated access times given in Figure 7.6, it can be seen that the relative performance of the scheduling algorithms are the same in the simulations and the real experiments. The experiments confirms MPScan* as the best algorithm for problems with less than approximately 1000 requests, and LOSS as the best algorithm for problem sizes between 1000 and 2000 requests.

From these figures, it can also be seen that the estimated and experienced access times are approximately equal. To get a better comprehension of the accuracy of the estimated execution times, Figure 7.9 shows the difference in percent between estimated and measured execution time for schedules produced by MPScan* and run on the MLR1 drive. A positive value in the figure indicates that the estimated execution time is a number of percent higher than the actual measured execution time. As the figure shows, most of the estimated execution times are within +/-5 percent of the measured execution time. As this is achieved without determining the important key points, we consider this to be a good result. Determining every key point, Hillyer and Silberschatz (1996b) experienced differences within +/-1 percent for the Quantum DLT 4000 drive, but at a preventive high

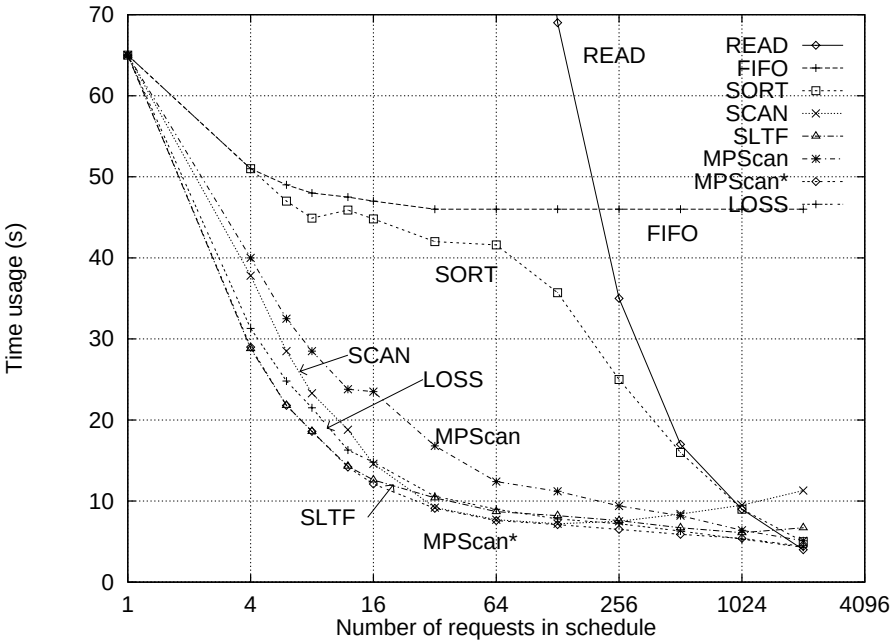


Figure 7.8 Average access time in seconds for each I/O request using different scheduling algorithms and for different problem sizes.

cost.

7.3.1 Access Time Improvements

The purpose of random I/O scheduling is to reduce the total execution time for a given combination of I/O requests. This will minimize the waiting time for the requesting application(s) and maximize the utilization of the (expensive) tape drives, which frequently are a bottleneck in tape-based storage systems. Figure 7.10 shows the relative and absolute improvements that the best scheduling algorithm MPScan*, gives compared to the non-scheduling approaches, which are random order (FIFO) execution of the requests, reading the entire tape (READ), or a favorable combination of FIFO and READ. As shown in the figure, MPScan* scheduling gives substantial benefits for all problem sizes ranging from two I/O requests and up to 2100 I/O requests. The maximum gain, compared to an optimal combination of FIFO and READ, is for a schedule of 196 requests, which is the point where FIFO and READ have the same performance. At this point, a MPScan* schedule executes in 20 minutes and 47 seconds, compared to an execution time of 2 hours and 27 minutes for the corresponding FIFO/READ schedule, saving more than two hours and seven minutes.

Comparing the results for MPScan* to the results for the other scheduling algorithms, we find that LOSS is the only scheduler that gives better results and

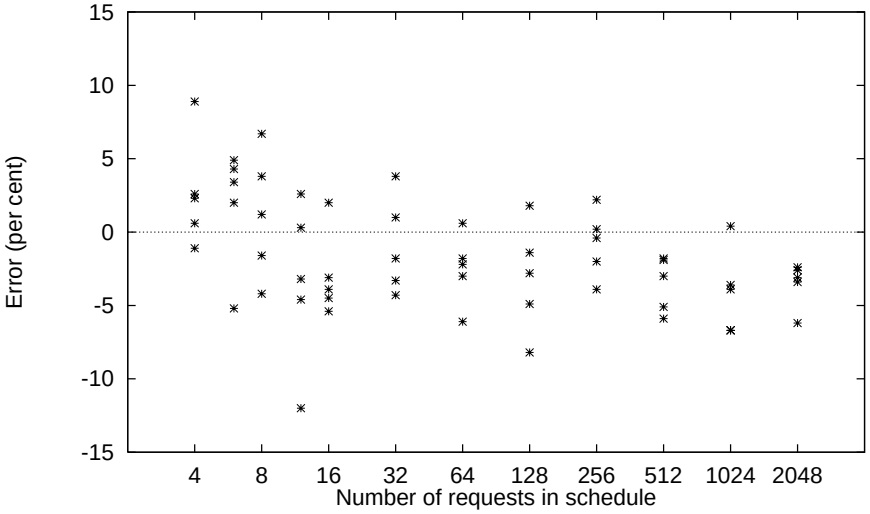
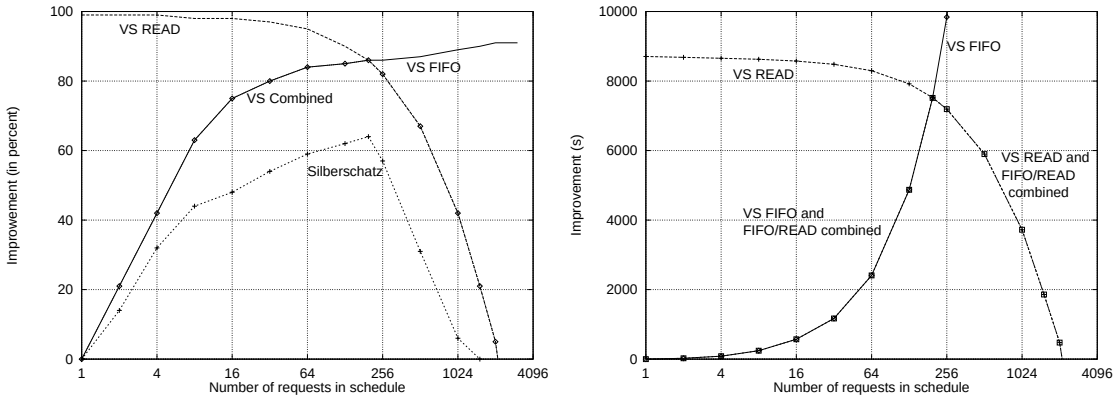


Figure 7.9 Deviation in estimated execution times for schedules produced by MPScan*. For each problem size, the results from performing five schedules on a Tandberg MLR1 are included.



(a) (b)

Figure 7.10 (a) Reduction in percent of total execution time using Tandberg MLR1 for MPScan* compared to FIFO scheduling, compared to READ and compared to an optimal combination of FIFO and READ. Results from Hillyer and Silberschatz (1996b) show their best results compared to an optimal combination of FIFO and READ. **(b)** Reduction in seconds of total execution time for MPScan* compared to FIFO scheduling, compared to READ and compared to an optimal combination of FIFO and READ.

No. of requests	MLR1	DLT 2000	DLT 4000
1	63.2	66.8	95
2	43.9	51.7	72
4	29.0	40.2	54
16	12.1	29.0	38
64	7.6	24.8	31
256	6.5	21.7	24
1024	5.4	11.0	13

Table 7.1 Average access time in seconds for the Tandberg MLR1 and DLT 2000 with the MPScan* algorithm, and DLT 4000 with the LOSS algorithm. The access times for DLT 4000 are found in (Hillyer and Silberschatz, 1996b).

only for schedules containing more than 1000 requests. For 1024 requests, the average improvement by use of LOSS is less than one percent. For 2048 request, the improvement increases to five percent. Unfortunately, the computational cost of LOSS is rather high for large schedules (see Figure 7.7), leading to a high initial latency. Unlike MPScan*, there is no easy way to start execution of any requests until the entire schedule has been computed.

Comparing MPScan* to SLTF, MPScan* produces schedules which are more than ten percent better in the entire interval from 24 to 2048 requests. A schedule of 128 requests has an average total execution time of 16 minutes and 55 seconds using SLTF, compared to 14 minutes and 26 seconds using MPScan*. For schedule sizes from about 25 requests to about 70 requests, the scheduler performing closest to MPScan* is SCAN.

7.3.2 Validation using the Quantum DLT 2000 Drive

To verify the generality of our access-time model and MPScan*, we have also used MPScan* and the other scheduling strategies to schedule random I/O requests on a Quantum DLT 2000 tape drive. The details from these experiments are found in Appendix C. As shown in Table 7.1, the results are similar, but not quite as good as the results for the MLR1 tape drive. The best result is for 150 I/O requests, where a 53 percent reduction of total execution time is achieved compared to FIFO/READ. As the DLT 2000 has fewer key points on each track, the key point distance is longer and the cost of many short seeks is going to be higher. The longer key point distance also decreases the accuracy of the access-time model, introducing more errors into the schedules.

7.3.3 Comparing Results

It is interesting to compare our results for the Tandberg MLR1 to the results that Hillyer and Silberschatz (1996b) achieved for the Quantum DLT 4000 tape drive. In Figure 7.10 (a) we have shown their best results compared to an optimal combination of FIFO and READ for the Quantum DLT 4000 drive. For any number of requests, we get significantly better results with the Tandberg MLR1 drive. On average, we reduce the total execution time with 23 percent more than they do for the DLT 4000 drive.

The average access times are much higher for the DLT 4000 drive than for the MLR1 drive. Table 7.1 shows the access times for the DLT 4000 drive with their best algorithm, LOSS, and the access times for the MLR1 drive with the MP-Scan* algorithm. A schedule of 196 I/O requests would for instance have a total execution time of 1 hour and 24 minutes on the DLT 4000 drive with the LOSS algorithm, compared to the less than 23 minutes on the MLR1 with MPScan*.

The Quantum DLT 4000 drive and the Tandberg MLR1 drive are comparable, except that the DLT stores 54 percent more data (20 GB versus 13 GB) and has a maximum seek time that is 43 percent longer (180 seconds versus 126 seconds). Even if we take the longer maximum seek time into full effect, the Tandberg drive still performs a schedule of 196 requests in less than half the time needed by the DLT 4000. The main reason why the MLR1 performs so much better, is that the Tandberg drive has many more *key points* (25 versus 13 on each track). The higher number of key points leads to a much shorter key point distance, which significantly reduces the cost of many of the shorter tape movements. Another factor is that the DLT 4000 seeks with 30 percent lower speed when searching for a position between two key points, while the MLR1 performs all operations at full speed.

7.4 Analysis of Tape Behavior

This section presents observations from studying the behavior of the scheduling algorithms in practical experiments with the Tandberg MLR1 drive. These results will explain some of the results presented earlier in this chapter.

In Figure 7.11, we show the measured seek time for each request in one schedule containing 512 tape requests for some of the schedulers. From this figure, it is worth noting that only the MPScan* and LOSS schedulers have approximately constant average seek times for all the requests in the schedule. MPScan and SLTF have about the same average seek times for the first part of the schedule, but for the last part of the schedule the seek times increase significantly. For MPScan the reason is that it is too *lazy*, only including requests which will not interrupt the streaming of the tape drive in the schedule. This leads to few requests in the last scans of the schedule, and to too many passes of the tape. For SLTF the reason is that it is too *greedy* and only considers one step at a time. The effect of the

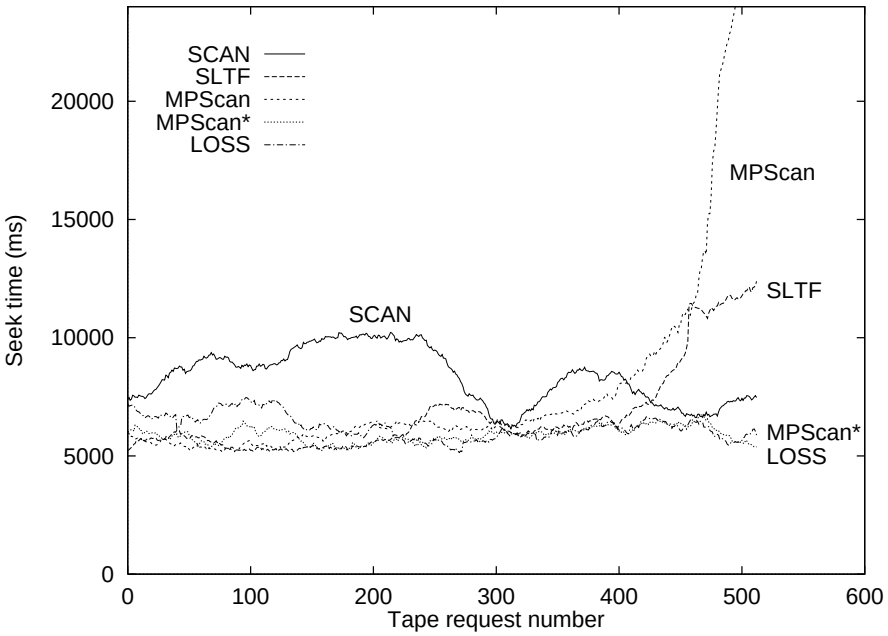


Figure 7.11 Measured seek times for each request in a schedule of 512 requests. The schedules are produced using SCAN, SLTF, MPScan, MPScan* and LOSS. The seek times have been run through a smoothening filter of width 60.

greedy behavior of SLTF can also be seen in Figure 7.12, where we have plotted the longitudinal seek pattern on the tape for the same schedule as in Figure 7.11. Note that although the SLTF algorithm does not impose any particular seek pattern on the tape, it still produces a seek pattern where the tape drive is mostly streaming for the first part of the schedule. As the number of non-scheduled requests gets lower, the algorithm starts to produce more and more costly seeks (see Figure 7.11). This is not the case for MPScan* where the long, costly seeks have been removed from the schedule. As can be seen in Figure 7.13, MPScan* manages to keep the streaming pattern for the entire schedule. The figure also shows "clusters of requests" along the tape scans. These are the positions in the schedule where the requests of the last scans of the initial MPScan schedule have been inserted.

7.5 Conclusions

In this chapter we have used the access-time model presented in the previous chapter to schedule random I/O requests against tape drives. Simulations and executions on tape drives have demonstrated results that are fully comparable to

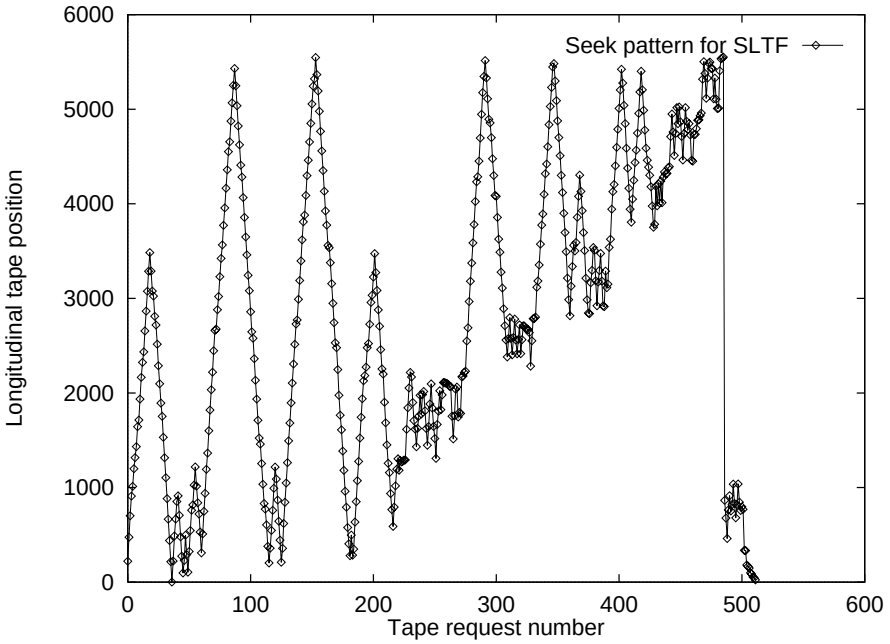


Figure 7.12 Physical seek pattern on the tape for a schedule of 512 requests produced by SLTF. The graph shows the longitudinal tape position for as the drive executes the requests in the schedule.

results that have been achieved with far more elaborate approaches. The main conclusion is that our low-cost model gives access time estimates that are good enough to facilitate efficient scheduling of random I/O requests, while still having a resource consumption well within practical limitations. When used for the Tandberg MLR1 tape drive, the accuracy of the time estimates of the schedules are within +/- 5 percent for most cases.

We have proposed a novel scheduling algorithm, Multi-Pass Scan Star (MP-Scan*), which makes good utilization of the streaming capability of the tape drive and avoids the pitfalls of naive multi-pass scan algorithms and greedy algorithms like Shortest Locate Time First. Compared with other scheduling algorithms, MP-Scan* gives shorter access times for all reasonable problem sizes and produces access patterns that are favorable to keep the drives in "streaming" operation. Use of the MPScan* algorithm have clearly demonstrated the usefulness of I/O scheduling, as up to 85 percent savings in execution time have been experienced, compared to no scheduling.

The applicability of our approach has been demonstrated by practical experiments on the Tandberg MLR1 drive and on the Quantum DLT 2000 tape drive. An interesting side effect of our work is the difference in random I/O performance that has been experienced for the Tandberg MLR1 and Quantum DLT 4000 drives. From the specifications, the Quantum DLT 4000 and Tandberg MLR1 tape drives

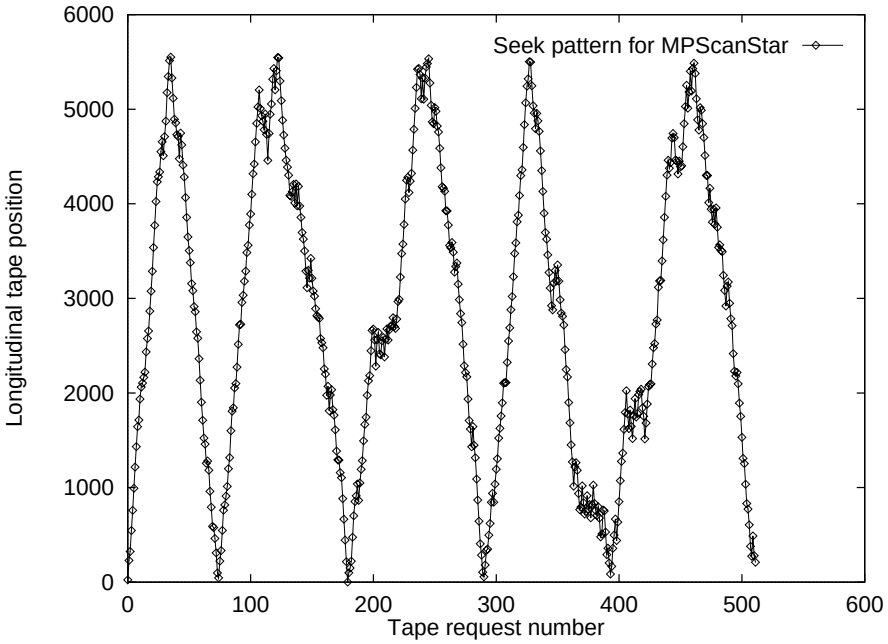


Figure 7.13 Physical seek pattern on the tape for a schedule of 512 requests produced by MPScan*. In this particular schedule, all requests are read in five scans through the tape.

seem to have very similar performance characteristics. For random access, however, the Tandberg drive has demonstrated superior performance.

Chapter 8

Retrieval of Multimedia Data Stored on Serpentine Tape

In the previous chapter, we studied scheduling of random I/O requests to a serpentine tape, and presented the results that could be achieved by using the MPScan* scheduler. In the study, all requests were for one data block on the tape. As performance criteria we used the average access time and the throughput. In this chapter, we study the problem of scheduling I/O requests for multimedia data stored on serpentine tape, using the Tandberg MLR1 tape drive as an example. Compared to the requests scheduled in the previous chapter, the size of most multimedia objects is much larger than one data block. We evaluate the different scheduling strategies for retrieval of multimedia objects of different sizes. This evaluation is performed by using both simulations and measurements on a Tandberg MLR1 drive. The purpose is to evaluate the properties of the MPScan* algorithm when retrieving data objects larger than one data block. As MPScan* is shown to be the best algorithm, we provide a detailed discussion of the Quality of Service that the use of this algorithm can provide for retrieval of image and video data.

To illustrate one possible application, we give an example. The Objects Collection at the Norwegian Folk Museum collection consists of 250,000 items which are photographed with a digital camera. On average, three images are taken of each object. For each image the museum wants to store both a high quality version (3.5 MB) and a JPEG compressed version (160 KB) for use on the WWW. The total size of the database will be 2.7 TB. A single 13 GB MLR1 tape would be able to store over 81,000 of the JPEG images. A user query requesting images of a "rocking chair" could for instance find 100 such images on a single tape cartridge, using a secondary index on type of object. To read these 100 images, which are scattered around the tape, without any attempts to optimize the access time, would take 3 hours and 20 minutes in the worst case and 1 hour and 15 minutes in the average case. Using the best scheduling algorithm, this access time can be reduced to 11 minutes and 40 seconds, which is only 16 % of the average case without opti-

Case	Data size	Objects on a tape
Compressed JPEG image	160 KB	79924
Uncompressed 600x800 image	1.44 MB	8881
High quality image	30 MB	426
1.5 Mbit/s video clip, 62 seconds	11.6 MB	1092
6 Mbit/s video clip, 62 seconds	46.5 MB	264

Table 8.1 The five different storage cases.

mizing. As this simple example shows, the random access performance of tape drives can be significantly improved by means of proper I/O scheduling.

The results from this chapter have been published in the paper "Analysis of retrieval of multimedia data stored on magnetic tape" presented at the 1998 International Workshop on Multi-Media Database Management Systems, held in Dayton, Ohio, August 1998 (Sandstå and Midtstraum, 1998).

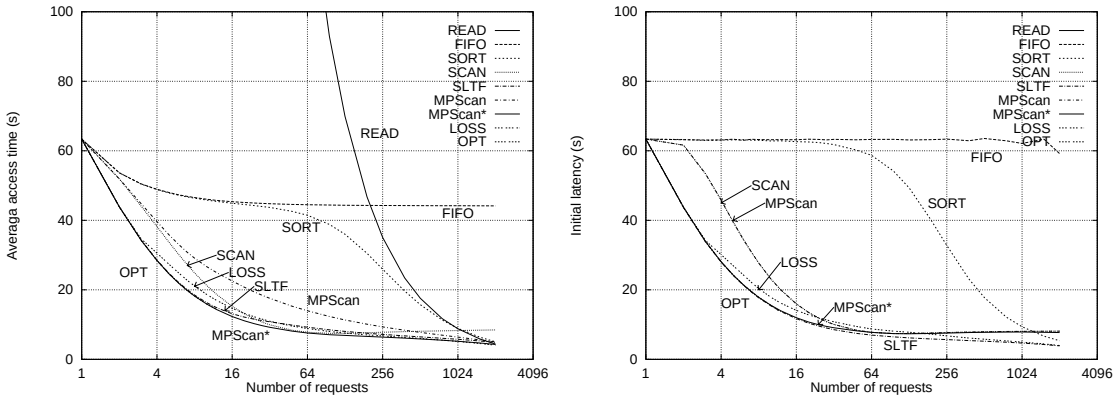
8.1 Storage Cases

To analyze the effect of scheduling in different situations, we use five different multimedia cases in the simulations. First, we consider storage of images of low, medium and high quality. Second, we consider storage of short video clips of medium and high quality. To have realistic data sizes for the video objects, we have analyzed television news from TV2 Norway and found that their typical news program consists of 12 news stories, with an average duration of 62 seconds each. In a television news archive application, a set of such news stories would be the typical result of a query (Hjelsvold et al., 1996). The details of the five storage cases are presented in Table 8.1. The bandwidths of the two video cases have been chosen to resemble typical MPEG-1 and MPEG-2 video streams.

8.2 Simulations of Scheduling Algorithms

Chapter 7 presented nine different scheduling algorithms for serpentine tape. In this section, we present the results from simulations of the scheduling algorithms used for retrieval of multimedia objects. The reason for simulating the algorithms is to compare the properties of the different scheduling algorithms for retrieval of media objects of different sizes. The most important properties of the scheduling algorithms are the *initial latency* until the first request has been completed and the *average access time* for the scheduled requests. The *CPU cost* for computing a schedule was studied in Section 7.2.1.

For each of the five storage cases, we consider a 13 GB MLR1 tape filled with corresponding media objects. The number of objects on a tape is given in Ta-



(a) Average access time

(b) Initial latency

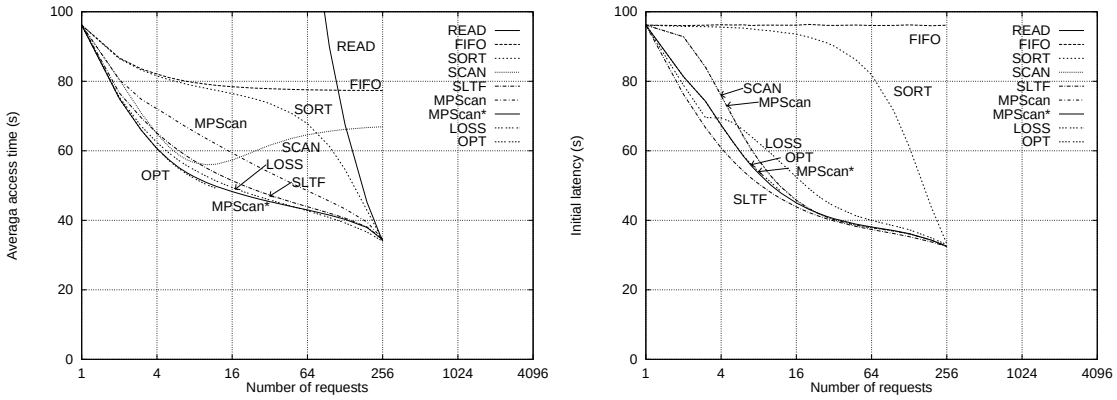
Figure 8.1 Average access time and initial latency for accessing images of 160 KB from the tape.

ble 8.1. To perform the simulations, we have made request lists containing from 1 to 2048 requests for distinct media objects on the tape. For schedules of lengths from 1 to 192 requests, we have repeated the simulation for 100,000 different request lists, for schedules from 256 to 2048 requests we have gradually reduced the number of request lists from 25,000 for 256 requests, to 500 for 2048 request. For the OPT algorithm, the largest request lists contained 12 requests and was run only 100 times due to the high CPU usage. All simulations started with the tape drive positioned at the beginning of the tape.

Figure 8.1(a) shows the average access times for tape requests for images of 160 KB. With only one request, there is nothing scheduling can do to improve the performance, and the average access time for one 160 KB image will be 65 seconds. For requests of more media objects, no scheduling (i.e., use of the FIFO strategy) would result in an average access time of 45 seconds. In this case, the average access times can be greatly reduced by using one of the better scheduling algorithms. For schedules with less than 12 requests, the average access time curves for OPT, SLTF and MPScan* overlap, and any of them could be used. For longer schedules, it is not feasible to use the OPT algorithm, and as long as the schedule contains less than 2100 requests, MPScan* gives the shortest access times. For longer schedules, the READ algorithm should be used.

Figure 8.1(b) shows the initial latency until the first image is made available to the application. As for access times, the best scheduling algorithms provide substantial benefits compared to the FIFO approach. For large request sizes, SLTF gives the shortest initial latency. This is due to the *greedy* behavior of always selecting the request with lowest seek time first (Hillyer and Silberschatz, 1996b).

To illustrate how the performances of the schedulers are influenced by the size



(a) Average access time

(b) Initial latency

Figure 8.2 Average access time and initial latency for accessing 6 Mbit/s video sequences of 62 seconds from the tape. The average object size is 46.5 MB.

of the requested media objects, Figure 8.2 shows average access times and initial latencies for the 6 Mbit/s video clips, which is the largest object size that has been investigated. Due to the much larger data size, the average access times and the initial latencies have increased. Comparing the different schedulers, there are three points worth noting. First, MPScan* now performs better than SLTF over the entire simulation range. Second, when a high percentage of the objects on the tape is requested, LOSS performs better than MPScan*. Third, SCAN performs much worse than for smaller objects, because it quite often has to rewind long distances to get to the next request.

The simulations for the intermediate object sizes show similar results. MPScan* gives the best results for all object sizes, and should be preferred over the other scheduling algorithms. In the remaining of this section we present more detailed results for this scheduler. Figure 8.3 shows the average access times for all the five storage cases in Table 8.1, using the MPScan* scheduler. The access time curve for the READ scheduler is included to indicate the point where it is sensible to change to this approach. The figure shows that the MPScan* scheduler provides substantial improvements for all data sizes, compared to not using a scheduler. Note that larger data sizes increase the access times, and reduce the number of requests that are necessary to make READ the best solution.

The MLR1 tape drive has a maximum sustained data rate of about 1.4 MB/s. Figure 8.4 shows the effective data rates that can be achieved for different object sizes and numbers of requests, using the MPScan* scheduler. For the small images of 160 KB, the effective data rate is quite low, from 2.5 KB/s for schedules of one image to about 30 KB/s for schedules of 1000 images. As the object

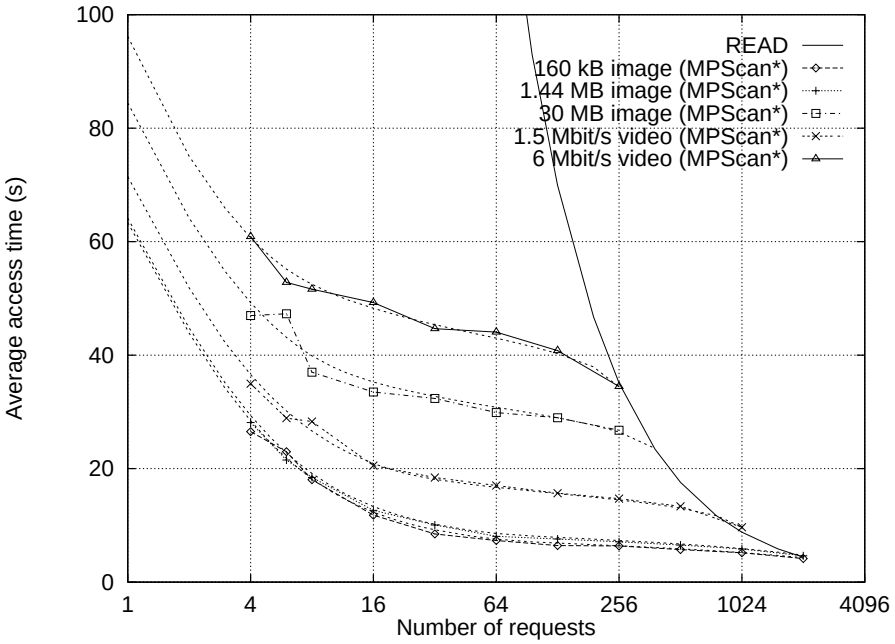


Figure 8.3 Simulated access times (dotted lines) using MPScan* compared to measured average access times.

size increases, the effective data rate increases. As an example, accessing sixteen 6 Mbit/s video clips produces a high sustained data rate of almost 1 MB/s, which is a 70% overall utilization of the maximum data bandwidth of the tape drive.

8.3 Validation of Simulation Results

To validate the simulation results, we have run schedules of varying problem sizes on a MLR1 tape drive. By doing this, we were able to verify that the behavior of the algorithms are similar to the simulation results, and that the estimated execution times are approximately equal to the measured execution times. Since the MPScan* scheduler is the preferred scheduler to use for most problem sizes, we only present results from executions of schedules produced by MPScan*.

In the experiments, we used the same storage cases as during the simulations. The tapes were filled with fixed sized data objects, according to the sizes given in Table 8.1, and rewinded to the beginning of the tape before the first tape operation was started. Each tape operation consisted of reading an object of a given size from the tape. For each schedule, the total time used to execute all the operations was measured.

Figure 8.3 presents measured average access times for schedule lengths from 4 to 2048 requests for each of the five storage cases. Comparing the measured

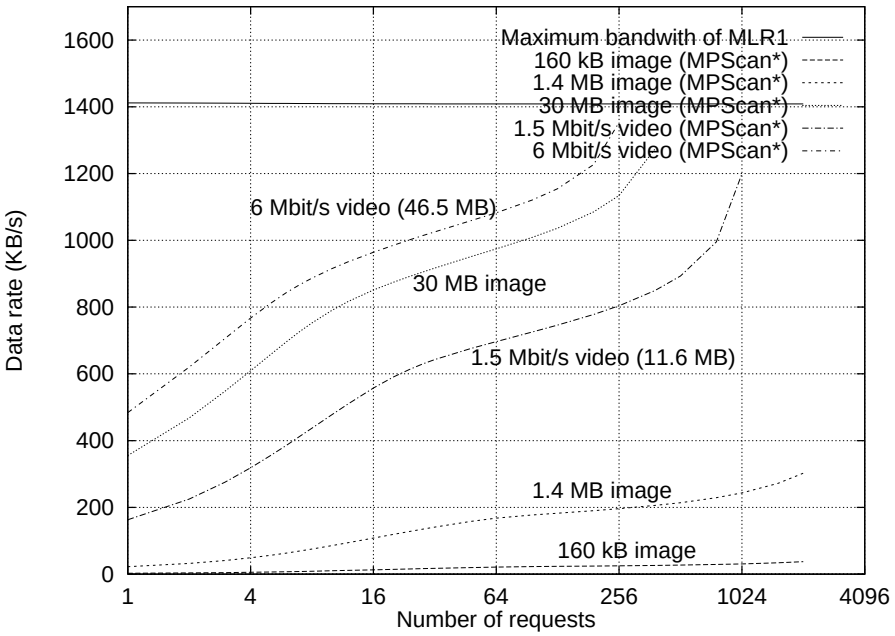


Figure 8.4 Effective data rates using the MPScan* scheduler.

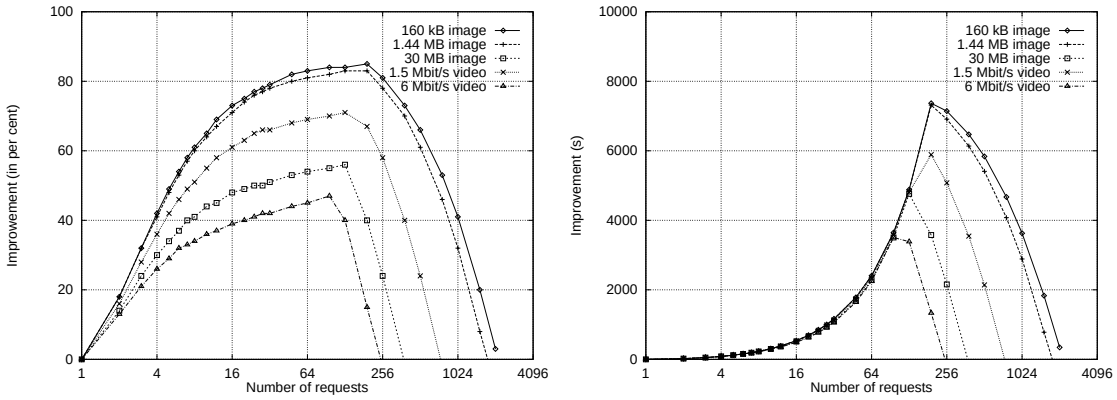
access times to the simulated access times for MPScan* shows that the simulated access times are good approximations of the measured access times.

To get a better apprehension of the accuracy of the estimated times, we have calculated the difference between the execution times estimated by the MPScan* scheduler and the measured execution times for each of the schedules. The average difference between estimated and measured times in our experiments was 3.2 %. This shows that the estimated scheduling time is a good estimate for the actual time to perform the schedule, and is good enough to facilitate efficient scheduling of queries for multiple media objects stored on a tape.

8.4 Access Time Improvements

The purpose of scheduling random I/O requests is to reduce the total execution time for the requests, in order to minimize the waiting time of the requesting application and to maximize the utilization of the tape drive. Figure 8.5 shows the relative and absolute improvements that the best scheduling algorithm, MPScan*, gives compared to the best non-scheduling approach, which is an optimal combination of FIFO and READ. From the figure, one should make the following observations:

1. For all object sizes, the advantage of MPScan* first increases with the number of request, then reaches a maximum gain at the point where READ starts



(a) (b)

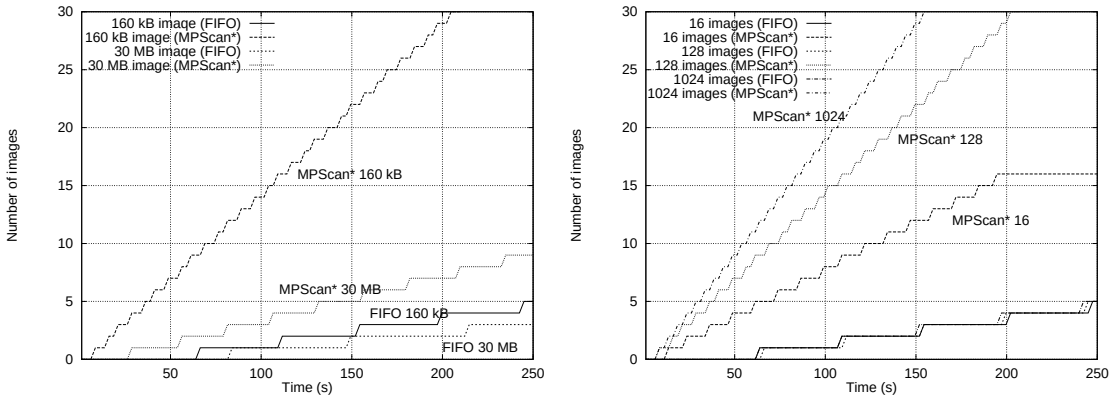
Figure 8.5 (a) Reduction in percentage of total execution time by using MPScan*, compared to an optimal combination of FIFO and READ scheduling. **(b)** Reduction in seconds of total execution time by using MPScan*, compared to an optimal combination of FIFO and READ scheduling.

to get better than FIFO, and from that point on, the relative advantage decreases until READ finally takes over as the best approach.

2. As the object size increases, the relative advantage of MPScan* decreases, and READ takes over as the best algorithm at a lower number of requests. This is intuitive, since the transfer time takes a higher share of the total access time as the object size increases.

3. Until the points where READ takes over from FIFO, the absolute improvements (in seconds) are largely independent of the object size and only a function of the number of requests. This implicates that the seek times for MPScan* are determined by the number of requests only, and do not depend on the media object size.

For small image files, MPScan* more than halves the total execution time of all schedules containing from 5 to 768 requests. The best results are achieved for schedules of 196 request, which are executed in 1/7 of the time, saving more than 7300 seconds (more than two hours). For large video files, the savings are more modest. The total execution times are reduced with more than 15% for all schedules containing from 2 to 192 requests. The best results are for schedules of 96 requests, which are executed in half the time, saving more than 3500 seconds (slightly less than an hour).



(a) (b)

Figure 8.6 (a) Number of retrieved images for FIFO and MPScan* for different image sizes. The scheduled request was for 128 images. **(b)** Number of retrieved 160 KB images for FIFO and MPScan* for different schedule lengths.

8.5 Quality of Service

When magnetic tapes are used to store multimedia objects, the Quality of Service (QoS) parameters for the retrieval of the objects are: 1) *initial latency*, which is the time from the data request is sent, until the first requested object is made available to the application, 2) *inter-arrival times*, which are the times between delivery of two consecutive requests, 3) *total execution time* for retrieving all requested objects, and 4) *resulting data rate*. A fifth QoS parameter being the degree of *variance* of the other QoS parameters. Since the use of image and video data have different characteristics, we have organized the discussion of QoS in two parts, first considering QoS in the context of image objects, and then discussing QoS in relation to video data. The rest of this section discusses QoS as perceived from a single user. If multiple users are issuing concurrent request to the same tape, the QoS will be lower and fairness of service has to be considered.

Image Objects

Assume a user of an application retrieving a number of images from an image collection. For her, the important QoS parameters will be the initial latency, the inter-arrival times, and the total execution time. Figure 8.6(a) shows the number of images that will be retrieved from a tape during the first four minutes using FIFO and MPScan*. The figure shows the retrieval rates for small (160 KB) images and large (30 MB) images when the user has requested 128 images for retrieval.

The user will experience much lower initial latencies if the application uses MPScan* instead of FIFO, in average 7 versus 64 seconds for 160 KB images, and 25 versus 83 seconds for 30 MB images. Also, the inter-arrival times are much lower for MPScan* than for FIFO, in average 7 versus 44 seconds for the 160 KB images, and 28 versus 65 seconds for the 30 MB images. As a result, the total time to retrieve 128 images of size 160 KB will be reduced from about one and a half hour to less than 15 minutes by changing from FIFO to MPScan*.

Another quality of MPScan* is that the variations in both initial latency and inter-arrival times are much lower than for FIFO. For larger request sizes, these typically vary with a few seconds using MPScan*, while for FIFO these times vary uniformly between a few seconds and up to two minutes.

Figure 8.6(b) shows how the retrieval rate for FIFO and MPScan* is influenced by the number of images included in a schedule. As the figure shows, FIFO is not influenced by the number of requested images. For MPScan*, both the initial latency and the inter-arrival time are reduced as the number of images in the schedule increases. During the first 100 seconds after the tape drive has started performing the tape operations, in average only one image will be made available to the user if FIFO is used. If sixteen images are requested, MPScan* will be able to deliver 8 of these in the same period of time.

Video Objects

Contrary to image data, it is possible to estimate the time a user is going to spend "consuming" a video object. If VCR operations like *fast forward* are ruled out, a video object is consumed at the *playback rate* of the video stream, e.g. 1.5 Mbit/s. This facilitates computation of the production to consumption ratio, *PCR*, and introduces a sixth QoS parameter, *Waiting time*. Waiting time is the total time that the user spends waiting for video data to play.

Figure 8.7 shows the retrieval of four video clips. After the data request has been issued, the user has to wait the initial latency period, until the first video clip has been delivered from the tape drive. The user then starts playback of the video stream, while the tape drive goes on to fetch the next video clip. The figure shows what happens next for different PCR values. If PCR < 1, a user, which performs one uninterrupted playback of all the video clips, will have to wait for every video clip that is retrieved (e.g., a total waiting time of 5 time units for PCR=0.5 in the figure). If PCR = 1, a user will only have to wait for the first video clip, and the maximum buffer space equals the size of the largest video clip. If PCR > 1, a user experiences only the initial wait period, and the amount of video in buffer grows with the number of retrieved video objects.

If buffer space is of little concern, one would want the PCR to be as high as possible, leading to the least total execution time and a minimum waiting time. If PCR ≥ 1, it would be possible to start video playback as soon as the tape drive starts to deliver video data. This would reduce the initial latency, and possibly

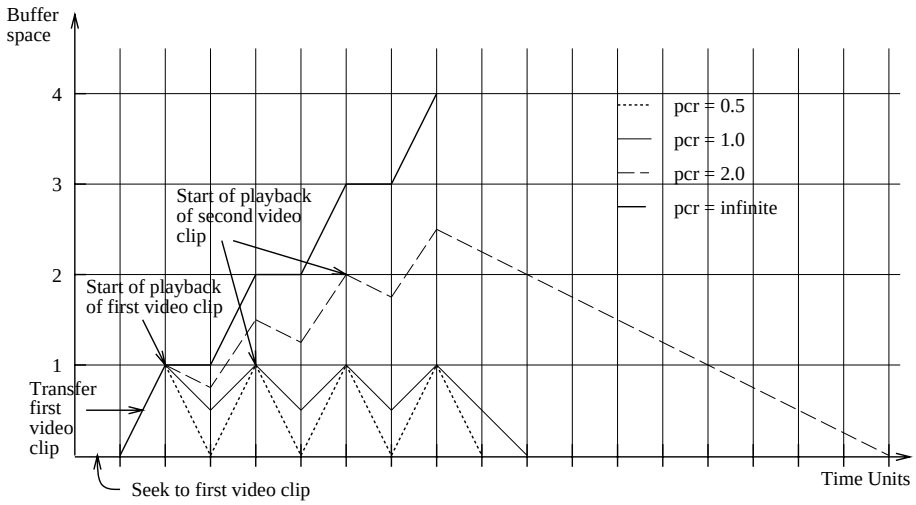


Figure 8.7 Retrieval and playback of four video clips for different PCR values.

the total waiting time, but to simplify the discussion, this possibility will not be discussed any further in this chapter. A more detailed discussion of *pipelining* is found in (Ghandeharizadeh et al., 1995).

Table 8.2 shows QoS parameters for retrieval of 1.5 Mbit/s and 6.0 Mbit/s video streams, respectively. The ”Must see” column gives the fraction of each video clip that a user has to play in order to have a PCR of 1.0, and thus avoid waiting for the next video clip. A “Must see” value greater than one means that the user has to play the clip more than once (or rather, spend more time than one full playback). Without any scheduling (FIFO), the tape drive barely reaches a PCR of 1.0 for video clips of medium quality, and the resulting QoS is going to be just acceptable. For a high quality stream, quite an amount of waiting time has to be expected. Utilizing MPScan* scheduling, the QoS improves to be quite good for medium quality video clips, and just acceptable for the high quality video. These QoS values are for video clips of 62 seconds. As long as the video data rate is less than the data transfer rate of the drive (1.4 MB/s or 12 Mbit/s), longer video clips would improve the QoS parameters (except total time), while shorter clips would make them worse (except total execution time).

8.6 Conclusions

In this chapter, we have investigated static scheduling of random I/O requests for multimedia data stored on magnetic tape using serpentine data layout. The results showed that in many cases, clever scheduling provides substantial savings

Scheduler	Number of requests	Initial latency (s)	Inter arrival time (s)	Total time (s)	Production rate (KB/s)	PCR	"Must see"
FIFO	1	72.0	–	72	161	0.86	–
	4	71.3	57.5	243	202	1.08	0.93
	16	71.3	53.8	878	216	1.15	0.87
	128	71.7	52.8	6777	220	1.17	0.85
MPScan*	1	72.0	–	72	161	0.86	–
	4	36.9	36.5	146	319	1.70	0.59
	16	21.3	20.6	330	565	3.01	0.33
	128	15.0	15.2	1945	763	4.07	0.25

(a) 1.5 Mbit/s

Scheduler	Number of requests	Initial latency (s)	Inter arrival time (s)	Total time (s)	Production rate (KB/s)	PCR	"Must see"
FIFO	1	95.5	–	96	488	0.65	–
	4	95.8	81.4	340	571	0.76	1.31
	16	94.7	77.9	1263	597	0.80	1.26
	128	95.6	76.8	9849	605	0.81	1.24
MPScan*	1	95.3	–	95	488	0.65	–
	4	66.0	59.9	245	776	1.03	0.97
	16	45.0	47.1	752	987	1.32	0.76
	128	36.0	39.2	5014	1187	1.58	0.63

(b) 6 Mbit/s

Table 8.2 Quality of Service parameters for 1.5 Mbit/s and 6 Mbit/s video streams.

in initial latency, average access times, and total execution times. As a result, we get much better utilization of the tape drives, which can give an improved Quality of Service for the users of applications that retrieve data from tapes. When used in hierarchical storage systems, the improved utilization of the tape drives can reduce the number of tape drives required and reduce the need for disk buffer, and hence reduce the total cost of the system.

Different scheduling algorithms have been implemented, and evaluated by simulations and by practical experiments on Tandberg MLR1 tape drives. Our new algorithm, MPScan*, is shown to give better retrieval performance than any of the other algorithms we have investigated with the exception of the optimal algorithm. For retrieval of small and medium sized images, we have shown that the average retrieval time can be reduced by more than 80 % for retrieval of 100 images. For retrieval of video sequences, the reduction in average retrieval time

is less, due to that the transfer rate dominates more as the size of the object increases. Although magnetic tape continues to be a rather slow random access storage medium, MPScan* provides a convenient way to make the most out of serpentine tape drives.

Three unsolved problems, which have not been considered in this chapter, are 1) the dynamic scheduling of requests which arrive during the execution of a schedule, 2) the possible mismatch between the scheduler's ordering of the requests and the user's priorities, and 3) fairness of service in case of multiple users.

Chapter 9

Simulation Model and Simulator of a Video Archive using Tertiary Storage

If you don't know what your program is supposed to do, you'd better not start writing it.

Edsger Wybe Dijkstra

This chapter presents the simulator that is used for performing the study of using tertiary storage in a video archive. During the design of a digital video archive, there will be many alternative solutions for which technologies to use, how to integrate the different technologies, and which algorithms to use. Each of these choices influence the total performance of the video archive. How the video archive is used, i.e., the load the users generate on the archive, has to be considered when designing the archive.

The architecture for the video archive we study in this thesis was presented in Section 5.1. Based on this architecture, we present the simulation model for the video archive simulator. This model shows the components used for simulating a video archive and presents the algorithms that are used for the different operations that the video archive performs. The simulator consists of two main parts. The first part is the tertiary storage system, which consists of library units containing tertiary drives and storage media. Each of the tertiary storage components used in the simulator is presented together with a performance model. The simulator has implemented detailed models for two tape drives and one DVD drive. The second part is the disk cache used for caching the most frequently used videos. Since the focus is on studying use of tertiary storage for storing and retrieving video in the video archive, we use a simple model of the disk cache. The model of the disk cache do not consider the costs and delays introduced by the video cache, but assumes it can handle both the video delivery to the users and the writing of video data transferred from the tertiary storage. The last section of this chapter contains an overview of the cost of using the alternative storage technologies.

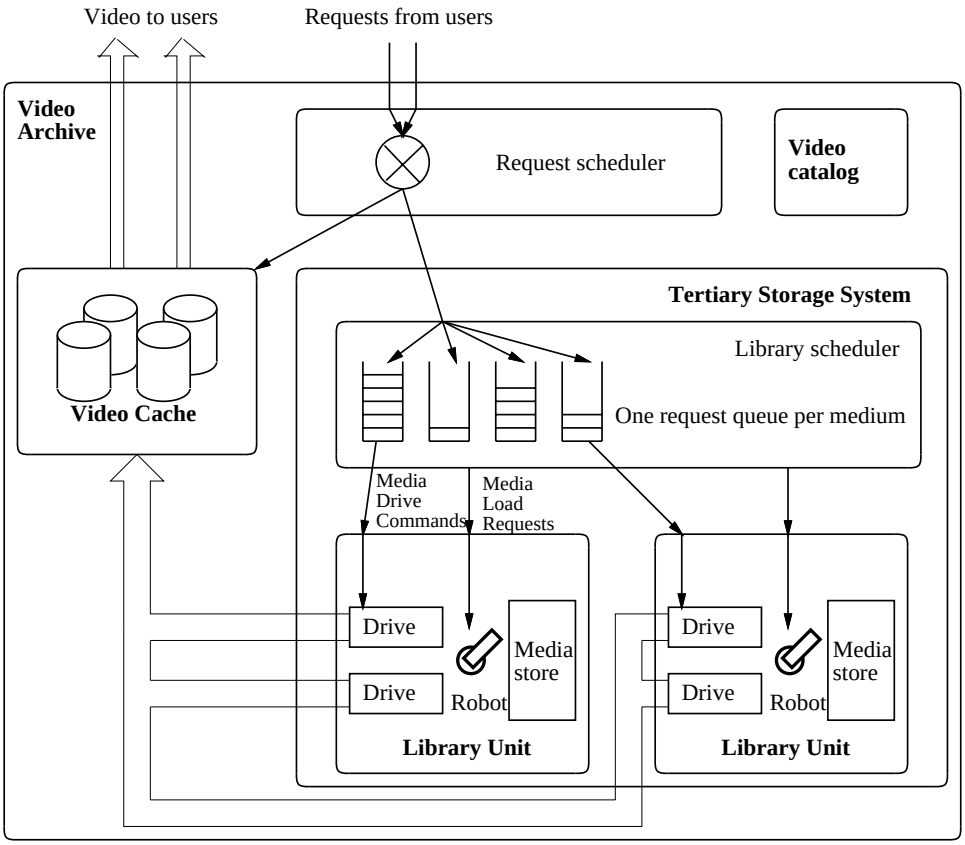


Figure 9.1 Overview of the main components included in the video archive simulator.

9.1 The Video Archive Simulator

In this section we present the simulation model used for simulating a digital video archive using tertiary storage and a disk based video cache. The simulation model is based on the architecture for the video archive presented in Section 5.1. The main components of the video archive together with algorithms and strategies used by the video archive are presented. We show how the different components are integrated and cooperate in order to execute requests for video data issued by the users.

The video archive simulator is *event driven*. It consists of simulated resources and processes, where the processes interact with each other and compete for the use of the simulated resources. Examples of processes are users that request video to be delivered, and drives and robots participating in the delivery of the video. The simulated resources are the storage media, the media drives, the robot mechanisms, and the disks used as video cache. The simulator has been imple-

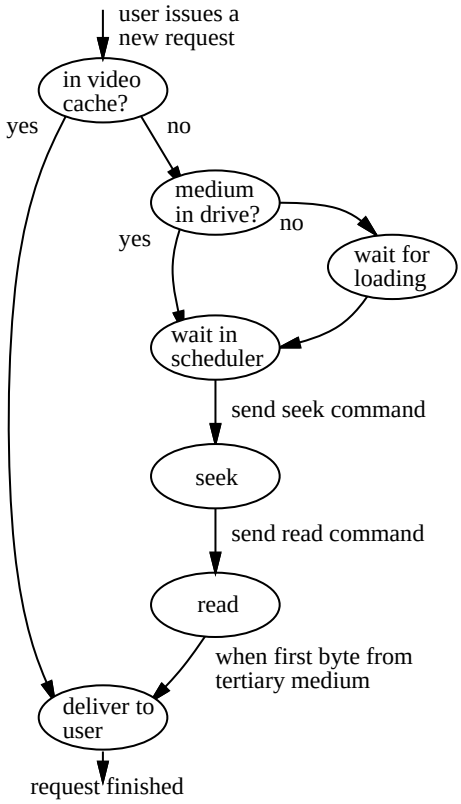


Figure 9.2 Diagram showing the states a user request for a video sequence goes through in the simulated video archive.

mented using C++ and the CSIM[1] library (Schwetman, 1996; Mesquite Software, Inc., 1994). The implemented simulator consists of approximately 15.000 lines of C++ code.

An overview of the main components of the simulator is shown in Figure 9.1. The main components are the *video archive* with a *video cache*, the *tertiary storage system*, *library units* containing a *robot*, *tertiary media drives* and *storage media*, and a *video catalog*. Each of these are presented in this section.

9.1.1 Video Archive

The *video archive* module is the main module in the simulator. It is responsible for simulating the complete video archive. Figure 9.1 shows the main components of the video archive module, which are the *request scheduler*, the *video catalog*, the *video cache* and the *tertiary storage system*.

[1]The CSIM library is a collection of objects for creating event driven simulators.

The users of the video archive issue requests for one or several video sequences. When a user requests a video sequence to be delivered, the *request scheduler* creates a *request object* that is responsible for simulating the delivery of this video sequence to the user. Figure 9.2 contains a state diagram for the possible steps and operations that have to be performed to execute this request. If the video sequence is already present in the video cache, the user is informed that the delivery of the video sequence is ready to be started. On instruction from the user, the video cache starts delivering the video to the user, using the bandwidth of the video sequence to determine the amount of time it will take to deliver this video sequence to the user.

If the requested video sequence is *not* present in the video cache, a request is issued to the *tertiary storage system* to fetch the video sequence from one of the tertiary media into the video cache. The system supports *pipelining* of the video delivery from the tertiary storage system via the video cache to the user (Ghandeharizadeh et al., 1995). As soon as the tertiary storage system starts writing the first part of the video sequence to the video cache, the video cache can start delivering the video to the user. The user is informed about this, and on request from the user, the video cache starts streaming the video to the user.

The Video Cache

The *video cache* is responsible for *delivering* the video to the users using the bandwidth of the video as the delivery rate. In a real system, the video cache would consist of a collection of machines and disks under the management of a video server. The reason for calling this a *video cache* is that the most popular video sequences should be present on disk to avoid having to retrieve them from the tertiary storage every time they are requested.

Since this study focus on use of tertiary storage for storing and retrieving video data, we use a very simple model for the video cache. The only simulation parameter is the total size of the cache. We do not consider any delays introduced by the video cache, and assume it can handle both the video delivery to the users and the writing of video data transferred from the tertiary storage without experiencing resource problems. We consider this a reasonable simplification in the simulation model of the video archive. Compared to the delays introduced by the tertiary storage system, the disk delays should be several magnitudes lower.

In a real video cache, the bandwidth of the disks would be a critical resource of the video cache. The cache consists of many disks, and since there exists several strategies for optimizing the use of the disk bandwidth and achieving load balancing between the disks, the bandwidth of the cache should be sufficient for most applications. This can be achieved by for instance use of striping (Özden et al., 1996b) or random data allocation (Santos et al., 2000). If we included the bandwidth of the video cache into the simulations, we had to make a much more detailed model of the video cache. We also had to consider data layout and disk

scheduling. This would make the simulation model more complex. The performance of the video cache would influence the performance results of the tertiary storage system. A analytical model of a disk cache used in a tertiary storage system for video is presented in (Chan and Tobagi, 1999). This model includes the delays introduced due to bandwidth contentions in the disk cache, but does not include any start-up delays due to buffering or disk scheduling strategies. The model assumes that the total aggregate disk bandwidth can be utilized for delivering the videos.

The video cache is used in Chapter 14 for studying the effect caching of the most frequently accessed video sequences has on a video archive. To illustrate how a more realistic video cache could have been designed and implemented, a more detailed model of a disk based video cache is presented in Section 14.5. This is used to evaluate some of the results and consequences of using this simple model of a video cache.

To avoid a long *warm up* period in the simulations, the disk cache is filled with video sequences before the simulation is started. By using the access distribution, video sequences are inserted into the cache until it is full. Thus when a simulation starts, the cache contains a representative set of mostly popular video sequences. To make room in the disk cache when a new video sequence is requested from the tertiary storage, one (or several) of the video sequences already present in the cache has to be removed. We use a Least Recently Used (LRU) algorithm to decide which video sequences to remove from the disk cache. Alternative cache replacement strategies for a disk based video cache are proposed and evaluated in (Ghandeharizadeh and Shahabi, 1994; Lau and Lui, 1996; Cha et al., 2002).

9.1.2 Tertiary Storage System

The module called the *tertiary storage system* simulates the part of the video archive that is responsible for storing and retrieving video sequences from the tertiary storage media. The module consists of a tertiary library scheduler and one or more library units containing the media drives and the storage media. A similar model for a tertiary storage system can be found in (Lijding, Jansen and Mullender, 2002b).

The tertiary library scheduler is responsible for managing and optimizing the use of the library units and the tertiary drives. The scheduler manages a request queue for each tertiary media that has pending requests. If an arriving request for a video sequence is the only request to this storage medium, a new request queue is created for this medium. The library scheduler also issues a load request to the library unit containing this medium. The scheduler uses a *minimum switching model* (Prabhakar et al., 1997). As soon as the requested medium is available in a drive, the library scheduler starts executing the requests in the corresponding request queue. Before the first request in the request queue is executed, the order of the requests in the queue is scheduled using a tape or disk scheduler. We

use the MPScan* scheduler presented in Chapter 7 for the Tandberg MLR1 drive, the SLTF scheduler for the Magstar drive (Hillyer and Silberschatz, 1998), and a FIFO scheduler for the DVD drive. If a new request arrives for a medium during execution of the request queue, this request is included in the request queue, and the request queue is re-scheduled. This is the same strategy as used by the *Bypass* scheduling policy (Christodoulakis et al., 1997), which gives priority to requests for media that are already loaded in a drive.

Each request is executed by sending a seek operation followed by a read operation to the drive containing the medium for this queue. If the tertiary storage system is used as a part of the video archive module, the requested video is delivered to the video cache. If no video cache is used, the video is delivered directly to the user. As soon as the drive starts delivering the video to the video cache or directly to the user, the user application is informed about that the video is ready for playback. In both cases, the delivery rate of the video data is the same as the transfer rate for the drive. When the last request in a request queue for a media is finished, a release operation is sent to the library unit for this medium.

The requests for loading media into drives are issued in a FIFO order. We do no scheduling or optimization of the order for requesting a medium to be loaded into a drive. Alternative scheduling strategies for scheduling the order of switching media in the drives were presented in Section 4.5.2.

9.1.3 Library Units

The *library unit* module simulates the behavior of a physical library unit. An overview of the components of a library unit that are included in the simulator, is given in Figure 9.3. It consists of a media store capable of storing a number of storage media, one or several media drives, and a robot mechanism moving media between the media store and the drives. The performance data for the robot used in the simulations is presented in Section 9.3.2. In this subsection we present how the simulated library units execute requests. Since we did not have the opportunity to perform experiments on real library units, we have implemented the library unit to perform close to what we would expect an ideal library unit to perform. We have implemented some optimizations that can improve the performance, all of which easily could have been implemented in real library units.

The library unit receives *load* and *release* requests from the *tertiary library scheduler*. The requests for loading a medium is mostly executed in a FIFO order, but in order to possibly reduce the number of media transfers, before a load request is queued in the FIFO queue, we check if the requested media already is in one of the drives and not used by any external process. If that is the case, the load request is fulfilled immediately. Likewise, every time the tertiary library scheduler issues a release request for a media, if there is a load request for the same media, this request is granted immediate access to the drive. This corresponds to using the *Bypass* scheduling strategy (Christodoulakis et al., 1997).

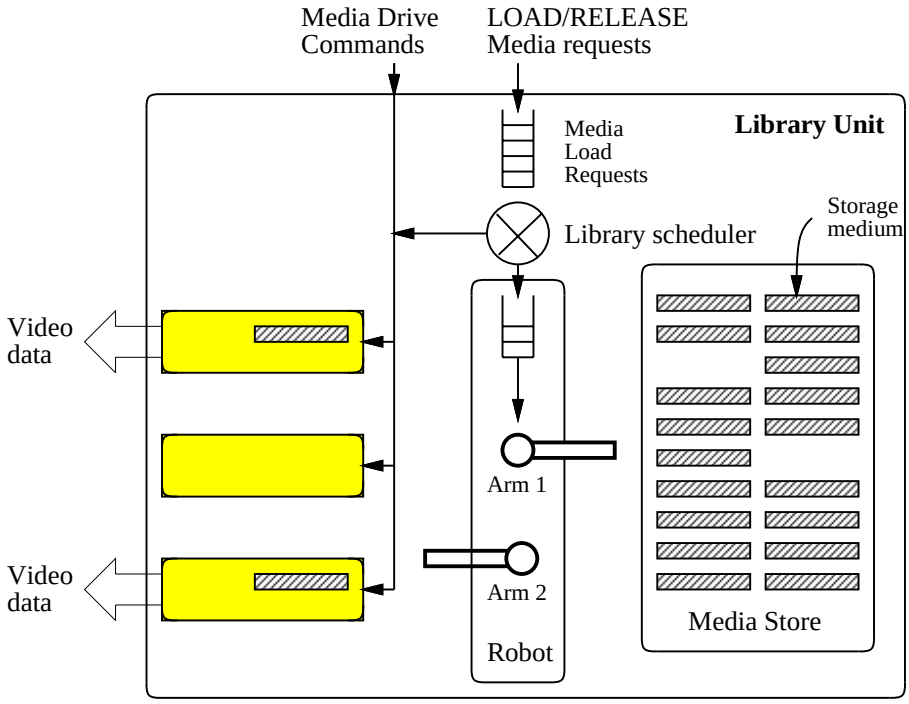


Figure 9.3 The main components included in the simulation module for a library unit.

When all the requests for a medium has been performed, and the library scheduler has issued a *release* command for this medium, one of four strategies for what to do with the medium can be followed:

1. Do nothing, just mark the drive as idle.
2. Rewind the medium.
3. Rewind and eject the medium.
4. Rewind, eject, and return the medium to the media store.

Which of these strategies to use, depends on how likely it is that the medium already in the drive will be reused before another medium will need the drive. If otherwise not stated, the last of these strategies is used. For most applications, it is not likely that the same medium will be needed before the drive has to be reused by another medium (Lijding, 2003). To reduce the response time when the next medium is to be loaded into the drive, we restore the medium back into the media store as soon as possible after the user has finished the use of it.

Robot

Most library units have a robot device that is capable of transferring only one medium at the time, although there also exists robot devices that can perform multiple media transfers concurrently. To make the simulator flexible enough to simulate library units with multiple transfer mechanisms, the robot device used by the simulator can be instantiated to have as many robot arms as desired (see Figure 9.3). For most applications, the robot device will not be a bottleneck. Still, having more than one robot arm can improve the performance. If the robot has two robot arms, these can cooperate when replacing a media in a drive. The first arm can do the job of removing the currently loaded medium from the drive, while the second arm fetches the requested medium from the store and inserts it into the drive.

The load operation is simulated as one move operations — move the arm from the current position to the media store, grab the medium, move it to the drive and insert the medium in the drive (possibly waiting if the drive contains a medium, which has to be unloaded first by another robot arm). The unload operation has to be divided into two parts. The first part of the operation moves the arm to the drive and waits there until the medium is fully ejected. In the simulations this operation takes 25 percent of the total unload time as given in Table 9.6. In the second part of the unload operation, the medium is grabbed and returned to the media store.

To optimize the use of the robot mechanism when there are several concurrent load/unload operations waiting to be performed, the robot uses the following strategies:

- The simulator can estimate rewind and eject times for the drives by using the performance models of the tertiary media drives in Section 9.3.1. It also knows the load/unload times of the robot mechanism. This knowledge is used to schedule the order the robot performs the operations in case of multiple concurrent operations. For example, if the robot is requested to unload the tape from two drives, where the first drive is in the process of ejecting the tape, while the second is still rewinding the tape, the robot mechanism should serve the first drive first. The simulator uses the estimated time for when a drive has ejected the medium to schedule the order of the requests for unloading and loading media.

- Load operations has preference over unload operations since a load operation means that some user is waiting for data.

These strategies will improve the usage of the robot, and may lead to higher performance of the library unit. If actual library units do not make such optimizations, it would still be possible to achieve the same optimizations. The program using the library could perform this scheduling of the requests before it sends

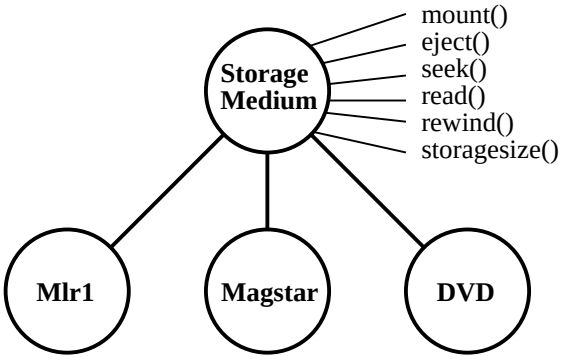


Figure 9.4 The media types implemented in the simulator.

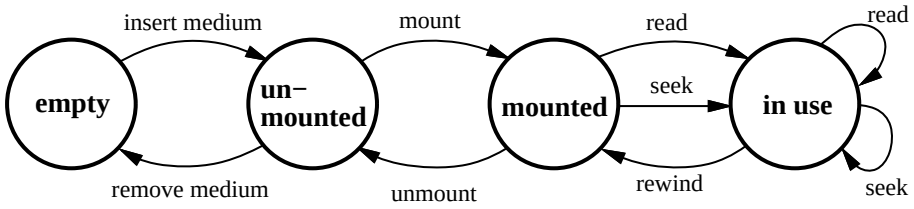


Figure 9.5 State diagram for the drives.

the requests to the library unit. An analytic model for a robot used in a tertiary jukebox is presented in (Lijding, Mullender and Jansen, 2002c). This model uses detailed measurements of the behavior of a real tertiary jukebox when estimating the amount of time for the different operations.

9.1.4 Drives and Media

The simulator uses a *generic drive* common for all the media types. The media specific properties are implemented in the individual *media types*. The advantages of this design, are that all media specific behavior are implemented in one module, and makes it easy to add new media types to the simulator without having to do changes to the media drive's behavior.

The media types implemented in the simulator are presented in Figure 9.4. They are implemented as subclasses of a base class named *Storage Medium*. To add a new media type to the simulator, all that is needed is to make a new subclass that implements the member functions of the base class. These member functions provide information about how much data each medium can store and information about the amount of time operations on this medium take. For the mount and unmount operations, we use the time values given in Table 9.3. For MLR1 media we use the access time model presented in Chapter 6 (Sandstå and Midtstraum, 1999b) to compute seek, read, and rewind times. The model is in-

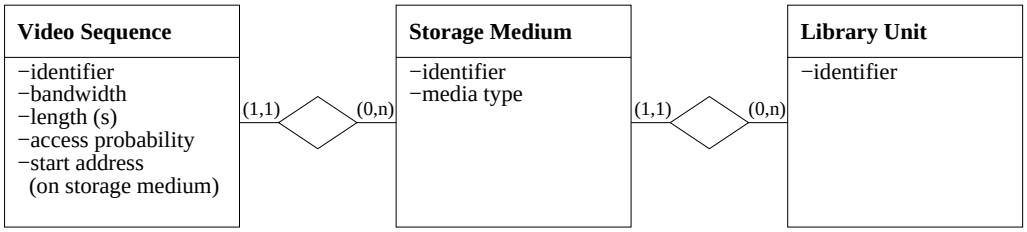


Figure 9.6 An ER model of the main content of the video catalog.

strumented using information about the start address of each track on a real tape. For Magstar we use the model in (Hillyer and Silberschatz, 1998) to estimate seek and rewind times and the given data rate of the drive to compute read times. We use a list of key points of a real tape to instrument the model[2]. For DVD we use the model in (Shastri et al., 1997) to compute seek and rewind times. Read times are computed using the data rate of a double speed DVD drive. More detailed models for the performance of the tertiary drives are presented in Section 9.3.1.

In Figure 9.5, a state diagram is given for the operations the media drive performs. All state transitions are triggered by external commands issued to the drive. Each state transition takes a given amount of time. These times are found using the member functions of the medium used in the drive. All operations on the drives are performed in the order of arrival, i.e., no internal scheduling is performed by the drive.

9.1.5 Video Catalog

The video archive stores video sequences, which can be from a few seconds up to a few hours. When a user issues a request, it will be for one or more video sequences. To keep track of the video sequences and where they are stored, the simulator contains a video catalog. Each video sequence is identified by a unique identifier. For each video sequence we store information about the bandwidth of the video, the length of the video sequence, and the access probability. In order to locate where a video sequence is stored, the video catalog also contains information about the storage medium each video sequence is stored on, the start address for the video sequence on the medium, and the number of data blocks required to store it. A data model for the information contained in the video catalog is presented in Figure 9.6.

Before a simulation starts, the video sequences are "created" and registered in the video catalog. They are also assigned to a tertiary storage medium. In the first parts of this study, the video sequences are assigned to storage media in the same order as they are created, thus two following video sequences are likely to be stored on the same storage medium. Later, we perform simulations

[2]The list of key points was provided by Bruce Hillyer.

where we study alternative allocation strategies. These allocation strategies are presented in Section 13.1. Each video sequence is stored entirely on one storage medium. If there is not room for it on the current storage medium, a new storage medium is created. The simulator does not support any form of replication of the video or complex storage layout schemes like media striping (Drapeau and Katz, 1993a; Drapeau and Katz, 1993b; Chervenak, 1994).

The storage media are assigned to a library unit. We have not included any mechanisms for exchanging media between library units, so when a medium is assigned to a library unit it stays in this library unit for the entire simulation. If nothing else is specified, a round-robin algorithm is used when assigning the media to the library units. Alternative strategies for assigning storage media to library units are presented in Section 13.1.

If we want to have different probabilities for accessing the video sequences, we assign an access probability to each video sequence. These access probabilities are used when the simulated users select which video sequence they want to watch. During simulations, the video catalog is used to select which video sequences to retrieve, for locating where they are stored, and determining which data rate should be used when sending the video to the user.

9.1.6 Users

The simulation load will be created by having simulated users sending requests to the archive. The archive can have any number of users, and each user can issue several concurrent requests. The behavior of the users will depend on the purpose of the simulation. To generate the workloads for the simulations, two different request generation strategies are used:

- **Closed Queueing Model.** The first strategy is to use a closed queueing simulation model (Jain, 1991, Chapter 32). It maintains a constant length queue of requests. As soon as one request is completed, a new request is generated and inserted into the queue. This strategy simulates a fixed number of I/O bound users/processes. It is a useful simulation strategy for obtaining information about the behavior of the video archive during maximum load. This simulation strategy is used for determining the maximum throughput of the system.

- **Open Queueing Model.** In order to get response time information as a function of the load on the tertiary system, we use an open queueing simulation model (Jain, 1991, Chapter 32). The arrival rate of requests is generated using a Poisson process where the interarrival time between requests is drawn from an exponential distribution. This simulation model gives a more realistic workload model of users of a video archive. It simulates a large number of users sending infrequent requests to the archive.

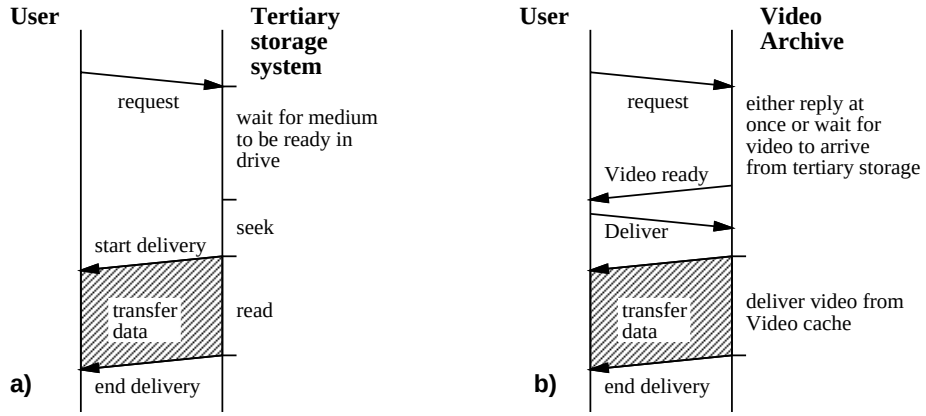


Figure 9.7 Interaction between users and **a)** the tertiary storage system, and **b)** the video archive.

In the simulations, we use either the *video archive* module (including the video cache) or just the *tertiary storage system* module. Depending on which of these being used, the user interaction and video delivery differs. Figure 9.7 shows the interaction between the user and the simulator for the two cases.

The main difference between these two cases is in how the video is delivered to the user. When only the tertiary storage system is used (see Figure 9.7(a)), the video delivery starts as soon as the tertiary storage system is ready to deliver the video. The video is delivered using the transfer speed of the tertiary drive. When using the video archive module (see Figure 9.7(b)), the video is always delivered from the video cache. After having issued a *request* for a video sequence, the user receives a notification as soon as the video is ready to be delivered. When the user issues a *deliver* command, the video is delivered to the user at the data rate specified for the video sequence.

9.2 Using the Video Archive Simulator

The video archive simulator is used in the following chapters for studying properties and performance of different video archive configurations. In this section, we give an overview of how the simulator is used for running the simulations. The implemented simulator exists in two versions:

- **librarysim:** This implements a simulator for the tertiary storage system. This uses the *tertiary storage system* module. No video cache is included. The videos are delivered directly to the user at the speed of the tertiary drive.
- **archsim:** In addition to simulating the tertiary storage system, this version includes *video archive* module and the video cache in the simulator.

Parameter	Description
hours	The number of hours of video.
length	The length of each video sequence.
bit rate	The bit rate of the video sequences.
drives	Number of tertiary drives per library unit.
technology	MLR1, Magstar, DVD or MLR3.
cache size	The size of the disk cache (in GB).
users[a]	The number of concurrent users.
requests[a]	The number of arriving requests per hour.
partitions	The number of partitions the video sequences should be divided into (see Section 12.1).
allocation	Storage allocation strategy (see Table 13.1).
seed	Initial seed for the simulator.

[a]Only one of *users* and *requests* may be specified for a simulation.

Table 9.1 Parameters for configuration of the simulator.

These two programs take a set of parameters for configuring the video content, the storage system configuration and the user load. An overview of the main parameters are presented in Table 9.1.

In the rest of this study, we use these two simulators for most of the experiments. In total, the thesis contains the results from approximately 70,000 runs of these simulators. We estimate having used more than 20,000 CPU hours for running the simulations. To produce the results, the simulators are used in three different ways. We describe these briefly here.

Simulation Method I

The simulator takes a video archive configuration and a user load as the parameters. Thus, *one* run of the simulator gives information about the performance of the video archive with the given load. To achieve statistical validity we use the confidence interval of the simulated response time as termination criterion for the simulation.

If otherwise not stated, statistical validity is ensured by verifying that the 90 % confidence interval is sufficiently tight. This is done by running the simulations until the size of the confidence interval is within 1% of the mean. The confidence intervals are computed using confidence interval functions provided by CSIM. These functions use a technique called *batch mean analysis* for computing confidence intervals (Schwetman and Brumfield, 1997; Jain, 1991, Chapter 25).

Simulation Method II

If the accesses to the stored video sequences are not uniform, the simulation results from running the simulator *one* time will be dependent of the initial allocation of the video sequences to storage media. If we have a *random* allocation of the video sequences to storage media, two following runs of the simulator will produce different results. To avoid that this influences the validity of the simulation results we perform multiple runs of the simulator.

The following pseudo code shows how this is done:

```
do {
  run simulator (random seed);
  compute mean values;
  compute confidence interval;
  numberOfRuns++;
} while (numberOfRuns < MinNumberOfRuns and
         confidence interval > requested confidence interval);
```

This ensures that the simulation results are based on a minimum number of runs where the simulator has a different allocation of the video sequences to storage media in each run. Each run of the simulator produces results that are statically valid for one allocation. After each run of the simulator, we compute the confidence interval for the mean of the average response time for all simulations. This is used as the termination criterion for when to stop the simulations.

In the simulations performed in this thesis we use a 90 % confidence interval for terminating the simulations. For running the simulator program itself, we terminate the simulator when the confidence interval is within 2 % of the mean (see Simulation Method I). For terminating the while loop we require that the confidence interval is within 3 % of the mean of all the results from the individual simulations. The simulator is run minimum ten times to ensure that we have enough data samples to compute the confidence interval.

Simulation Method III

The simulator takes the user load as in parameter. This load is given as the number of concurrent requests or as an arrival rate for the requests. As the main result from running the simulator we get the average response time for this user load. In some of the experiments we want to do the opposite. We want to give an average response time as in parameter and get the corresponding load as a result. This will be used for determining the throughput of the system.

To use the two simulation programs for this purpose, we use a binary search strategy to find the load that produces the given response time. Pseudo code for this strategy is given in Figure 9.8. It takes the response time limit as in parameter. For each step in the binary search we use the simulator for finding the average response time for the current throughput. The binary search continues until we

```
minRequests = 1;       // Search interval - lower bound
maxRequests = 20000;   // Upper bound

while (maxRequests > minRequests) {
  throughput = (maxRequests-minRequests)/2;
  responseTime = run_simulator(throughput);
  if (responseTime < ResponseTimeLimit){
    minRequests=throughput;
  }
  else{
    maxRequests=throughput;
  }
}
```

Figure 9.8 Pseudo code for determining the throughput of the archive for a given response time limit.

have found the throughput that gives an average response time closest to the given response time limit.

It is important to note that the function `run_simulator` either uses simulation method I or simulation method II to find the average response time.

9.2.1 Statistics

The simulator collects statistics about the performance of the video archive. Table 9.2 presents the main statistical values which are collected by the simulator. For all time values, the minimum, maximum, average, variance and standard deviation is collected. Most of the gathering of statistical data during runs with the simulator is done by using statistical functions from the CSIM library.

An example of the output from one run of the archive simulator is given in figure 9.9. It shows the results from running the simulator with a video archive containing 10000 hours of 5 Mbit/s video and where the length of each video is 60 seconds. Each library unit contains four MLR1 drives, and a 1000 GB video cache is used. The load on the system is 500 requests per hour. In this run of the simulator, the average response time was found to be 16.7 seconds.

9.3 Performance Models for Drives and Library Units

In this section, we present performance models for the tertiary devices used in the simulations. To compare the performance between a DVD based and a tape based archive, we use one *generic* DVD drive and two different tape drives. The reason for using two different tape drives is that the performance vary much between

Module	Statistical values	
Video Archive:	Response time	Time from the user issues a request until the video cache is ready to start deliver it to the user.
	Throughput	Number of requests fulfilled per hour.
	Simulation time	Total simulated time.
Video Cache:	Hit rate	
Tertiary Storage System:	Response time	Time from the user issues a request until the first part is delivered to the video cache (or to the user).
	Access time	Time from the user issues a request until the entire video sequence is read from the tertiary media.
	Throughput	Number of requests fulfilled per hour.
Tertiary Library Units:	Operations	Number of robot operations.
	Wait queue	Average length of wait queue for robot operations.
	Utilization	Robot utilization.
Tertiary Media Drives:	Operations	Number of operations for each drive.
	Wait queue	Average length of wait queue.
	Utilization	Utilization for each drive.

Table 9.2 The main statistics gathered by the simulator.

```
staude% archsim -h 10000 -l 60 -b 5000 -d 4 -c 1000 -r 500
-m MLR1 -p 3 -A 7
# Media type       : MLR1
# Number libraries : 9
# Drives per lib   : 4
# Loaders per lib  : 1
# Partitions       : 3
# Allocation method: Allocate_Uniform
# Storage media    : 1730
# Datavolume       : 20965
# Cache size       : 1000
# Load: requst/hour: 500
# Request length   : 60
# Simulation time  : 1.12372e+06
# Number of samples: 156600
# Requests per hour: 501.692
# Datarate         : 4.98643 MB/s
# Average wait time: 16.7314    (156600)
# => Tertiary      : 98.5683    (26582)
# Drive utilization: 0.126156
# Robot utilization: 0.0209938
# Robot oper/hour  : 170.05
# Cache hit percent: 83.0224
# Storage cost     : 277699
# Cost per GB      : 13.2458
# Cost per stream/h: 553.525
# Wait time library: 98.5683
# Access time Libr : 124.023
# Run length accur : 0.0099987
```

Figure 9.9 An example output from one run of the archive simulator. The input parameters to the simulator are presented in Table 9.1. The detailed report generated by the CSIM library is not included.

different tape drives. This is mainly due to the design of the drive and the technology used by the drive. The first tape drive is the Tandberg MLR1 (Tandberg Data, 1996). This is the same drive as used in Chapter 6 to 8 when investigating serpentine tape as a storage medium for multimedia data. Based on the performance parameters, this drive can be characterized as a *traditional* tape drive. The goal has been on optimizing storage capacity and transfer bandwidth. The second drive used in this study, is the IBM Magstar MP. This drive has been designed to be used in a tertiary library system with frequent changes of the tapes. The goal has been to reduce the tape switch time.

Tertiary library units consists of drives, a robot mechanism and a media store. The media store stores the media which are not currently mounted in one of the drives. The robot mechanism is responsible for moving media between the media store and the drives. To simulate the robot mechanism, we use performance specification for a *generic* tertiary library unit. The performance parameters of this library unit are presented later in the section.

A performance model must include information about the main operations the storage devices perform, how these operations are related to each other, and the resources and the amount of time each operation takes. In the simulations each operation is given a cost, i.e., the amount of time it takes to perform the operation. We call the corresponding simulation parameter the performance parameter for the operation. For requests handled by the tertiary storage system, the following operations (performance parameters) will influence the time to access a video sequence:

1. **Mount time.** When a new medium is inserted into a drive, the drive has to mount the medium.

2. **Unmount time.** The time used by the drive to unmount and eject a medium. This does not include rewind of the medium if that should be necessary.

3. **Seek time.** The amount of time the drive uses to seek to the beginning of the requested video sequence. This takes typically a few hundred milliseconds for a DVD drive, and from a few seconds up to a few minutes for tape drives.

4. **Transfer time.** The time a drive uses to read the video sequence and deliver it to the requesting entity (the user or the disk cache).

5. **Rewind time.** Before a storage medium can be unmounted from the drive, the drive's read/write head must be repositioned to a position where it is safe to remove the media from the drive. In the case of tape, the tape has to be wound back to either the start of the tape, or to a load position. In the case of DVD, the drive's read head is usually moved to a parking position. We use the seek time for seeking back to the start position as an estimate for the rewind time.

	Tandberg MLR1	IBM 3570 Magstar MP, model B	Generic DVD-ROM
Capacity per media	12.1 GB (13 Gbytes)	4.6 GB (5 Gbytes)	4.38 GB (4.7 Gbytes)
Transfer rate	1.41 MB/s (1.5 Mbytes/s)	2.1 MB/s (2.2 Mbytes/s)	2.64 MB/s (2.77 Mbytes)
Seek time	1–125 s	0–42 s	100–436 ms
Time to mount media	avg. 31 s	avg. 5.6 s	avg. 3 s
Time to unmount media	avg. 5 s	avg. 4.0 s	avg. 2 s

Table 9.3 Values for the main performance parameters for the drives used in the simulations.

6. **Robot transfer time.** If the medium storing the requested video sequence is not in a media drive, it has to be fetched from the media store and inserted into a drive. If the drive contains a medium, this medium has to be unmounted and returned to the store before a new medium can be inserted into the drive. The two main tasks of the robot are:

   (a) **Load medium.** The robot transfers a storage medium from the media store to the drive. This operation is often called the *pick* operation.

   (b) **Unload medium.** The robot transfers a storage medium from the drive back to the media store. This operation is often called the *place* operation.

   For most tertiary robots these operations take from a few seconds up to a half minute.

9.3.1 Performance Model for Tertiary Drives

This subsection presents the performance parameters for the three media drives used in the simulations. The values for the main performance parameters are given in Table 9.3. We use fixed values for performance parameters except for the seek time. Using a constant for the transfer rate is justified by the fact that all the drives read the media with a constant data rate. The load and unload operations are mostly mechanical operations. Johnson and Miller (1998b) and Chervenak (1994) present measurements of load and unload times for tape drives, which are shown to be nearly deterministic with a low variance between measurements. This justifies using constants for load and unload times in the simulations. Since seek times vary much depending on the start and end positions for the seek, we use a seek time model for each of the three drives. Table 9.4 contains minimum,

	Tandberg MLR1	IBM Magstar MP	DVD drive
Minimum	0.81	0.0	0.100
Mean	63.0	12.3	0.267
Maximum	125	26.1	0.436

(a) Seek times for seeks from start of medium

	Tandberg MLR1	IBM Magstar MP	DVD drive
Minimum	0.81	0.0	0.100
Mean	44.0	15.0	0.211
Maximum	126	42.2	0.436

(b) Seek times for random seeks

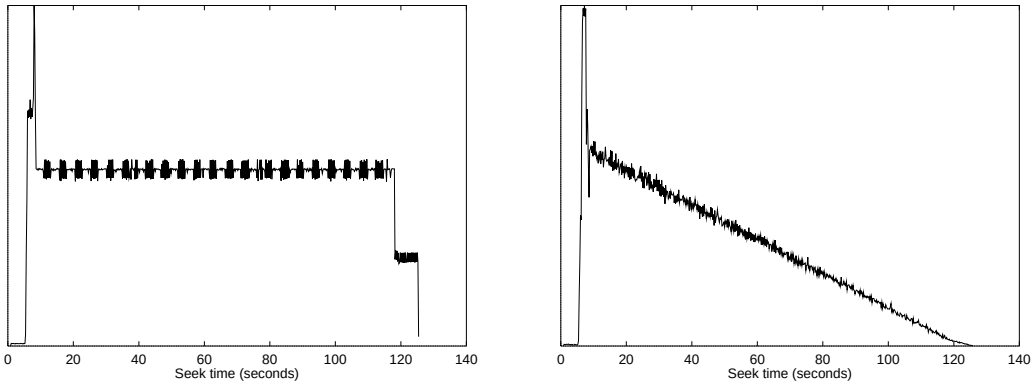
Table 9.4 Seek time characteristics for the three drives used in the simulations. The values are computed using the seek time model for each of the drives. All numbers are given in seconds.

maximum and the mean seek time for seeks starting at address zero, and for random seeks for the three drives. The details of the seek time models will be presented in the following sections.

Tandberg MLR1

The Tandberg MLR1 (Tandberg Data, 1996) tape drive uses serpentine data layout. It is based on the 13 GB QIC standard (QIC Development Standard, 1994), making it possible to store 12.1 GB (13 Gbytes) per tape (without compression). According to the drive specifications, it can deliver (read/write) a maximum sustained data rate of 1.4 MB/s (1.5 Mbytes/s) to/from the host computer (Tandberg Data, 1996). To estimate seek and transfer times, we use the access-time model developed in Chapter 6. The access-time model is instrumented with track addresses from one of the MLR1 tapes used when validating the access-time model. For the first seek after a tape is inserted into the drive, the seek time vary between 0.8 seconds and 125 seconds, with an average of 63 seconds (see Table 9.4). In Figure 9.10 we have plotted the distribution of seek times for seeks from the start of the tape and for seeks between two random positions on the tape.

To improve the performance of the drive in cases where there are multiple requests for video sequences on the same tape, we use the MPScan* scheduler to optimize the retrieval order. The MPScan* scheduler was presented in Chapter 7.



(a) From start of tape

(b) Random seeks

Figure 9.10 Distribution of seek times for the Tandberg MLR1 drive. The seek times are computed using the model in Chapter 6. The resolution along the X-axis is 200 ms.

The mount and unmount times for Tandberg MLR1 given in Table 9.3, are found by measuring a series of mount and unmount operations performed on the drive, and then computing the average value. For the mount operation we measured the time from we manually inserted the tape into the drive until the drive accepted the first SCSI command. For the unmount operation we measured the time from we issued an eject command until the tape was fully ejected.

IBM 3570 Magstar MP

The IBM 3570 Magstar MP tape drive is optimized for random I/O. The physical data layout on the tape is designed to give short seek and rewind times in order to reduce the switch time when used in a tape library. As shown in Figure 9.11, it uses a modified serpentine pattern for the data tracks on the tape. When the tape is loaded in the drive, the read/write head is positioned on the middle of the tape. Thus, the average seek time for the first access is reduced by fifty percent.

In this study we use the Magstar model made by Hillyer and Silberschatz (1998) to estimate seek times for this drive. They have used the IBM Magstar MP model B in their study. This drive stores 4.6 GB (5 Gbytes) on each tape and has a transfer rate of 2.1 MB/s (2.2 Mbytes/s). There exists a faster version of the drive, model C, capable of storing 4.6 GB per tape and having a transfer rate of 6.0 Mbytes/s.

In the simulator we use the same source code[3] as used in (Hillyer and Silber-

[3]The source code used for estimating seek times on the Magstar drive has been provided to us by Bruce Hillyer.

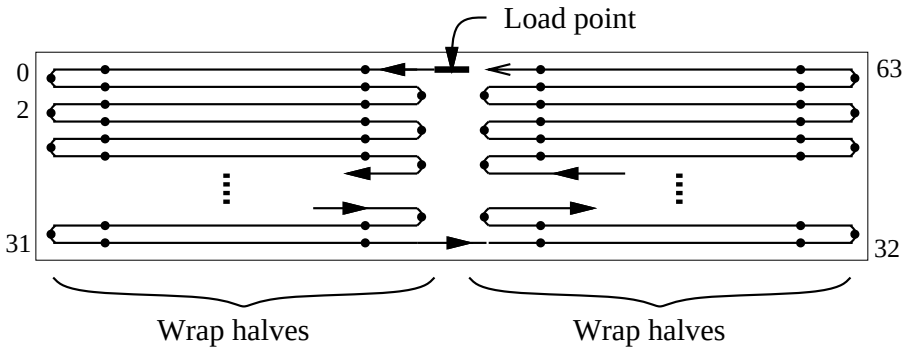
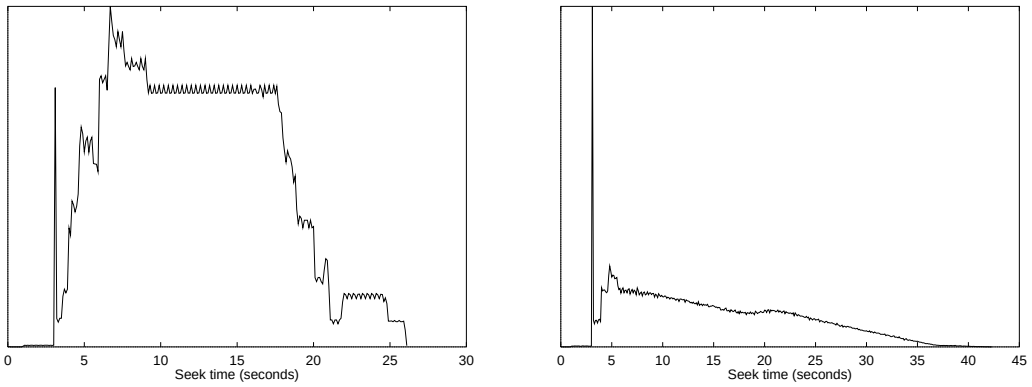


Figure 9.11 The serpentine layout pattern of the IBM Magstar MP drive. This figure is based on the presentation of the drive given in (Hillyer and Silberschatz, 1998). The *dots* marked along the tracks indicate the key points used in the seek time model.



(a) From start of tape

(b) Random seeks

Figure 9.12 Distribution of seek times for the IBM Magstar MP drive. The seek times are computed using the model in (Hillyer and Silberschatz, 1998). The resolution along the X-axis is 100 ms.

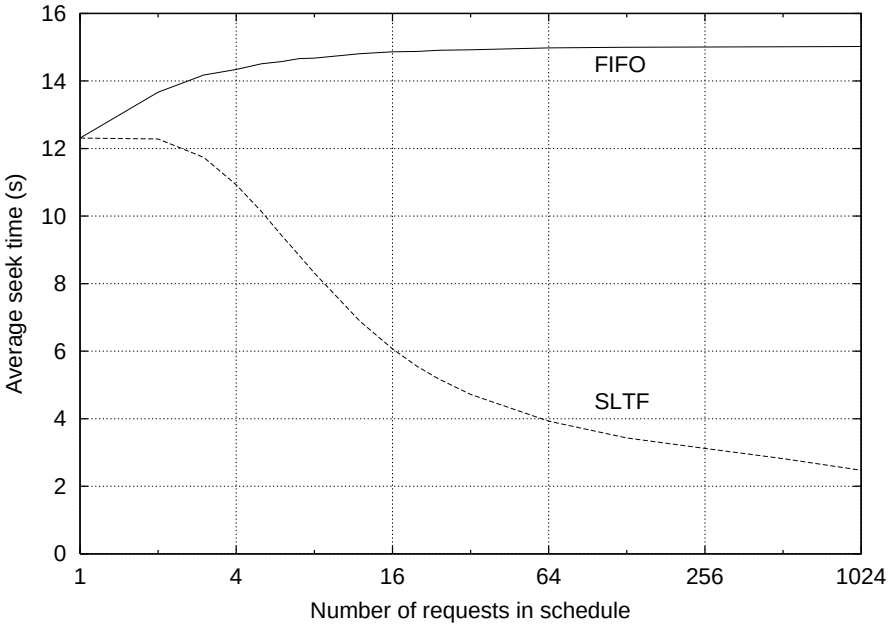


Figure 9.13 The average seek times for the Magstar MP drive when using the FIFO and SLTF schedulers for reorganizing multiple requests.

schatz, 1998) to estimate seek times. The Magstar drive supports compression, and in the experiments done by Hillyer and Silberschatz, they used slightly compressed data on the tape. All data was compressed to 7/8 of the original size, and thus they were able to store more than 4.6 GB on each tape. Since video data usually is compressed in advance, we compensate for the built in compression in the model (and source code) to get the storage capacity and seek times the drive would give if compression was turned off. Figure 9.12 presents the distribution of seek times we get when using the model without any kind of scheduling of the requests. By comparing this figure to the similar figure for the Tandberg MLR1 (see Figure 9.10), it can be seen that the seek times for the Magstar drive is about one fifth of the seek times of the MLR1 drive.

To improve the performance of the Magstar drive, Hillyer and Silberschatz (1998) studied several scheduling algorithms. In our simulations, we use the SLTF scheduler when there are multiple requests for data on one tape. Figure 9.13 contains average seek times for the Magstar MP drive using the SLTF scheduler to determine the retrieval order for different request list sizes. We have also included the seek time we would experience if no scheduler was used (i.e., performing the requests in FIFO order).

The values for mount and unmount times for the Magstar MP drive given in Table 9.3 are found in (Strategic Research, 1996).

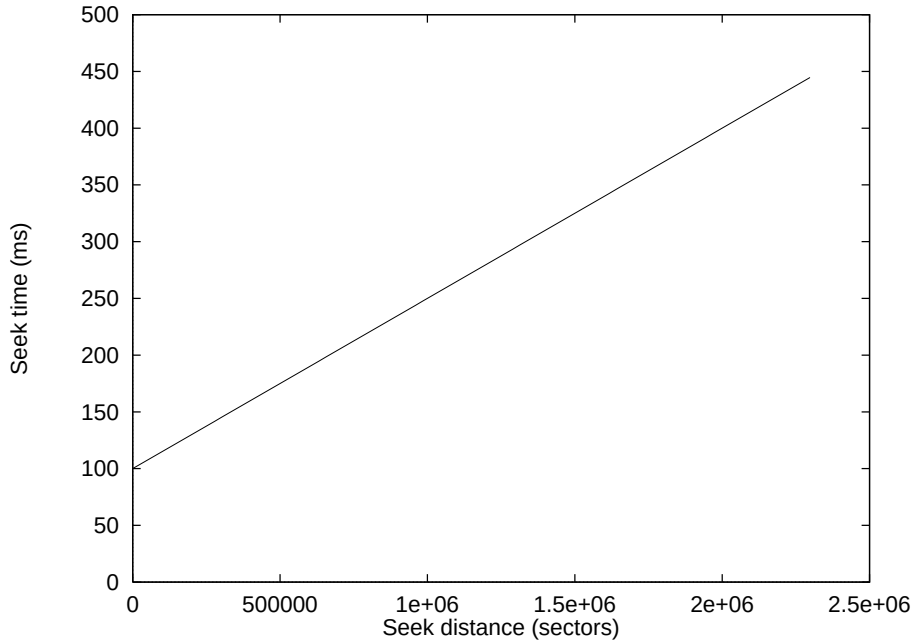


Figure 9.14 Model for seek times for the DVD drive used in the simulations. The model is based on the DVD model in (Shastri et al., 1997).

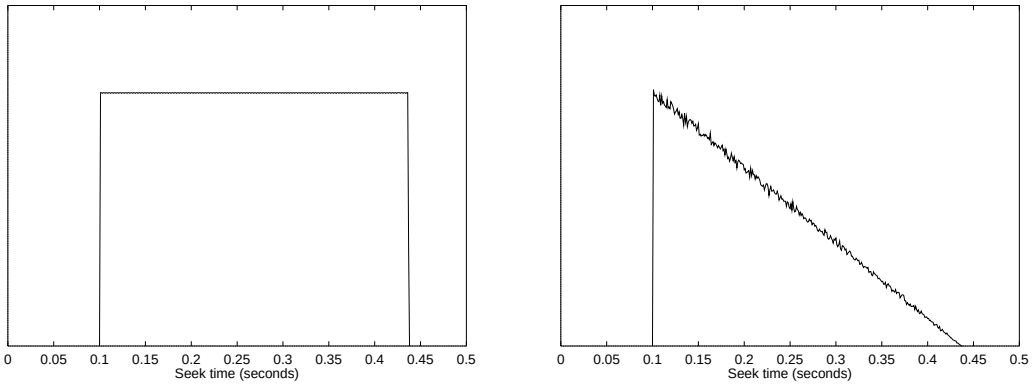
DVD drive

The DVD technology was presented in Section 3.3.1, with storage capacities ranging from 4.38 GB to 15.9 GB. In our simulations, we use the standard storage capacity for a single-sided, single-layer DVD. Thus, each DVD medium is able to store 4.38 GB (4.7 Gbytes) of video data. In the simulations, we assume that only one side of the DVD disks is used for data storage.

To estimate response times for the DVD drive, we use a generic access time model like the one used in (Shastri et al., 1997). This access-time model was originally developed for modeling seek times for a CD-ROM drive (Shastri et al., 1996a). Since CD-ROM and DVD drives basically use the same technology for seeking on a disc, this model was later used to estimating seek times for DVD-ROM drives (Shastri et al., 1997). The model estimates the seek time by the following simple formula (Shastri et al., 1997):

$$t_s = \alpha d_s + \beta$$

where d_s is the number of sectors between the start and end positions, and α and β are constants. As can be seen from this formula, the estimated seek time is a linear function of the number of sectors between the start and end positions. This is not in correspondence with most models for magnetic disks, where the seek time



(a) From start of media

(b) Random seeks

Figure 9.15 Distribution of seek times for the DVD drive. The seek times are computed using the model in (Shastri et al., 1997). The resolution along the X-axis is 1 ms.

is a linear function of the number of tracks the disk head has to pass (Ruemmler and Wilkes, 1994).

For the constants α and β used in the model, Shastri et al. (1997) use the values of 0.0012 ms/sector and 100 ms respectively. This is the same values as used in their CD-ROM model (Shastri et al., 1996a). The estimated value for the constant α used in (Shastri et al., 1997) is based on a disc having approximately 300,000 sectors (i.e., a CD-ROM). It is obviously wrong to reuse the value for the constant α found by using a CD-ROM without adjusting it to the much higher number of sectors on a DVD. A 4.38 GB DVD disc has approximately 2,300,000 sectors each containing 2048 bytes of data. To get results valid for a DVD drive, we use the number of sectors for a DVD disc for estimating the constant. With 2,300,000 sectors, the constant α gets the value of 0.00015 ms/sector. With this new value for α, the seek time for a seek across a given number of sectors are presented in Figure 9.14. The distribution of seek times for seeks starting on the beginning of the disk and for random seeks are given in Figure 9.15.

The standard transfer rate for a DVD-ROM drive is 1.3 MB/s. In the simulations we assume we have a drive capable of reading the disc at twice the normal speed, giving us a transfer rate of 2.6 MB/s. We use this transfer rate for computing the transfer time of a video sequence.

9.3.2 Performance Data for Tertiary Library Units

To build a tertiary storage system containing more storage media than drives, the common way is to integrate the drives and the storage media in a tertiary

	Tandberg TDS 1440	IBM Magstar MP 3575	Pioneer DVD-ROM Jukebox
Number of media	10–40	60–324	100
Total storage capacity (GB)	121–484	276–1490	438
Number of drives	1–4	2–6	1–4
Maximum bandwidth (MB/s)	1.5–6.0	4.4–13.2	2.6–10.4
Average load medium	avg. 9.9 s	< 4 s	< 3 s
Average unload medium	avg. 9.9 s	< 4 s	< 3 s

Table 9.5 Technical data for three representative tertiary library units using the three storage media used in this part of the thesis. The data for the Tandberg TDS 1440 is found in (Tandberg Data, 1997). For the IBM MP 3575 and the Pioneer DVD-ROM Jukebox, the information is found on the respective companies' product presentations on the World Wide Web.

Generic Library Unit	Tape	DVD
Number of drives	1–32	1–32
Maximum number of media	200	200
Average time to load medium	4 s	3 s
Average time to unload medium	4 s	3 s

Table 9.6 Performance data for the generic library units used in the simulations.

library unit. In this section we present the technical performance data for the library units we use in our simulations. Initially, we planned to use performance data for actual library units for the three drives presented earlier. Data for three such library units are given in Table 9.5. As we can see from this table, the data vary rather much between the different models. Most of the differences in the technical specifications are not a result of the storage technology used, but due to mechanical design decisions made by the producer of the library unit. The purpose of these experiments is to compare different storage technologies. To avoid that design decisions of the library units influence the results, we use a *generic library unit* instead of actual library units.

The performance data for the generic library unit we use in the simulations is presented in Table 9.6. For the DVD library unit, we use lower latencies for the robot mechanism than for the tape library unit. A DVD disk weighs less than a tape, and should be easier to move faster. The thickness of a DVD disk is also

	Tandberg MLR1			IBM Magstar			DVD drive		
Operation	block	1 min	2 h	block	1 min	2 h	block	1 min	2 h
Robot load	4	4	4	4	4	4	3	3	3
Drive mount	31	31	31	5.6	5.6	5.6	3	3	3
Search	63	63	63	12.3	12.3	12.3	0.27	0.27	0.27
Read	0.023	24.8	3057	0.015	17.1	2045	0.012	13.5	1625
Rewind	63	63	63	12.3	12.3	12.3	0.27	0.27	0.27
Drive unmount	5	5	5	4	4	4	2	2	2
Robot unload	4	4	4	4	4	4	3	3	3
Total	170	195	3227	42	59	2087	12	25	1637

Table 9.7 Cycle time for retrieving one block of data (32 KB), a one minute video sequence, and a two hours video sequence using each of the three media drive types in the generic tertiary library unit. All numbers are given in seconds.

lower than for a tape, making it likely that the distance the disk has to be moved by the robot device will be lower. We use constant values for the time to perform the load and unload operations. This is justified by experiments performed by Johnson and Miller (1998b) and by Chervenak (1994). A more detailed presentation of how this generic tertiary library unit performs the load and unload operations was given in Section 9.1.

To end this section about the performance of the hardware devices, we show some examples of the average cycle time when this generic library is used for fetching one data block, one minute of video, and two hours of video from a tertiary storage media using each of the three media drives. In this example, videos with an average data rate of 5 Mbit/s are used. The results are presented in Table 9.7. This table gives the number of seconds the library unit is occupied retrieving a data segment of the three different sizes. We have assumed there is only one drive in the library unit and that it is necessary to change the medium for each request.

These numbers are more illustrative if we convert them into number of video sequences the library units can deliver during one hour. Using a MLR1, a Magstar, and a DVD drive, the library unit can retrieve 18, 61, and 144 one minute long video sequences per hour. If we retrieve two hour long video sequences, the corresponding numbers are 1.1, 1.7, and 2.2 video sequences per hour. The main point to note is that the number of requests a library unit with this configuration can deliver is rather low for all three drive configurations. If we compare the drive technologies, we see that the tape based drives suffer much due to the long seek time when retrieving short video sequences. When retrieving one minute

	Cost per medium	Capacity	Cost per MB	Media to store 1 TB	Cost per TB
Seagate Barracuda 18LP	$ 800	16.9 GB	$ 0.046	61	$ 48500
Tandberg MLR1	$ 50	12.1 GB	$ 0.0040	85	$ 4230
IBM 3570 Magstar MP	$ 60	4.6 GB	$ 0.013	223	$ 13360
DVD-R	$ 40	4.38 GB	$ 0.0089	234	$ 9350

Table 9.8 Cost of storage media.

Drive model	Cost
Seagate Barracuda 18LP	$ 800
Tandberg MLR1	$ 1500
IBM 3570 Magstar MP Model B1A	$ 8500
DVD-ROM	$ 300
DVD-R	$ 5000

Table 9.9 Prices for the drives. The price of a DVD-R drive is included since at least one such drive is needed to write the video to the DVD-R disks.

long video sequences, the DVD based library unit is able to deliver eight times more video sequences than the MLR1 based library unit. When retrieving two hour long video sequences, the relative performance between a library unit using a DVD drive and a library unit using a MLR1 drive is reduced to two to one in favor of the DVD drive.

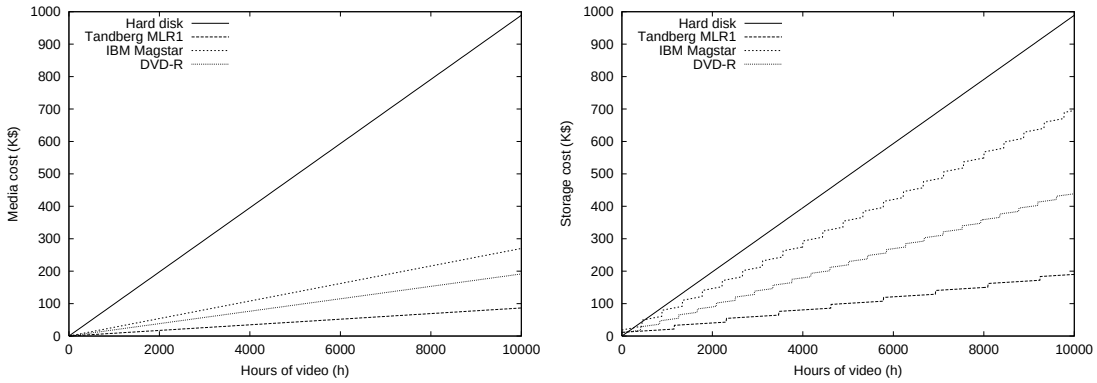
9.4 Cost of Storing Digital Video

So far, we have assumed that use of tertiary storage is the least expensive way to store large amounts of digital video. However, we have not investigated how much it will cost to store the video, or how much that can be saved by use of tertiary storage instead of secondary storage. In this section, we use some examples to investigate the cost of storing digital video. We compare the different storage technologies with respect to the cost of *storing* the video. The cost of *delivering* the video is not considered. This will by studied further in Chapter 10 and 14.

All prices used in this section are from 1998 and are given in American dollars. The prices are found by checking several suppliers on the World Wide Web. Since prices vary between the different suppliers, the prices presented here are representative prices. In most of the price lists we have studied, the price is given for one piece of the item. If we actually should build a video archive, we would

Library unit	Cost	Drives	Maximum media	Storage capacity	Cost per TB
Tandberg TDS 1440	$ 21.000	4	40	484 GB	$ 44.400
IBM MP 3575-L32	$ 62.000	2	324	1490 GB	$ 42.600
Pioneer DVD	$ 12.500	0	100	438 GB	$ 29.200

Table 9.10 Cost data for library units. The Tandberg TDS 1440 is delivered with 4 MLR3 drives. The IBM library unit is delivered with two IBM 3570 Magstar model C drives. The Pioneer library unit is delivered with no drives. The cost per TB is computed using the cost of the library unit divided by the maximum storage capacity.



(a) Media cost

(b) Media, drive, and library cost

Figure 9.16 a) The cost of the storage media to store digital video. The data rate for the video is 5 Mbit/s. **b)** Including the cost of necessary tertiary libraries. Each library contains one drive and can hold 200 storage media. The cost numbers are given in thousand dollars.

Technology	Number of media	Media cost	Number of libraries	Library cost	Total cost
Seagate Barracuda 18LP	1237	990	—	—	990
Tandberg MLR1	1731	87	9	104	191
IBM 3570 Magstar MP	4500	270	23	426	696
DVD-R	4788	192	24	247	439

Table 9.11 Summary of the number of media and library units that are required to store 10000 hours of 5 Mbit/s video. The cost of the media and library units are included. These numbers are given in thousand dollars. For each library unit, the cost of one media drive is included.

buy in large quantities. This would probably give us lower prices than we are using in this section.

In Table 9.8, we present prices for the storage media for the three drives presented in the previous section. We have also included the price for a hard disk drive. Hard disk is an alternative storage medium to use instead of tertiary storage. We us the Seagate Barracuda 18LP in our examples. This hard disk stores 18.2 Gbytes. By calculating the price per MB for other members of the Barracuda disk family with different storage capacities, the 18 Gbytes disk is the optimal drive when the storage cost is the most important criterion. Table 9.8 gives the price per medium, the price per MB of storage, the number of storage media needed to store one TB of data, and the total cost for storing one TB of data. From the table we see that using the Tandberg MLR1 gives the lowest storage cost. The storage cost using this drive is about one tenth of the cost of using hard disks. Using DVD is about twice as expensive as using tape. It should be noted that we in this example use writable DVD-R disks, which cost more than read-only DVD-ROM disks. The storage cost of using tapes for the Magstar drive is higher than using DVD-R disks, and more than three times higher than using MLR1 tapes.

Table 9.8 gave the cost of storing one megabyte and one terabyte of data. To visualize the cost of storing video, we have in Figure 9.16(a) plotted the cost of the storage medium for storing a given number of hours of digital video using each of the storage media in Table 9.8. In this example we use video with a data rate of 5 Mbit/s. Storing 10000 hours of this video quality requires 20.5 TB of storage. Figure 9.16(a) shows that storing 10000 hours of video costs $ 990.000 using hard disks. Using MLR1 tapes the media cost is reduced to about $ 87.000. For Magstar tape and DVD-R, the corresponding costs are respectively $ 270.000 and $ 192.000. The number of media and the total cost for storing this amount of video is given in column two and three in Table 9.11.

To access the video stored on the tertiary storage media, the storage media

must be included in library units containing drives. In Table 9.9, we present prices for drives able to read the tertiary storage media used. The price for a DVD-R drive is included for reference since at least one such drive is needed in the system to be able to perform the initial writing of the DVD disks. For reading the video from the DVD disks, it is likely that less costly DVD-ROM drives will be used.

Table 9.10 contains prices for the three tertiary library units presented in Table 9.5. Just as the technical specifications of the library units are very different, the prices differ much between the library units. For the same reasons as stated in Section 9.3.2, we use a *generic* library in this section. The specifications for the generic library is given in Table 9.6, and use a representative price of $ 10,000 for the library unit *without* any media drives.

In Figure 9.16(b) we have plotted the cost of storing the same amount of video as in Figure 9.16(a), including the cost for the necessary tertiary libraries. The number of libraries is determined by how many libraries are needed when each library stores 200 media. The number of library units required to store 10000 hours of video is given in the fourth column of Table 9.11. The cost of having one drive in each library is included in the figure. Hard disks, which already have a drive for each medium, do not need to be contained in a library unit. In this example, even though not entirely correct, we use the same storage cost for hard disks as in Figure 9.16(a). We will comment further on this in Section 9.4.2. For the tertiary storage media, the figure shows that the total storage cost is more than doubled by including the cost of library units and drives. Column 5 and 6 of Table 9.11 contains the cost of the library units and the total cost of storing 10000 hours of digital video. The least increase in the cost is for the Tandberg MLR1 where the price is 120 percent higher. The reason is that the Tandberg MLR1 has the highest storage capacity per medium, and thus requires the least number of library units. For the Magstar and DVD, the total storage cost is increased by 160 and 130 percent respectively .

In Figure 9.17 we compare the price of using the three tertiary storage media to the price of using hard disks. The figure is based on the numbers given in Table 9.11, and contains bars for both the media cost and the total cost of libraries, drives, and media. The bars give the cost in percent relative to the cost of using hard disks. If we only consider the media cost, using MLR1 media costs only nine percent of the cost of using hard disks. For Magstar and DVD the corresponding numbers are 27 and 19 percent. By including the cost of library units and drives, we get more realistic numbers. MLR1 still provides the least expensive storage alternative. Compared to hard disks, the cost of using MLR1 is only 19 percent. Using DVD costs approximately 44 percent compared to hard disks. Magstar is the most expensive of the three tertiary storage technologies. The cost of using this technology is approximately 70 percent of the cost of using hard disks.

In addition to provide storage capacity, the storage system also provides access to the data. The bandwidth of the different storage technologies varies much.

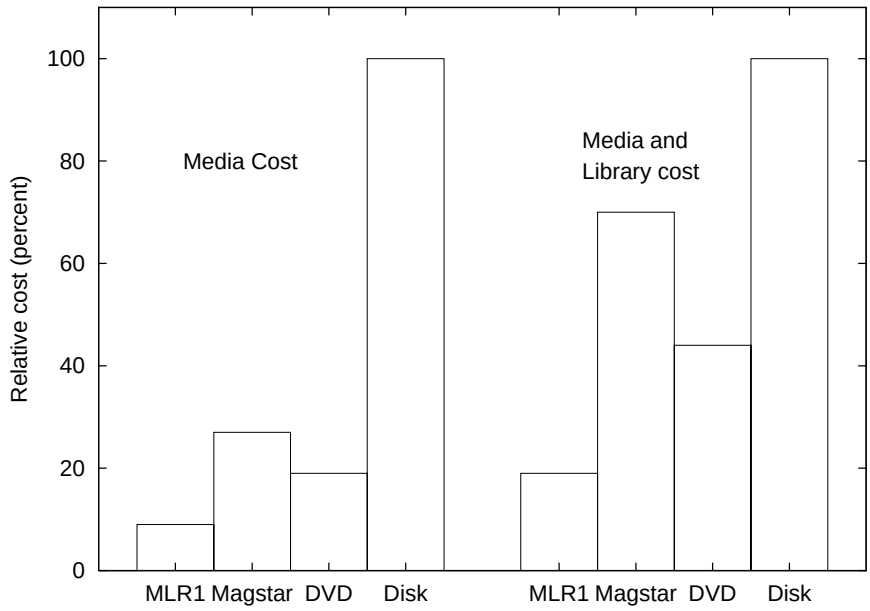


Figure 9.17 The cost of using different tertiary storage technologies compared to using hard disks. The figure is based on the cost numbers in Table 9.11.

In Table 9.12 we have included both storage and bandwidth data for the storage system configuration used as example in this section (as specified in Table 9.11). The bandwidth numbers are computed by multiplying the bandwidth for *sequential* reading of each drive with the number of drives in the system. These numbers are only achievable if the drives spend no time for repositioning. It is important to note that the bandwidth data for the tape and DVD-based systems only contains one drive in each library. The table also contains cost numbers for storage and for bandwidth. These numbers are found by taking the total cost for the different systems and dividing it by respectively the storage capacity and the bandwidth of the system. The storage cost numbers have the same ratio between the storage technologies as given in the right part of Figure 9.17. The bandwidth of the disk-based system is between 200 and 1000 times higher than the bandwidth of the tape and DVD-based systems. Each disk having its own drive while the tape and DVD libraries only contain a single drive. Similarly, the bandwidth cost for the tape-based systems is approximately 200 times higher than for the disk-based system and approximately 100 times higher for the DVD-based system.

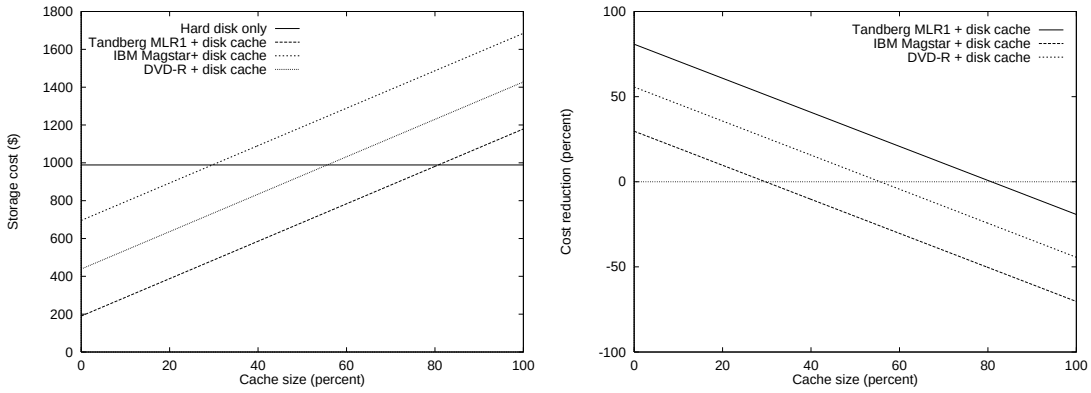
Technology	Storage (GB)	Storage cost ($/MB)	Bandwidth (MB/s)	Bandwidth cost ($/(MB/s))
Seagate Barracuda 18LP	20905	0.046	12370	80
Tandberg MLR1	20945	0.0089	12.7	15040
IBM 3570 Magstar MP	20700	0.032	48.3	14410
DVD-R	20971	0.020	63.0	6970

Table 9.12 Summary of storage, bandwidth, and cost data for the storage system configurations presented in Table 9.11. The bandwidth for the DVD and tape-based system is computed using the transfer rates given in Table 9.3. For the disk drive, an average transfer rate of 10 MB/s is used.

9.4.1 Storage Cost for Using a Disk Cache

The main reason for considering using tertiary storage in a digital video archive is to reduce the total cost. If tertiary storage is not used, the video must be stored on secondary storage, i.e., hard disks. As shown in Figure 9.16(b), replacing the hard disks by tertiary storage devices can reduce the storage cost by up to 80 percent. This can be a good solution if the video content is infrequently accessed and the rather large response time is not a problem. If the archive is frequently used for interactive retrieval of video sequences, it can be necessary to use a disk cache. The use of a disk cache is studied in more detail in Chapter 14.

The size of the disk cache should be determined by studying the distribution of the accesses to make sure that the most frequently used video sequences are stored in the cache. Another factor for deciding the size of the disk cache is also how much the owner of the archive is willing to pay to reduce the average response time. In Figure 9.18(a) we have plotted the cost of storing 10000 hours of 5 Mbit/s video data including the cost of a disk cache. The x-axis gives the relative size of the disk cache. The figure shows the total storage cost for the archive as we vary the size of the disk cache between zero and the total size the archive. For reference, we have included the cost of using disk as the only storage medium for the video content. This figure shows that by using Tandberg MLR1 tapes, the total storage cost is favorable to a disk-only system as long as the size of the disk cache is less than about 80 percent of the total required tertiary storage. For DVD, the hard disk cache must be less than 55 percent. With the relatively higher cost of the Magstar tape drive, the disk cache must be less than 30 percent of the total data volume in order to make it less expensive than having all data on hard disks. The same can be found by studying Figure 9.18(b). This figure shows how many percent of the total cost that can be saved by using tertiary storage compared to a disk-only system.



(a) (b)

Figure 9.18 a) The added cost by including a disk cache in the system. The size of the disk cache is varied between zero and 100 percent of the total size of the archive. The costs are computed for a video archive containing 10000 hours of video. **b)** The percent of the total cost that can be saved by using tertiary storage compared to a disk-only system as a function of the relative size of the disk cache.

9.4.2 Further Considerations

In this section we have only included the minimum cost for creating a video archive. Only storage for one copy of the video content is included. In a real video archive, storage media will due to tear and wear become unreadable, and we must be able to recreate the lost data. Traditionally, this problem is solved by having one extra copy of the data (backup). In the case where all video is already stored using tertiary storage, this requires us to double the amount of tertiary storage needed. If the entire video archive is stored on hard disk only, to have safe backup will require the same amount of tertiary storage. An alternative to backup can be to store enough redundancy in the data to be able to recreate the lost data. For hard disks this can be done using layout of the data on disks similar to RAID (Patterson et al., 1988). It is also possible to achieve this by using a similar redundancy schema for tertiary devices (Chervenak, 1994).

In each library unit, we have only included one drive. For an interactively used archive, this will in most cases be not be enough. To handle a higher number of requests, more drives can be added to each library unit. Adding more drives increases the storage cost ($/MB) while decreases the cost of bandwidth ($/(MB/s)). In Chapter 10 and 14, we study how the number of drives influences the performance and cost of tertiary storage systems.

Further, we have not included the cost of all the extra peripheral equipment needed to build a working archive. First of all, we need a computer (or more

likely multiple computers) to manage the stored video and to be responsible for delivering the video to users on request. But also accessories like cables, host interfaces, power supplies and cabinets can contribute considerably to the total cost. If we for example have a 1 TB disk cache, this will consist of 61 disks each storing 18 Gbytes. To connect this number of disks to a computer can be expensive. In some cases cables, cabinets, power supplies and host adapters will cost as much as the disks themselves.

One important factor that must be considered when comparing the cost and performance numbers is that the performance of hardware is rapidly changing. In this part of the thesis we compare different storage technologies. It is important to be aware that for the tape-based systems we use performance data for technology that is approximately two years old (1996) while for the hard disks and DVD we use performance data for today's versions (1998). For both MLR1 and Magstar there are newer versions available. If we had used these newer versions, the storage cost would have been lower since the new versions are able to store more data on each medium. Thus, the required number of media and the number of library units would have been reduced. The newer versions also have a higher bandwidth, which would lead to lower bandwidth cost.

Chapter 10

Performance Analysis of Library Units used for Storing and Retrieving Digital Video

I haven't lost my mind, it's backed up on tape somewhere.

Peter da Silva

In Chapter 8, we studied the performance we could get from using *one tape drive* containing *one storage medium* for retrieving video sequences. We showed that in the case of multiple concurrent requests, both the throughput and the response time can be substantially improved by scheduling the requests. In this chapter, we go one step further and investigate the performance it is possible to get from *one library unit* when used for storing and retrieving digital video. In the study, library units utilizing the following three tertiary storage technologies are evaluated and compared:

1. **Tandberg MLR1.** This is a general purpose tape drive based on serpentine tape technology optimized for providing safe storage at a low cost. A detailed performance model of this drive was developed in Chapter 6 (Sandstå and Midtstraum, 1999b).

2. **IBM 3570 Magstar MP.** This tape drive is also based on serpentine tape technology. It is optimized for random I/O requests. A detailed performance model is found in (Hillyer and Silberschatz, 1998). An overview of this model was presented in Section 9.3.1.

3. **DVD.** A generic 2X DVD drive is used. The performance model of the DVD drive is based on the performance model found in (Shastri et al., 1997), which was briefly presented in Section 9.3.1.

The library units contain one or more tertiary drives, multiple storage media, and a robot mechanism for transporting storage media to and from the drives. The simulator presented in the previous chapter is used for conducting the experiments and for producing the results and the performance data.

In order to evaluate and compare the different tertiary storage technologies, we study the factors most closely related to the *performance* and *cost* of the storage system:

- **Throughput.** In the simulations, we define the *throughput* of the library unit as the number of video sequences of a given length the system is able to deliver during a given period of time. Since retrieval of long video sequences may take a considerable amount of time, we use *one hour* as time unit when presenting throughput numbers.

- **Response time.** We use the response time observed by the users as the response time of the library unit. This response time is measured as the amount of time it takes from the user issues a request for a video sequence, until the tertiary storage system *starts* to deliver the requested video to the user. The *mean* (average) response time of the requests is used when measuring and presenting response times from simulations.

- **Cost of storage and delivery.** In Section 9.4, we discussed the cost of *storing* video using tertiary libraries and storage media. Only the necessary hardware needed for storing the video data was included in the cost of the system. To achieve the throughput and response time requirements set by the users of the archive, it can be necessary to include multiple drives in the library unit, or to use an alternative tertiary storage technology. This increases the cost of the storage system. In order to compare different technologies and configurations with regards to the cost of both *storing* and *delivering* the video sequences, we use the following two criteria (Chervenak, 1998):

  - **Storage cost as function of throughput or response time requirements.** This is determined by configuring the system so that it is able to fulfill the decided *throughput* or *response time* requirements, and then compute the storage cost in dollars per GB of video.
  - **Cost per stream.** We compute cost per delivered video sequence as the total cost of the storage system divided by the *throughput* of the system (Chervenak, 1998).

In most of this chapter, we consider the library unit as a black box in order to determine the *external* performance characteristics of the library unit. The goal is to study and compare the performance that can be achieved by using the three different tertiary storage technologies. We determine what are the limiting factors

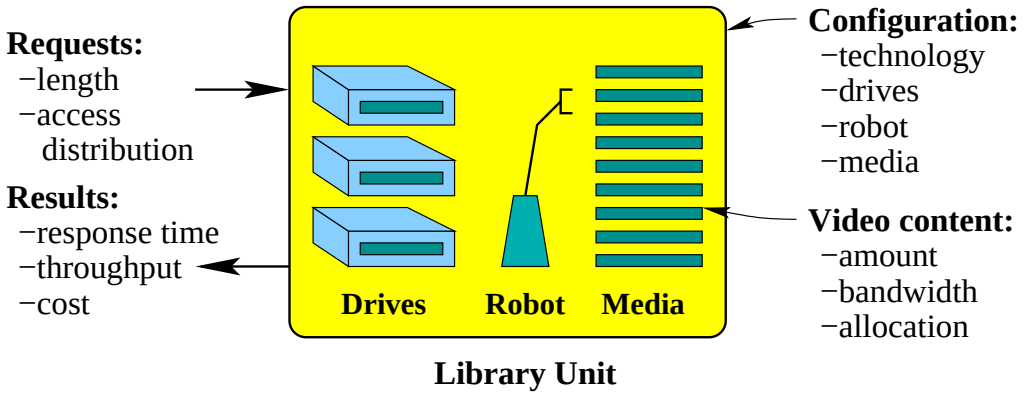


Figure 10.1 The simulation model and the main parameters that characterize the workload used in the simulations.

for different library unit configurations and user loads. The main variables we vary in the simulations are the size of the retrieved video sequences, the load on the system, the number of drives, and the tertiary technology used. In Chapter 11, we study in more detail the effect the individual operations of the media drives and the robot mechanism has on the performance of the library units.

10.1 Simulation Model

Before we can start running simulations, a model for the workload to be used in the simulations has to be defined. For the simulations performed in this chapter, the main parameters are the library unit configuration, the amount of video, the size of the video sequences, and the load generated by the users. These parameters are given as input to the simulator. Figure 10.1 gives an overview of the library unit and the simulation parameters which are further described in the following paragraphs.

10.1.1 Library Configuration

In this part of the study, we simulate one library unit. For each run of the simulator, the following configuration variables must be determined:

- **Drives.** Three tertiary drives are simulated: Tandberg MLR1, IBM 3570 Magstar MP, and 2X DVD. The number of drives is from one to 32 drives.
- **Robots.** The library unit contains one robot mechanism.
- **Storage media.** The library unit contains a fixed number of 200 storage media.

	Length of video sequences		
Technology	One minute	Ten minutes	One hour
MLR1	1155	1130	1000
Magstar	440	432	400
DVD	416	400	400

Table 10.1 The amount of video stored in each library unit. The numbers are given in hours of 5 Mbit/s video that can be stored on 200 storage media.

As we do not use a video cache in the study of library units, all requested video sequences have to be fetched from tertiary storage.

10.1.2 Video Content

The library unit contains videos which the simulated users issue requests for. Before each simulation is started, the library unit is filled with video data. The videos used in the simulations performed in this chapter have the following characteristics:

- **Bandwidth.** Video with a bandwidth of 5 Mbit/s is used in the simulations. Using MPEG-2 compression, this corresponds to video of PAL quality.
- **Length of video sequences.** In each simulation, the library unit is filled with video sequences of the same length. In most of the chapter, video sequences of two different lengths are used. The two lengths are *short* video sequences of one minute and *long* video sequences of one hour. These two cases were selected in order to reduce the number of simulations, but also because they represent different use of a video archive. For a television news archive, typical queries result in retrieval of a set of short news stories (Hjelsvold et al., 1996). By analyzing a small set of evening news from TV2 Norway, we found the average length of a news story to be 54 seconds (Hjelsvold, 1995). Retrieval of one hour video sequences corresponds roughly to a video-on-demand-like use of the archive.
- **Amount of video data.** In the simulations, the library unit contains the maximum amount of video that can be stored on 200 storage media. The number of hours of video that can be stored in the library unit for different video sequence lengths and storage technologies is given in Table 10.1.
- **Access locality.** In this chapter, we use a uniform access model. All video sequences have the same probability of being accessed. The performance

of a video archive when the access distribution is non-uniform is studied in Chapter 12.

- **Allocation.** Each video sequence is allocated to only *one* storage medium as explained in Section 9.1.5. We do not utilize any distribution or replication of videos across multiple storage media. Strategies for allocating video sequences to storage media and library units in a tertiary storage system is evaluated in Chapter 13.

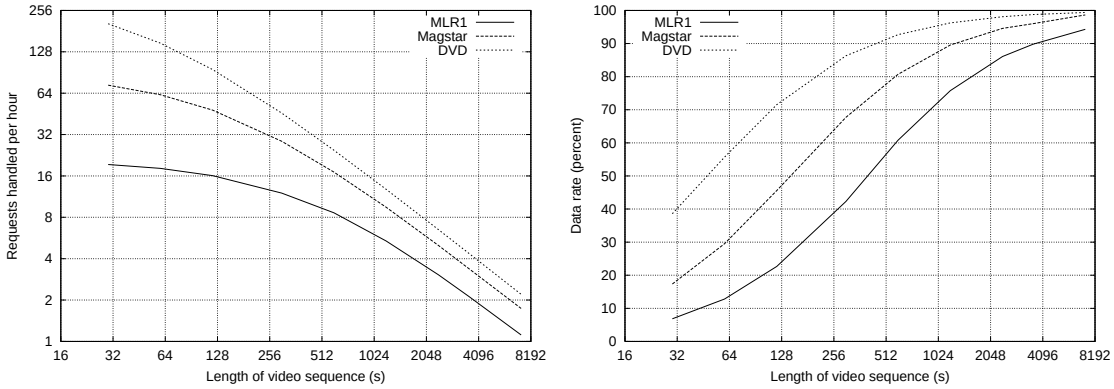
10.1.3 User Load

The load generated by the users of a video archive is very dependent of the type of video archive. A television news archive used by the archive staff will likely have a very different load than a video-on-demand server or a video archive available to everybody on the Internet. In our simulations, we do not try to model real user behavior or any particular video archive application. As described in Section 9.1.6, we use general statistical models to generate the load on the system. For each simulation, the load generators are instrumented to generate the desired load on the library unit.

10.2 Throughput of Library Units

In this section, we investigate the throughput of different library units. The maximum throughput of a library unit has to be larger than the number of requests generated by the users in order to avoid long queues of waiting requests. The main factors that determine the throughput of a library unit are the mount, seek, and rewind times and the transfer rate of the tertiary media drives and the performance of the robot mechanism. For a library unit containing a single drive, the robot mechanism should not be the limiting resource, but as more drives are included in the library unit, it might become a bottleneck. The mount, seek, and rewind times are independent of the size of the retrieved video sequence, while the transfer time is proportional to the size of the video sequence.

We start with investigating the throughput of library units containing a single media drive. The purpose is to compare the relative performance of the storage technologies when used for storing and retrieving video sequences of different lengths and to evaluate which factors that determine the throughput. The throughput of a library unit can be increased by including multiple drives in the library unit. In the second part of this section, we study how the throughput scales as more drives are included in the library unit. The main goal is to determine the factors that limit the number of drives it is useful to include in a library unit.



(a) Throughput (b) Data rate utilization

Figure 10.2 a) The number of requests a library unit containing one media drive can handle per hour as a function of the length of the requested video sequences. **b)** The effective data rate a library unit containing one media drive can deliver as a function of the length of the requested video sequences. The data rate is given as percentage of the maximum transfer rate of the drive.

10.2.1 Single Media Drive in the Library Unit

The simplest configuration of a library unit contains a single media drive. To determine the throughput of library units using each of the three tertiary storage technologies, we simulate a system where users are retrieving video sequences. In each simulation, the video sequences have the same length. To get throughput numbers as a function of the length of the video sequences, we perform simulations for lengths from 30 seconds up to two hours. We use a closed queueing model as described in Section 9.1.6 for generating the load on the library unit. A single user is repeatedly requesting one random video sequence. As soon as the current request completes, a new request is generated. This model corresponds to having an infinite long line of waiting requests which are scheduled in a FCFS order. The reason for limiting the number of active requests to one in this simulation is to avoid that re-ordering of the requests by the library and media schedulers influences the throughput. Allowing multiple concurrent requests would have increased the probability of multiple requests executed per media load, and thus resulted in higher throughput numbers.

The simulator keeps track of the number of executed requests and the simulated time. Based on these two numbers, the throughput of the library unit is computed. The resulting throughput curves are presented in Figure 10.2(a). This figure shows the throughput for each of the three types of media drives as a func-

	Length of video sequences			
Technology	One minute	Ten minutes	One hour	Transfer rate
MLR1 : DVD	1 : 8.2	1 : 2.9	1 : 2.1	1 : 1.8
Magstar : DVD	1 : 2.4	1 : 1.4	1 : 1.3	1 : 1.2
MLR1 : Magstar	1 : 3.4	1 : 2.0	1 : 1.6	1 : 1.5

Table 10.2 Comparing the relative throughput between the three types of media drives for retrieval of video sequences of one minute, ten minute and one hour. The ratios for the transfer rate of the drives are based on the transfer rates given in Table 9.3.

tion of the length of the requested video sequences. It is important to note that the axis are logarithmic. Thus, the performance differences between the three drive technologies are larger than it might seem from a quick glance at the figure.

As the throughput curves show, using a single DVD drive provides the highest throughput, Magstar is second, while using a MLR1 drive gives the lowest throughput. To illustrate the relative throughput ratios between the three drive technologies, we give some examples in Table 10.2. These numbers show that the relative difference in throughput between the drives decreases as the length of the retrieved video sequences increases. For retrieval of short video sequences of one minute, the library unit containing one DVD drive is able to serve eight times more requests than a library unit containing one MLR1 drive and 2.4 times more requests than a library unit containing one Magstar drive. If the length of the requested video sequences is one hour, the ratio between the the library unit containing a DVD drive and the library unit containing a MLR1 drive is reduced to two. The reason for the reduced difference between the DVD and tape drives is that the transfer rate dominates more on the total retrieval time when the length of the video sequences increases. The last column of Table 10.2 contains the ratios between the transfer rates of the three drives. If we compare the columns containing numbers for throughput ratios with the transfer rate column, we see that the throughput ratios approach the transfer rate ratios as the length of the video sequences increases.

Figure 10.2(b) shows the data rates a library unit using the three different drives is able to deliver as a function of the length of the video sequences. The data rates are given as a percentage of the maximum transfer rate of the drives. The figure shows that the utilization of the drives' transfer rate increases as the length of the video sequences is increased. It also shows that the system is able to utilize the transfer rate of the DVD drive better than the transfer rate of the Magstar and MLR1 drives. This is due to the fact that the DVD drive spends much less time seeking compared to the tape drives. For instance, the library unit

achieves an average transfer rate of 70 percent of the drives' transfer rate for two minutes long video sequences for the DVD drive, five minutes for the Magstar drive and 15 minutes long video sequences for the MLR1 drive.

This far, we have studied the throughput of library units containing a single MLR1, Magstar, or DVD drive. We have shown that the relative difference in throughput between the three drives is dependent on the size of the requested video sequence. As the length of the retrieved video sequence is increased, the difference in throughput between the technologies decreases. In Chapter 11, we study in more detail the effect that the mount and seek time, the transfer rate, and the robot have on the throughput of a library unit. In this section, we continue by studying how the throughput of a library unit can be increased by including multiple media drives.

10.2.2 Multiple Media Drives in the Library Unit

The performance of a library unit can be increased by including more than one media drive in each library unit. The number of media drives that can be included in a library unit is normally limited by the physical size of the library unit and the design of the robot mechanism. More media drives require the robot to be able to move media over longer distances and thus require more complex mechanical operations. With more drives in the library unit, the number of media transportations will increase. Thus, the robot mechanism might become the resource that limits the linear scalability in throughput as more drives are added. In our experiments, we simulate library units containing from one to 32 media drives.

The purpose of this study is to investigate how the throughput scales when increasing the number of drives in the library unit, and to determine which factors limit the number of drives that is useful to have in a library unit. To study this, we perform simulations that show how the throughput of the library units increases as more media drives are included in the library units. As shown in the previous subsection, the throughput is highly dependent of the length of the requested video sequences. In order to show the effect the size of the requested video sequences has on the scalability, we perform the simulations for video sequences of one minute and one hour length.

Simulations

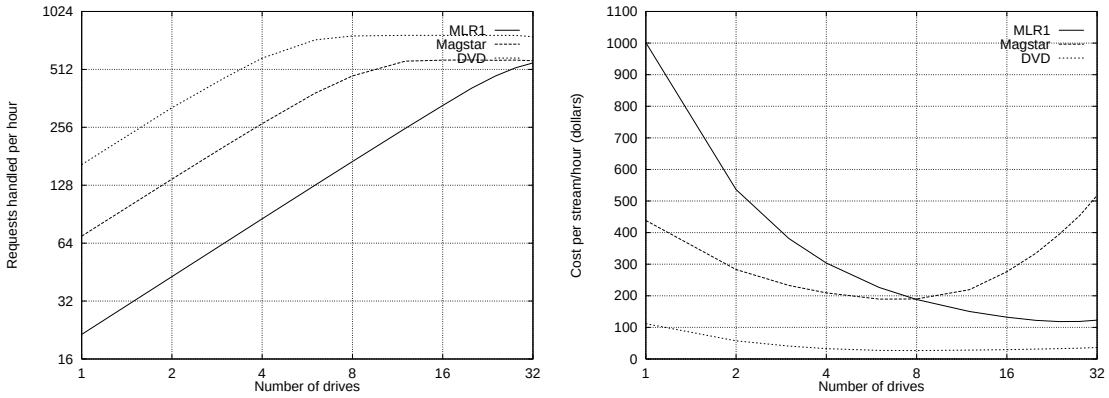
To find the throughput limits, we use a closed queueing simulation model with a load of 50 concurrent requests, i.e., at all times during the simulations there are 50 requests that are either being handled by the drives or the robot, or are waiting for a drive or the robot to become idle. The reason for having more concurrent requests than there are drives, is to avoid that some of the drives become idle. As a result of having concurrent requests is that there will be *queueing effects*. Con-

current requests increase the likelihood of multiple requests for the same storage medium. Since all requests for the same storage medium are executed during the same media cycle, the average amount of time to perform each request is reduced (Prabhakar et al., 2003). Thus, the more concurrent requests, the higher throughput numbers we can get for the library unit. The choice of having 50 concurrent requests makes it unlikely that any of the maximum 32 drives are idle, while still keeping the probability of multiple requests to each medium at a reasonable level. The simulation results show that the average number of requests served per media cycle is approximately 1.27. The queueing effects could have been avoided by not allowing the system to perform multiple requests during one media cycle. However, removing this optimization would have resulted in a slightly lower throughput.

The results from simulation of retrieving video sequences of one minute and one hour are shown in Figure 10.3 and 10.4. These figures contain plots of the throughput, the *cost per stream*, and the drive and robot utilization as a function of the number of media drives in the library unit. The drive utilization is given as the fraction of the time the drive is busy. In addition to the transfer time, this also includes time for mounting, unmounting, seeking, and rewinding. Similarly, the robot utilization is given as the fraction of the time the robot is busy either transporting a medium between a drive and the media store, or waiting for a drive to eject the medium.

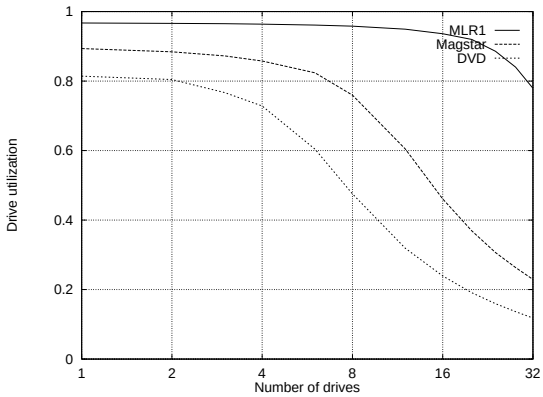
Retrieval of Short Video Sequences

When retrieving short video sequences, Figure 10.3(a) shows that the throughput increases linearly when we add the first few drives. But as the number of drives becomes higher, the effect of adding a new drive decreases. This is most visible for the DVD drives and least visible for the MLR1 drives. We see the same effect in Figure 10.3(c) where the drive utilization decreases as we increase the number of drives. The reason is that when retrieving short video sequences, the cycle time for retrieving one video sequence is short, and thus, the number of media that have to be transported between the drives and the media store is large. The robot mechanism becomes the bottleneck of the system. By studying the robot utilization in Figure 10.3(d), we see that the robot becomes highly utilized as we add more drives to the library. The robot has an utilization of 80 percent for four DVD drives, seven Magstar drives, and 24 MLR1 drives. Adding more drives than this to a library unit which primarily is delivering short video sequences results in poor utilization of the drives. We see the same effect by studying the cost per stream in Figure 10.3(b). As we add more drives, the cost per stream decreases until the robot becomes saturated. From that point, the cost per stream increases as we add more drives. The cost increases most for the Magstar drive since this drive is the most costly, and least for the MLR1 drive since the robot does not get saturated until we have included 24 drives in the library unit.

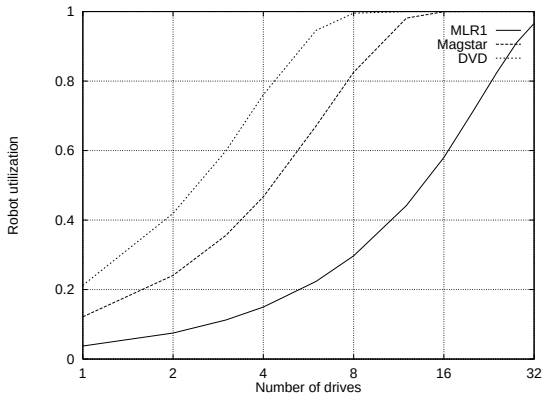


(a) Throughput

(b) Cost per stream

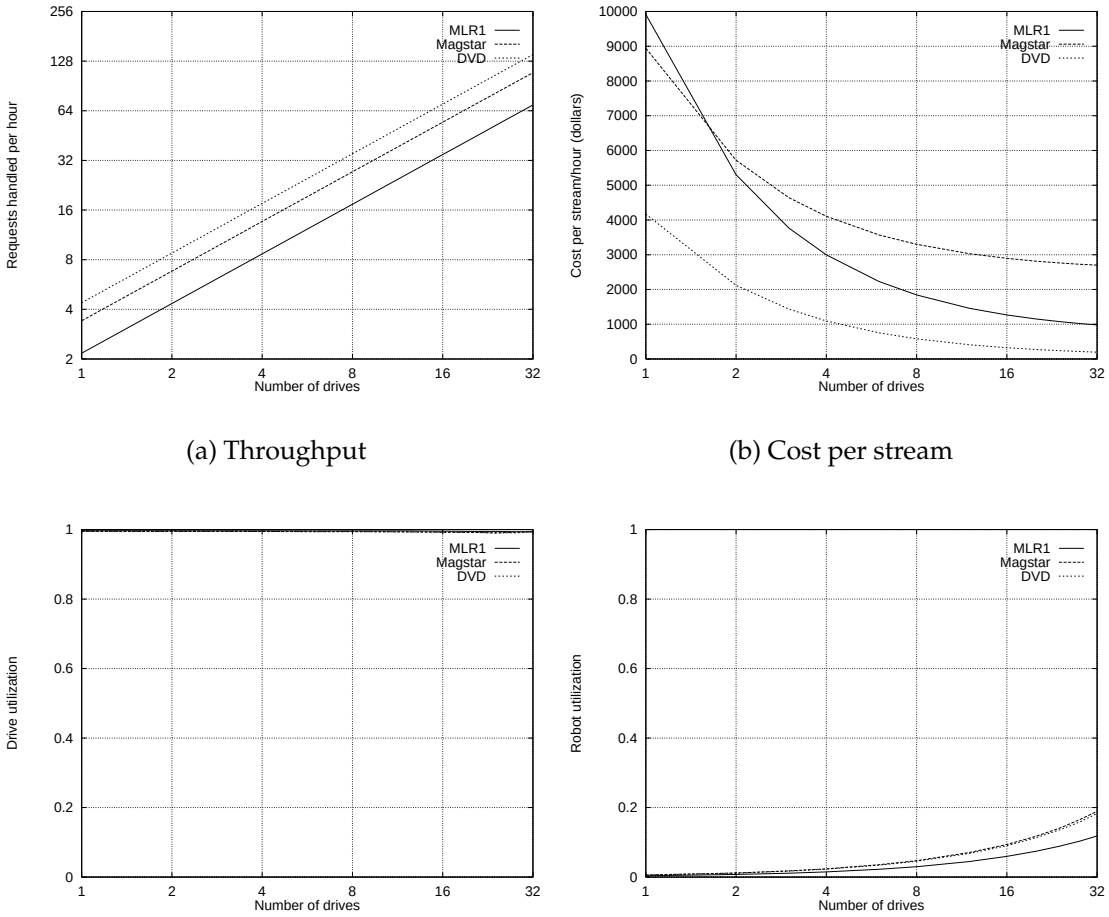


(c) Drive utilization



(d) Robot utilization

Figure 10.3 a) The throughput of a library unit as function of the number of media drives included in the library unit and storage technology. Each request is for a video sequence of *one minute*. **b)** The cost of delivering video sequences, given as the total cost divided by the throughput. **c)** and **d)** gives the corresponding utilization for the drives and the robot.



(a) Throughput

(b) Cost per stream

(c) Drive utilization

(d) Robot utilization

Figure 10.4 a) The throughput of a library unit as function of the number of media drives included in the library unit and storage technology. Each request is for a video sequences of *one hour*. **b)** The cost of delivering video sequences, given as the total cost divided by the throughput. **c)** and **d)** gives the corresponding utilization for the drives and the robot.

Retrieval of Long Video Sequences

Figure 10.4(a) shows the throughput numbers for retrieval of video sequences of one hour. The throughput curves show that the throughput scales linearly as we add more drives to the system. When retrieving long video sequences, the robot is no longer a bottleneck. As a result, we achieve a high drive utilization also when the library unit contains a larger number of drives. For retrieval of video sequences of one hour, the robot is able to handle more than 32 drives without becoming the bottleneck of the system (see Figure 10.4(d)).

Cost of Video Delivery

In Section 9.4, we discussed the cost of storing video using tertiary library units, and showed that using MLR1 drives and media gave the lowest storage cost. In the simulation results we have included the cost of *delivering* video sequences (see Figure 10.3(b) and 10.4(b)). If we compare the three media drives by using the *cost per stream* as criterion, we find that DVD drives give the least cost per requested video sequence for all the cases we have simulated. If we compare the two tape drives, we find that the Magstar drive gives the lowest *cost per stream* when there are few drives in the library unit. For a library unit containing many drives, the *cost per stream* is lower when using MLR1 drives. The reason for this is the higher cost of the Magstar drive. With few drives in a library unit, the fraction of the total cost that the drives account for is comparable to the cost of the library unit itself. As the number of drives increases, the cost of the drives becomes more and more dominating. Thus, the lower cost of the MLR1 drive makes this technology scale better with regards to the cost per delivered video sequence.

Another point worth noting when comparing the cost of using MLR1 and Magstar drives, is where the *cost per stream* curves cross. For retrieval of short video sequences, we can use up to eight Magstar drives in the library unit while still having a lower cost per delivered stream compared to using the MLR1 drives. For retrieval of video sequences of one hour, only when using one Magstar drive in the library unit will the cost per delivered stream be lower than when using MLR1 drives. The reason for this difference, is that the throughput of the Magstar drive compared to the MLR1 drive is comparatively higher for shorter video sequences than for longer video sequences. As shown in Table 10.2, for retrieval of one minute video sequences, the Magstar drive is able to retrieve 3.4 times more video sequences per hour than the MLR1 drive. For retrieval of video sequences of one hour, the ratio between these two drives is reduced to 1.6.

An evaluation of the cost of using tape libraries for delivery of video is performed by Chervenak et al. (1995). The cost per stream using both low-cost Exabyte EXB 120 and high-performance Ampex DST600 tape libraries containing four tape drives is evaluated. For retrieval of 3 Mbit/s video sequences of 100 minutes they found the cost per stream to be in the range of $ 20,000. The corresponding cost numbers from our simulations are $ 3000 using the MLR1 drive

and $ 4000 using the Magstar drive (see Figure 10.4(b)). The main reason why we achieve a lower cost per stream compared to Chervenak et al. (1995) is that the transfer rate of the tape drives used in our study is 3–4 times higher than the transfer rate of the drives used in their low-cost library, while compared to their high-performance library, the cost of our library units is much lower.

10.3 Response Times for Library Units

For the individual user of the library unit, the throughput might not be the most important performance variable, as long he or she gets the job done. For the user, the response time is usually a more important criterion for deciding how satisfied he or she will be with the performance of the storage system. In this section, we investigate the response time of library units using the different tertiary storage technologies. The purpose is to evaluate how the response time is influenced by the load on the library unit. To do this, we study how the response time changes under different load scenarios and different library unit configurations. We also study the distribution of the response time for the different storage technologies. Finally, we study the response time during scaling of both the number of drives in the library unit and the load.

The load on a storage system will likely vary rather much. There may be peak periods where the total load can get close to the maximum throughput or even higher. During these periods there will be contention for the resources in the library unit, and requests will be delayed due to queueing, resulting in higher response times. In our simulations, we use an open queueing model as explained in Section 9.1.6 to generate the user load. A Poisson process generates requests for a random video sequence where the interarrival time of the requests are drawn from an exponential distribution. In the presentation of the results, the load is given as *requests per hour*.

We measure the response time as the amount of time from the user issues the request until the system *starts* to deliver the video to the user. The storage devices used in the simulations have a transfer rate larger than the bit rate of the video sequences, so as soon as one of the tertiary storage drives starts to deliver the video sequence to the user, it is capable of delivering the video faster than the user consumes it. When presenting the results from the simulations, the *mean* (average) response time is used in most of this section. The reason for using the mean response time is to make the presentation of the results simpler and easier to comprehend since each measurement can be represented by one number. Alternative representations could be to include the standard deviation or the interquartile range in the presentation of response times. In Section 10.3.2, we give some examples of the distribution of the response times for the library units.

10.3.1 Response Times as Function of Load

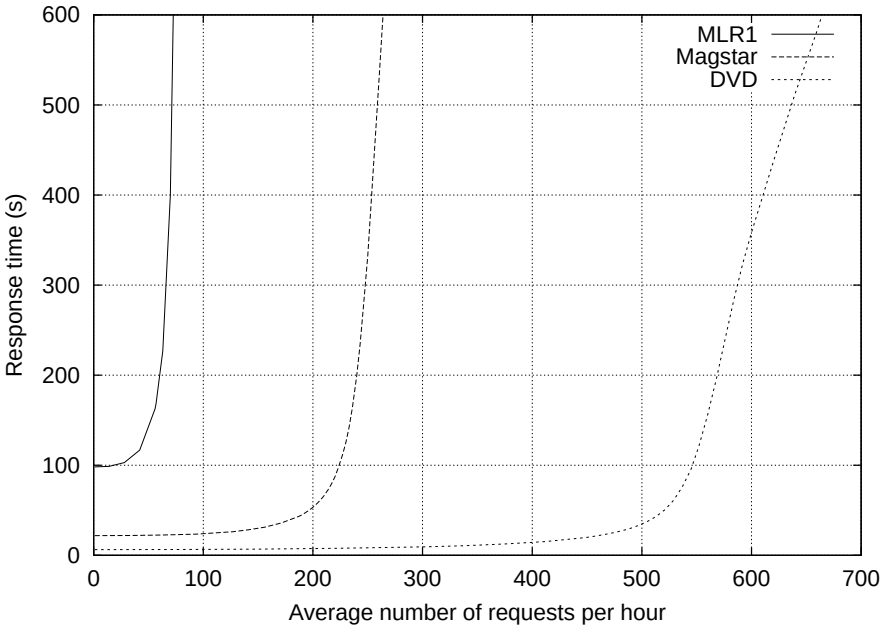
As we increase the load on a library unit, we expect the response time to increase. It is therefore interesting to study how the response time increases as a function of the load on the system. We do this by studying the response time of a library unit containing four tertiary drives. The reason for using four drives is that for retrieval of short video sequences this is the largest number of DVD drives that does not make the robot mechanism become the bottleneck in the system (see Figure 10.3(d)). In the last part of this section, we study the response time for library units containing a different number of media drives.

Retrieval of Short Video Sequences

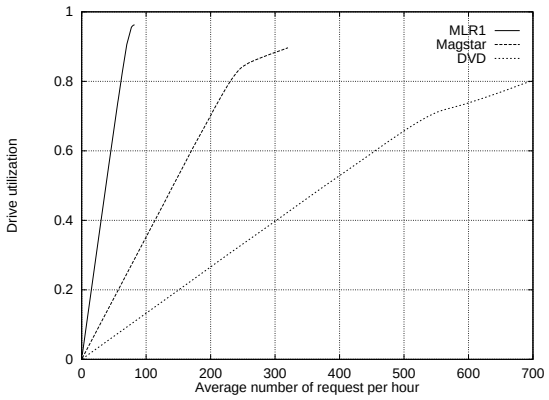
We start by studying the response time when retrieving short video sequences of one minute. Figure 10.5(a) shows the average response time as a function of the load on the system. For all three drive types, the response time curve is relatively constant until it reaches a point where the response time increases rapidly. By studying this figure (and the numbers used for plotting it), we see that during low load, the average response time is 98 seconds when using MLR1 drives, 22 seconds when using Magstar drives, and 6.3 seconds when using DVD drives. Thus, for a lightly loaded system, the response time when using DVD drives is only one sixteenth of the response time when using MLR1 drives and about 30 percent of the response time when using Magstar drives.

The increase in the response time is due to resource contention in either drives or the robot mechanism. Figure 10.5(b) and 10.5(c) show the utilization of the drives and the robot mechanism. For the MLR1 and Magstar drives, the response times start to increase due to high drive utilization. For the DVD drive, the main reason the response time start to increase is due to high robot utilization.

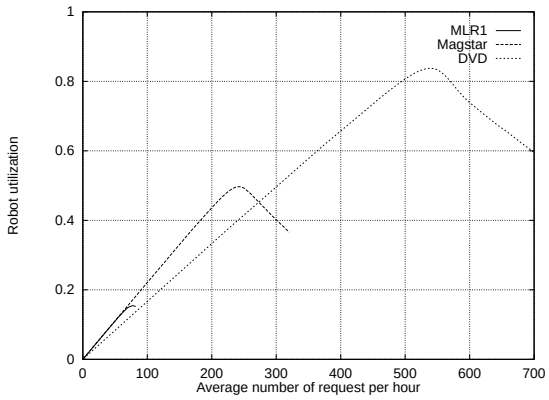
One interesting point worth noting in Figure 10.5(b) and Figure 10.5(c) is that when the load becomes very high for the Magstar and DVD drives, the drive utilization increases less, and the robot utilization actually decreases. Why does this happen? The reason is that as the response time increases, the number of requests waiting for a drive to become idle will increase. The more waiting requests, the more likely there will be multiple requests waiting for the same storage medium to be loaded. Performing two requests on the same medium is less costly than performing two requests on two different media. As a result, the average time the media drive uses per request will go down, and the number of media changes will decrease even if the number of executed requests increases. As an example, using four DVD drives and having a load of 500 requests per hour, the average number of requests per media change will be 1.03. If the load is increased to 600 requests per hour, the average number of requests per medium change will be 1.35. With this increase in the number of concurrent requests to each medium, the robot is able to handle 30 percent more requests with no increase in the utilization. This leads to the drop in the robot utilization curves seen in Figure 10.5(c).



(a) Average response time



(b) Drive utilization



(c) Robot utilization

Figure 10.5 The response time as a function of the number of requests per hour. Each request is for a one minute video sequence. The library unit contains four media drives. The corresponding utilization for the drives and the robot mechanism are as shown in **b)** and **c)**.

	Limit on average response time						Maximum
Technology	10 s	20 s	30 s	1 min	2 min	3 min	throughput
MLR1	—	—	—	—	44	58	86
Magstar	—	—	150	205	228	237	267
DVD	319	450	489	528	551	564	586

(a) Retrieval of one minute long video sequences

	Limit on average response time						Maximum
Technology	10 s	20 s	30 s	1 min	2 min	3 min	throughput
MLR1	—	—	—	—	2	4	9
Magstar	—	—	3	6	7	8	14
DVD	3	6	7	9	11	12	18

(b) Retrieval of one hour long video sequences

Table 10.3 The number of requests a library unit containing four drives can handle during one hour, while keeping the average response time less than the time given in the column header. The rightmost column contains throughput numbers from Section 10.2.2.

Throughput given Response Time Limits

So what is the maximum number of requests a library unit containing four drives can handle using the three different drive technologies? In the previous section we found the throughput by applying a very high load and measuring the number of requests the library was able to handle during one hour. We did this without considering the response time. In this section we show some examples of the throughput for the library units given an upper limit on the average response time.

Based on requirements for a maximum average response time during periods with peak load, the throughput of the library unit can be determined. This corresponds to drawing a horizontal line through Figure 10.5(a). In Table 10.3(a) we show the number for some selected response time limits. For example, if the users allow an average response time of three minutes during peak periods, four MLR1 drives will be able to handle 58 requests, four Magstar drives will be able to handle 237 requests, and four DVD drives will be able to handle 564 requests. If the limit is reduced to one minute, it is not possible to use MLR1 drives. Using

Technology	Lightly loaded	Loaded	Heavily loaded
MLR1	29	44	58
Magstar	119	178	237
DVD	282	424	564

Table 10.4 The number of requests per hour used in the simulations for generating the response time distributions in Figure 10.7. The *heavy load* gives an average response time of 3 minutes. The numbers for the *heavy load* are taken from Table 10.3(a).

Magstar drives, the system will be able to handle 205 requests per hour, and using DVD drives, the system will able to handle 528 requests.

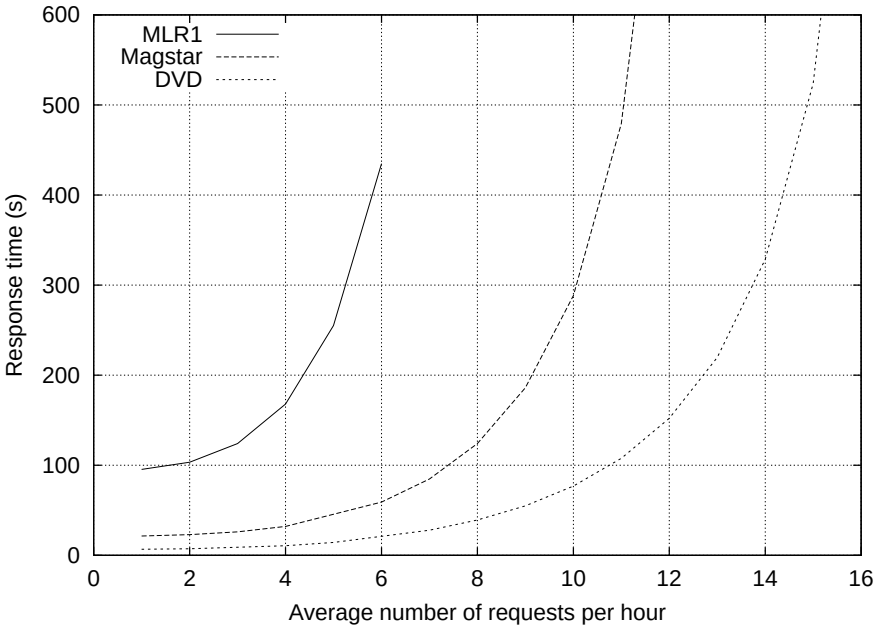
In the last column of Table 10.3, we have included the throughput numbers found in Section 10.2.2. If we compare the throughput number found when not considering any restriction on the average response time, we see that these are much higher. Thus, the utilizing the throughput found in the previous section would lead to very high response times for the library unit.

Retrieval of Long Video Sequences

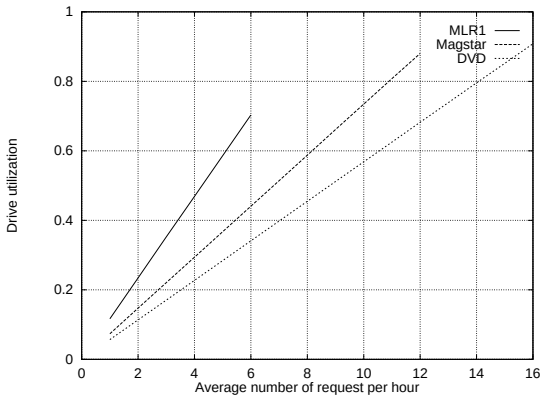
To study the response time when retrieving longer video sequences, we have performed the same simulations for video sequences of one hour. These simulations show that we get very much the same behavior and results as for short video sequences. The average response times for retrieval of video sequences of one hour are included in Figure 10.6. Since retrieving longer video sequences occupies the media drives for a longer period of time, the resource that limits the number of requests the library unit is able to deliver is the media drives. In Table 10.3(b), we have included the number of requests for one hour long video sequences the system is able to handle during one hour. We use the same limits on the response time as we did for retrieval of video sequences of one minute. If we study the numbers in these two tables, we find that the comments we made for retrieval of short video sequences also are valid for retrieval of longer video sequences.

10.3.2 Response Time Distribution

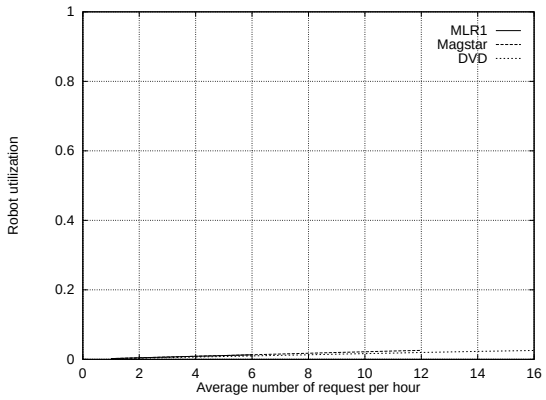
This far we have looked at the *average* response time observed by the user. But how much does the response time vary? What is the maximum response time the user will experience? We answer these questions by giving a few examples of the distribution of the response times for different loads. To represent a *heavily loaded* library unit, we use the number of requests per hour that results in an average response time of 3 minutes. These numbers are given in the last column



(a) Average response time

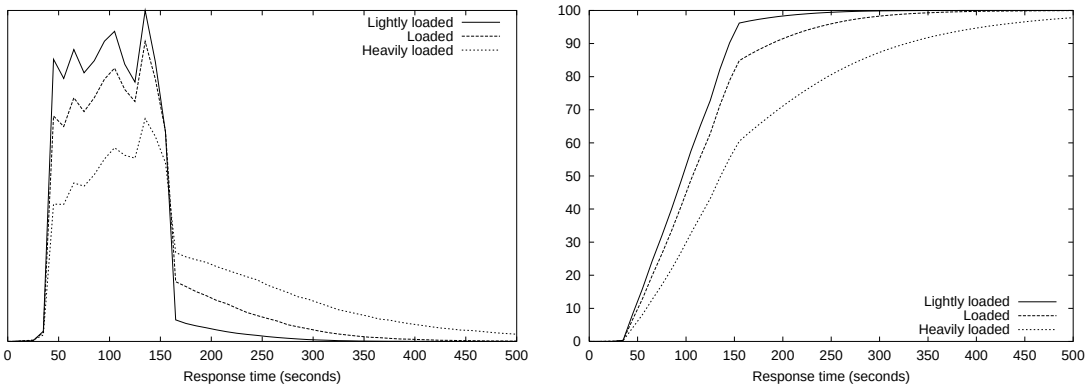


(b) Drive utilization

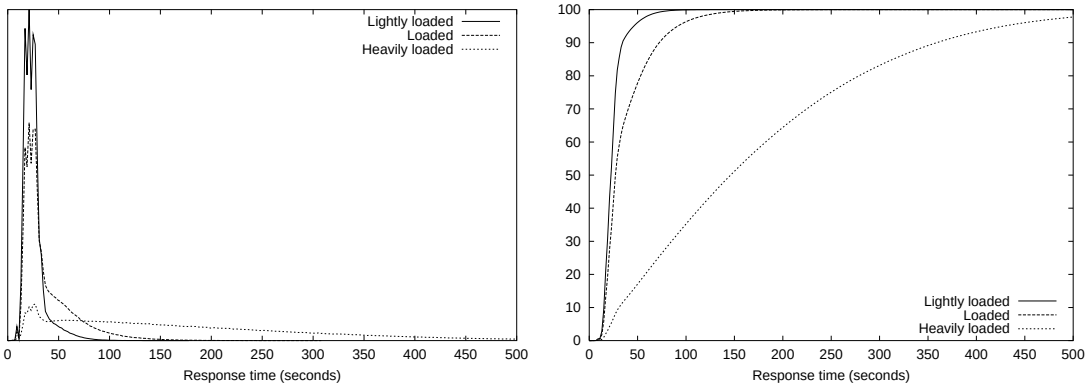


(c) Robot utilization

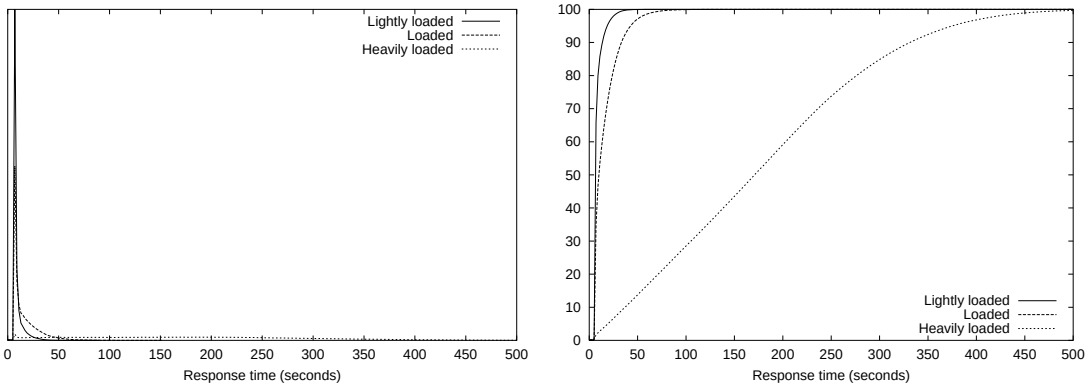
Figure 10.6 The response time as a function of the number of requests per hour. Each request is for a one hour video sequence. The library unit contains four media drives. The corresponding utilization for the drives and the robot mechanism are as shown in **b)** and **c)**.



(a) MLR1 (10 second resolution)



(b) Magstar (2 second resolution)



(c) DVD (2 second resolution)

Figure 10.7 Distribution and cumulative distribution curves for the response time of 1,000,000 requests for video sequences of one minute. The load for each curve is given in Table 10.4. The resolution along the X-axis is given for each of the plots. The figures show only the part of the graphs where the response time is less than 500 seconds.

	Lightly loaded			Loaded			Heavily loaded		
Technology	avg	min	max	avg	min	max	avg	min	max
MLR1	103	1.6	511	120	3.5	781	175	1.1	1218
Magstar	25	0.4	176	38	0.0	306	176	0.3	998
DVD	8.8	0.1	88	16	0.1	150	180	0.1	671

Table 10.5 The average, minimum, and maximum response times (in seconds) for the response time distributions presented in Figure 10.7.

of Table 10.4. Based on these numbers of requests, we select a load of 75 percent of these to represent a *loaded* system, and 50 percent of these to represent a *lightly loaded* system.

For each of these load levels, we simulate 1,000,000 retrievals of video sequences of one minute. For each request, we record the response time. The distributions of the response times are presented in Figure 10.7. Both normal distribution curves and cumulative distribution curves are included in the figure. Table 10.5 contains the average, minimum, and maximum response times for each of the simulations.

The main conclusions that can be made from studying the response time distributions are that the variance is large, and that variance increases as the load on the library unit increases. For a *lightly loaded* library unit, the width of the interval containing most of the response times is much narrower for DVD and Magstar compared to MLR1 due to the much larger seek time of the MLR1 drive. As the load increases, the width of the interval containing most of the response times increases for all three media drives.

For the *heavily loaded* distribution, the average response time is the same for all three media drives. Still, the library unit is relatively more loaded when using DVD and Magstar drives compared to using MLR1 drives. This can be seen from the cumulative distribution curves where DVD and Magstar both have a larger fraction of the requests above 180 seconds compared to MLR1. The reason is that the library unit using MLR1 drives has a high response time also when it is not loaded, and less load is needed on a system utilizing MLR1 drives to increase the average response time to 180 seconds. We observe the same by studying the shape of the distribution curves. For DVD and Magstar, the response times are almost evenly distributed across the response time interval when using the *heavy load*.

	Short video sequences			Long video sequences		
Technology	Lightly	Loaded	Heavily	Lightly	Loaded	Heavily
MLR1	7	11	14	0.5	0.75	1.0
Magstar	29	44	59	1.0	1.5	2.0
DVD	70	106	141	1.5	2.3	3.0

Table 10.6 The number of requests per hour *per tertiary drive* used in the simulations for generating the response time curves in Figure 10.8. The *heavy load* gives an average response time of 180 seconds. The numbers for the *heavy load* are based on the numbers found in Table 10.3.

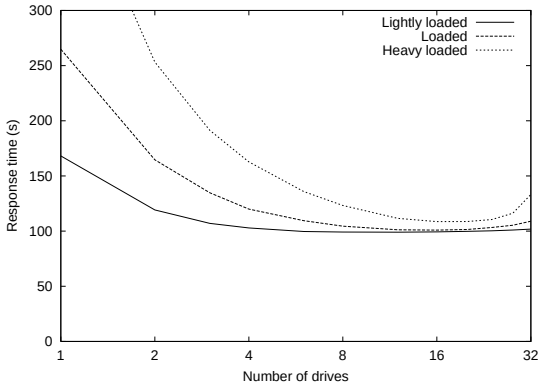
10.3.3 Scalability

This far, we have studied the response times of library units containing four media drives. To handle higher loads while avoiding increase in response time, more drives can be added to the library unit. In this subsection, we study the scalability when adding more drives with regards to the response time. The purpose is to investigate how the response time is influenced when both the number of drives and the load are increased correspondingly. By including more drives, the theoretical transfer capacity of the library unit is increased. The throughput of the robot mechanism stays constant, and might limit how much of the drives' transfer capacity that can be utilized.

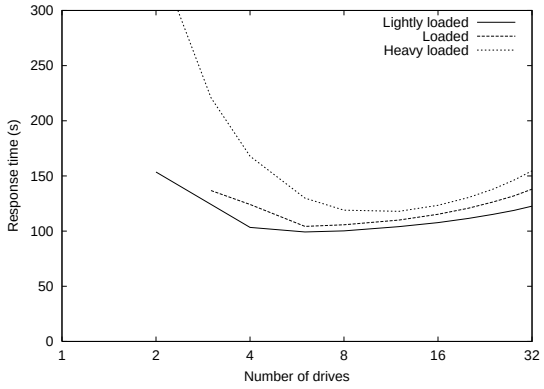
To study how the response time is influenced by the number of drives in the library unit, we perform simulations where we scale both the number of drives and the load on the library unit. In the simulations, we keep the generated load *per tertiary drive* constant. As we increase the number of drives, the user load is increased correspondingly. Thus, if the performance of the library unit scales linearly, we should get approximately the same response time for all combinations of drives and user loads.

The simulations are performed using the same load levels as when studying the response time distribution earlier in this section. The *heavy load* is based on the throughput number for a library unit with an average response time of three minutes. These numbers are found in Table 10.3. Based on these numbers, the numbers of requests per hour in Table 10.6 is used as the load per tertiary drive in the simulations. For each of the three load levels, we run simulations with one to 32 drives in the library unit. The simulation results are presented in Figure 10.8. The results from simulating retrieval of short video sequences are presented in the left part of the figure and the results from retrieval of long video sequences are presented on the right part of the figure.

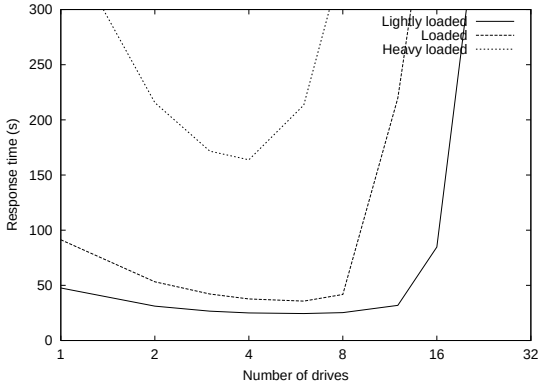
The first observation we make from studying the response time curves is that



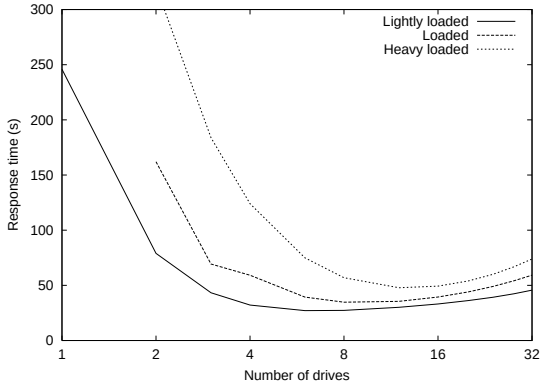
(a) MLR1 – short video sequences



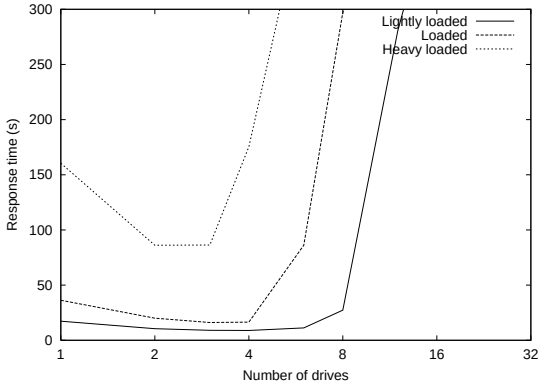
(b) MLR1 – long video sequences



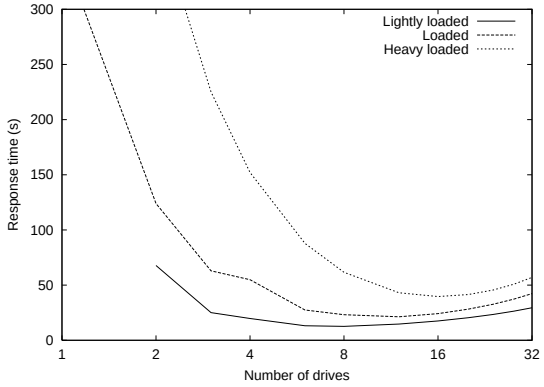
(c) Magstar – short video sequences



(d) Magstar – long video sequences



(e) DVD – short video sequences



(f) DVD – long video sequences

Figure 10.8 Response time as function of the number of drives in the library unit. The load *per drive* is constant. The curves on the left side of the figure are for retrieval of video sequences of one minute, and on the right side are for retrieval of video sequences of one hour.

the response time does not stay constant. Thus, the performance with regards to the response time does not scale linearly as we add more drives to the library unit. All curves show the same shape. For library units containing few drives, the response time is reduced when adding more drives. The reason for the response time decreasing as we add more drives is that it becomes less likely that all drives are busy when a new request arrives. Thus, the probability for new a request having to wait for another request to finish before it can be executed is reduced.

As the number of drives increases further, the response time increases. For retrieval of short video sequences, the reason for the increased response times is that the robot mechanism becomes the bottleneck. The is in correspondence with the results we found in Section 10.2.2 (see Figure 10.3). For retrieval of long video sequences the robot mechanism is not limiting the performance of the library unit. Nor is it the drive resources that lead to increased response time. The reason for the increased response time is contention for the storage media. There is a constant number of media in the library unit. As the number of requests increases, the likelihood for the requested media being occupied by another request increases. Since the amount of time for retrieving a one hour video sequence is rather long, these requests can be delayed for a considerable amount of time. This leads to increased average response time.

Based on the results found from studying the curves in Figure 10.8, it is possible to give an interval for the number of drives per library unit that gives the best response time when the load per drive is kept constant. For retrieval of short video sequences, the curves show that when using MLR1 drives the optimal number of drives is in the interval between six and thirty drives per library unit. For Magstar and DVD, we achieve the lowest response times when the number of drives is between two and eight. For retrieval of long video sequences, the best response times are achieved for library units containing between six and thirty drives.

10.4 Storage Cost and Performance

In the previous sections, we showed that using DVD drives in a library unit gives the best performance both regarding maximum throughput and response times. As Table 10.2 shows, the throughput of a library unit using DVD drives is from two to eight times higher than a library unit using MLR1 drives and from 1.3 to 2.4 times better than a library unit using Magstar drives. We showed that the response time using DVD drives was only one sixteenth of the response time using MLR1 drives, and about 30 percent of the response time using Magstar drives for a lightly loaded library unit.

Usually, there is a correlation between performance and cost. In Section 9.4 we studied the *storage cost* without considering that the library should fulfill any throughput or response time requirements. From the result in Figure 9.16(b) we

concluded that using MLR1 drives and media in the library unit gave the least storage cost. The main reason for MLR1 giving the least storage cost was the much larger storage capacity per medium compared to DVD and Magstar. Using Magstar drives and media was the most costly storage alternative due to the costly drives.

From this, we can conclude that if storage cost is the only (or main) parameter when selecting a storage technology, we should use MLR1 drives and media. If performance is the main parameter, we should select DVD drives and media for storing the video. In real situations, the choice might not be that easy. Most users want both high performance and low storage cost. This makes the selection of which storage technology to use more difficult. In this section we compare the storage technologies with regard to both performance and cost.

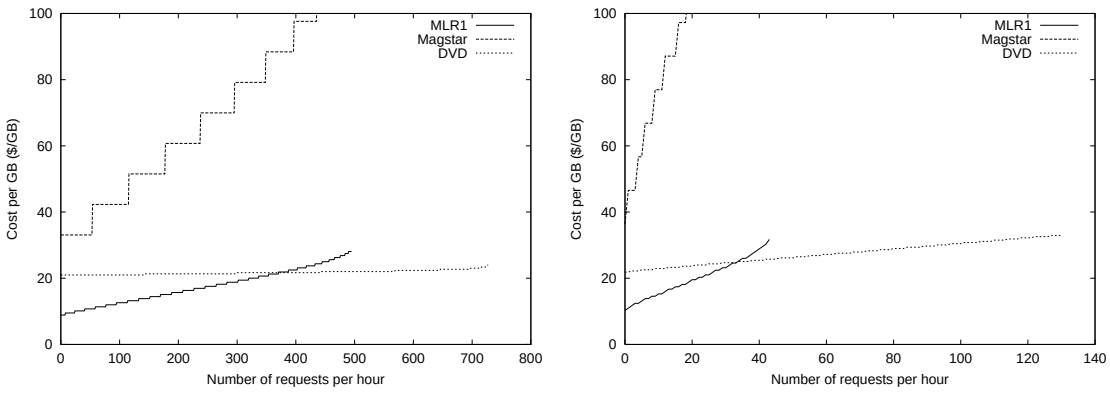
To make a sensible decision on which tertiary storage technology to use in a given project, it is necessary to determine the requirements the system has to fulfill. The main issues are the amount of storage needed for storing the video and the requirements users have regarding throughput and response time. The amount of storage needed is determined by the number of hours of video and the data rate for the video.

The throughput of the system should be high enough to satisfy the maximum number of concurrent video sequences the users may request. As shown in the previous section, the response time is dependent on the load on the system and increases rapidly when we approach the throughput limit of the library unit. In this section, we use the *maximum average response time* that will be accepted during *maximum* load as the response time requirement the system has to fulfill. An alternative would have been to use the average response time as criterion. The reason for not using the average response time in this discussion, is that we then would have to define what would be the average load. As shown earlier in this chapter, the length of the retrieved video sequences influences how the different media drives perform. In this section we show results for retrieving video sequences of one minute and one hour.

The goal of this section is to show how the storage cost is dependent on both the selected technology and the performance requirements put on the library unit. We present the results using the storage cost per GB of video as the main criterion. Informally, we use the following function to compute the storage cost of a library unit given the required throughput and maximum average response time requirement:

$$\textit{storage cost}(\textit{throughput}, \textit{max average response time}) = \frac{\textit{media cost} + \textit{library cost} + n \times \textit{drive cost}}{\textit{amount of video (in GB)}} \quad (10.1)$$

where the throughput is given in requests per hour. n is the number of media drives that is necessary in order to deliver the requested number of video se-



(a) One minute (b) One hour

Figure 10.9 The cost for storing the video for different retrieval rates. The cost is given in dollars per GB of stored video. In the simulations, 180 seconds were used as the value for the *maximum average response time* requirement.

quences per hour while fulfilling the *maximum average response time* requirement. The media cost and library cost is computed for a full library unit.

To find the storage cost as a function of the number of requests the system is able to handle, we use the video archive simulator. As the value for the *maximum average response time* requirement, we use 180 seconds in the experiments in this section. Selecting a different value for this (as long as it is larger than the MLR1 response time of 98 seconds), will have some impact on the results, but not alter any of the main conclusions.

In order to show how the storage cost increases by adding more drives to the library unit, we run simulations with from 1 to 32 media drives in the library unit. In each simulation, we find the average number of requests per hour that is the maximum load the library unit can handle without breaking the *maximum average response time* requirement. For this number of requests, we compute the storage cost using Equation 10.1.

The results of the simulations are presented in Figure 10.9. This figure shows the storage cost per GB as a function of the maximum number of requests the library unit is able to deliver with the given *maximum average response time* requirement. The steps in the curves are related to adding more drives to the library unit in order to handle more requests. Each step corresponds to adding one drive. These curves should be interpreted as follows. When the user knows what the maximum load she wants the library unit to be able to handle during peak load, e.g., 200 requests for short video sequences, she can find (by looking in Figure 10.9(a)), that using MLR1 drives gives the lowest storage cost, $ 16 per GB. Using DVD drives or Magstar drives, the storage cost will be $ 21 and $ 61

per GB, respectively.

The following points must be taken into account when interpreting the results from the simulations presented in this section:

- If the user wants to keep the *maximum average response time* low, it is possible that some of the storage alternatives can not be used at all. E.g., if a user requirement says that the *maximum average response time* should be less than 60 seconds, it will not be possible to use MLR1 drives. As shown in Figure 10.5(a), it is not possible to achieve an average response time of less than about 98 seconds when using MLR1 drives.

- This discussion uses the average response time during the *maximum load* on the library unit to decide how many drives have to be used in order to satisfy the *maximum average response time* requirement. The average response time during the *average load* is not included in this discussion. It is important to remember that even if one library unit using DVD drives and one library unit using MLR1 drives during high load might have the same average response time, when these are lightly loaded, the average response time will always be lower for the system using DVD drives.

- As Table 10.1 shows, the amount of video we are able to store in a library unit is different for the different media types, and thus to store a given amount of video might require a different number of library units depending on which media type we decide to use. This influences the number of requests each library unit has to serve.

The main conclusions we get from studying the curves in Figure 10.9 are as follows. First, the Magstar drive can not compete with the two other media drives regarding the cost of storing the video. Second, as long as the maximum number of requests per hour are reasonable low, using MLR1 drives and media gives the lowest storage cost. As the load increases, using DVD drives and media becomes the least costly storage alternative. For example, if we retrieve short video sequences of one minute each, Figure 10.9(a) shows that as long as the maximum number of requests per hour is less than 370, using MLR1 drives is the least costly. If the maximum number of requests needed to be served is larger, DVD drives should be used. At a maximum load of 370 requests per hour, 21 MLR1 drives are needed. Alternatively, three DVD drives give the same average response time and the same storage cost.

10.5 Conclusions

In this chapter, we have evaluated three different tertiary storage technologies for use in library units that are used for storing and retrieving video data. The different storage technologies and different library unit configurations have been

evaluated with regards to throughput, response time, and cost of storage and delivery. The main conclusions from studying properties of library units using the three tertiary storage technologies are as follows:

- **Throughput.** For retrieval of short video sequences, library units using tertiary disk drives are likely to outperform tape based systems due to the much shorter seek times of the disk drives compared to the tape drives. In our study, we have shown that for retrieval of one minute long video sequences, a library unit containing 2X DVD drives outperforms a library unit containing MLR1 tape drives with a factor of eight. For retrieval of long video sequences, the transfer rate of the tertiary drives becomes the dominating factor. As the length of the retrieved video sequences increases, the impact of the long seek and rewind times of the tape drives becomes less important. We have shown that for long video sequences, the relative throughput ratio between the library units becomes approximately equal to the relative transfer rate ratio of the tertiary drives. For example, retrieving video sequences of one hour, the ratio between a library unit using 2X DVD drives and a library unit using MLR1 drives is reduced to two[1].

- **Response time.** Disk based tertiary drives outperform tape drives with regards to the response times that is possible to achieve for a library unit. Our simulations show that the best response time performance is achieved by using DVD drives, which provide an average response time less than seven seconds. By using the Magstar tape drive, which is optimized for random I/O requests, the average response time will be approximately 22 seconds. Using MLR1 tape drives in the library unit is only a choice if the users of the system accept to have an average response time of about 98 seconds.

- **Scalability.** We have studied the throughput and response times when scaling the number of drives in the library unit. For retrieval of short video sequences, the scalability is limited by the number of media exchanges the robot mechanism is able to handle. The higher number of requests the tertiary drives are able to execute, the lower number of drives is cost efficient to include in the library unit. For retrieval of long video sequences, the robot mechanism is not a bottleneck, and the throughput scales linearly with the number of drives included in the library unit.

- **Scalability and response time.** Our experiments show that for a low number of drives in the library unit, scaling the number of drives and the load correspondingly, reduces the average response time. Thus, to ensure that the library unit is able to handle random variations in the user load without increasing the average response time, it is useful to include at least a few drives in the library unit.

[1]The transfer rate of the 2X DVD drive is 1.8 times higher than the transfer rate of the MLR1 drive.

- **Storage cost.** If the goal is to have a cheapest possible storage alternative, the amount of data that can be stored on each medium and the cost per medium determine which tertiary storage technology to use in the library unit. In our study, the MLR1 technology provides the lowest storage cost due to that it has the lowest storage cost per MB, but also has the highest storage capacity per medium. With a high storage capacity per medium, the cost of the library unit, the robot mechanism, and the drive is amortized across a higher number of MBs of storage. A library unit containing one MLR1 drive will be able to deliver about eight video sequences of one minute per hour or less than one video sequence of one hour per hour given that the average response time should be less than three minutes.

- **Maximize performance.** If the goal is to maximize the performance of the system, the drive technology with the highest throughput should be used. For retrieval of short video sequences, the drive with the shortest mount and seek times should be selected. For retrieval of long video sequences, the drive with the highest transfer rate should be selected. In our study, the DVD drive provides both the shortest seek times and the highest transfer rate. With a requirement that the average response time should be less than three minutes, a library unit containing four DVD drives is able to deliver approximately 560 video sequences of one minute per hour. When retrieving long video sequences, the robot mechanism is not a bottleneck, and the number of drives can be increased. With 32 DVD drives in the library unit, it is able to deliver approximately 130 video sequences of one hour per hour.

- **Cost of storage and retrieval.** As shown in the last section, the choice of storage technology that gives the lowest cost for storing and delivering video data depends on the expected load on the system. In our study, we have shown that as long as the number of accesses is limited, using MLR1 drives in the library unit gives the lowest cost for the system. If the number of requests is high, using DVD drives will be most cost efficient.

The choice of which tertiary storage technology to use for storing digital video will likely be based on several factors. In this chapter we have studied properties related to the performance and cost of three different tertiary storage technologies. We have shown that the choice of storage technology depends on throughput and response time requirements set by the users of the storage system. In addition to the performance and cost criteria that have been studied in this chapter, factors like reliable storage and availability have to be taken into account when selecting the storage technology to use.

Chapter 11

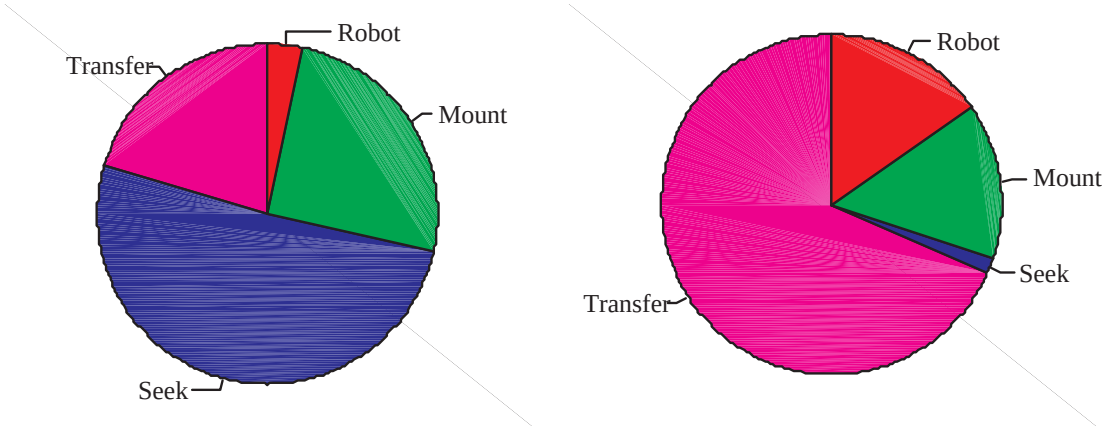
The Effect of Performance Improvements

In the previous chapter, we studied the throughput and the response time of library units using the performance data and models presented in Chapter 9. In this chapter, we continue this work, and study the individual operations performed by the tertiary storage devices and investigate how these influence the total performance of the library unit. We do this by studying how improvements in each of the operations affect the throughput and the response time of the library unit. The purpose of doing this study is to find out which operations should be improved in order to improve either the throughput or the response time when the library unit is used for retrieving video sequences. The results of this study should also be useful for evaluating how new and improved versions of the tertiary storage devices will influence the performance.

This study should help us answer questions like: What will the effects of reducing the seek time or increasing the transfer rate of the tertiary media drives be? What will the effect of increasing the speed of the robot mechanism be? As an example, assume we have a tertiary video archive using MLR1 drives (or 2X DVD drives). Further, assume this archive is storing short news sequences. How will the throughput and response time be affected by upgrading to a newer tape drive in the MLR/SLR series (or to a faster DVD drive)?

11.1 Improving Performance Parameters

The main operations performed by the tertiary drives and the robot mechanism were presented in Section 9.3. As Table 9.3 shows, there are large differences between the three tertiary media drives. Figure 11.1 shows the relative amount of time each of the main operations takes when retrieving a one minute video sequence using an MLR1 drive and a DVD drive. As the figure shows, there are major differences in how large fraction of the total access time each of the



(a) MLR1 (b) DVD

Figure 11.1 The distribution of the total time to retrieve a one minute video sequence using a MLR1 drive and a DVD drive.

operations takes when retrieving short video sequences. For retrieval of long video sequences, the transfer time will be dominating.

In order to study the effect each of the operations has on the total performance, we run simulations where we double the performance of the operation. The following four cases are studied:

1. **Seek** and **Rewind times**. The seek times and rewind times are halved.
2. **Transfer rate**. The transfer rate of the media drives is doubled.
3. **Mount** and **Unmount times**. The time used by the drives for mounting and unmounting the media is halved.
4. **Robot mechanism**. The time the robot mechanism uses for moving a storage medium between a drive and the media store is halved.

For each of these cases, we study the effect it has on the throughput and the response time of the library unit. In the simulations, the library unit contains four media drives and the amount of video as given in Table 10.1.

11.2 Throughput

By improving the performance of one or several of the operations the library unit performs, we expect the throughput to be increased. To study how much each of the operations contribute to the increase in throughput, we run simulations

Improved	Short video sequences			Long video sequences		
performance variable	MLR1	Magstar	DVD	MLR1	Magstar	DVD
None	77	250	561	8.6	13.6	17.5
Seek time	114	308	564	8.9	13.7	17.5
Transfer rate	83	288	634	15.7	26.2	34.6
Mount time	84	268	592	8.7	13.6	17.6
Robot performance	79	273	674	8.6	13.6	17.6
Combined	154	500	1123	17.1	27.2	35.1

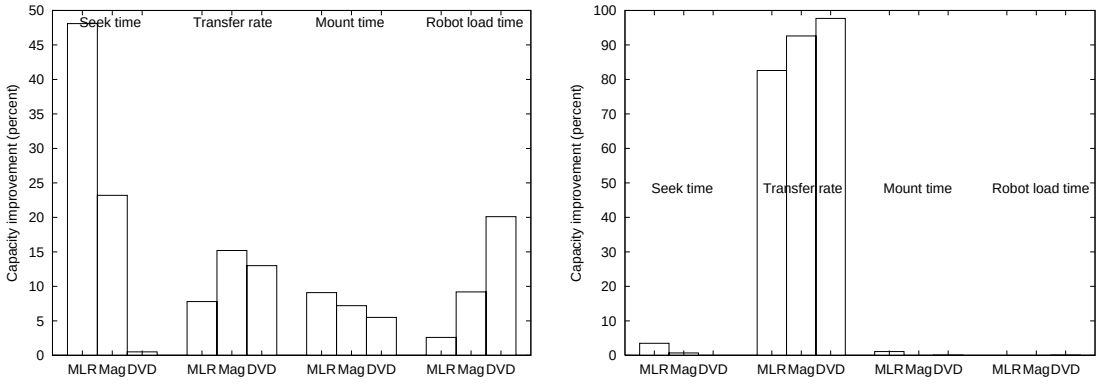
Table 11.1 The throughput of a library unit containing four media drives when doubling the performance of the four main performance variables. The throughput is given in number of video sequences a library unit containing four media drives can deliver during one hour.

for each of the four cases presented in the previous section. A closed queueing simulation model with 20 concurrent requests is used as load. We also perform one simulation where none of the operations are improved and one simulation where the performance of all operations are doubled. The first of these two is used as the basis for computing the relative throughput improvement in percent. The last is used for validating that the total throughput of the system is doubled when doubling the performance of all operations.

The throughput numbers from the simulations are presented in Table 11.1. To get a more comprehensible view, the relative performance improvements are presented in Figure 11.2. This figure shows how much the throughput can be increased in percent by doubling the performance of each of the operations.

Figure 11.2(a) shows the results when the users are retrieving short video sequences of one minute. It shows clearly the effect of tape being a sequential medium and DVD a random access medium. By doubling the seek speed, the throughput of the MLR1 drive is increased by almost 50 percent. Using the Magstar drive, the throughput is increased by about 20 percent. For DVD the effect of improving the seek speed has hardly any effect.

For DVD the greatest improvements come from improving the speed of the robot mechanism and the transfer rate. The reason that increasing the speed of the robot has much larger effect for the DVD than MLR1 and Magstar, is that the time spent by the robot transferring the medium from the media store to the drive counts for a larger portion of the total access time for DVD than for the two tape based systems (see Figure 11.1), but also due to that the robot mechanism is highly utilized when the system is highly loaded (see Figure 10.3(d)). Similarly, the reason for the much larger improvements for DVD by doubling the transfer



(a) Short video sequences

(b) Long video sequences

Figure 11.2 The throughput improvements in percent by doubling the performance of the four main performance variables of the library unit. The graphs are based on the throughput numbers in Table 11.1.

rate is due to that it counts for a larger part of the total access time.

As Figure 11.2(b) shows, the transfer rate of the media drives is the only performance variable that has any real impact on the throughput for retrieval of long video sequences. By doubling the transfer rate of the media drives, we are close to doubling the throughput of the library unit.

11.3 Response Time

To see how improvements in the performance of the media drives and the robot mechanism influence the response time observed by the users, we run simulations where the load on the system is varied from a very lightly loaded system to a heavily loaded system. The load is generated using an open queueing simulation model where the interarrival time between the requests was drawn from an exponential distribution as explained in Section 9.1.6. We use the same strategy as in the previous section, and run simulations for each of the four cases presented earlier in the chapter.

When there is no resource conflicts between concurrent requests, the response time will be the amount of time the robot uses to move a medium from the media store to an idle drive ($robot_t$), the time used by the media drive to mount the medium ($mount_t$), and the time the drive uses to seek to the start of the requested video sequence ($seek_t$). Thus, for an idle (or lightly loaded) library unit, the response time will be:

$$response\ time_{idle} = robot_t + mount_t + seek_t$$

It should be noted that for a small fraction of the requests, the storage medium will already be in the media drive, and thus the robot and mount times will be zero.

As the request rate increases, the probability that requests have to wait due to resource conflicts increases. To account for this, we have to include the amount of time a request is delayed due to resource conflicts ($queue_t$) in the response time:

$$response\ time = robot_t + mount_t + seek_t + queue_t$$

When we improve the performance of one or more of the operations the library unit performs, the average values for the components in the above expression for the response time will change. If we label the average values experienced using the original performance parameters with the subscript $t0$ and the values experienced using improved performance parameters with the subscript ti, we can define the improvement in response time as:

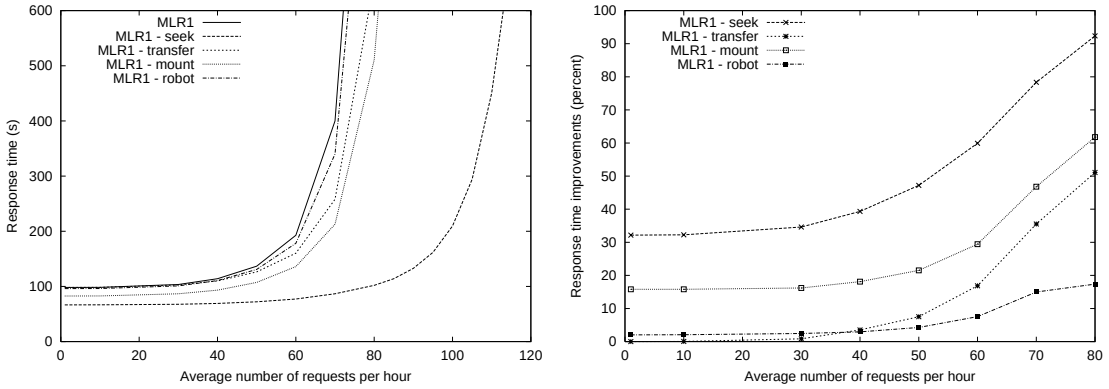
$$response\ time\ improvement = \frac{robot_{ti} + mount_{ti} + seek_{ti} + queue_{ti}}{robot_{t0} + mount_{t0} + seek_{t0} + queue_{t0}} - 1 \qquad (11.1)$$

We use this equation when analyzing the results obtained in the simulations.

11.3.1 Retrieval of Short Video Sequences

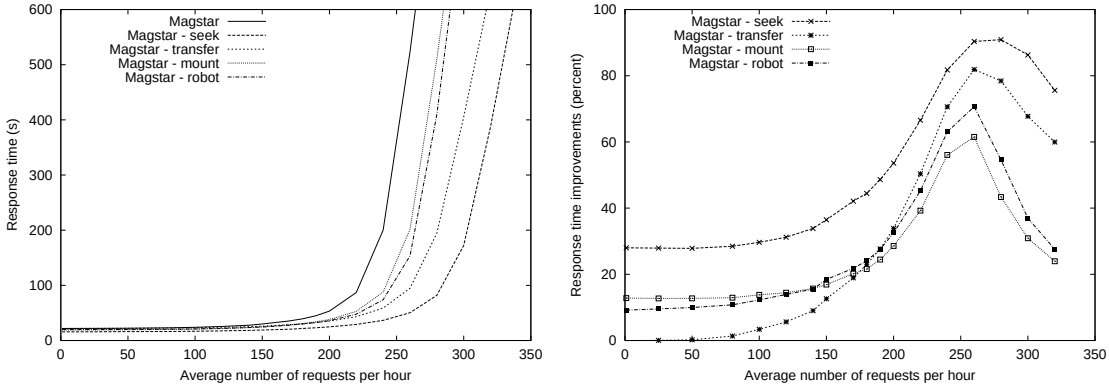
The results for retrieval of *short* video sequences of one minute using MLR1, Magstar and DVD drives are presented in Figure 11.3, 11.4, and 11.5 respectively . For each of the three drive types, we show both the actual response times and the relative improvements in the response times as a function of the load on the system. It is important to note that the x-axes do not cover the same load interval for the two graphs included for each drive type. For the response times, we show the entire simulation interval. The response time improvements are relative to the response time we get without any performance optimization. Thus, the second graph only covers the load interval where it was possible to get response time measurements using the non-optimized library unit.

We start by looking at the improvements experienced by a lightly loaded library. When this case there will be very little resource contentions and the time each request spends waiting for resources (the $queue_t$ in Equation 11.1) will be close to zero. For the tape based systems, the seek time is the performance parameter that has the greatest impact on the response time. As Figure 11.3(b) and Figure 11.4(b) show, by halving the seek time of the drives, the response time is reduced by about 30 percent for the MLR1 and Magstar drives. For library units using DVD drives, reducing the seek time has hardly any effect (see Figure 11.5(b)). For the tape based systems, the second most important performance



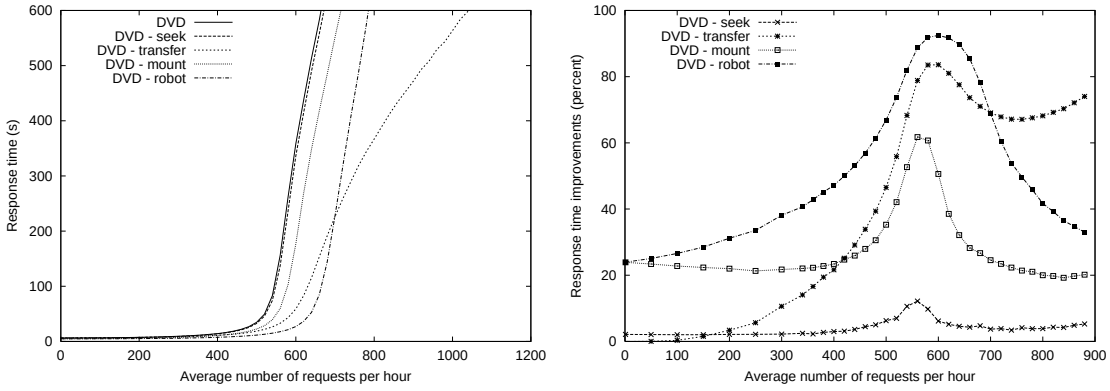
(a) Response time (b) Response time improvement

Figure 11.3 Improvements in average response time when using MLR1 drives and doubling the performance of the main operations of the drives and/or robot device. The lengths of the retrieved video sequences are one minute. The library unit contains four media drives.



(a) Response time (b) Response time improvement

Figure 11.4 Improvements in average response time when using Magstar drives and doubling the performance of the main operations of the drives and/or robot device. The lengths of the retrieved video sequences are one minute. The library unit contains four media drives.



(a) Response time

(b) Response time improvement

Figure 11.5 Improvements in average response time when using DVD drives and doubling the performance of the main operations of the drives and/or robot device. The lengths of the retrieved video sequences are one minute. The library unit contains four media drives.

parameter is the mount/unmount time. By halving this, the response time can be reduced by about fifteen percent. For DVD, the two performance parameters having the greatest influence on the response time are the mount/unmount time and the time used by the robot transporting the media. Halving each of these can lead to an improvement of about 25 percent on the response time.

As the figures show, the improvement increases as the load on the system increases. To explain the shape of the response time curves, we use Figure 11.6. This figure shows a generalized response time improvement curve. We have divided the load on the x-axis into three intervals. The first interval covers the load levels where the library is lightly loaded. For a lightly loaded system with few requests queued due to lack of resources, the improvement only affects the currently executed requests. In the second load interval, the relative response time improvement increases as the load is increased. For a loaded system where requests are delayed due to resource conflicts, the reduced execution time also benefits all requests waiting for a tertiary library drive to become idle. Thus, the $queue_{ti}$ in Equation 11.1 increases less than the $queue_{t0}$. For example, improving the transfer rate has no effect on the response time when the system is lightly loaded, but as the load on the system increases, improving the transfer rate reduces the amount of time the drive uses to transfer data, and thus reduces the amount of time other requests have to wait for the drive to become idle. At the end of interval 2 in Figure 11.6, the library unit with the original performance parameters is close to reaching its throughput limit, leading to huge response times, while the optimized system still is able to handle the load without too large re-

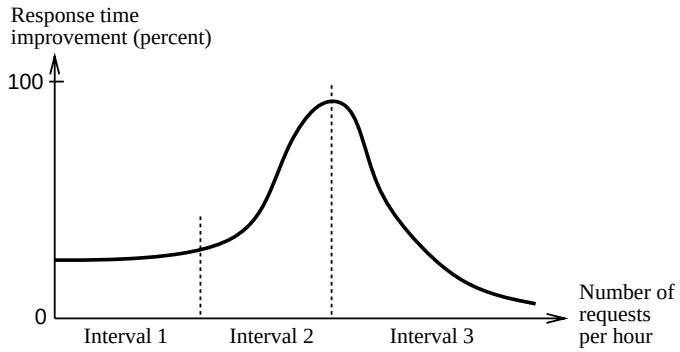


Figure 11.6 A generic response time improvement curve for retrieving short video sequences as a function of the load on the library.

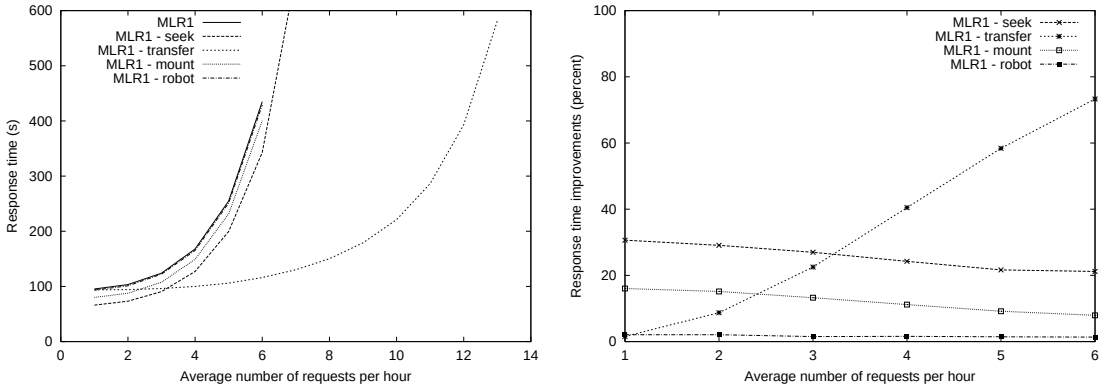
sponse times. In interval 3, the improved system is also reaching its throughput limit, and the relative response time improvement will decrease. It is worth noting that this is only visible in the response time improvement curves for Magstar and DVD (see Figure 11.4(b) and Figure 11.5(b)). The reason we do not observe this for the MLR1 drive, is that the slope of the response time curve is much larger for the MLR1 drive than the other two drives when it becomes heavily loaded.

11.3.2 Retrieval of Long Video Sequences

The response times and the response time improvements when the library units are used for retrieving long video sequences of one hour is presented in Figure 11.7, 11.8, and 11.9. In Section 11.2 we showed that for retrieval of long video sequences, the only performance parameter having any impact on the throughput of the system was the transfer rate. For a lightly loaded system, the transfer rate has very little impact on the response time. As long as the system is lightly loaded, the seek time is the most important performance parameter, which influences the most on the response time for the MLR1 drive and the Magstar drive. For the DVD drive, the mount time and the robot time are the performance parameters that have the largest impact on the response time. As the system becomes more loaded, the transfer rate is the most important performance parameter for all three drive types. This is a result of the transfer time being the main cost of accessing a one hour video sequence.

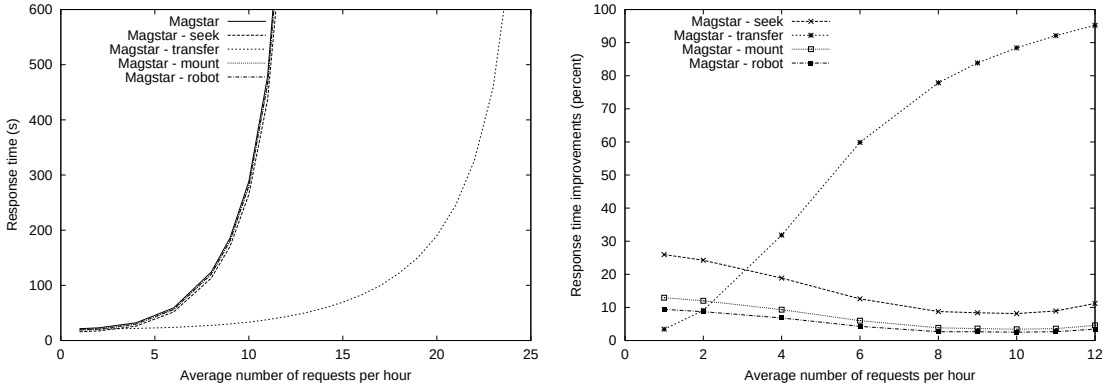
11.4 Conclusions

In this chapter, we have investigated how improvements in the main operations of the media drives and the robot mechanism influence the throughput and the



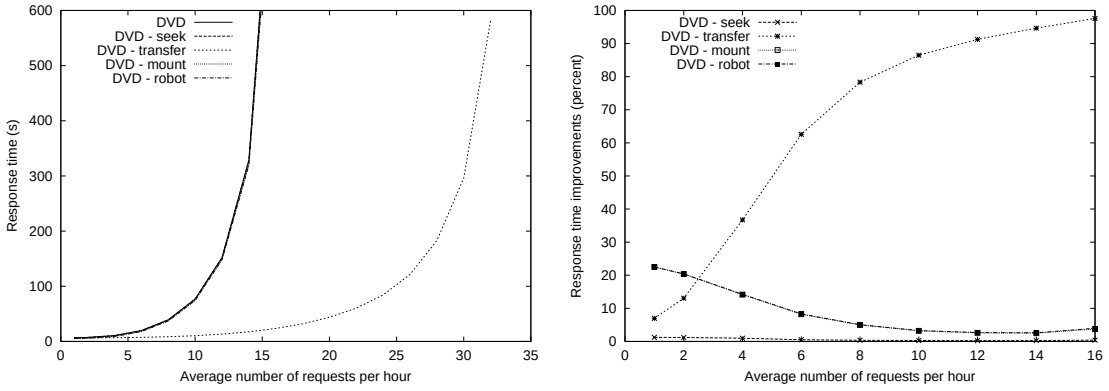
(a) Response time (b) Response time improvement

Figure 11.7 Improvements in average response time when using MLR1 drives and doubling the performance of the main operations of the drives and/or robot device. The lengths of the retrieved video sequences are one hour. The library unit contains four media drives.



(a) Response time (b) Response time improvement

Figure 11.8 Improvements in average response time when using Magstar drives and doubling the performance of the main operations of the drives and/or robot device. The lengths of the retrieved video sequences are one hour. The library unit contains four media drives.



(a) Response time

(b) Response time improvement

Figure 11.9 Improvements in average response time when using DVD drives and doubling the performance of the cost variables of the drives and/or robot device. The lengths of the retrieved video sequences are one hour. The library unit contains four media drives. Note that the curves for improvements in mount and robot exchange times in Figure 11.9(b) overlap.

response times of a library unit. Table 11.2 contains an overview of the potential improvements that can be achieved by improving each of the main operations.

The main results can be summarized as follow:

- Retrieving short video sequences using tape drives, the seek time is the operation that should be improved in order to increase the throughput and reduce the response time.
- Retrieving short video sequences using DVD drives, the transfer rate is the operation that should be improved in order to increase the throughput of the library. To reduce the average response time, reducing the mount and robot times gives the greatest benefit.
- To increase the throughput when retrieving long video sequences, only the transfer rate of the media drives has any effect. For a lightly loaded library using tape drives, improvements in the seek and mount times have the largest effects on the average response time. For a DVD based library, improvement in the mount and robot times have the largest impact on the response times. As the load on the library increases, the improvement in the transfer rate has largest impact on the average response time for both tape and DVD drives.

This shows that there is no single operation that is the most important to improve in order to increase the performance of a library unit when using it for

Performance	Tape		DVD	
parameter	Throughput	Response time	Throughput	Response time
Seek time	+++	+++	0	0
Transfer rate	++	0/++++[a]	++	0/++++[a]
Mount time	+	++	+	+++
Robot time	+	+	+++	+++

(a) Retrieval of one minute long video sequences

Performance	Tape		DVD	
parameter	Throughput	Response time	Throughput	Response time
Seek time	0	+++	0	0
Transfer rate	++++	0/++++[a]	++++	+/++++[a]
Mount time	0	++	0	++
Robot time	0	+	0	++

(b) Retrieval of one hour long video sequences

[a]Highly dependent on the load.

Table 11.2 The table gives an overview of the improvement potential in throughput and response time when doubling the performance of the four main operations of a library unit. + = 5-10%, ++ = 10-20%, +++ = 20-50%, ++++ = more than 50% performance improvement.

storing video. It depends on the type of media drive, the size of the requested video sequences, the load on the library, and whether the goal is to increase the throughput of the library unit or to reduce the average response time. The results presented in this chapter should be useful when evaluating how improvements in tertiary storage technologies will affect the throughput and the response time of a tertiary storage system.

Chapter 12

Effect of Access Distributions on the Performance of a Video Archive

We are learning to generate data more rapidly than we can move it.

Reagan Moore, Thomas A. Prince and Mark Ellisman
in *Data-Intensive Computing and Digital Libraries*
(Moore, Prince and Ellisman, 1998)

In the previous chapters we studied the performance of a tertiary storage system consisting of one library unit. We showed that the different storage technologies have very different performance characteristics and costs when used for storing and retrieving video sequences. In this chapter we continue the study of using tertiary storage in video archives. We investigate how the access distribution for the video sequences influence the performance of the video archive.

This far, we have assumed that all video sequences stored in the video archive are accessed with the same access probability. The simulated users have picked a random video sequence to watch. In real life, most users do not want to watch a random video sequence. Most likely, they have some preference for which or what kind of video they want to watch. The selected video sequence(s) may also be the result of a database query. These user preferences will make some video sequences more popular than other, and these will be accessed more frequently. For example, in a television news archive, the last month's news stories will likely be accessed more frequently than older news stories. In a video-on-demand server, the latest Oscar winners are likely to be requested a lot more often than the Norwegian movie "Hud"[1].

To model this kind of user behavior, *non-uniform* access distributions must be used. In this chapter, we use three different access distributions and evaluate the effect these have on the throughput and the response time of a video archive.

[1]Famous for being the Norwegian movie seen by fewest people in movie theaters.

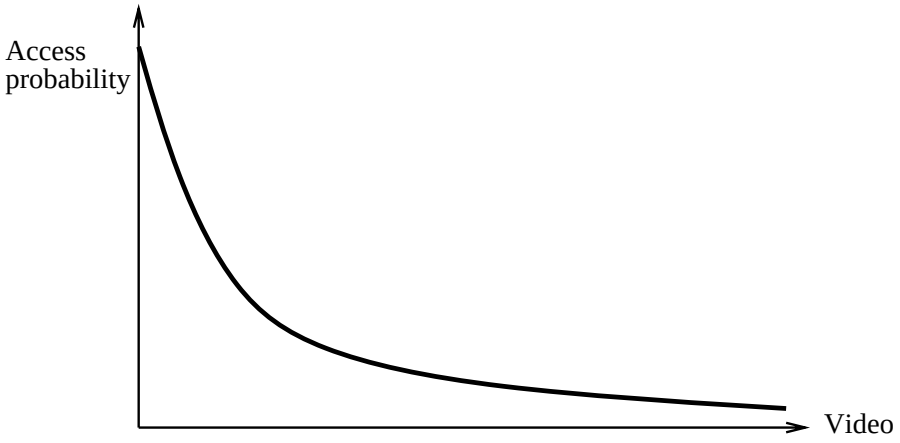


Figure 12.1 An example of a non-uniform access model. The video sequences are ordered by decreasing probability of being accessed.

12.1 Access Distributions

A access model contains information about how often the different videos are accessed. It should contain the access probability of the individual videos and information about how related videos are accessed. To model realistic user behavior and locality of the accesses to a video archive is in general difficult, as it is highly dependent on the use of the archive. The access pattern changes over time, and even different parts of a day may have different access patterns (Griwodz, Bär and Wolf, 1997).

In most applications, some of the video sequences are more popular than others, and are consequently accessed more frequently. To model such a situation, a non-uniform access model as the one illustrated in Figure 12.1 must be used. Most research involving the access pattern for video use statistical models for the access distribution. The most frequently used model is the Zipf's Law distribution. Chervenak (1994) provides one example that shows how a movie rental history approximately matches the Zipf's Law distribution. Another frequently used model for the access distribution is to divide the videos into multiple popularity classes (Chan and Tobagi, 1999). Within each popularity class, the access probability is uniform.

The goal of this section is not to study access distributions, but to investigate the influence non-uniform access distributions have on the performance of the tertiary storage system, and in particular how it affects the throughput and response times. In this study, the following access distributions are used:

- **Uniform Access Distribution.** All video sequences in the archive have the same access probability. This is the access distribution we have used in all

Partition	Storage percent	Access percent
Hot	1	81
Warm	9	9
Cold	90	10

Table 12.1 Access probabilities for the 90/9/1 access distribution.

experiments in the previous chapters.

- **80/20 Access Distribution.** The video sequences are *logically* partitioned into two sets. The first set contains 20 percent of the video sequences. The remaining 80 percent of the video sequences are included in the second set. Of all the requests, 80 percent goes to the first of the two sets. As a consequence, the video sequences in the first set will be accessed sixteen times more often than the video sequences in the second set. We refer to the two sets as the *warm* and *cold* set, respectively.

- **90/9/1 Access Distribution.** To get an even more skewed access distribution, we use an access distribution where the data accesses are grouped into three partitions. The video sequences are *logically* partitioned into three sets, which contain the *hot*, *warm*, and *cold* video sequences. The *cold* set contains 90 percent of the video sequences. Of the last ten percent, one percent is assigned to the *hot* set and nine percent to the *warm* set. The access probabilities are summarized in Table 12.1. With this access distribution, the video sequences in the *hot* set will be accessed 81 times as often as the video sequences in the *warm* set, and 729 times as often as the video sequences in the *cold* set.

We are not doing any allocation to storage media or library units based on access probability. The *hot* and *warm* video sequences will be randomly distributed among the less popular video sequences. Storage allocation based on access probability will be evaluated in Chapter 13.

12.2 Video Archive based on Tertiary Storage

With non-uniform access distributions, the number of accesses to the individual storage media will vary. Due to this, it is likely that the utilization of the media drives within a library unit becomes more uneven. Further, if the video archive contains multiple library units, it is likely that the load on the different library units may become skewed. In order to include potential effects of uneven loads between library units, we extend the simulated tertiary storage system to include multiple library units.

Length	Technology	Media	Libraries	Drives
One minute	MLR1	1730	9	36
	Magstar	4546	23	92
	DVD	4800	24	96
One hour	MLR1	2000	10	40
	Magstar	5000	25	100
	DVD	5000	25	100

Table 12.2 The number of media and library units required for storing 10000 hours of 5 Mbit/s video. Each library unit contains four media drives.

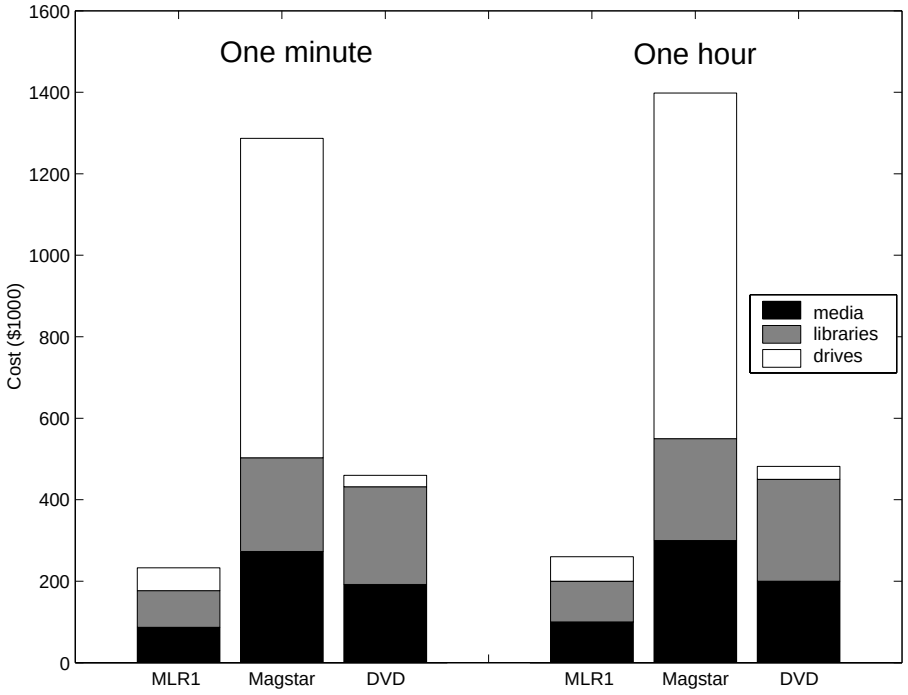


Figure 12.2 The cost of the video archive configurations given in Table 12.2.

In this and the following chapters, we simulate a video archive that stores 10000 hours of video. The data rate of the digitized video is 5 Mbit/s. Totally, this requires approximately 20.5 TB of tertiary storage. To store this amount of video, the number of storage media will be from about 1730 when using MLR1 tapes, up to about 4800 when using DVD-R disks. When each library unit can contain up to 200 storage media, we have to use nine library units when using MLR1 tapes, 23 library units when using Magstar tapes, and 24 library units when using DVD disks. The number of required media and library units are given in Table 12.2. This table contains the number of required media and library units for storing both short video sequences of one minute and long video sequences of one hour. The numbers are slightly larger when storing long video sequences since the storage media will not be completely filled[2]. In the simulations performed in this chapter, each library unit contains four media drives. The cost of the storage media, drives, and library units is presented in Figure 12.2.

12.3 Throughput

To investigate the effect non-uniform access distributions have on the throughput of a tertiary storage system, we compare the throughput of the tertiary storage system presented in the previous section using each of the three access distributions. As a criterion for determining the maximum throughput, we use the number of requests that can be retrieved while keeping the average response time less than three minutes.

This far, the allocation of video sequences to storage media has not influenced the performance of the system, since the access distribution has been uniform. We have been able to determine the response time of a given configuration and a given load by running a single simulation. Since we introduce non-uniform access distributions for the video sequences and use a *random* allocation of the video sequences to storage media and library units, the allocation will be different from one simulation to another. Thus, the initial storage allocation may influence the performance of the tertiary storage system. To avoid this to influence the simulation results, we run multiple simulations for each configuration and load, and use the mean of the result from the individual simulations. To determine the throughput numbers presented in this section, the simulations are performed using *Simulation Strategy III* presented in Section 9.2.

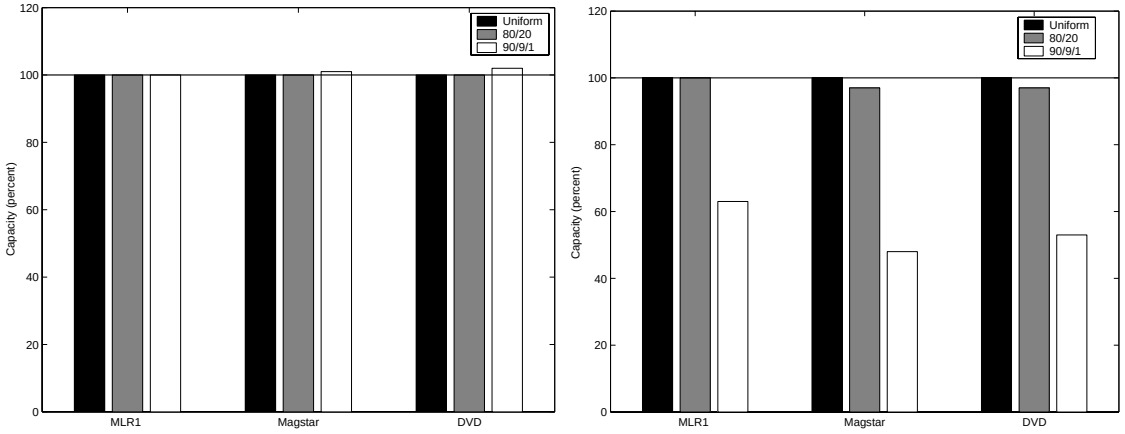
Retrieval of Short Video Sequences

In Table 12.3, the throughput for the storage system is presented for retrieval of short video sequences of one minute and long video sequences of one hour. Figure 12.3 contains a graphical presentation of the relative throughput of the system

[2]For the same reason, these numbers also differ from the numbers given in Table 9.11.

Length	Tertiary technology	Access distribution		
		Uniform	80/20	90/9/1
One minute	MLR1	527	528	529
	Magstar	5468	5476	5518
	DVD	13554	13553	13853
One hour	MLR1	41	41	26
	Magstar	223	217	106
	DVD	311	302	166

Table 12.3 Number of requests per hour that can be satisfied with an average response time of less than 3 minutes. The results are presented graphically in Figure 12.3.



(a) One minute (b) One hour

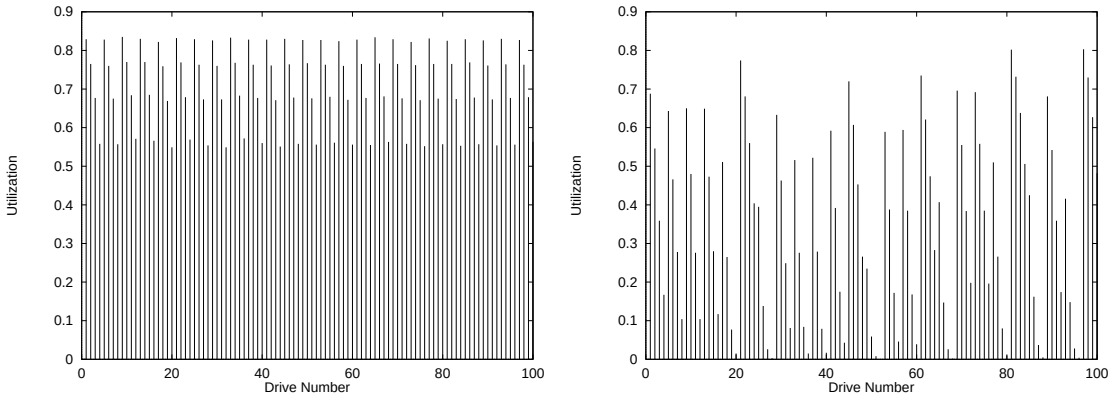
Figure 12.3 Graphic presentation of the throughput of the tertiary system for the three access distributions. The throughput is given in percent of the throughput for the uniform access distribution. The percentages are computed using the numbers in Table 12.3.

when using each of the three access distributions. As Figure 12.3(a) shows, the throughput is largely uninfluenced by the access distribution when the tertiary storage system is used for retrieval of short video sequences. As the skewness increases, there is a small increase in the throughput. This is due to increased probability of multiple users accessing the same video sequence, and thus multiple users can be served during each load cycle. For instance, for the throughput given in Table 12.3 and using DVD drives, the number of requests served per loaded DVD is 1.16 when the access distribution is uniform. With a 90/9/1 access distribution, this is increased to 1.24 requests per load of a DVD. The increase in throughput is largest for DVD drives. The DVD system serves many more requests at the throughput limit than the tape based systems, and the probability of having multiple requests for the same DVD disk is higher than for the tape based systems.

Retrieval of Long Video Sequences

For retrieval of long video sequences, Figure 12.3(b) shows that the throughput decreases as the access distribution becomes very skewed. For the 90/9/1 access distribution, the throughput of the storage system is approximately halved compared to the uniform access distribution. The main reason for the much lower throughput is that the *hot* video sequences are *randomly* distributed over the storage media. As a result, some storage media will be *hotter* than other storage media. For these storage media there will be a queue of waiting requests, which results in increased average response time and reduced overall throughput. Thus, it is neither lack of drive resources nor robot mechanism resources that are the limiting factor for the throughput.

The limiting factor when retrieving long video sequences will be the drives that have to serve the few storage media that contain multiple hot video sequences. To illustrate this, we use Figure 12.4. This figure shows the average utilization of each of the DVD drives in the storage system for two simulations using respectively the uniform and the 90/9/1 access distribution. The DVD system contains 25 library units, each having four DVD drives. Both simulations have an average response time of approximately three minutes. The number of requests handled with the uniform access distribution was 311. With the 90/9/1 access distribution it was reduced to 166 requests per hour. Figure 12.4(a) shows that when the access distribution is uniform, the load on the library units and the drives in the system is approximately equal. The reason we have a pattern where the utilization of the drives goes down for every fourth drive, is that each library unit contains four drives. In our simulator, the first drive in the library unit is used if it is idle, thus the first drive will be more utilized than the next drive in the library unit. Figure 12.4(b) shows the utilization of the drives when the 90/9/1 access distribution is used. As indicated by this figure, the utilization of the drives vary much. Some drives are highly utilized, while some are



(a) Uniform access distribution

(b) 90/9/1 access distribution

Figure 12.4 The average utilization of each of the one hundred DVD drives in the storage system when using two different access distributions. Both simulations have an average access time of approximately three minutes.

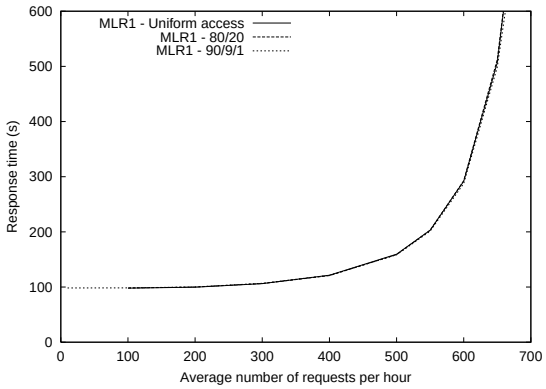
hardly used at all. If we group the drives by the library unit they are contained in (i.e., make groups of four drives), the figure shows that the utilization of the library units also vary much. The highest utilized drives are in the library units that have been allocated most video sequences from the *hot* partition. Since we use an upper limit on the average response time, these drives are limiting the throughput of the tertiary storage system.

Several strategies might be used to overcome this problem. The hot video sequences can be replicated on multiple storage media (Chervenak, 1998; Hillyer et al., 1999), the initial allocation of videos to storage media and library units can be based on access probability, the videos can be reallocated based on the access probability (Nemoto and Kitsuregawa, 1999), or the hot videos can be cached using a disk-based video cache.

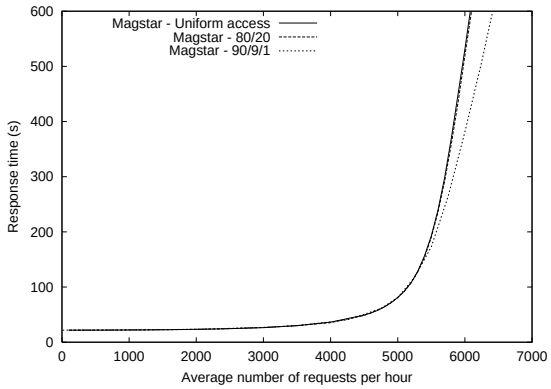
12.4 Response Time

To investigate the effect non-uniform access distributions have on the response time, we perform the same experiments as we did when evaluating the response time of a single library unit. But due to the non-uniform access distribution and the random allocation of videos to storage media, we have to run multiple simulations for each user load. This is explained as *Simulation Strategy II* in Section 9.2.

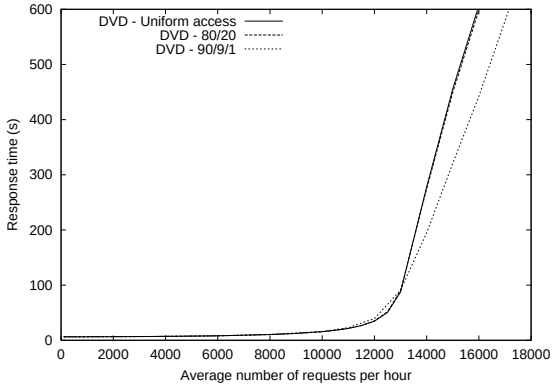
The response time curves as a function of the load for the three access distributions are given in Figure 12.5 and Figure 12.6. Figure 12.5 contains the response time curves for retrieval of short video sequences of one minute using each of



(a) MLR1

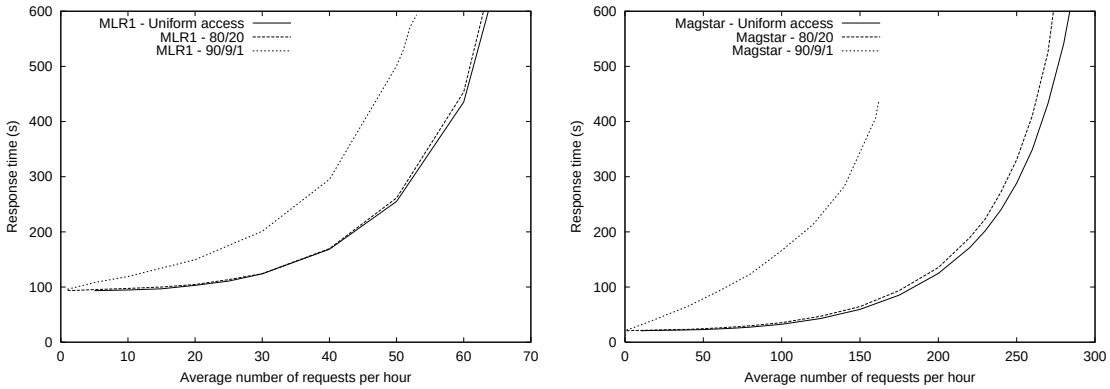


(b) Magstar



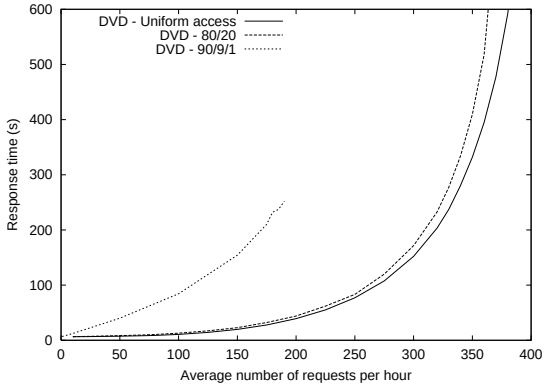
(c) DVD

Figure 12.5 Response times for retrieval of one minute video sequences using the different access distributions.



(a) MLR1

(b) Magstar



(c) DVD

Figure 12.6 Response times for retrieval of one hour video sequences using the different access distributions.

Access distribution	MLR1		Magstar		DVD	
	Light load (158)	Loaded (527)	Light load (1640)	Loaded (5468)	Light load (4066)	Loaded (13554)
Uniform	98.9	179.5	22.8	179.6	7.1	179.1
80/20	98.8	178.0	22.7	176.1	7.1	179.9
90/9/1	98.9	174.6	22.7	165.4	7.1	145.8

(a) One minute

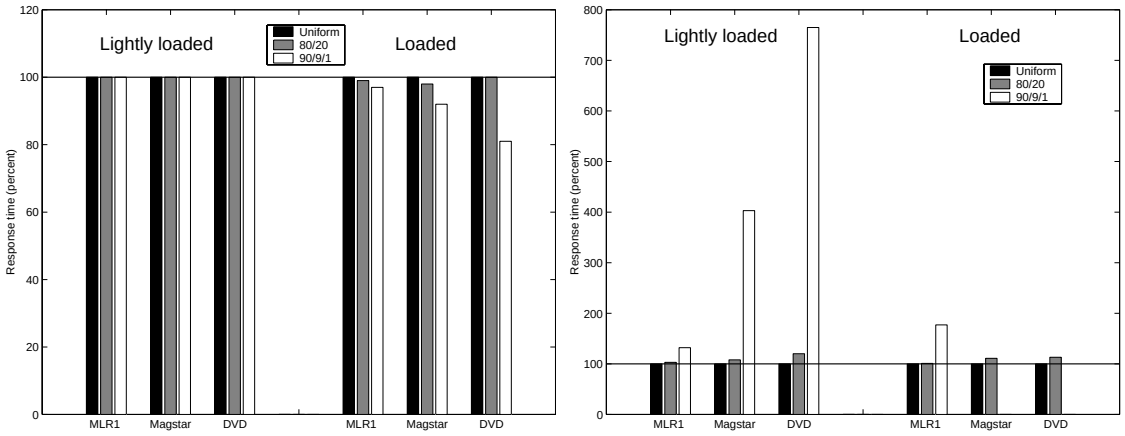
Access distribution	MLR1		Magstar		DVD	
	Light load (12)	Loaded (41)	Light load (67)	Loaded (223)	Light load (93)	Loaded (311)
Uniform	95.5	174.6	25.1	179.6	10.0	178.0
80/20	98.1	176.9	27.2	198.7	12.0	201.4
90/9/1	126.2	308.2	101.2	—	76.5	—

(b) One hour

Table 12.4 Response times in seconds for the two cases using each of the three access distributions. The number of requests per hour is included in parentheses in the top of each column. The results are presented graphically in Figure 12.7.

the three media drives, while Figure 12.6 contains the same results when retrieving long video sequences of one hour. By studying these curves, we get the same main results as for the throughput. For retrieval of short video sequences, skewed access distributions have a small positive effect on the average response time. For retrieval of long video sequences, skewed access distributions increase the average response time, particularly for high request rates.

To be able to quantify more precisely the influence the two non-uniform access distributions have on the average response time, we compare response times for two cases. These cases represents a *lightly loaded* and a *loaded* storage system. For the *loaded* system, we use the number of requests the system is able to handle with an average response time of three minutes with the uniform access distribution (i.e., the throughput numbers for the uniform access distribution in Table 12.3). As load for a *lightly loaded* system, we use a request rate that is thirty percent of the request rate for the loaded system. The request rates are given in the heading of Table 12.4. This table contains the average response times when applying these



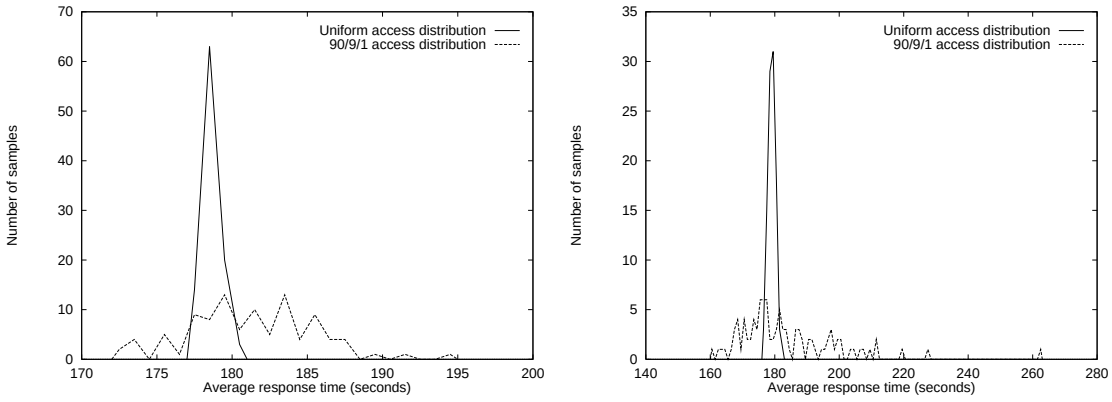
(a) One minute (b) One hour

Figure 12.7 Graphic presentation of the response times for tertiary system for the three access distributions. The response times are given in percentage of the response time for the uniform access distribution. The percentages are computed using the number in Table 12.4. Note that for retrieval of long video sequences, bars are not included for the Magstar and DVD drive in the loaded case.

loads to the archive. The results are presented graphically in Figure 12.7.

As shown in Figure 12.7(a), for retrieval of short video sequences, the average response time is largely uninfluenced by the access distribution. For the *lightly loaded* case, we observe no change in the response time (at least not more than what can be considered within the uncertainty of the simulation results). For the *loaded* case, the average response time is improved as the access distribution get more skewed. For the 90/9/1 access distribution, the average response time is reduces by three percent using the MLR1 drive, eight percent using the Magstar drive, and 19 percent using the DVD drive compared to the uniform access distribution.

For retrieval of long video sequences of one hour, the situation is exactly opposite. Just as for the throughput, as the access distribution becomes more skewed, the average response time increases. For the *lightly loaded* case, the average response time increases by 32 percent using the MLR1 drive, 300 percent using the Magstar drive, and 670 percent using the DVD drive when comparing the results for the 90/9/1 access distribution to the similar results for the uniform access distribution. As explained when discussing the throughput of the storage system, this is due to some of the storage media becoming hotter than the other media, and the drives serving these media become the limiting resource in the system. For the *loaded* case the difference between the uniform and the 90/9/1 access



(a) One minute (b) One hour

Figure 12.8 The distribution of the average response times for one hundred *random* storage allocations using the *uniform* and the 90/9/1 access distribution. The DVD drive was used in these simulations.

distributions is even larger. For Magstar and DVD the response times becomes infinite for the 90/9/1 access distribution, i.e., the storage system is not able to handle the load.

12.5 Variance in Performance

As stated earlier in this section, when the access distribution no longer is uniform and the video sequences are allocated randomly to storage media, the performance of the storage system becomes dependent on the initial allocation to storage media. Thus, the average response time and throughput of the system vary depending on how good (or bad) the initial allocation of the video sequences to storage media was.

To visualize this, Figure 12.8 presents the distribution of the average response time for one hundred random storage allocations. We use the DVD drive in this example, since the skewness of the access distribution has the highest influence on the performance when using DVD drives. To show how the skewness of the access distribution increases the spread of the average response time, we run simulations using the uniform and the 90/9/1 access distribution for retrieval of short and long video sequences. As load on the system we used the throughput numbers found in Table 12.3, thus this experiment is run for a loaded archive. For a less loaded archive, the spread in the average response time would be lower.

The main points to observe in Figure 12.8 are that the spread of the average response times becomes larger as the access distribution becomes more skewed,

and that the spread is much larger for retrieval of long video sequences compared to retrieval of short video sequences. As a consequence of this, it can be important to carefully choose the initial strategy for allocating video sequences to storage media if the access distribution is non-uniform. If not, one might create a tertiary storage system which performs worse than expected due to *bad luck* in the initial allocation of the video sequences to storage media and library units.

12.6 Conclusions

In this chapter we have studied how non-uniform access distributions influence the throughput and response time of a video archive based on tertiary storage. The main results of our study can be summarized as follows:

- **The uncertainty in the performance increases.** The average performance becomes dependent on the initial allocation of video sequences to storage media and library units when the access distribution is non-uniform. The more skewed the access distribution is, the larger will the spread of the performance be.
- **Retrieval of short video sequences.** Non-uniform access distributions has little effect on the throughput of the tertiary storage system when used for retrieval of short video sequences. Neither has it any effect on the response time as long as the system is lightly loaded. As the system becomes more loaded, the response time is slightly improved due to increased probability that multiple users are requesting the same or multiple video sequences on the same storage medium.
- **Retrieval of long video sequences.** As the access distribution becomes more skewed, the throughput of the storage system is reduced and the average response time is increased. This is due to the fact that some of the storage media become *hotter* than the other. The drives that have to serve these storage media become the limiting resource in the storage system.

The main conclusion is that non-uniform access distributions may lead to lower performance compared to the performance when the accesses are uniformly distributed. In the next chapter, we continue the study of non-uniform access distributions to see if it is possible to take advantage of having knowledge about how the videos are accessed. We study allocation strategies for videos to storage media based on the access probability and investigate how this can improve the performance of a video archive.

Chapter 13

Allocation Based on Access Probability

In the experiments performed in the previous chapters, the video sequences was allocated to storage media and library units without considering the access probability of the video sequences. Thus, the hot video sequences have been randomly allocated to storage media. This *random* allocation can produce every possible allocation of the video sequences, from allocations that make the storage system perform very well to allocations that make the system perform poorly. The previous chapter showed that as the access distribution became more skewed, the throughput and average response time became more unpredictable.

In this chapter, we investigate strategies for allocating video sequences to storage media and library units, and the effect these have on the performance of a video archive. The main objective is to determine what is the best allocation strategy to use in a digital video archive. We also attempt to determine whether the choice of an allocation strategy depends on the usage of the archive or not, and whether it depends on the criteria we use for comparing the allocations strategies. We use throughput and average response time as the two main criteria for comparing allocation strategies.

The results from this study should be useful for improving the throughput and response time of a tertiary storage system. This chapter provides information about which allocation strategies are best for different usage patterns. The results can be used both when performing the initial allocation of videos to storage media and library units, as well as when adding new videos to the archive. It can also be used for reallocating the video sequences in order to improve the performance of the video archive. By using one of the recommended allocation strategies, allocations that lead to low throughput, high response times, and poor utilization of the drives and robot mechanisms can be avoided.

In this chapter, we only study *static* allocation of videos to storage media and library units. As soon as a video is stored on a medium and this medium has been allocated to a library unit, it will not be reallocated. In most video archives,

the access probability of the videos will likely change over the time (Griwodz et al., 1997). To avoid that this reduces the performance of the archive, it might be necessary to reallocate videos to other media or other library units. One proposed strategy for doing this is the *Hot declustering* strategy proposed by Nemoto and Kitsuregawa (1999). This strategy is used for migrating storage media between library units in order to improve the load balancing in the tertiary storage system.

We do not consider replication of videos on multiple storage media. In our study, only one copy of each video is stored in the tertiary storage system. Hillyer et al. (1999) study how replication of hot data objects can be used for improving the performance of a tertiary library unit. Their conclusion is that with no replication, the hot data object should be stored at the *beginning* of the tape, while with replication (either partial or full replication), the replicated data object should be stored at the *end* of the tape. A more dynamic replication strategy is *Hot replication* (Nemoto and Kitsuregawa, 1999). During the initial writing of a tape, the tape is not completely filled. An area on the end of the tape is left unused. When there is an idle tape drive in the system, this is used for filling the end of the tape with data that is currently hot. Thus, in this way they claim to reduce the average access time since they are able to cluster together hot data objects. As the access probability changes, the data objects stored on the end of the tape can be overwritten. The authors specify that for this to improve the performance it is required that the tapes have multiple load/eject zones. With the serpentine model presented in Chapter 6, it should be possible to extend the *Hot replication* strategy to be used with serpentine tape.

13.1 Allocation Strategies

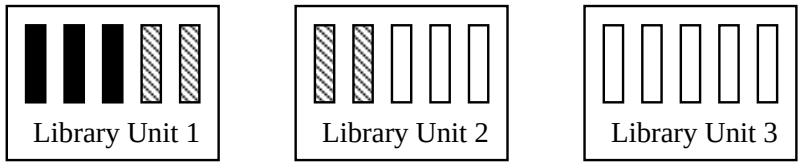
In our study of different allocation strategies, we use the allocation strategies presented in this section. Each allocation strategy must address the following three problems:

1. For each video sequence, decide which storage medium it should be stored on.

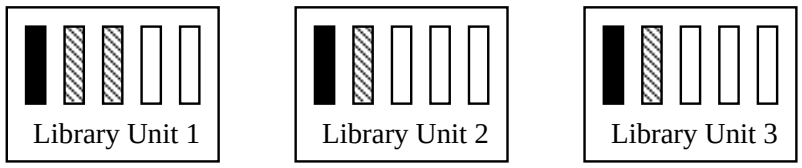
2. Given the set of video sequences to be stored on a storage medium, decide the location the video sequences should have within the medium.

3. For each storage medium, decide which library unit this should be stored in.

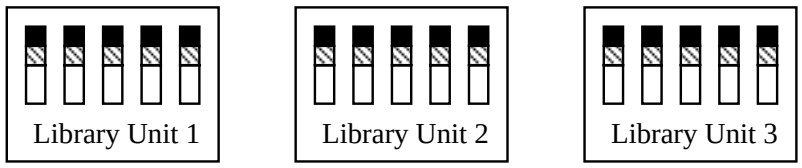
In the presentation of the allocation strategies we use an access distribution where the video sequences are partitioned into three sets based on access probability. As in the previous chapter, we call these the *hot*, *warm*, and *cold* partitions:



(a) Hot Library



(b) Hot Media



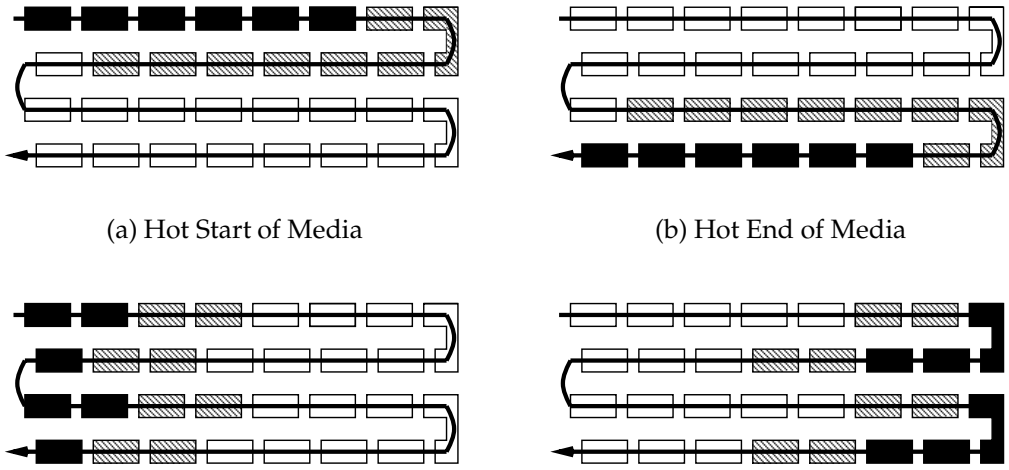
(c) Uniform distribution of hot, warm, and cold video sequences

Figure 13.1 Illustration of three strategies for allocating video sequences to storage media and library units. The archive consists of three library units and fifteen storage media. $\frac{3}{15}$ of the video sequences are hot (marked as black) and $\frac{4}{15}$ are warm (marked as grey). The remaining $\frac{8}{15}$ are cold (marked as white).

1. **Random.** Initially, the hot and warm video sequences are randomly distributed between the cold video sequences. They are then allocated round-robin to storage media and library units. This is the allocation strategy that we have used in the previous chapters.

2. **Uniform.** The hot and warm video sequences are uniformly distributed among the library units and the storage media. Within each storage medium the most frequently accessed video sequences are stored with equal distance between them.

3. **Hot Library.** The video sequences are sorted based on access probability. The video sequences is then assigned to storage media by filling one medium completely before starting on the next medium. The storage me-



(a) Hot Start of Media

(b) Hot End of Media

(c) Hot Minimum Seek Time

(d) Hot Maximum Seek Time

Figure 13.2 Example illustrating allocation of video sequences to a serpentine tape using different allocation strategies. The color coding of the video sequences are as in Figure 13.1.

dia are assigned to library units by filling one library unit completely before starting on the next library unit. As a result, the hottest video sequences are stored in one (or a few) of the library units. This is illustrated in Figure 13.1(a).

4. **Hot Media.** Just as in the Hot Library strategy, the hottest video sequences are allocated to *hot* storage media. The storage media are allocated round-robin to storage libraries, as illustrated in Figure 13.1(b). This strategy is similar to the allocation strategy for data objects to disks in tertiary storage systems proposed by Christodoulakis et al. (1997).

The following four allocation strategies distribute the hot, warm, and cold video sequences evenly over the storage media (see Figure 13.1(c)). They differ in the placement of the hot and warm video sequences on each medium. The strategies are illustrated in Figure 13.2.

5. **Hot Start of Media.** The hottest video sequences are allocated at the *logical beginning* of the storage medium as shown in Figure 13.2(a). This is the same strategy as proposed by Christodoulakis et al. (1997) for allocating data objects to tapes in storage systems using tape drives that always rewind to the physical beginning of the tape.

6. **Hot End of Media.** The hottest video sequences are allocated at the *logical end* of the storage medium as shown in Figure 13.2(b).

Allocation strategy	Abbreviation
Random	RAN
Uniform	UNI
Hot Library	HLI
Hot Media	HME
Hot Start of Media	HSM
Hot End of Media	HEM
Hot Minimum Seek Time	MIN
Hot Maximum Seek Time	MAX

Table 13.1 Abbreviations used in figures and tables for the different allocation strategies.

7. **Hot Minimum Seek Time.** The hottest video sequences are allocated to the storage medium based on the *shortest seek time* from the medium's loading point. This is illustrated in Figure 13.2(c).

8. **Hot Maximum Seek Time.** The hottest video sequences are allocated to storage medium based on the *highest seek time* from the loading point as illustrated in Figure 13.2(d).

To make figures and tables more readable, we use abbreviations for the name of the allocation strategies in this chapter. These abbreviations are defined in Table 13.1.

It is worth noting that for the DVD drives, the *Hot Start of Media* and *Hot Minimum Seek Time* strategies are identical due to the linear relationship between positions on the DVD and the corresponding seek times. Similarly, the *Hot End of Media* and the *Hot Maximum Seek Time* strategies are identical when used for DVD drives. It is also worth noting that in order to use the *Hot Minimum Seek Time* and *Hot Maximum Seek Time* allocation strategies for the tape based systems, it is necessary to have a model which provides accurate *seek time* estimates for the tapes. The model presented in Chapter 6 is used for the MLR1 drive and the model presented in (Hillyer and Silberschatz, 1998) is used for the Magstar drive.

To investigate how the performance of the tertiary storage system is influenced by the allocation strategy, we compare the performance of each of the allocation strategies to the performance achieved using the *Random* allocation strategy. In the experiments presented in this chapter, we use the 90/9/1 access distribution presented in Section 12.1. As in the previous chapter, the tertiary storage archive contains 10000 hours of 5 Mbit/s video and each library unit contains four media drives (see Table 12.2). We perform the same experiments as we did when studying the effect of different access distributions. First, we study the influence

	One minute			One hour		
Allocation	MLR1	Magstar	DVD	MLR1	Magstar	DVD
RAN	529	5518	13853	26	106	166
UNI	532	5488	13619	27	113	182
HLI	80	328	860	4	10	15
HME	1260	8430	15025	6	58	94
HSM	651	6429	13752	37	119	183
HEM	491	4912	13587	32	108	183
MIN	1003	6883	13752	37	119	183
MAX	295	4656	13587	18	108	183

Table 13.2 Number of requests that can be satisfied with an average response time of less than 3 minutes. These numbers are presented graphically in Figure 13.3.

the allocation strategy has on the throughput of the storage system. Second, we present the results from investigating the effect different allocation strategies have on the response time. All experiments are performed for retrieval of both short video sequences of one minute and long video sequences of one hour.

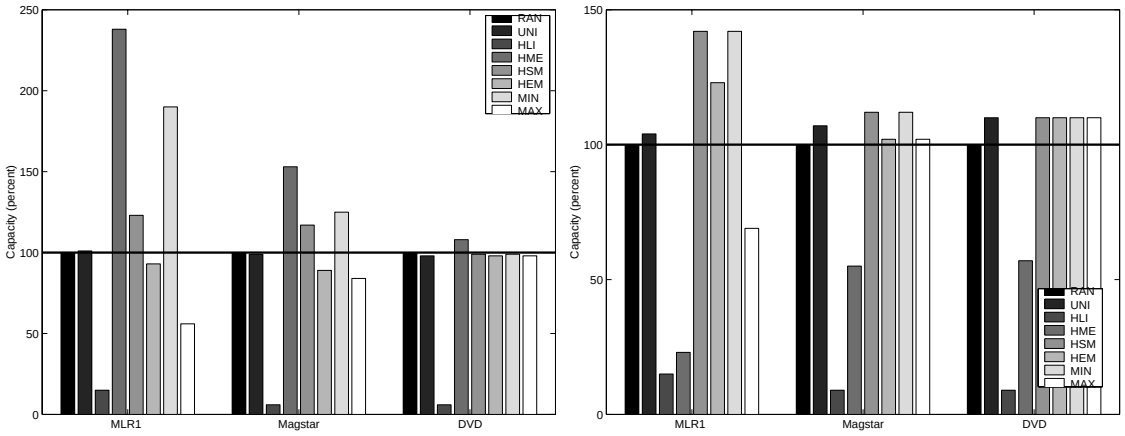
13.2 Throughput

We use the same criterion for the throughput as in the previous chapter. By use of simulations, we determine the maximum number of requests the storage system can perform while keeping the average response time less than three minutes.

The throughput for the video archive found using this requirement is presented in Table 13.2. To make the performance of each of the allocation strategies more easy to compare, we present the relative throughput graphically in Figure 13.3. The throughput using each of the allocation strategies are presented in percent relative to the throughput using the *Random* allocation strategy.

The reason for the difference in throughput for the different allocation strategies is in how the resources of the library units are utilized. The allocation strategies influence the *seek time* differently and some of them may lead to *contention* for either one or more storage media, drives, or the robot mechanism. From studying the throughput in Figure 13.3, the following can be seen:

- If we order the allocation strategies by the throughput numbers, the order is the same for all three media drives when the workload is the same. It should be noted that the order when retrieving short video sequences is different from the order when retrieving long video sequences. The relative



(a) Short, one minute

(b) Long, one hour

Figure 13.3 Throughput improvement by using the six allocation strategies. The improvement is given in percent improvement compared to using no allocation, and are based on the throughput numbers in Table 13.2.

differences between the allocation strategies are largest for the MLR1 drive and least for the DVD drive. This is a consequence of the larger seek times of the tape based systems. Thus, achieving a *good* allocation strategy is more important when using sequential storage devices with long seek times.

- In all cases, the *Hot Library* strategy gives the lowest throughput since the library unit containing the hot video sequences becomes the bottleneck.

- For retrieval of short video sequences, the *Hot Media* strategy gives the highest throughput. The reason is that for a large number of the requests, the media containing the hot video sequences is already loaded in one of drives. As long as the drives containing the *hot* media manage to execute the requests, this is the best allocation strategy for short video sequences. We will later show that this allocation strategy is not the best when we increase both the number of drives in the library units and the load on the system (see Figure 13.8 in Section 13.4).

- For retrieval of long video sequences, the *Hot Media* strategy is a bad strategy to use. The reason is that for retrieval of long video sequences, each drive is occupied for a rather long time (in the order of 20 minutes). In our simulator, each request is delivered independently of other request for the same video sequence. There is no batching of requests for the same video sequence. Multiple hot video sequences stored on the same medium makes

the drive containing this medium become the limiting resource in the system.

- For retrieval of long video sequences, the *Hot Start of Media* and *Hot Minimum Seek Time* strategies give the highest throughput. The reason for getting the same throughput numbers for these, is that for long video sequences, only one hot video sequence is stored on each medium, and the position with the shortest seek time will be the start of the medium.

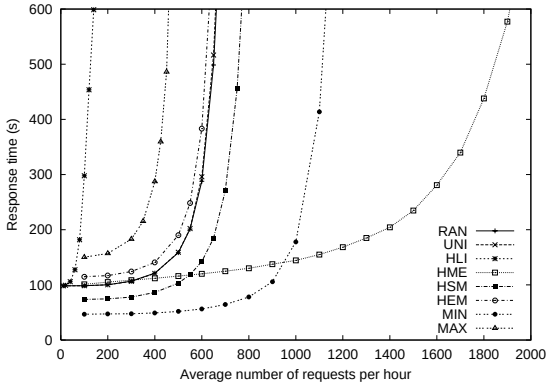
The last four allocation strategies (*Hot Start of Media*, *Hot End of Media*, *Hot Minimum Seek Time*, *Hot Maximum Seek Time*) only differ in placement of the hot (and warm) video sequences on the medium. For DVD all of these give approximately the same throughput. This is due to the very short seek times of the DVD drive, and thus the placement of the video sequences is of little importance for the throughput. For the Magstar and MLR1 drives this is not the case. The placement of the more popular video sequences within the medium has great influence on the throughput. For the MLR1 the throughput when using the *Hot Minimum Seek Time* allocation strategy is 3.4 times higher than when using the *Hot Maximum Seek Time* allocation strategy for retrieval of one minute long video sequences. For retrieval of long video sequences, the throughput of the *Hot Minimum Seek Time* allocation strategy is 2.1 times higher than the throughput of the *Hot Maximum Seek Time* allocation strategy. This last number might be surprising, since for retrieval of long video sequences the seek time should have less influence on the throughput. The reason for the low throughput when using the *Hot Maximum Seek Time* allocation strategy for the MLR1 is that the long seek time of approximately two minutes is close to the requirement that the average response time should be less than three minutes.

13.3 Response Time

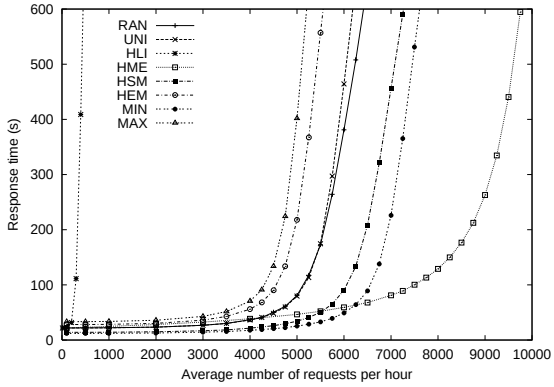
To investigate the effect of different allocation strategies on the response time, we performed the same experiments as when studying non-uniform access distributions in the previous chapter. For each of the allocation strategies, we find the average response time as a function of the load. These response time curves are presented in Figure 13.4 for retrieval of one minute long video sequences, and in Figure 13.6 for retrieval of one hour long video sequences.

Retrieval of Short Video Sequences

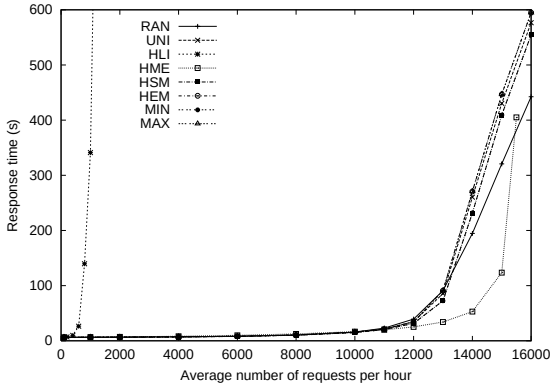
One of the questions stated in the beginning of this chapter was to study if the best allocation strategy was dependent on the load. For retrieval of short video sequences this is the case. Figure 13.4 shows that for most of the load interval, the allocation strategy that gives the lowest response times for the tape-based systems



(a) MLR1

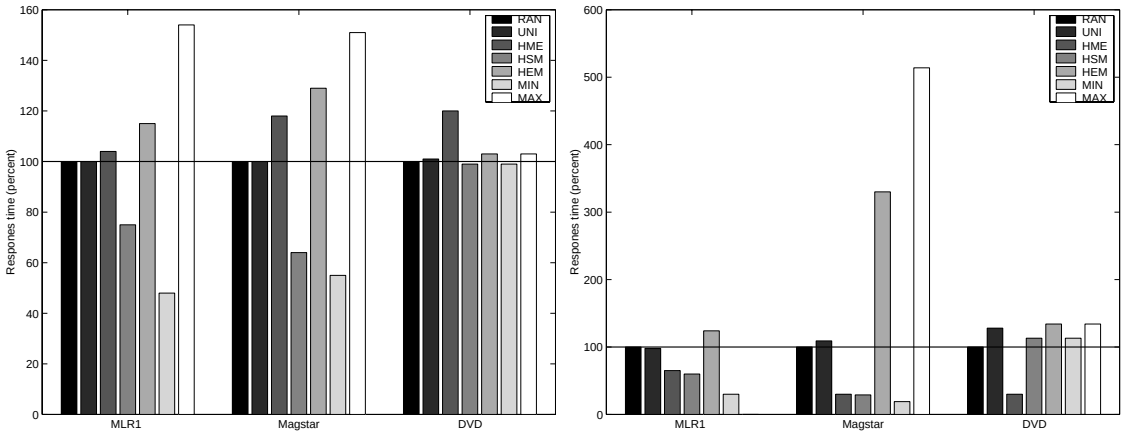


(b) Magstar



(c) DVD

Figure 13.4 Response times for different allocation strategies for retrieval of video sequences of one minute.



(a) Lightly loaded

(b) Loaded

Figure 13.5 Average response times for retrieval of short video sequences of one minute when using the different allocation strategies. The response times are given in percent relative to the response time of the *Random* allocation strategy. The numbers in Table 13.3(a) are used when computing the relative response times. The *Hot Library* strategy is not included.

is the *Hot Minimum Seek Time* strategy. For the archive using DVD, any of the allocation strategies that have a uniform distribution of the hot video sequences to storage media perform well. For a highly loaded system, the *Hot Media* allocation strategy provides the lowest response times. When the number of requests is low, the *Hot Media* strategy is not able to compete due to the long seek times on the tapes. But as the load increases, the response time of the other allocation strategies increases more rapidly than the response time of the *Hot Media* strategy. The reason is that for the *Hot Media* strategy, the *hottest* media will stay permanently in a drive. These media will have multiple concurrent requests and since a scheduler is used when there are multiple requests, the average seek time is reduced. Being able to serve multiple requests from the same medium also reduces the number of media exchanges. The same conclusion was reached by Hillyer et al. (1999) when studying allocation of data objects to tape. We discuss the *Hot Media* strategy further in the next section when we study the scalability of the allocation strategies.

Table 13.3 contains response time measurements for the *loaded* and *lightly loaded* cases used in Section 12.4. For the *loaded* case, we use the number of requests the system can handle with an average response time less than three minutes when using the *Random* allocation strategy. To get response time measurements for the *lightly loaded* case, we reduce the load to 30 percent of the *loaded* case. In Fig-

Allocation strategy	MLR1 Light load (159)	MLR1 Loaded (529)	Magstar Light load (1655)	Magstar Loaded (5518)	DVD Light load (4156)	DVD Loaded (13853)
RAN	99.2	178.6	22.8	174.7	7.1	178.3
UNI	98.8	175.3	22.9	191.2	7.2	228.4
HLI	711.1	—	—	—	—	—
HME	103.2	115.3	26.8	52.0	8.5	53.4
HSM	74.7	107.9	14.5	50.3	7.0	201.4
HEM	114.4	211.2	29.3	575.7	7.3	239.3
MIN	47.3	52.7	12.6	33.4	7.0	201.4
MAX	153.0	—	34.5	897.8	7.3	239.3

(a) One minute

Allocation strategy	MLR1 Light load (8)	MLR1 Loaded (26)	Magstar Light load (32)	Magstar Loaded (106)	DVD Light load (50)	DVD Loaded (166)
RAN	114.6	177.2	55.8	181.4	40.8	187.4
UNI	114.6	171.7	56.2	165.7	39.6	155.3
HLI	—	—	—	—	—	—
HME	199.6	625.2	94.7	429.0	80.3	519.1
HSM	62.1	119.4	45.4	153.7	39.5	155.2
HEM	96.7	152.6	65.8	174.4	39.6	155.2
MIN	62.1	119.4	45.4	153.7	39.5	155.2
MAX	154.2	210.0	65.8	175.4	39.6	155.2

(b) One hour

Table 13.3 Response times for the two cases using each of the allocation strategies. The number of requests per hour is included in parentheses in the top of each column. The results are presented graphically in Figure 13.5 and Figure 13.7.

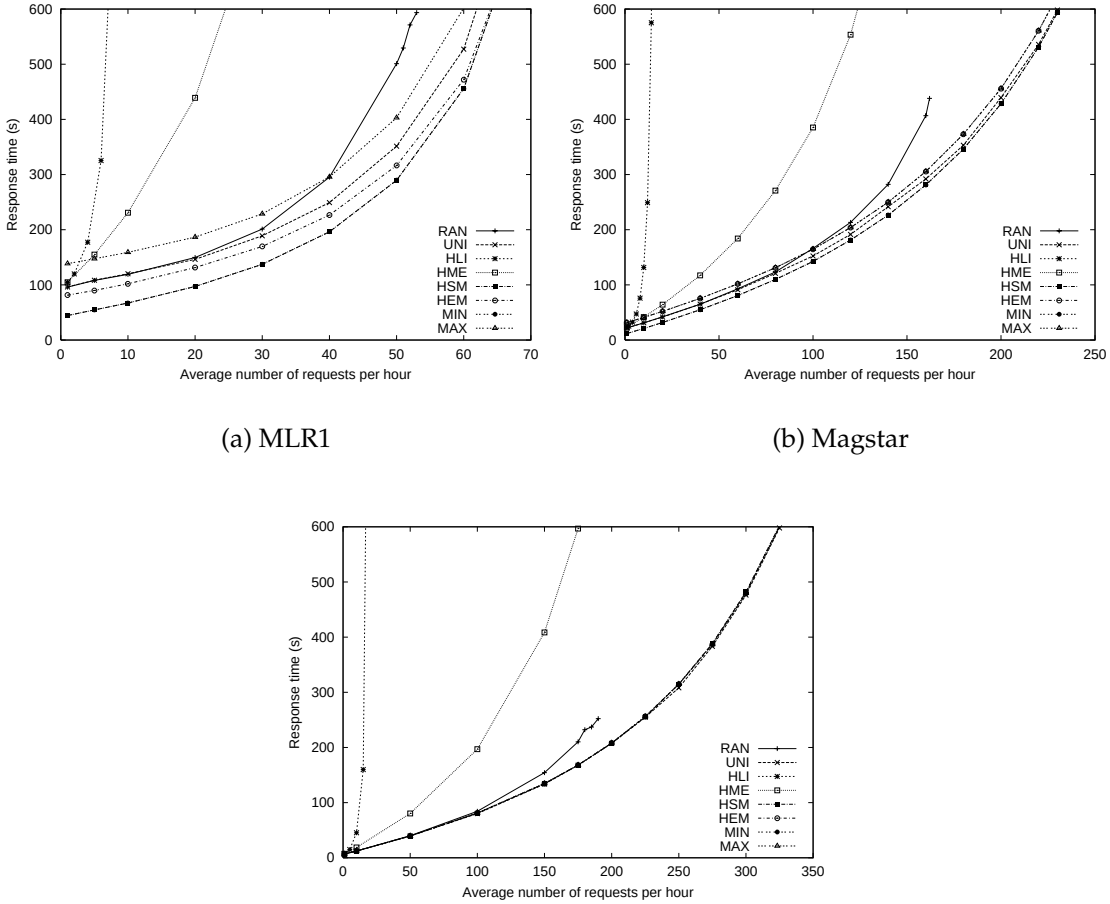
ure 13.5, the response time of the allocation strategies is presented relative to the response time of the *Random* allocation strategy.

Using the tape-based systems for retrieval of short video sequences, the average response time can be reduced from about 50 percent for a lightly loaded system up to about 80 percent for a loaded system by using the *Hot Minimum Seek Time* allocation strategy instead of the *Random* allocation strategy. Using the DVD drive, all allocation strategies with the exception of the *Hot Library* strategy, have approximately the same average response time as long as the system is lightly loaded. As the load on the system increases, the results are more surprising. The *Hot Media* strategy gives the lowest average response time, approximately 70 percent lower than the *Random* allocation strategy. What might not be obvious, is that the second best allocation strategy is the *Random* allocation strategy. The average response time by using this is about 10–25 percent lower than using any of the allocation strategies that distribute the popular video sequences uniformly across the storage media. The reason is that the robot mechanism is approaching its performance limit. When this happens, it will be advantageous to be able to execute multiple requests during each load/unload cycle. For this reason, the *Hot Media* allocation strategy gives the best results. In average, more than 4 video sequences are delivered per loaded DVD. Also the *Random* allocation strategy will result in some media containing multiple hot video sequences, and the likelihood of having multiple requests for the same media will be higher than for the allocation strategies where the hot video sequences are uniformly distributed over the media.

Another observation is that for the loaded case, even the very short seek time for the DVD drive has impact on the average response time. Figure 13.5(b) shows that the two strategies that place the hot video sequences at the start of the medium perform better than the two strategies that place the hot video sequences on the end of the medium. The different placement of the video sequences on the DVD results in a difference in the average response time of almost 20 percent.

Retrieval of Long Video Sequences

Figure 13.6 contains the response time for retrieval of one hour video sequences, while Figure 13.7 shows the relative performance of the allocation strategies. As seen from these figures, for retrieval of long video sequences, the *Hot Minimum Seek Time* and *Hot Start Media* allocation strategies give the lowest response times for the entire load interval. For the DVD, the three other strategies which distribute the more popular video sequences uniformly to the storage media have the same performance.

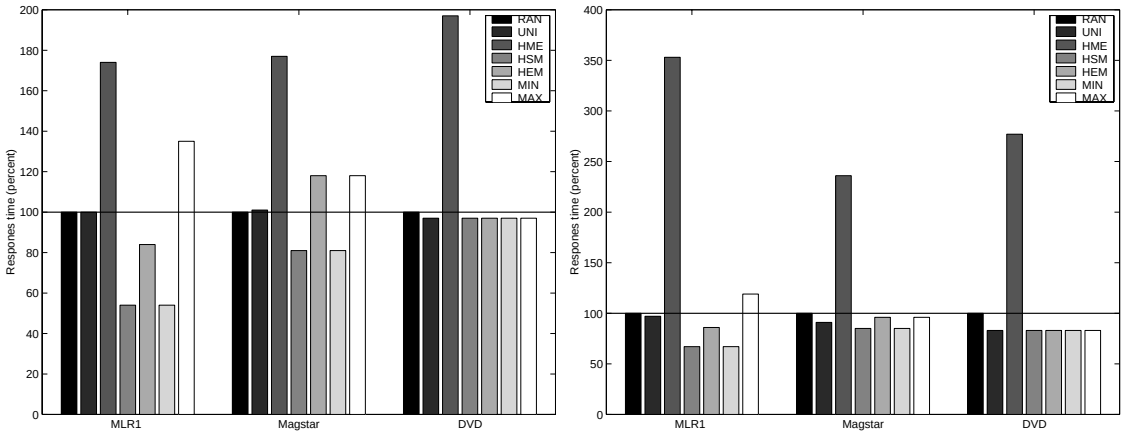


(a) MLR1

(b) Magstar

(c) DVD

Figure 13.6 Response times for different allocation strategies for retrieval of video sequences of one hour.



(a) Lightly loaded (b) Loaded

Figure 13.7 Average response times for retrieval of long video sequences of one hour when using the different allocation strategies. The response times are given in percent relative to the response time of the *Random* allocation strategy. The numbers in Table 13.3(b) are used when computing the relative response times. The *Hot Library* strategy is not included.

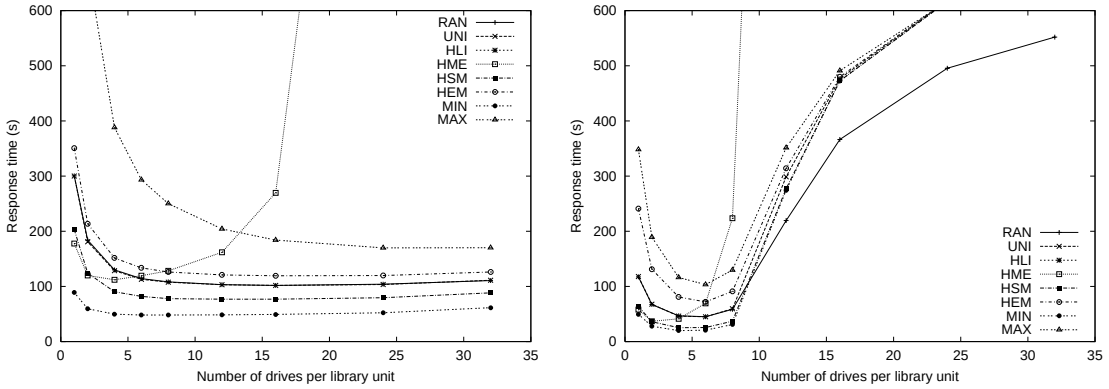
13.4 Scalability by Utilizing more Media Drives

The results shown this far has been for a tertiary storage system where each library unit contained four media drives. The performance of the storage system can be increased by including more drives in each library unit. To investigate whether the results presented this far in this chapter are valid as the number of drives are increased, we study the performance of the allocation strategies when we increase the number of drives and the load correspondingly. Thus, if we double the number of drives per library unit, we also double the load on the system.

Retrieval of Short Video Sequences

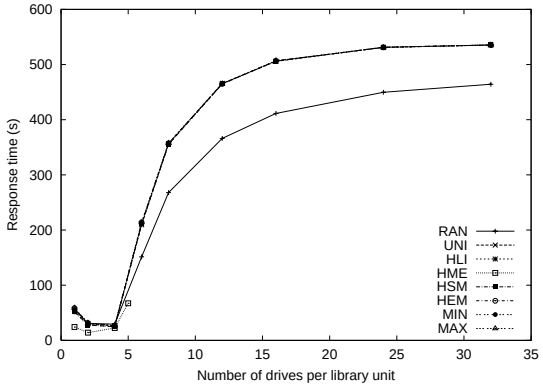
For retrieval of one minute video sequences, we use 12 requests per hour for each MLR1 drive, 48 requests per hour for each Magstar drive, and 120 requests per hour for each DVD drive. The average response time as a function of the number of media drives in the library unit is presented in Figure 13.8. The main points these figures illustrate are:

- Having few drives in each library unit (one or two), results in large average response times. By including more drives, the average response time can be reduced.



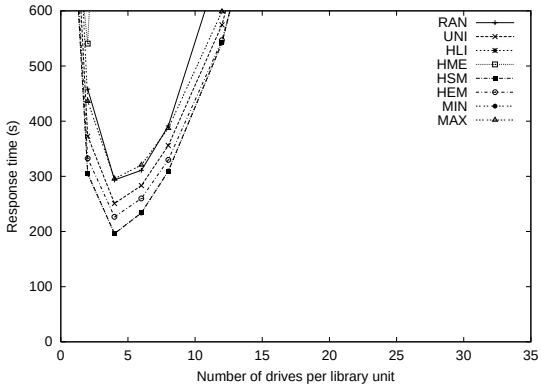
(a) MLR1

(b) Magstar

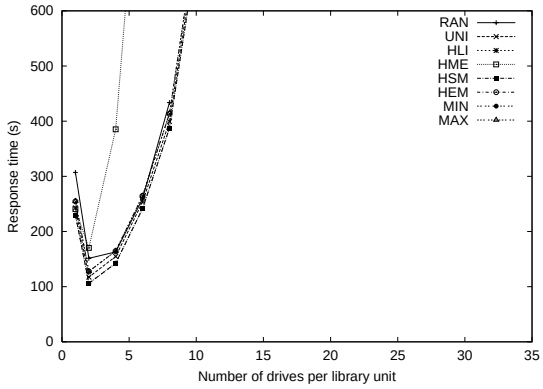


(c) DVD

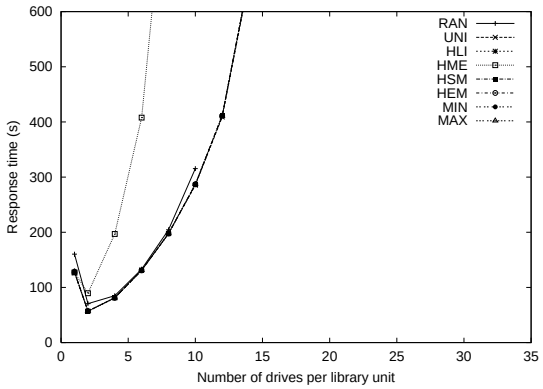
Figure 13.8 Response times for retrieving short video sequences of one minute as a function of the number of drives in each library unit. The average load is constant *per drive*, with 12 requests per hour for each MLR1 drive, 48 requests per hour for each Magstar drive, and 120 requests per hour for each DVD drive.



(a) MLR1



(b) Magstar



(c) DVD

Figure 13.9 Response times for retrieving long video sequences of one hour as a function of the number of drives in each library unit. The average load is constant *per drive*, with on average one request per hour for each drive.

- We have earlier shown the *Hot Media* strategy to perform well with regards to the throughput of the system. This figure shows that this is correct for a system containing a small number of drives, but as the number of drives and the load increase, the drives containing the hot video sequences become the bottleneck. Thus, the *Hot Media* strategy does not scale well when the load on the system increases. The other allocation strategies scale much better, with the exception of the *Hot Library* allocation strategy.

- In Section 10.2.2, we showed that including more than four DVD drives or eight Magstar drives in a library unit, could make the robot mechanisms become the bottleneck (see Figure 10.3(d)). This is the reason for the rapid increase in response time in Figure 13.8(b) and Figure 13.8(c). As seen earlier, when the robot mechanism becomes the bottleneck, the *Random* allocation strategy gives better average response time than the strategies where the hot video sequences are uniformly distributed over the storage media.

An optimal strategy for allocating data objects to disks in a tertiary storage systems containing a single disk drive is presented in (Christodoulakis et al., 1997). This strategy is similar to the *Hot Media* strategy evaluated in this chapter. As Figure 13.8(c) shows, for a tertiary storage system using DVD disks, the *Hot Media* strategy provides the lowest response times for a system containing one drive. The allocation strategy presented in (Christodoulakis et al., 1997) is generalized to support multiple disk drives given that it is always the disk containing the least popular data objects that is replaced when a new disk needs to be loaded. Our simulation results show that this strategy becomes less optimal as the number of drives is increased. The reason other strategies outperform the *Hot Media* strategy in our simulations is that we use a better replacement strategy than to always replace the least popular disk. The replacement strategy used in (Christodoulakis et al., 1997) is optimal with regards to the number of disk loads, but may result in a low utilization of the drives.

Retrieval of Long Video Sequences

The simulation results for retrieval of one hour long video sequences are presented in Figure 13.9. The load used in these simulations is in average one request per hour per drive. From the shape of the response time curves, it is obvious that the system does not scale when we increase the number of drives and the load correspondingly. From our study of scalability of the number of drives in a library unit in Section 10.2.2 we know that the robot is not the limiting resource (see Figure 10.4). Since we are adding drive resources at the same rate as we increase the load, it should not be the drives either that limit the throughput. Data from the simulations shows that the drive utilization is rather low, but very skewed within each library unit. The reason for this is that with 10000 hours of video and using the 90/9/1 access distribution, only 100 media contains a *hot* video. Thus, for the

	Short video sequences		Long video sequences	
	Throughput	Response time	Throughput	Response time
RAN				
UNI				
HLI	$\div\div$	$\div\div$	$\div\div$	$\div\div$
HME	$\div\div/++$[a]	$+/\div$[a]	$\div\div$	$\div\div$
HSM	$+$	$+$	$+$	$++$
HEM				
MIN	$+$	$++$	$+$	$++$
MAX				

[a] Highly dependent on the skewness of the access distribution and the load.

Table 13.4 Summary of the performance of the allocation strategies studied in this chapter.

Magstar and DVD archive, there are only four media in each library unit containing a hot video. The drives that have to serve these media become the bottleneck in the system. As the load increases, these drives will not be able to serve all requests for the hot media, and thus will make the average response time increase rapidly.

13.5 Conclusions

Table 13.4 gives a summary of the properties of the allocation strategies we have studied in this chapter. The most important recommendations from the work are as follows:

- Avoid allocations that make one library unit hotter than the other (e.g., the *Hot Library* strategy).
- The experiments show that the importance of using a good allocation strategy is more important for sequential storage media than for random access storage media. This is due to that the placement of a video influences more on the access time for storage media with high seek and rewind times.
- The *Hot Media* allocation strategy can give very good results regarding the throughput for retrieval of short video sequences, but it is a "dangerous" strategy to follow, because the drives containing the hot video sequences might become the bottleneck (see Figure 13.8).

- Overall, the best strategy to use for allocation of video sequences is to use the *Hot Minimum Seek Time* strategy for allocating the video sequences to the storage media. This is the same conclusion as reached by Hillyer et al. (1999) when studying placement of hot data object in a tape-based library unit with no use of data replication.

The last point answers the initial question we started this chapter with: *What is a good allocation of video sequences to storage media and library units?* Of the allocation strategies studied here, in most cases using the *Hot Minimum Seek Time* allocation strategy results in the highest throughput and lowest average response times. But as shown in this chapter, there are usage scenarios where other allocation strategies perform better than the *Hot Minimum Seek Time* strategy.

An important observation from several of the experiments is that with an access distribution as skewed as the one used in this chapter, one or a few drives in each library unit may become the bottleneck and thus limiting the performance that can be achieved. This is due to the fact that we do not perform any caching of videos in secondary storage. If we had cached the hottest videos on disk, the load on the drives serving the hottest video sequences would have been greatly reduced. In the next chapter, we continue this study by investigating the effect of caching the most frequently accessed videos in a disk cache.

Chapter 14

Caching Video Sequences using a Disk Cache

You get what you pay for.

Anonymous

In the previous chapters, we have studied the performance of tertiary storage systems used for storage and retrieval of digital video. All requests were executed by retrieving the requested video sequence from a tertiary storage medium. For some applications, a video archive using only tertiary storage might provide the required throughput or be the only storage alternative due to budget constraints. For other applications, the performance of a tertiary storage system is not sufficient. In particular, the long response time of a tertiary storage system limits its use in interactive video archive applications. In this chapter, we continue our study of storage systems for video archives by investigating how the performance can be improved by utilizing secondary storage as part of the storage system.

To improve both the throughput and the response time of a video archive, magnetic disks can be used instead of tertiary storage devices. While this increases the performance by orders of magnitudes, it also increases the cost of the video archive significantly. A less costly way to improve the performance is to use magnetic disks for storing the most frequently accessed video sequences. This is commonly referred to as a *cache*. Caching is a well-known technique used at all levels in a storage hierarchy for improving the performance. The goal of using a cache is to achieve an average access time close to the access time of the faster storage medium, while keeping the storage cost for the system close to the storage cost of the slower storage medium. Since the cache in our study consists of magnetic disks and is used for storing video sequences, we interchangeably refer to it as a *disk cache* or a *video cache*.

The purpose of this chapter is to investigate some of the effects the use of a disk cache may have on the performance and cost of a video archive based on tertiary storage. The main issues studied in this chapter are:

1. **Cache hit rate and performance.** In order to benefit from using a disk cache, it is necessary that a high number of the requests can be served from the cache. Two of the main factors that determine the fraction of the requests that can be delivered from the disk cache are the size of the cache and the access distribution. We study what influence these two factors have on the cache hit rate and how this may improve the performance of the video archive and reduce the cost of retrieving videos.

2. **Disk cache size versus tertiary bandwidth.** Both increasing the size of the disk cache and increasing the bandwidth of the tertiary storage system may improve the performance of a video archive. We study the relationship between increasing the size of the disk cache and increasing the number of tertiary drives used in the library units.

3. **Allocation strategies.** In Chapter 13, we investigated strategies for allocating videos to storage media and library units in a tertiary storage system. In this chapter, we study the same allocation strategies in order to determine whether the conclusions made in Chapter 13 are influenced by introducing a disk cache.

The focus in this chapter is on how the performance and cost of a video archive using tertiary storage can be improved by using a disk cache and how inclusion of a disk cache affects the tertiary storage system. We assume a very simple model for the disk cache, and do not study the disk cache or technologies for building disk caches in detail. In Section 14.5, we provide a detailed example of how a disk cache might be implemented.

14.1 Simulation Overview

The video archive simulator was presented in Chapter 9. As shown in Figure 9.1, the tertiary storage system is augmented by including disk drives. These disk drives constitute the video cache, and is used for caching the most recently accessed video sequences. If a requested video sequence is available in the video cache, it is delivered from the cache. If the video cache does not contain a copy of the requested video sequence, it is retrieved from the tertiary storage system. For the videos retrieved from the tertiary storage system, *pipelining*[1] is used for reducing the response time (Ghandeharizadeh et al., 1995; Chan and Tobagi, 2003). As soon as the first blocks of the video sequence are available in main memory or in the disk cache, the streaming of the video to the user is started.

In order to compare results in this chapter to results presented in the previous chapters, we use the same amount of video in the archive. Thus, the simulated archive contains 10,000 hours of 5 Mbit/s video. By including a disk cache in

[1] In (Chan and Tobagi, 2003) this is referred to as *stage-streaming*.

the video archive, the size of the cache becomes a new configuration parameter that must be set in order to optimize the performance and cost of the archive. To cover the most realistic cache sizes, we vary the size of the cache from 0.1 to 30 percent of the total storage requirement. With a total amount of 20.5 TB of video data, the cache size will be from 20 GB to 6000 GB. For the tertiary storage system, we use the same configurations as in the previous chapters. In most of the simulations, each library unit contains four media drives. In order to reduce the amount of simulation time, we use the *Uniform* allocation strategy (defined in Section 13.1) in most of the simulations. In Section 14.4, we study the effect of different allocation strategies, and will in particular compare the use of the *Uniform* allocation strategy to use of other allocation strategies.

14.1.1 Evaluation Criteria

As in the previous chapters, we use throughput, response time, and cost as the main evaluation criteria. The introduction of a disk cache does not change how we compute the throughput or cost numbers for the system, but for measuring and interpreting the response time there are a few issues that must be taken into account.

The response time is measured as the amount of time from the user requests a video sequence until the video cache starts delivering the video to the user. Since a requested video sequence either is available in the disk cache, or has to be retrieved from the tertiary storage system, the following two cases are used for computing response times:

1. For video sequences already present in the video cache, the response time is zero.

2. For video sequences that must be retrieved from tertiary storage, we use the response time of the tertiary storage system. Pipelining is used (Ghandeharizadeh et al., 1995; Chan and Tobagi, 2003), and our model assumes that the disk cache does not introduce any significant delay to the response time.

To use a zero response time for the disk cache is obviously not correct. The main reason for doing this is that the response time of a disk-based system is several orders of magnitudes lower than the response time of the tertiary storage system (tens of milliseconds compared to seconds). The main goal of this study is to investigate how the performance of a tertiary storage system can be improved by using a disk cache. To make a more realistic performance model of the disk cache would complicate the overall simulation model, and make the results of this study depend of the architecture and strategies used by the disk cache for delivering video sequences to users. In Section 4.4, we gave an overview of some of the technologies and strategies that can be used for implementing a disk-based storage system for use as a video cache. In Section 14.5, we provide a more detailed

model of one possible disk cache design and evaluate the resources required for delivering video from the disk cache.

One problem related to using the average response time as a performance criterion is that retrieval of video sequences either includes only the disk cache, or both the tertiary storage system and the disk cache. Thus, the measured response times fall into two sets. The first set contains the video sequences that can be delivered from the disk cache. The second set contains the video sequences that must be retrieved from tertiary storage. Since, (hopefully) most of the requests can be served directly from the disk cache, the average response time will be rather low even when the response time for the requests that have to be served from the tertiary system is high. Thus, for a user requesting one video sequence, she will very seldom experience the *average* response time. Frequently, video playback may start immediately, but in some cases she has to wait a rather lengthy time. Because of this, we present a few examples that include both the average response time of all the requests and the average response time of the requests that have to be retrieved from tertiary storage.

14.2 Cache Hit Rate and Performance

For a video archive using a tertiary storage system as the main video store and a disk cache for caching the most used videos, either the tertiary storage system or the disk cache may become the bottleneck that limits the performance of the archive. With our model for the disk cache, the performance of the archive will be limited by the performance of the tertiary storage system. Thus, the main factor influencing the average response time and limiting the throughput of the video archive is the fraction of the requests that require access to the tertiary storage system.

For any system utilizing a cache, it is important that as many as possible of the requests can be served from the cache. The fraction of the requests that can be delivered from the cache is called the *cache hit rate*. In general, the cache hit rate is determined by the following factors:

$$\textit{cache hit rate} = f(\textit{data volume}, \textit{cache size}, \textit{replacement policy}, \textit{access distribution})$$

In this section, we study how these factors influence the cache hit rate, the performance and the cost of a video archive. In this study, the factors will take on the following values:

- **Data volume.** The amount of video data is constant. As in the previous chapters, the video archive contains 10000 hours of 5 Mbit/s video. The configurations for the tertiary storage system are given in Table 12.2. Each library unit contains four tertiary drives.

- **Cache size.** The size of the disk cache is varied between 20 GB and 6000 GB, i.e., the disk cache is able to store from 0.1 percent to about 30 percent of the videos.

- **Replacement policy.** The LRU replacement policy is used for determining which video sequences to remove from the cache in order to make room for new video sequences. The granularity for replacement is entire video sequences. Alternative replacement policies for use in a disk cache have been proposed by (Ghandeharizadeh and Shahabi, 1994; Lau and Lui, 1996; Cha et al., 2002). Also, most of the caching strategies for caching parts of videos in main memory presented in Section 4.3 could have been extended to use in a disk cache. Most of these strategies are page based and do not consider entire videos for replacement. We selected to use LRU due to its simplicity and since we wanted a replacement policy that replaced entire video sequences in the cache. We do not use any form for *trail-deletion* to reduce the amount of space occupied by each video sequence (Chan and Tobagi, 2003).

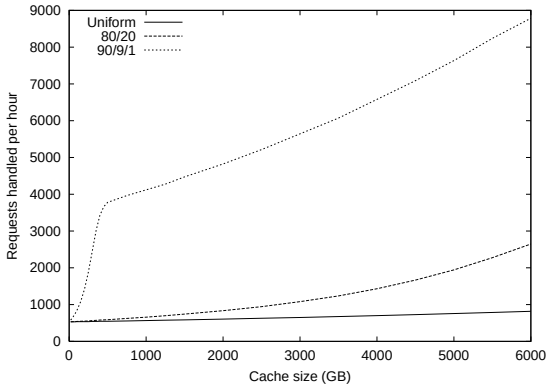
- **Access distribution.** For studying the effect different access distributions have on the cache hit rate and the performance, we use the *Uniform*, *80/20*, and *90/9/1* access distributions described in Chapter 12.

As the total amount of video data and the replacement strategy are fixed in this study, the focus of this section will be on studying the effect the cache size and access distribution has on the cache hit rate, the performance, and the cost of a video archive.

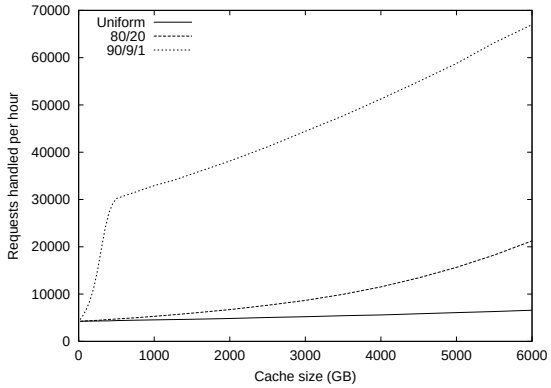
14.2.1 Throughput

To study the effects the cache size and the access distribution have on the performance of a video archive, we study the throughput of the video archive. The main goal is to show the main consequences of the cache size and the access distribution, not to study the performance of the system in details.

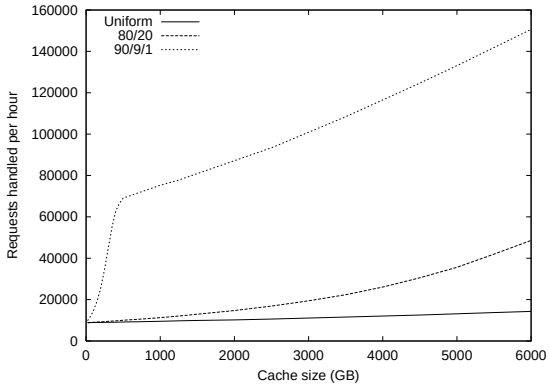
As in previous chapters, the criterion used for the throughput is the number of requests that can be served with a limit on the average response time. This far, we have mostly used 180 seconds as the response time limit. In this section, we use response time limits that are proportional to the response time of an idle tertiary storage system using the different tertiary storage technologies. In order to be able to compare some of the results in this section to results from previous chapters, 180 seconds are used as the average response time limit for the archive using MLR1 drives. This is 85% higher than the average response time of an idle library unit using MLR1 drives. For the archives using Magstar and DVD drives, we selected 41 and 12 seconds as the response time limits, which are 85% higher



(a) MLR1 – limit=180 seconds

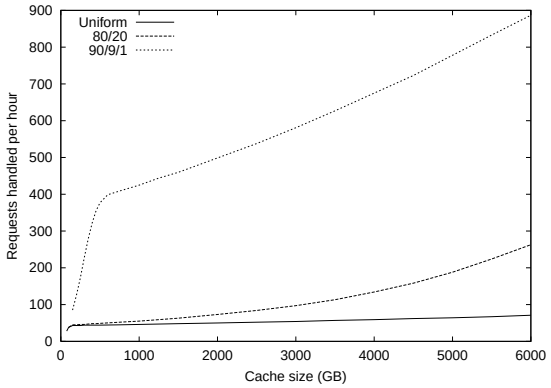


(b) Magstar – limit=41 seconds

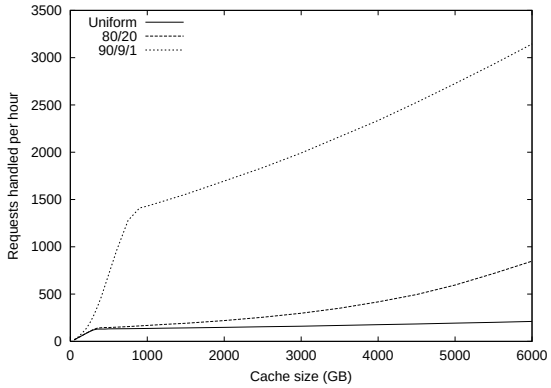


(c) DVD – limit=12 seconds

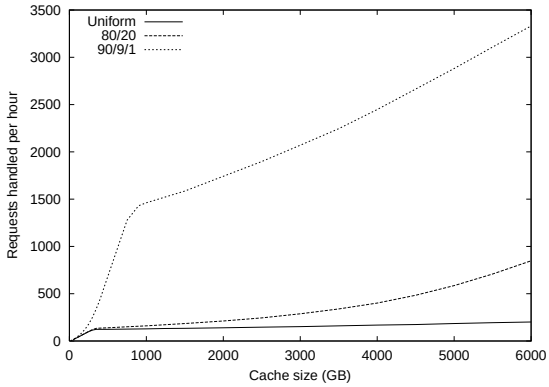
Figure 14.1 Throughput as function of cache size for three different access distributions. Each request is for a one minute video sequence. Limits for the average response time are 180 seconds for MLR1, 41 seconds for Magstar, and 12 seconds for the DVD drive.



(a) MLR1 – limit=180 seconds



(b) Magstar – limit=41 seconds



(c) DVD – limit=12 seconds

Figure 14.2 Throughput as function of cache size for three different access distributions. Each request is for a one hour video sequence. Limits for the average response time are 180 seconds for MLR1, 41 seconds for Magstar, and 12 seconds for the DVD drive.

than the response time of an idle library unit using the corresponding tertiary storage technologies.

To determine the throughput of the video archive, we perform simulations for each of the access distributions. The cache size is used as the independent simulation variable. The resulting throughput numbers for retrieval of video sequences of one minute and one hour are presented in Figure 14.1 and Figure 14.2.

From studying the throughput curves for the three access distributions, the following can be observed:

- **90/9/1.** In all figures, it is obvious where all the video sequences of the *hot* partition more or less always are present in the disk cache. This happens when the disk cache has a size of approximately 500 GB. The *hot* partition contains one percent of the videos, or 210 GB of video data. The reason the size of the cache has to be about twice as large as the *hot* partition, is due to that the *hot* video sequences have to compete with the *warm* and *cold* video sequences for space in the cache.

- **80/20.** For the 80/20 access distribution, the size of the *warm* distribution is about 4 TB. Thus, in our simulations, most of the accesses to *warm* video sequences can be served from the disk cache when the size of the disk cache is 6 TB. With this size of the cache, 74 percent of the requests are delivered from the cache.

- **Uniform.** With a uniform access distribution, the throughput curves in Figure 14.1 and 14.2 show that the number of requests that can be served is rather low compared to the other two access distributions. A 6 TB disk cache is able to store 29 percent of the video sequences.

It is important to note that the throughput curves in these figures are dependent on the limit specified for the average response time. If we had specified a higher or lower limit on the average response time, the throughput for the archive would have become slightly higher or lower.

Cache Hit Rates

The reason for the increased throughput as more storage space is added to the disk cache or as the access distribution becomes more skewed is that a higher fraction of the popular video sequences are present in the cache. Figure 14.3 contains the cache hit rate for *all* throughput curves in Figure 14.1 and Figure 14.2. With a cache size of 1 TB, which corresponds to five percent of the total amount of video data, for the 80/20 access distribution only 15 percent of the requests can be served directly from the cache. For the 90/9/1 access distribution, 83 percent of the requests can be served from the cache. If the cache size is increased to 6 TB, the cache hit ratio is increased to 74 percent for the 80/20 access distribution, and

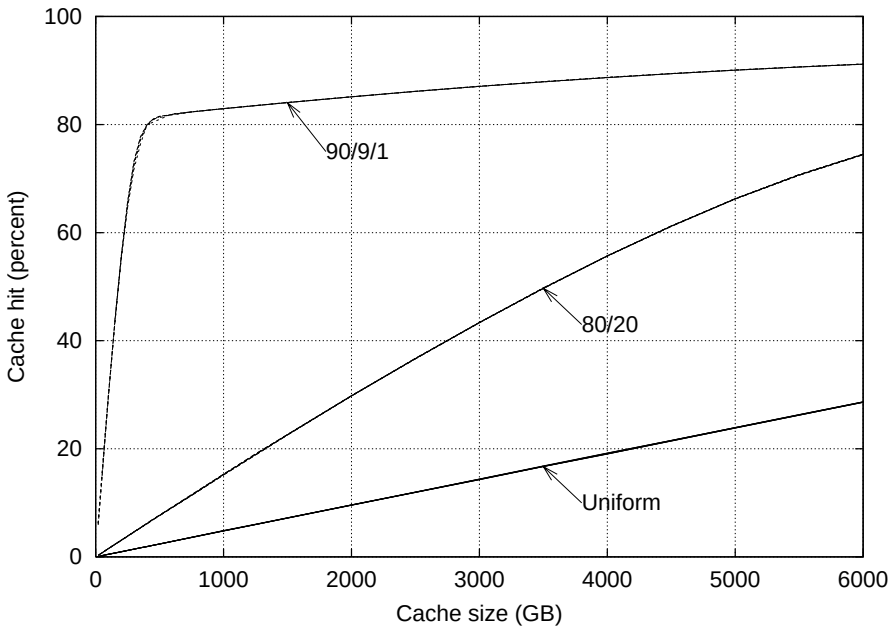


Figure 14.3 The cache hit probability of the three access distributions given as a function of the size of the disk cache. The figure contains the cache hit curves for all throughput curves in Figure 14.1 and 14.2.

to 91 percent for the 90/9/1 access distribution. For the uniform access distribution, with cache sizes of 1 TB and 6 TB, respectively 5 and 29 percent of the requests can be served directly from the cache.

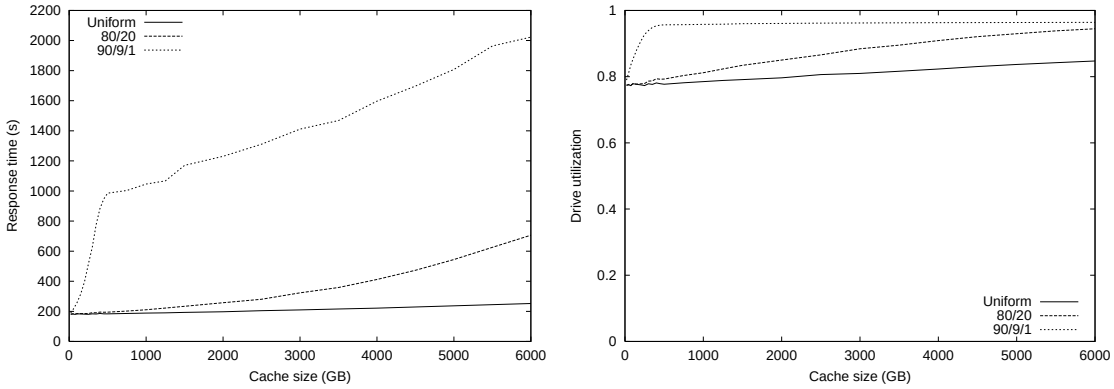
From studying the throughput curves in Figure 14.1 and 14.2 and the cache hit rates in Figure 14.3, we see that the size of the disk cache and the access distribution have the following effects on the performance of the storage system:

1. Both the size of the disk cache and the access distribution have a large impact on the achieved throughput.

2. The more skewed the access distribution becomes, the higher throughput can be achieved, and the smaller can the cache be and still provide a high cache hit rate.

3. For access distributions close to being uniform, the effect of using a disk cache is small.

4. The cache hit curves overlap for each of the access distributions. This shows that the cache hit rate is independent of both tertiary storage technology and the length of the requested video sequences.



(a) Response time, tertiary storage system

(b) Drive utilization

Figure 14.4 (a) Average response time for the requests that must be served from tertiary storage. **(b)** Utilization of the tertiary drives. These figures are based on the simulations for the throughput numbers in Figure 14.1(a) using MLR1 drives in the library units.

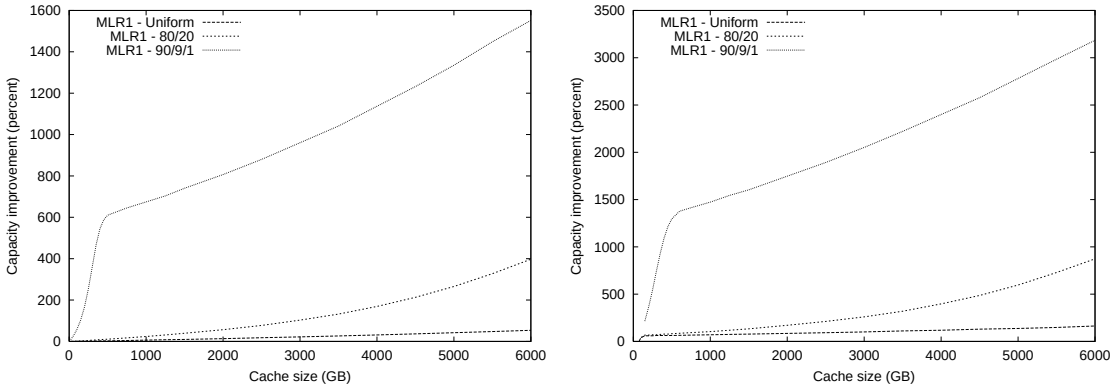
We have kept the average response time constant in these simulations. As throughput and cache hit rate increase due to a larger cache size or a more skewed access distribution, the following happens to the performance of the tertiary storage system:

- The response time of the requests served by the tertiary storage system will increase. This is due to that a smaller fraction of the total number of requests have to be retrieved from tertiary storage. An example of how the response time increases as the cache size is increased is given in in Figure 14.4(a).
- The utilization of the tertiary storage system is increased. An example of the tertiary drive utilization as the cache size is increased is given in Figure 14.4(b).

Throughput Improvement

To illustrate the throughput improvement that can be achieved by using a disk cache compared to the throughput of using only the tertiary storage system, we show one example. In Figure 14.5, the throughput improvement in percent is given for a system utilizing MLR1 drives. The throughput improvement is computed using the throughput numbers in Figure 14.1(a) and Figure 14.2(a) and the throughput of the corresponding tertiary storage system as given in Table 13.2.

These two figures show that the relative throughput can be increased much by using a disk cache. Using a 1 TB disk cache improves the throughput with almost



(a) One minute (b) One hour

Figure 14.5 The throughput improvement in percent by using a video cache compared to a system with no caching. For the system with no cache, the throughput numbers for the *Uniform* allocation strategy in Table 13.2 are used. The video archive contains four MLR1 drives in each library unit.

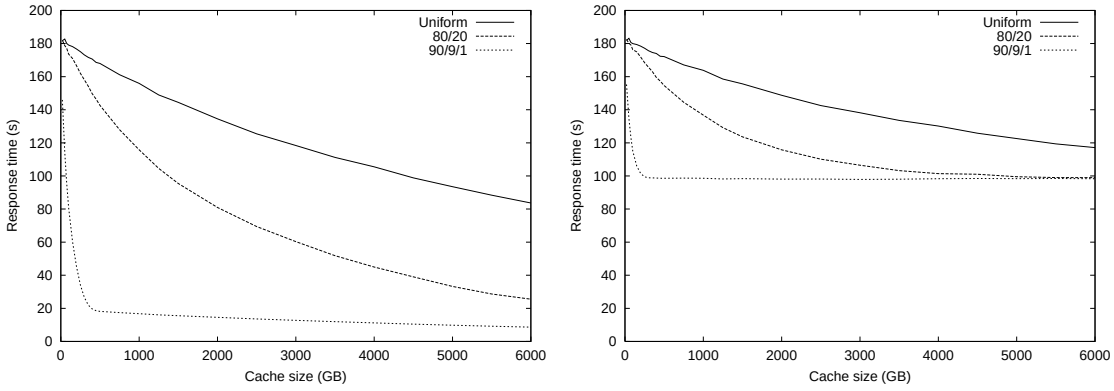
700 percent when retrieving short video sequences accessed with a 90/9/1 access distribution. If the video sequences are one hour long, the improvement is almost 1500 percent. If the access distribution is 80/20, the corresponding improvement numbers for retrieval of short and long video sequences are 23 and 104 percent respectively.

The improved throughput for long video sequences compared to short video sequences is mainly due to the throughput numbers for the tertiary storage system used as a reference. With a maximum average response time of 180 seconds for the tertiary storage system, the drive utilization is lower for retrieval of long video sequences compared to retrieval of short video sequences. Thus, there are more free resources in the tertiary storage system that can be utilized when the disk cache is added.

For a video archive using Magstar or DVD drives, the throughput improvement in percent would be in the same range as the throughput improvement when using MLR1 drives.

14.2.2 Response Time

Just as the cache size and access time distribution have a huge impact on the throughput of a video archive, the same factors will result in improved response times. The purpose of this subsection is to show some examples that illustrate how the response time is influenced by both the cache size and the access distribution.



(a) Response time, all requests

(b) Response time, tertiary storage system

Figure 14.6 Average response time for a video archive using MLR1 drives as function of cache size and access distribution. The load is constant, 532 requests per hour for one minute long video sequences.

To illustrate the effect the cache size and the access distribution have on the response time, Figure 14.6(a) shows the response time for a video archive using MLR1 drives for the three access distributions. The load on the archive is constant. The main conclusion from studying this figure is that the more skewed the access distribution is, the more effect do we get from including a disk cache. In this example, by including a 1 TB disk cache, the response time is reduced by 91 percent when the access distribution is 90/9/1, 36 percent when the access distribution is 80/20, and by 14 percent when the access distribution is uniform.

There are two reasons for the large reduction in average response time:

1. As the cache hit rate is increased due to a larger disk cache or a more skewed access distribution, the number of requests that can be served directly from the cache increases.

2. As the number of requests that can be served from the cache increases, the load on the tertiary storage system is reduced. This results in lower average response time for the requests that have to be served by the tertiary storage system. This is shown in Figure 14.6(b), which shows the response time of the requests to the tertiary storage system.

It is important to note that these response time curves are dependent on the load on the archive. Thus, with a different load, the reduction in the average response time would be different. Still, these results show that the size of the cache has a large impact on the response time, and the more skewed the access distri-

Cache size		Short video sequences			Long video sequences		
Percent	GB	MLR1	Magstar	DVD	MLR1	Magstar	DVD
0	0	98.1	21.8	6.3	96.2	21.7	6.7
1	200	43.8	10.0	2.9	43.7	9.7	2.9
2	400	19.6	4.4	1.3	19.0	4.2	1.3
5	1000	16.8	3.7	1.1	16.1	3.6	1.1
10	2000	14.7	3.3	1.0	14.0	3.1	0.9
20	4000	11.0	2.5	0.7	10.6	2.4	0.7
30	6000	8.7	1.9	0.6	8.4	1.8	0.6

Table 14.1 Average response times for a lightly loaded video archive for different cache sizes. The requests are generated using the 90/9/1 access distribution.

bution is, the smaller may the disk cache be and still result in a large reduction in the average response time.

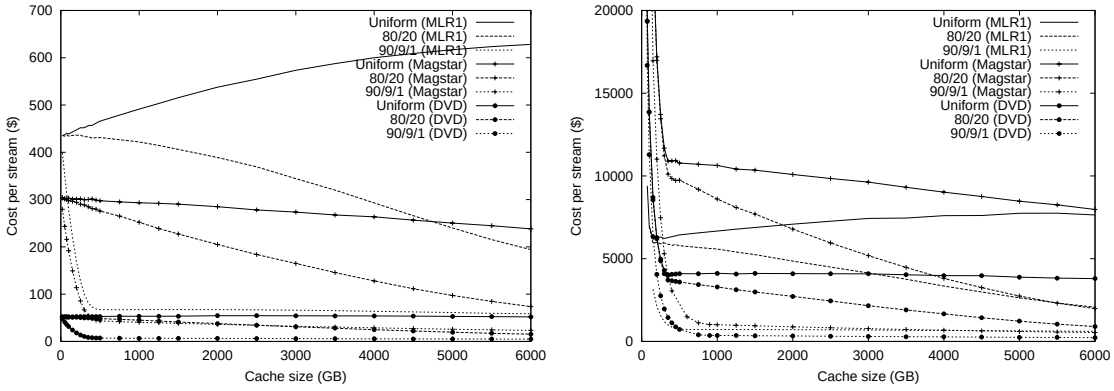
For the other storage technologies and for different lengths of the video sequences, the reduction in response time will be similar. Table 14.1 contains the average response times for a lightly loaded video archive using each of the storage technologies. The 90/9/1 access distribution is used. These numbers show that the relative reduction in the average response time is largely unaffected by the type of drive or the size of the requested video sequences.

14.2.3 The Cost of Using a Disk Cache

As expected, and confirmed by simulations, caching the most frequently accessed video sequences on hard disks, greatly improves both throughput and response time of a video archive. However, despite the recent years' dramatically reduced prices, a disk cache will still represent a significant part of the total cost of a video archive. The purpose of this subsection is to give an example showing how the cost of storing and retrieving videos is influenced by using a disk cache. We use the same archive configuration as when studying throughput and response times.

Cost per Stream

We use the *cost per stream* to represent the cost of retrieving videos stored in the video archive (Chervenak, 1998). In Figure 14.7 the cost per stream for the throughput simulations in Figure 14.1 and 14.2 is presented. Since the cost per stream is dependent on the throughput of the system, it is important to keep in mind that the actual values presented in this figure are dependent on the limit we



(a) One minute

(b) One hour

Figure 14.7 The cost per stream for the three different access distributions using each of the tertiary media drives. The cost per stream is based on the throughput simulations in Figure 14.1 and 14.2.

have chosen for the average response time. From studying this figure, the main observations are as follows:

- When the access distribution is non-uniform, the cost per stream can be substantially reduced by including a disk cache. In our simulations, the results from using the 90/9/1 access distribution show that by including a 1 TB disk cache, the cost per stream is reduced by 85 to 90 percent for retrieval of both short and long video sequences using any of the tertiary storage technologies.

- If the access distribution is uniform, the effect on the cost per stream of including a disk cache is more uncertain. In our example, the cost per stream increases for an archive using MLR1 drives, while it decreases for an archive using Magstar drives. The reason for this difference is the high cost of the Magstar drive, which makes the bandwidth of this drive more expensive compared to the storage cost of the hard disks. For an archive using DVD disks, there is a slight increase in the cost per stream when increasing the size of the disk cache.

- Using DVD drives in the tertiary storage system provides the lowest cost per delivered video sequence. For retrieval of short video sequences, using MLR1 drives results in the highest cost per delivered video sequence, while for retrieval of long video sequences, using Magstar is most expensive. As the length of the videos increases, the relative difference between using tape drives and DVD drives is reduced.

The reason the cost numbers in Figure 14.7(b) are very high for small cache sizes, is that the size of the disk cache limits the number of video sequences that can be stored in the cache, and thus also limits the number of delivered video streams.

Storage Cost

For some applications, the storage cost is the most important criterion for selecting which tertiary technology to use. In Section 9.4, we gave an overview of the storage cost for using the different storage technologies we are discussing in this thesis and the cost of including a disk cache. The total cost of the video archive is not dramatically increased by including a disk cache. For the archive configurations studied in this section and using MLR1, Magstar, and DVD drives, the storage cost is increased by 20, 4, and 10 percent respectively by including a 1 TB disk cache.

It is important to note that in this section we have studied the cost of a video archive that contained four tertiary drives in each library unit. It is not necessarily four drives per library unit that is the optimal choice. In Section 14.3, we investigate how to select the cache size and the number of tertiary drives in order to reduce the cost of the video archive.

14.2.4 Validation of the Simulated Video Cache

We end this section with a validation of the simulated video cache and the corresponding cache hit rates. Common for the three access distributions used in this section is that they divide the total data volume into one or several partitions, where the access distribution *within* each partition is uniform. Given this, the LRU buffer model developed by Bhide, Dan and Dias (1993) can be used for computing the *cache hit probability* as a function of the cache size. We refer to this model as the BDD LRU buffer model. We use the opportunity of having both simulated cache hit probabilities and an analytical model for the cache hit probability to validate that the implemented simulator for the video cache performs correctly with regards to the LRU replacement strategy.

Figure 14.8 contains the cache hit rates from the throughput simulations presented in Figure 14.1 and 14.2. As this figure shows, the cache hit curves for the different access distributions overlap. The BDD LRU buffer model is used for computing the corresponding cache hit probabilities. The computed cache hit probabilities are included in Figure 14.8 as points along each of the cache hit curves. The figure shows that the values computed by the BDD LRU buffer model and the simulated values overlap. By studying the numbers behind the cache hit curves, we find that the difference between the computed values and the simulated values is less than one percent when the size of the video cache is larger than 100 GB. If the disk cache is less than 100 GB and is used for storing video sequences of one hour, we find larger differences between computed and simu-

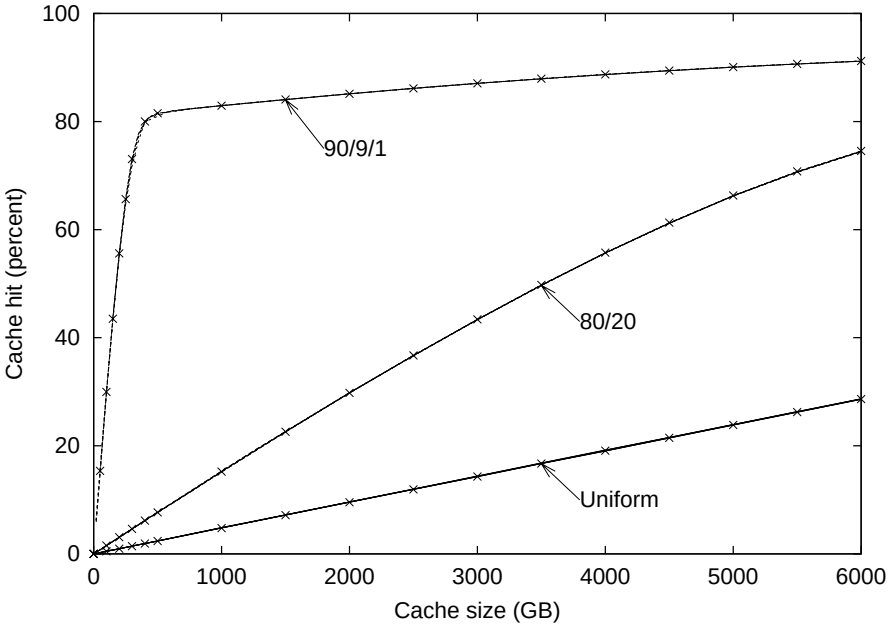


Figure 14.8 Cache hit probability as a function of the size of the disk cache. The curves contain the results from the simulations in Figure 14.1 and 14.2. The points along the curves are computed using the BDD LRU buffer model (Bhide et al., 1993).

lated cache hit probabilities (up to three percent). The reason is that the cache can only hold a few video sequences of this size, and as soon as one video is stored in the cache it will be locked in the cache for at least one hour. The negative effect of having long video sequences occupying a large fraction of the cache when the cache size is small could have been reduced by use of trail-deletion (Chan and Tobagi, 2003). This would have reduced the contention for cache space, but would also reduce the cache hit rate.

14.3 Disk Cache Size and Tertiary Bandwidth

In the previous section, we studied the effect of adding a disk cache to a tertiary storage system in order to improve the performance and reduce the cost of retrieving videos. We showed that increasing the size of the disk cache, improves both throughput and response time. But as shown in earlier chapters, there are other factors that influence the performance and cost. In our simulation model the three main factors that determine the performance and cost of the video archive are:

1. **Tertiary storage technology.** As we have seen in this and the previous chapters, the choice of tertiary storage technology has a large impact on the performance and cost of the tertiary storage system.

2. **Tertiary bandwidth**. The number of tertiary drives determines the bandwidth of the tertiary storage system. Increasing the number of drives in each library unit does not increase the cache hit rate, but increases the number of requests that can be retrieved from tertiary storage, i.e., those requests that correspond to cache misses.

3. **Disk cache size.** As shown in this chapter, increasing the size of the disk cache improves the cache hit rate, and thus increases the number of requests that can be delivered directly from the disk cache. Increasing the size of the disk cache, will usually also increase the bandwidth of the cache.

Since both adding hard disks and tertiary drives increase the performance of the system, the interesting question is: *Should we add more hard disks to the disk cache, or should we include more drives in the tertiary library units in order to improve the performance of a video archive?*

The purpose of this section is to study the relationship between the size of the disk cache and the number of tertiary drives in the tertiary storage system. We use the relationship between the cost and the throughput of the system as the main criterion for determining whether it is better to add more hard disks or more tertiary drives. The goal is to determine how the size of the disk cache and the number of drives are related, and to determine how this can be used to build a video archive with a given performance a the lowest possible cost.

A similar study has been performed by Chan and Tobagi (1999). They study how to design hierarchical storage systems for use in video servers. Their focus is on determining the required bandwidth and storage space given the performance requirements. They develop an analytical model for the performance of a disk cache and a tertiary library unit. Compared to our model of the tertiary storage system, they use a very simple model for the library unit. They only consider the bandwidth of the tertiary drives and do not include any delays due to mount and seek times or the robot. They also assume that the aggregate tertiary bandwidth can be used for reading a single video from the tertiary library. This model combined with simulations is used to determine the required bandwidth and storage capacity of the disk cache and the required bandwidth of the tertiary library unit. One of their main conclusions is that *a higher tertiary bandwidth can generally be traded with a lower secondary storage, and vice versa* (Chan and Tobagi, 1999). In (Chan and Tobagi, 1999), the videos are completely staged in the disk cache before streamed to the user. In (Chan and Tobagi, 2003), the analytical model is extended to support *stage-streaming* in order to reduce the response time.

Drives:	1, 2, 4, 6, 8, 10
Cache sizes (GB):	200^2, 300, 400, 500, 1000, 2000, 3000, 4000

Table 14.2 The number of drives per library unit and cache sizes used in the simulations in Section 14.3.

Example

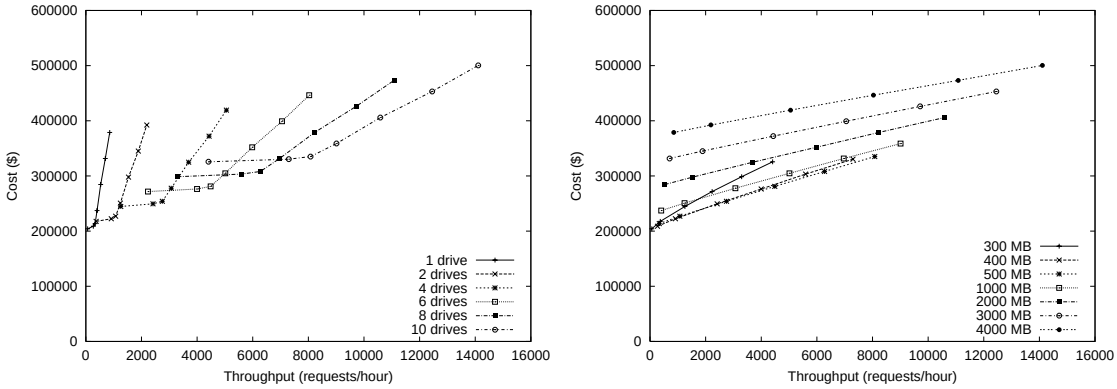
We start with an example to illustrate this issue. Figure 14.1(a) shows the throughput of a video archive utilizing nine library units each containing four MLR1 drives. The size of the disk cache is varied from 20 GB to 6 TB. Assuming the access distribution to be 90/9/1, Figure 14.1(a) shows that using a 1 TB disk cache, the archive is able to handle approximately 4100 requests for short video sequences during one hour. If the use of the archive increases, the throughput has to be increased correspondingly if we want to avoid increasing the average response time. As Figure 14.1(a) shows, by doubling the size of the disk cache the throughput can be increased from about 4100 requests per hour to about 4800 requests per hour, or a 17 percent throughput increase. One terabyte of hard disk is rather costly. With the price of hard disks we use in our simulations (see Table 9.8), one terabyte of disk costs $ 48,500. Could the performance of the archive be improved more by using the money on other hardware equipment than disk drives? The simulations performed in this section will answer this question.

14.3.1 Retrieval of Short Video Sequences

In order to study the relationship between cost and performance of a video archive using a disk cache and a given tertiary storage technology, we study the throughput of the archive as a function of the number of tertiary drives used in each library unit and the size of the disk cache. To reduce the simulation time, we perform the simulations for a limited number of combinations of drives per library unit and disk cache sizes, as given in Table 14.2. For each of the three tertiary drive types, we determine the throughput for each combination of the number of tertiary drives and cache sizes by means of simulations. We use the 90/9/1 access distribution and 30 seconds as the limit for the average response time.

Tandberg MLR1

Figure 14.9 contains the simulation results when using the MLR1 drive for retrieval of short video sequences of one minute. Since the goal is to study the relationship between the performance and the cost of the system, we present the simulation results as *parametric graphs*. The independent simulation variable (number of drives or cache size) is not shown on either axis. As we change the value of



(a) Total cost as function of the cache size

(b) Total cost as function of the number of tertiary drives

Figure 14.9 Throughput and storage cost of a video archive when varying the size of the disk cache and the number of drives. The figures contain simulations results using 1, 2, 4, 6, 8, and 10 MLR1 drives per library unit, and cache sizes of 300, 400, 500, 1000, 2000, 3000, and 4000 GB. The size of the requested video sequences is one minute.

the simulation variable, it traces a curve for the relationship between the *throughput* and the *cost*. In Figure 14.9(a) we present the *storage cost* as a function of the throughput using the cache size as the independent variable. For each of the different number of drives in each library unit, we make a separate line. Thus, on each line, the first point corresponds to a cache size of 300[2] GB and the last point corresponds to a cache size of 4000 GB. In Figure 14.9(b), the same simulation results are presented, but this time we use the number of drives in each library unit as the independent variable. Thus, each line now corresponds to a fixed cache size. It is important to note that the curves in Figure 14.9(a) and 14.9(b) are based on the same simulation results, but different ways to present the results are used.

The first important observation in Figure 14.9(a) is that when using MLR1 drives, we should let the requested throughput determine the number of drives in each library unit. For example, if the expected maximum load on the system is between 2000 and 3000 requests per hour, the figure shows that four drives per library unit give the lowest cost. (It is important to note that Figure 14.9 does not contain simulation results for three and five drives per library unit). Second, from Figure 14.9(a), we observe that for a given number of drives, there is only a limited part of the cache size curve where it gives the lowest cost. By counting the marks on the cache size curve and looking up the corresponding value

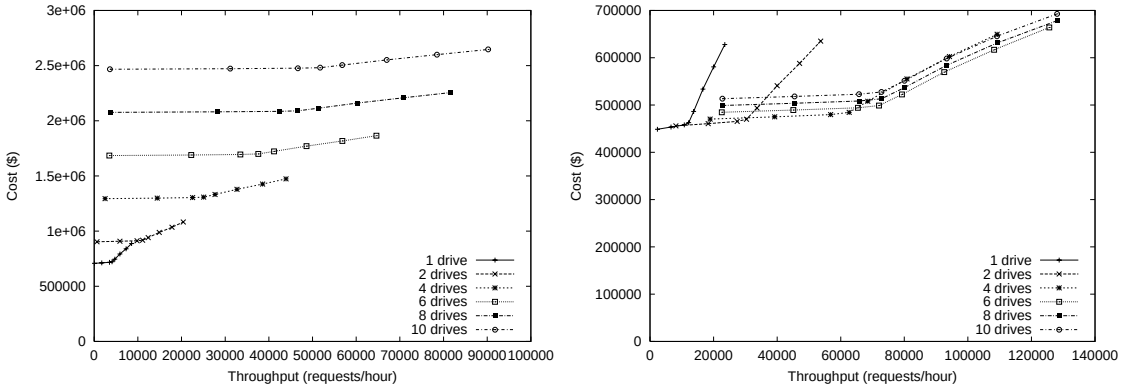
[2]For retrieval of short video sequences using MLR1 drives, a disk cache of 200 GB was not enough to retrieve video sequences with the given throughput criterion.

in Table 14.2, we find that for all curves in Figure 14.9(a), the cache size should be between 300 GB and 1000 GB. If a smaller cache size is considered, we should instead reduce the number of MLR1 drives. Similarly, if a larger cache size is considered, increasing the number of tertiary drives would be a better option. These two observations become even more clear when we study Figure 14.9(b). This figure shows that of the selected cache sizes we have used in these simulations, 500 GB is the cache size that gives the least total cost. Thus, using MLR1 drives for retrieving short video sequences, the optimal configuration of a video archive with regards to both cost and performance should have a disk cache somewhere between 300 GB and 1000 GB. Then, the number of MLR1 drives should be selected so that the archive fulfills the throughput requirement.

Before studying the simulation results for Magstar and DVD, we briefly return to the example we started out with. Figure 14.1(a) showed that by doubling the disk cache (from one TB to two TB), the throughput would increase by 17 percent. With the exception for having used a lower limit on the maximum average response time, the curve for 4 drives in Figure 14.9(a) contains the same simulation results as the curve in Figure 14.1(a). The fourth bullet on this curve gives the throughput and cost for an archive with a 1000 GB disk cache. The fifth bullet gives the throughput and cost when the cache size is doubled. As the figure shows, the cost increases approximately by $ 47,000, or by 17 percent. Instead of buying one TB of extra hard disks, we could have increased the number of tertiary drives to six. (Five drives would probably have been enough to increase the throughput by more than 17 percent, but since we do not have simulation results using five drives, we use the results for six drives in this example.) By studying the numbers behind the curve using six tertiary drives in Figure 14.9(a), we find that the throughput increases by 64 percent (from 3050 requests to 5000 requests per hour), while the cost only increases by 10 percent. Thus, we save $ 20,000 by increasing the number of tertiary drives in the library unit by two instead of doubling the cache size.

IBM Magstar

For an archive using Magstar drives, the situation is very different. Figure 14.10(a) contains the simulation results using Magstar drives. The cache size is the independent simulation variable along each curve in the figure. As Figure 14.10(a) shows, for the cache sizes used in these simulations, it is most cost efficient to increase the size of the disk cache instead of increasing the number of drives in the tertiary library units. Thus, an owner of a video archive using Magstar drives should start with one drive per library unit. As the throughput requirement increases, she should add hard disks to the disk cache until the size of the disk cache is at least 4 TB. Only if the load increases further, she should add more tertiary drives. The reason for using a larger disk cache when using Magstar drives compared to using MLR1 drives is the higher cost of the Magstar drive compared to



(a) Magstar (b) DVD

Figure 14.10 Throughput and storage cost of a video archive using Magstar and DVD drives when varying the size of the disk cache and the number of drives as specified in Table 14.2. The size of the requested video sequences is one minute.

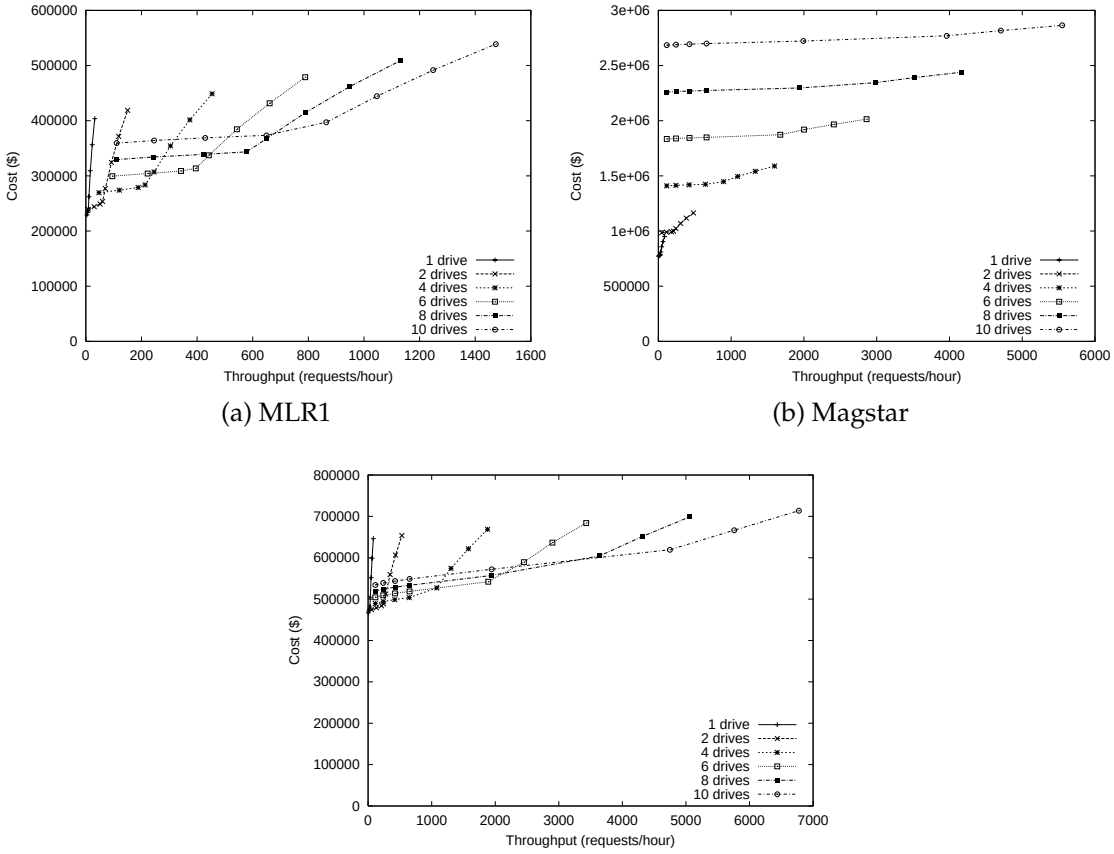
the cost of the MLR1 drive (see Table 9.9 for prices used for the tertiary drives).

DVD

Figure 14.10(b) contains the corresponding simulation results using DVD drives in the library units. In Section 10.2.2 we concluded that with the robot specification used in these simulations, four DVD drives was the maximum number of drives each robot could handle efficiently when retrieving short video sequences. The simulation results in Figure 14.10(b) show the same. Using eight or ten DVD drives in each library unit adds almost nothing to the throughput, and only increases the cost. Thus, using more than six DVD drives in each library unit is wasted resources. For an archive configuration where the number of drives is six or less, the curves in Figure 14.10(b) show approximately the same behavior as for the MLR1 drive. Ideally, the cache size should be in the interval from 300 GB to 500 GB, and then the number of DVD drives should be high enough to achieve the necessary throughput. Only if more than six drives are needed, the size of the disk cache should be increased.

14.3.2 Retrieval of Long Video Sequences

We performed the same simulations for retrieval of long video sequences of one hour. For all three tertiary drives, the results are presented in Figure 14.11. By studying these figures, we find that using the three different tertiary drives for re-



(a) MLR1

(b) Magstar

(c) DVD

Figure 14.11 Throughput and storage cost of a video archive using each of the three drive types when varying the size of the disk cache and the number of drives as specified in Table 14.2. The size of the requested video sequences is one hour.

	Libraries	Drives per library	Cache size	Total cost	Cost per stream
MLR1	9	10	2000	405900	0.0077
Magstar	23	2	500	917400	0.017
DVD	24	1	400	458100	0.0087

(a) Short video sequences

	Libraries	Drives per library	Cache size	Total cost	Cost per stream
MLR1	10	10	2000	444400	0.085
Magstar	25	4	2000	1494400	0.28
DVD	25	4	1000	527200	0.10

(b) Long video sequences

Table 14.3 Archive configurations and cost numbers used in the examples.

trieval of long video sequences gives similar performance results as when using them for retrieval of short video sequences. The main difference is for the DVD drive. For retrieval of short video sequences (see Figure 14.10(b)), the robot was the limiting resource when increasing the number of drives in each library unit. As shown in Figure 14.11(c), for retrieval of long video sequences the robot is not a limiting factor for the performance. We do not go further into details about the simulations results in Figure 14.11, but only briefly summarize the main results. Using MLR1 and DVD drives, the size of the disk cache should be between 400 and 1000 GB, and the number of tertiary drives should be selected so that the requested throughput is achieved. Using Magstar drives, a large disk cache should be used (larger than four TB), and the number of tertiary drives should be kept as low as possible.

14.3.3 Case Study

In this section, we have investigated the cost of a video archive as a function of the requested throughput and archive configuration (size of disk cache and tertiary drive type). To make some of these cost figures more comprehensible, we end this section with two small examples.

The first example is a news archive storing short video clips. We assume the amount of video and access distribution are the same as we have used earlier in this chapter. The average size of the requested video sequences is one minute. The

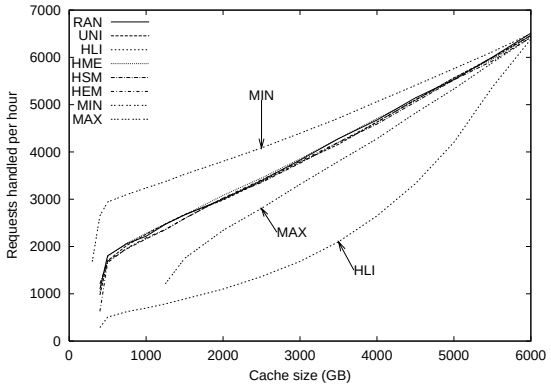
owner wants the archive to be able to handle maximum 10,000 requests per hour. By studying the numbers behind the curves in Figure 14.9 and 14.10, we find the least costly configurations for the archive using each of the three tertiary media drives. These configurations are given in Table 14.3(a). To make this example more realistic, we assume the average utilization to be twenty percent. Thus, in average, the system delivers 2000 news clips to users each hour. Further, if we assume that each of the hardware components of the archive have an average life time of three years, we can compute the average cost of retrieving one video sequence. These numbers are given in the last column of Table 14.3(a). Using the least costly configuration, the archive cost contributes with $ 0.0077 to the total cost of delivering one news clip of one minute to a user.

In the second example, the archive is used for a video-on-demand service. The requirement is that it should be able to handle a maximum load of 1000 requests for movies per hour. Although most movies are longer than one hour, we use one hour as the average movie length since this is the size we have used in our simulations. We assume that in average 200 movies are requested each hour. From Figure 14.11, we get the possible archive configurations given in Table 14.3(b). As in the first example, we assume an average discount time for the hardware components to be three years. Thus, the archive cost per movie delivered to a user can be computed as the total cost of the archive divided by the number of movies delivered during three years. These costs are given in the last column of Table 14.3(b). Today, it costs about two or three dollars to rent a movie in a movie store. Compared to this, an archive cost of $ 0.085 per movie is only about four percent of today's video rental cost.

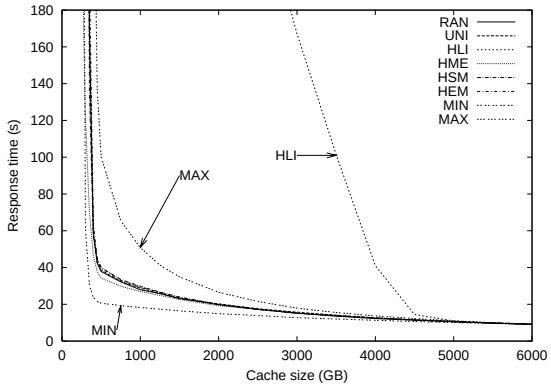
In both of these examples, we have only considered the cost of the storage equipment. This is only one of several factors which add to the final cost of delivering a news clip or a movie to a user. We have not considered cost of the video server software and hardware or network cost. For example, having a network infrastructure capable of delivering a total of 5 Gbit/s of data to users of distances up to several kilometers are still rather costly. Also the cost of operating such a service and royalties to content owners add substantially to the total price.

14.4 Effect of Data Placement on Tertiary Media

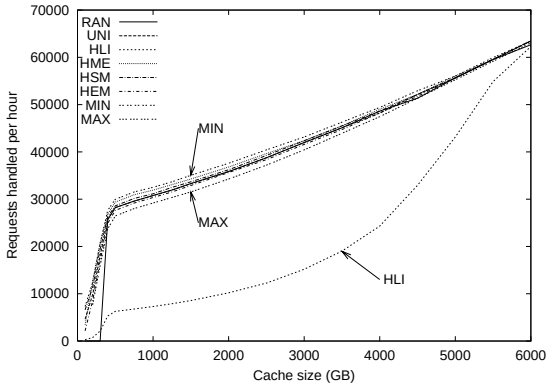
We end this investigation of the use of a disk cache with a study on the effect different allocations of the video sequences to tertiary storage media and library units have on the performance of the archive. In Chapter 13, we saw that the throughput of a tertiary storage system using the best allocation strategy could be as much as ten times higher than the same system when the worst allocation strategy was used. In this section, we perform similar investigations in order to find out if and how the results from Chapter 13 are changed by the introduction of a disk cache.



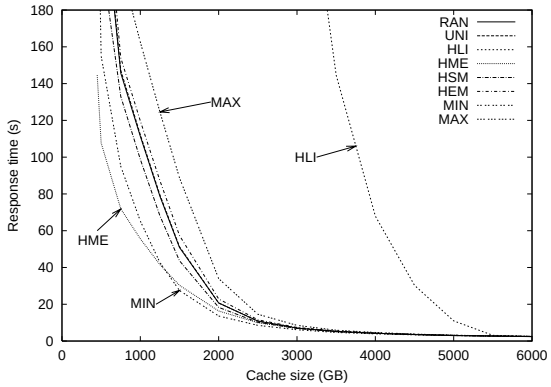
(a) MLR1 - Throughput, limit=20 seconds



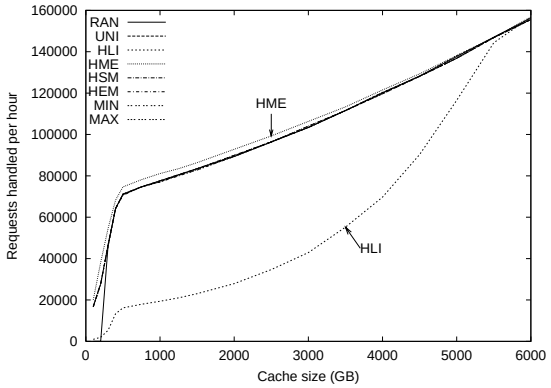
(b) MLR1 - Response time, requests=3000



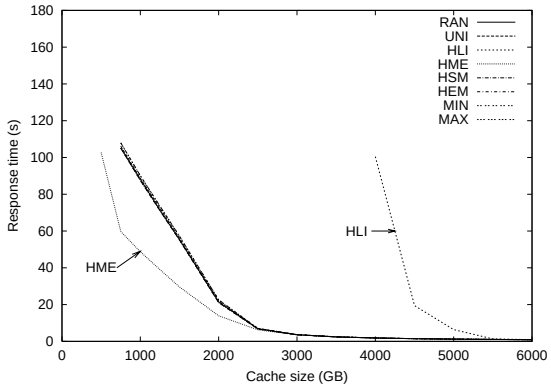
(c) Magstar - Throughput, limit=20 seconds



(d) Magstar - Response time, requests=36000

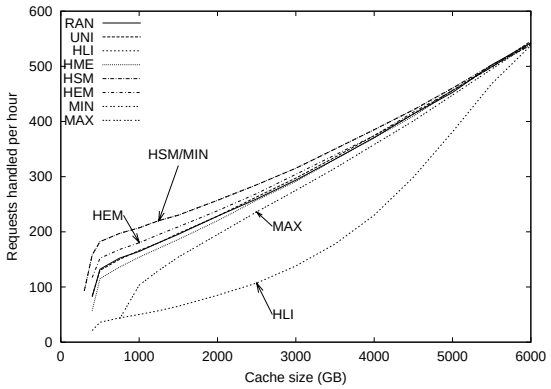


(e) DVD - Throughput, limit=20 seconds

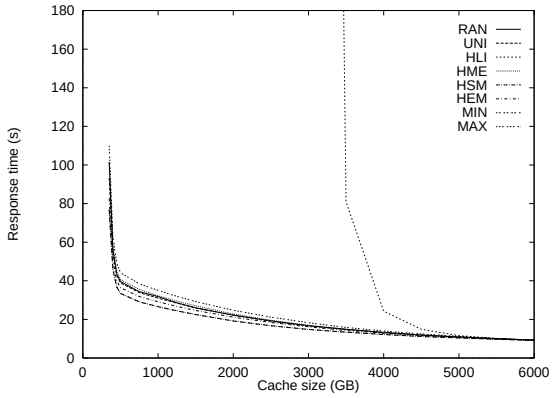


(f) DVD - Response time, requests=90000

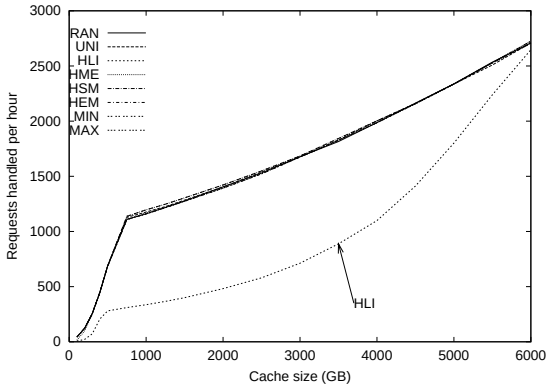
Figure 14.12 Throughput and average response time as function of cache size for the different allocation strategies. Each request is for a one minute video sequence. For the throughput simulations, the average response time limit used is 20 seconds.



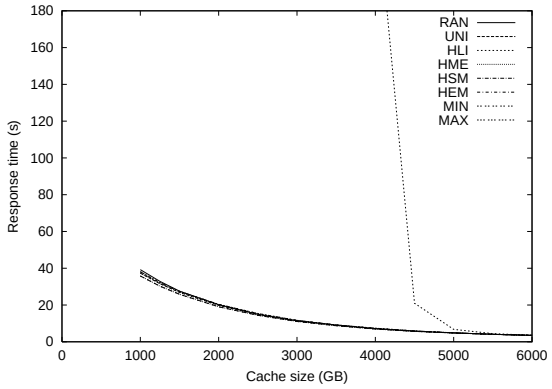
(a) MLR1 - Throughput, limit=20 seconds



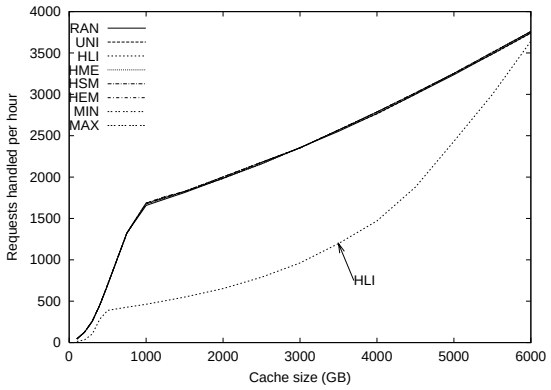
(b) MLR1 - Response time, requests=250



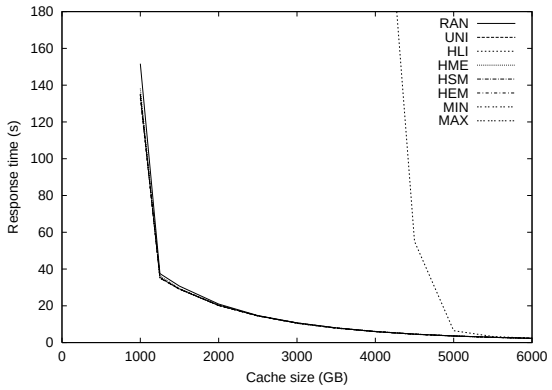
(c) Magstar - Throughput, limit=20 seconds



(d) Magstar - Response time, requests=1400



(e) DVD - Throughput, limit=20 seconds



(f) DVD - Response time, requests=2000

Figure 14.13 Throughput and average response time as function of cache size for the different allocation strategies. Each request is for a one hour video sequence. For the throughput simulations, the average response time limit used is 20 seconds.

For each of the allocation strategies presented in Section 13.1, we perform simulations to determine the throughput and response time as a function of the cache size. The throughput curves when retrieving short video sequences are presented in the left column in Figure 14.12. We use an average response time limit of 20 seconds as the criterion for the maximum throughput the archive can handle. In the right column of the figure, we present response time curves for *one constant* user load. The corresponding simulation results when retrieving video sequences of one hour are presented in Figure 14.13.

Effect of Using Allocation Strategies

For all simulations, the curves in Figure 14.12 and 14.13 show that the assignment of video sequences to tertiary storage media and library units is important for the performance of the system. The difference in the performance of the allocation strategies are larger for small cache sizes. For retrieval of short video sequences using MLR1 drives in the library units, the throughput of the best allocation strategy is up to six times higher than using the worst allocation strategy (see Figure 14.12(a)). As the size of the disk cache is increased, the relative performance difference becomes less, and with a cache size of 6 TB, the difference in performance between the allocation strategies is less than two percent. The reason the difference between the allocation strategies decreases as the size of the disk cache increases, is that most of the *hot* and *warm* video sequences will be present in the video cache. Only *cold* video sequences have to be fetched from tertiary storage. Thus, the access probability for the video sequences being retrieved from tertiary storage will be equal, and thus are not influenced by allocation strategy. Actually, the allocation strategies that optimize the access time for *hot* and *warm* video sequences, e.g., the *Minimum Seek Time* and the *Hot Start of Media*, now perform slightly worse than the corresponding access strategies which put the *hot* and *warm* video sequences on the physical or logical end of the medium (e.g., the *Maximum Seek Time* and the *Hot End of Media*). Note that this is only the case in this study since the video sequences in the *cold* partition have the same access probability.

Comparing the Allocation Strategies

If we study the throughput and response time curves in Figure 14.12 and 14.13 in order to find what are good and what are bad allocation strategies, we find that for most of the allocation strategies there is little difference in the performance. However, some of the allocation strategies perform notably better or worse than the rest. For retrieval of short video sequences, the *Minimum Seek Time* strategy yields the highest throughput and shortest response times for the tape based systems. For the DVD based system, the *Hot Media* strategy gives the best results. This might be surprising, since all the hot videos should be available in the cache. The reason the *Hot Media* strategy performs better is that the hot media is not

used at all. This reduces the number of DVD's that are in use, and as a result, the average number of requests served per load of a DVD is increased. For retrieval of long video sequences, the *Minimum Seek Time* and *Hot Start of Media* give the best performance. These two allocation strategies give the same results due to identical assignment of video sequences to tertiary drives when only a few video sequences can be stored on each storage medium.

Compared to the *Uniform* or *Random* allocation strategy, to use the *Minimum Seek Time* strategy can notably increase the performance for the tape based systems. For example, an archive using MLR1 drives and a one terabyte disk cache for storing short video sequences, the throughput can be improved by 46 percent by using the *Minimum Seek Time* allocation strategy compared to using a random allocation of the video sequences (see Figure 14.12(a)). Similarly, as Figure 14.12(b) shows, the average response time can be reduced from 28 seconds to 18 seconds, an reduction by 36 percent. For an archive using Magstar the throughput improvements by using the *Minimum Seek Time* strategy is 6 percent. Similarly, for an archive using DVD, the throughput improvements by using the *Hot Media* strategy is only 5 percent better than using the *Uniform* or *Random* allocation strategy.

Comparing Results from Cache versus Non-Cache Study

If we compare these results to what we found in the similar study for the tertiary storage system in Chapter 13, the main results are the same, with one important exception. For retrieval of short video sequences without using a disk cache, the *Hot Media* allocation strategy gave the best performance for a tertiary storage system containing four media drives per library unit. The reason for this was that each time a *hot* tertiary media was loaded into a drive, multiple requests could be executed. When introducing the disk cache, these *hot* tertiary media are hardly used since the video sequences stored on them already are present in the disk cache.

14.5 Evaluation of the Video Cache Model

In this chapter, we have deliberately used a very simple model for the disk cache. The only parameter specifying the disk cache has been the amount of video data that can be stored in the cache. We have assumed the disk cache could deliver all the requested videos and neglected any delays introduced by the disk cache in the simulation model.

The reason for using such a simple model was mainly motivated by the following two reasons: First, by including a more detailed model of the disk cache in the simulation model would have complicated the simulator even more. Second, and most importantly, by including a more realistic model for the disk cache, the

disk cache would influence the performance results and might become the bottleneck in the storage system. The simulation results would become dependent on the architecture and the performance characteristics of the disk cache.

In real systems, it is likely that the disk cache will be one of the components that limit the performance of a video archive server, but in our experiments the main purpose was to study the behavior and performance of the tertiary storage system.

The purpose of this section is to provide a more detailed analysis of how a disk cache may influence the performance. We present a model of a disk cache in order to illustrate the main operations the disk cache performs and the resources it will consume in order to deliver the video streams to the user. An analytical model of a tertiary storage system caching videos on disks are found in (Chan and Tobagi, 1999; Chan and Tobagi, 2003).

14.5.1 Response Time

In our model we have assumed that the disk cache does not incur any delay to the response time experienced by the user. For the requests served from tertiary storage, we have used the response time of the tertiary storage. For the requests served from the disk cache, we have used a zero response time. This is an optimistic assumption. The purpose of this subsection is to provide data about what would be a more realistic assumption for the response times.

Requests Served by the Disk Cache

The response time for requests served by the disk cache becomes highly dependent on the strategies used for designing and implementing the disk storage system. In Section 4.4, we gave an overview of the main strategies for designing a storage system for a video server. For an archive of the size studied in this chapter, it is likely that the video server will consist of multiple machines. In order to achieve an even load on the server machines and the hard disks, the videos may be distributed over multiple machines and disks by using some kind of *striping* (Özden et al., 1996b) or *random data allocation* (Santos et al., 2000). Common to most disk based storage systems for video is that before streaming of video to the user can be initiated, one or a few blocks of video data must have been read from disk to a main memory buffer, and that the storage system must provide either statistical or deterministic guarantees that it is able to continue retrieving data from disk at minimum the same rate as the bandwidth of the video. To guarantee this, many systems use round-based scheduling of the video streams. In such a system, the initial delay is highly dependent on the length of each slot and when the first free slot becomes available. A more detailed presentation of the factors that influence the response time is found in (Chang and Garcia-Molina, 1997b).

Analytical model. Chan and Tobagi (1999) present an analytical model for the average start-up delay for a disk cache. This model assumes the disk cache has bandwidth to serve a fixed number of video streams. The model estimates the amount of time a new request has to wait for available bandwidth in order to be started. The model shows that as the number of streams the disk cache is able to deliver increases, the higher can the utilization of the system be without increasing the initial response time. For a lightly loaded system, the initial latency will be very low, but then increases rapidly when there are resource conflicts. As an example, for a system capable of delivering 80 concurrent video streams, with a utilization of 80 percent, the average response time will be about 10 seconds. The model does not include the initial delays imposed by the disk scheduling strategy or time needed for reading the first data buffers from disk.

Practical experiments. The Tiger video server uses striping of the video and round-based scheduling for reading of data from disk (Bolosky et al., 1996). In order to reduce the initial latency, *thrifty scheduling* is proposed (Douceur and Bolosky, 1999). Using this scheduling strategy, at 90 percent utilization, the system is able to deliver videos with an average response time of 1.38 seconds. The Tiger video server has been deployed in several trials with hundreds of users. From these experiments, it has been reported that the average response time for starting a new video has been in the order of a few seconds (Jones, 1997). For the RIO Multimedia Storage System, which uses random data allocation, even lower response times have been reported (Santos and Muntz, 1998). For a system that utilize 80 percent of the disk bandwidth, the "guaranteed" delay introduced by the disk system is approximately one second. By use of replication of video data, the "guaranteed" delay is reduced to approximately 200 milliseconds. For experiments performed using the Elvira II video server, the average response time when starting a new video was approximately 1.1 seconds (Brataas, Klovning, Sandstå and Torbjørnsen, 1998b).

Requests Served by the Tertiary Storage System

The simulation model assumes that *pipelining* is used for the videos delivered from the tertiary storage (Ghandeharizadeh et al., 1995; Chan and Tobagi, 2003). Thus, the streaming of the video to the user can start as soon as the first blocks of video data are present either in main memory or in the disk cache. Assuming the video can be delivered directly from main memory without having to be written to disk first, all tertiary drives used in this study are capable of delivering the first 128 KB of a video stream to memory without adding more than 100 ms to the response time.

If the video data first has to be written to the disk cache, and then read from the disk cache before the data can be streamed to the user, the delay would become noticeable higher and be dependent on the strategies and design used by the disk

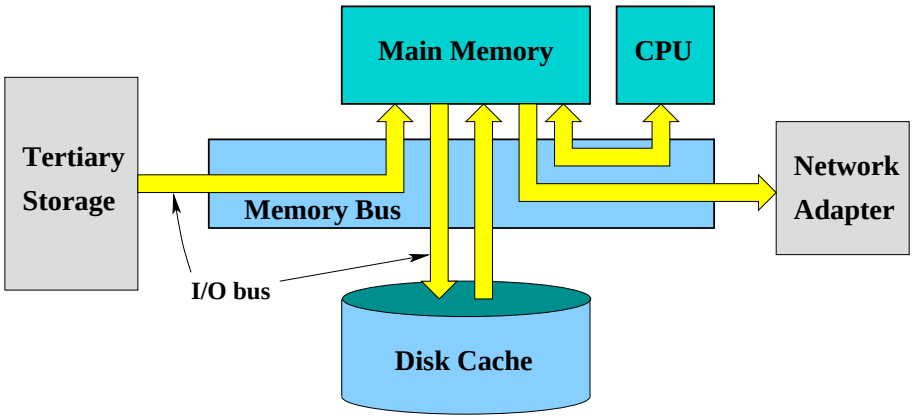


Figure 14.14 The main components for the storage system in a video archive server and the main data flow for transporting video data.

cache. The Elvira II video server uses this strategy, but it would have been easy to extend it to deliver the first part of a video directly from main memory without going through the disks.

Conclusions

These examples show that it is possible to implement a video cache with an average response time of less than two seconds. For the videos that can be delivered from the disk cache it is likely that they would have a response time in the order of one to two seconds. For the videos that have to be retrieved from tertiary storage, the extra delay could have been less than 100 ms. As a result of the delays introduced by the disk cache, the response times presented in this section would have been slightly higher if we had included a more realistic model for the disk cache in the simulation model.

14.5.2 Disk Cache Model

In Chapter 9, the simulated disk cache was presented. The only resource included in the simulation model was the amount of storage space. In this subsection we present a model of one possible implementation of a disk cache. The purpose is to provide a more detailed presentation of how a disk cache can be implemented and the main resources consumed during video delivery. We use this model to give some examples of the resources needed for the disk cache used in the simulations presented earlier in this section.

We use the same architecture as for the video archive simulated in this chapter, but add more details about the internals of the disk cache. An overview of

the video cache is presented in Figure 14.14. The main hardware components needed to implement the video archive consists of the tertiary storage system, the collection of hard disks, main memory, the memory bus, I/O buses, the network adapter, and the CPU. In the model, the load on the system is given by the arrival rate of λ requests per second. Each request is for a video sequence with a bandwidth of b bytes per second and having a length of l second. The cache hit probability is given by α. A similar model of the data transport in a video server using tertiary storage can be found in (Doğanata and Tantawi, 1996).

From Tertiary Storage to Disk Cache

The fraction of the requests that can not be served directly from the disk cache must be read from tertiary storage to the disk cache. As shown in Figure 14.14, the video data will be transferred to the disk cache by first transferring it from the tertiary drive via an I/O bus and the main memory bus to main memory. From main memory it will be written to disk via the main memory bus and an I/O bus.

The required data rate for transporting the video is $\lambda(1-\alpha)bl$ byte/s. Thus, this is the required bandwidth for the tertiary storage devices, the tertiary I/O bus, secondary I/O bus, and hard disks. In addition, the data is transported twice on the memory bus.

From Disk Cache to the Network

The disk cache must deliver all the video sequences. To do this, the disks must deliver λbl byte/s. The data will be transferred from the disks via an I/O bus and the main memory bus to main memory.

From main memory the data should be sent to the client using the network. This also requires moving the data. There exists network technologies like VIA and Infiniband, which can do this in user space and thus only require one transfer of the data across the main memory bus. Also operating systems that support zero-copy data paths can do the sending using one copy of the data (Halvorsen, Plagemann and Goebel, 2003b). A more standard implementation is to use the OS's communication subsystem. This will likely require the data to be transported three times over the main memory bus. The copying from user space through the network stack is likely to require use of the CPU, and thus the data must be transferred twice over the memory bus, and then, the data must be transferred from the protocol stack's message buffers to the network adapter. And finally, the network must support a bandwidth of at least λbl byte/s.

Resource Usage

Table 14.4 contains an overview of the data rates the main components in a video archive must be able to transport. These expressions give average values for the data rates. Due to variations in the arrival of new requests, there will be periods

	Read	Write	Transfer
Tertiary storage	$\lambda(1-\alpha)bl$		
Disk	λbl	$\lambda(1-\alpha)bl$	
Main memory	$\lambda(3-\alpha)bl$	$\lambda(3-\alpha)bl$	
Main memory bus			$\lambda(6-2\alpha)bl$
Secondary I/O bus			$\lambda(2-\alpha)bl$
Tertiary I/O bus			$\lambda(1-\alpha)bl$
Network			λbl

Table 14.4 Amount of data transfer performed by the main components in the video archive.

where the load will be higher. We use this model of the data transfer in a video archive to provide some examples of the work the video cache has to perform.

For a video cache that should deliver a high number of video sequences, it is likely that the video server will consist of multiple machines. In order to achieve load balancing across the machines, there will probably be need for redistribution of some of the video data that is read from tertiary storage. The model does not include any resources for redistribution of video data between machines.

14.5.3 Throughput

In this chapter we have presented numerous figures for the throughput of a video archive using a disk cache. In the evaluation of the throughput that could be achieved by using a disk cache in Section 14.2 we showed that the video archive could deliver up to 150,000 one minute video sequences or 3329 one hour video sequences per hour. In Section 14.3, we showed examples with even higher throughput numbers.

In this subsection, we use one example from our simulations to illustrate the work that has to be performed by the video cache and present some possible design alternatives for how such a video cache could be designed. We use the configuration used for evaluating the throughput of a video archive using DVD drives and a disk cache for delivering video sequences of one hour. The throughput is presented in Figure 14.2(c). In this example, we assume the video sequences are accessed using the 90/9/1 access distribution since this gives the highest load on the storage system.

We use the analytical model presented in the previous section to estimate the data transfer rate the main components in the system have to handle. As input to the model we use the capacity number and corresponding cache hit rates from using the 90/9/1 access distribution in Figure 14.2(c). The data rates, which the tertiary storage system, the disk cache, the main memory bus, and the network

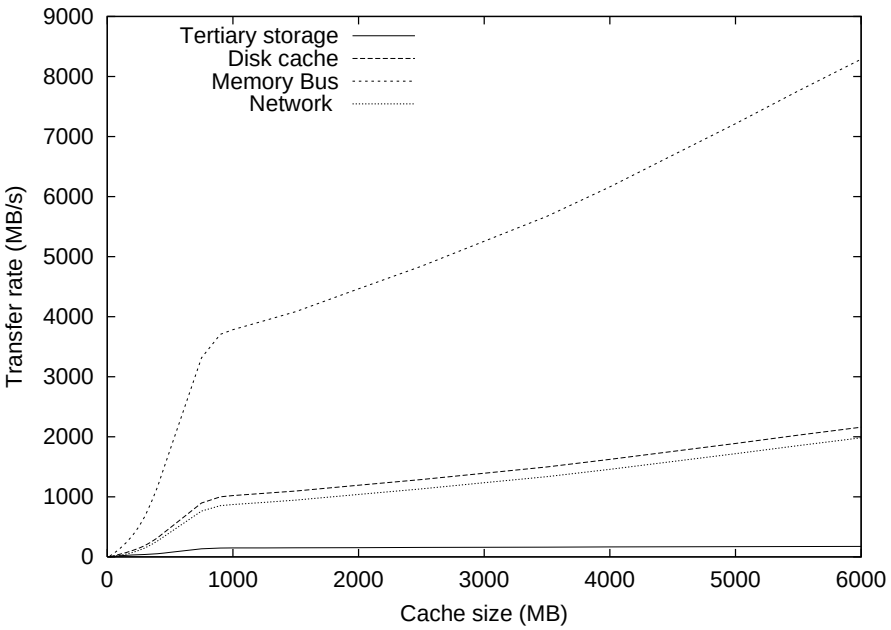


Figure 14.15 Data rates for the main components in the video archive. The load and the cache hit rates are taken from the 90/9/1 access distribution in Figure 14.2(c).

has to handle is plotted in Figure 14.15. In the remainder of this section we discuss and give an example of how a system handling this load can be realized. In the example we focus on a system containing a 6000 GB disk cache. In average, this system should be able to deliver 3329 one hour long video sequences per hour. The detailed results from the simulations shows that the maximum number of concurrent requests was 3565.

From Tertiary Storage to Disk Cache

The tertiary storage system consists of 25 library units each containing four DVD drives. Each DVD drive has a transfer rate of 2.64 MB/s (see Table 9.3). Thus, the maximum transfer rate of the tertiary storage system is 264 MB/s. In average, not all of this will be used. With a 90/9/1 distribution and the given size of the cache, 91 percent of the requests will go to the tertiary storage system. In average, the tertiary storage system will be delivering 300 video sequences per hour, or a data rate of 179 MB/s.

The video data will be transferred to the disk cache by first transferring it from the tertiary drive via an I/O bus and the main memory bus to main memory. Since each library unit contains four drives delivering a maximum data rate of 11.56 MB/s, the I/O bus could be realized using a wide SCSI-2 bus. From main

memory, the data will be written to disk via the main memory bus and the disk systems I/O bus.

Disk Cache

The disk cache must in average deliver 3329 video streams. In addition, it must store the 300 videos transferred from tertiary storage. To do this, the disks must read/write 2.1 GB/s. The data will be transferred from the disks via an I/O bus and the main memory bus to main memory.

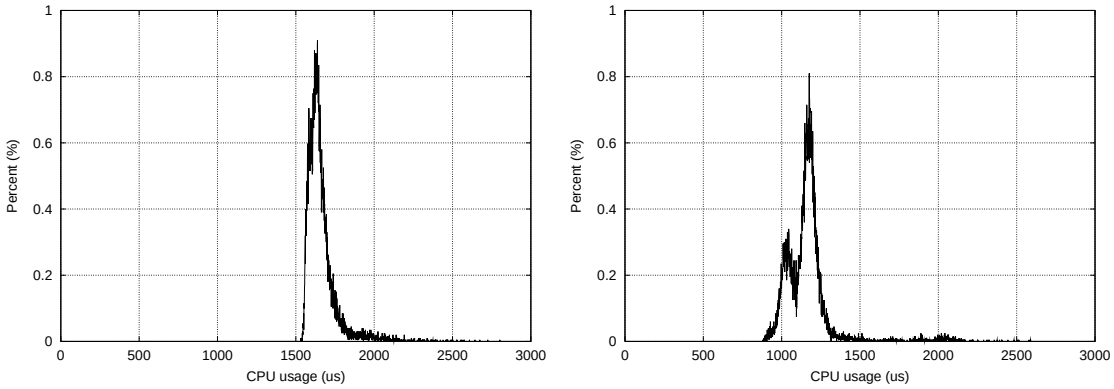
The disk cache should be able to store 6000 GB of data. Hard disks are available in a wide variety of specifications with regards to storage capacity, transfer rate and cost. In this thesis we have used the Seagate Barracuda 18LP as a representative hard disk for use in the disk cache (see Table 9.8). This disk stores 16.9 GB of data and has a specified transfer rate of 18 MB/s. If this disk is used for the disk cache, 355 disks are required. This will give the disk cache a theoretical transfer rate of approximately 6.2 GB/s, which should be more than enough for transferring the video to and from main memory. The challenge is how to organize the disks and to ensure that the load is distributed evenly across the disks.

Network Subsystem

Each video stream requires 5 Mbit/s of network bandwidth. With an average of 3329 video streams, a total network bandwidth of 16.6 Mbit/s is required. In the Elvira video servers, we have used OC-3 ATM adapters with a theoretical transfer rate of 155 Mbit/s. In a real application, each ATM adapter should be able to deliver at least 100 Mbit/s. With the recent improvement in Ethernet technology, 100 Mbit/s and 1 Gbit/s Ethernet might be an alternative technology at a lower price. Thus, assuming that it is possible to get an even load on the system, 166 ATM-interfaces each delivering 100 Mbit/s of video data or 208 100 Mbit/s ethernet interfaces each delivering 80 Mbit/s of video data should be able to deliver the 3329 video sequences to the users.

Main Memory System

The main memory and the main memory bus are the components in the system that must provide the highest transfer capacity. As shown in Figure 14.15, for the system used in this example, the main memory system must handle a transfer rate of 8.1 GB/s. Since the main memory bus also transfer code and other data, the aggregate bandwidth of the memory buses in the system must be considerably higher than this. This transfer rate is higher than most single-bus systems can handle, and it is likely that in order to support this transfer rate multiple machines must be used.



(a) Disk segment

(b) Network container

Figure 14.16 a) The distribution of CPU usage for reading 128 KB segments from disk. **b)** The distribution of CPU usage for sending an 8 KB container as a UDP message on ATM. The measurements are performed using the Elvira II video server running on a 125 MHz HyperSparc processor. Due to overhead in the measurements of the CPU usage when sending UDP messages, this figure shows values that are approximately 300 us higher than the real CPU usage.

Most of the experiments using the Elvira II video server were performed using Sun Enterprise 2 machines. The memory bus on these machines has a sustained transfer rate of 1.24 GB/s. Today (2004), corresponding machines have a memory bandwidth of up to to 9.6 GB/s.

CPU Usage

The last main resource used during video delivery is the CPU, which runs the video server software that administers the video delivery. It is hard to estimate the amount of CPU resources required in a video server. To give some indication of the amount of CPU required for delivering a video stream, we provide an example for the Elvira II video server. The main operations performed by the video server during video delivery is to read disk segments from the disk and into memory, and deliver the video containers from main memory to the network:

- **Disk reading.** The average CPU usage for reading one 128 KB segment from disk to main memory was measured to be 1.65 ms. Figure 14.16(a) contains the distribution of CPU usage per disk segment read from disk.
- **Network sending.** The disk segments contain 8 KB containers, which are sent to the ATM network adapter. The average CPU usage for sending one

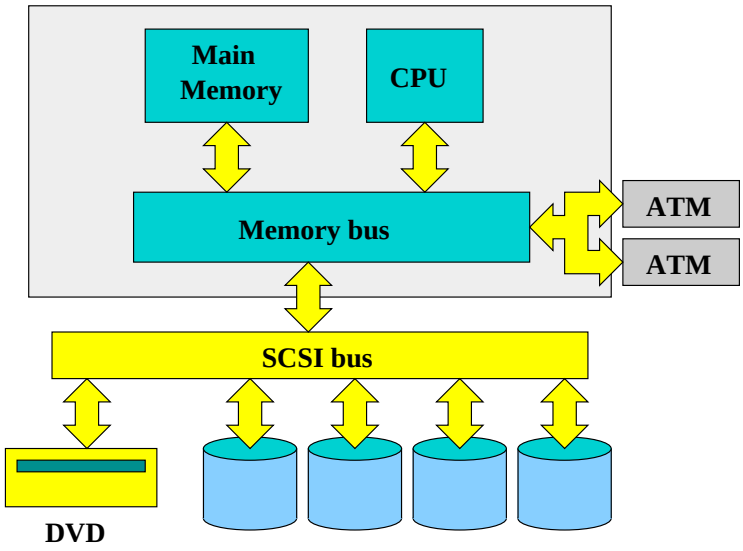


Figure 14.17 The main components of one machine in a cluster based video server.

container was measured to be 0.82 ms. Figure 14.16(b) contains the distribution of CPU usage per container sent to the network.

These measurements were performed using a 125 MHz HyperSparc processor (which was state-of-art in 1996). Delivering one 5 Mbit/s video stream requires 4.8 disk segments to be read from disk and 77 containers to be sent to the network each second. Thus, using this processor, each video stream would use approximately 71 ms of CPU, making it able to deliver approximately 14 video streams of 5 Mbit/s. Compared to the processor used in the measurements for Elvira II, today's (2004) processors have a processing capacity that is 20-30 times higher.

Video Server Configuration

We end this section with giving one example of how a video server capable of handling the more than 3300 video streams we have used as example in this section can be implemented. There are many alternative ways to build such a server. It could consist of a single machine, e.g., a multiprocessor[3], a parallel machine like the nCube, or it could be built using a cluster of many smaller machines. The Elvira II server is an example of the last architecture, and we use this as the example. The purpose is to give an example of how such a video archive server can be implemented, not to provide an optimal video server configuration.

[3]In 1999, one Sun Enterprise 10000 server equipped with 924 disk with a total storage capacity of 8.4 TB was capable of processing more than 3 GB/s of data read from the disks, mainly as 1 MB data segments (Sun Microsystems, 1999).

Earlier in this section we have established that the system should contain 100 DVD drives (distributed into 25 library units), 355 hard disks and minimum 166 ATM-interfaces (or more than 200 100 Mbit/s ethernet interfaces). If this server was built using technology from 1998, one possible configuration would be to use 100 server machines. Figure 14.17 contains an overview of one such machine. Each machine would be equipped with:

- Four SCSI disks, each responsible for an average data transport of 6.1 MB/s.
- One DVD drive. An advantage of having each library unit connected to two computers is that the video would be available even if one of the computers would become unavailable.
- One SCSI Fast-40 I/O bus capable of transporting 80 MB/s.
- Two ATM-adapters.

In average, each machine would be responsible for delivering 33 video streams. The data transfer on the main memory bus would require a bandwidth of 83 MB/s. To ensure load balancing, it is likely that some form of striping would be used. This might increase the load on the system since the videos read from tertiary storage would have to be distributed on the machines.

If this system were to be built today (2004), a smaller number of machines and hard disks would have required. The SCSI Fast-40 I/O bus would likely have been replaced by a Ultra320 SCSI bus. The ATM-adapter would have been replaced by a 1 Gbit/s ethernet interface.

Concluding Remarks

In this subsection we have presented one example of how a disk cache could have been implemented. In the example we have used performance data from one of our simulations. The example shows that with the disk model used as an example in this thesis, the disk cache should be able to deliver the required amount of video data.

We have used the highest capacity number found from using DVD drives in the simulations performed in Section 14.2 as the example. This example results in the highest load on the system. We have shown that with the available resources in the disk cache, mainly the disk bandwidth, the system should be able to deliver the required amount of data. Although this is the example with the highest load on the system, we could also have made an example where the system probably would not be able to deliver the number of video streams given by the simulation results. For instance, with a 1 TB disk cache, the system should be able to deliver approximately 1500 video streams. Although the load is halved, the number of disks in the system is reduced six times. Figure 14.15 shows that the disk system

would be required to transfer in average 1020 MB/s of video data. With the selected disk, this would not be possible. To overcome this problem, the aggregate transfer rate of the disk system had to be increased, by either using faster disks or increasing the number of disks (possibly using smaller disks).

14.6 Conclusions

In this chapter we have evaluated the use of hard disks for caching the most frequently used videos in a video archive based on tertiary storage. We have studied three different issues related to introducing a disk cache in a tertiary storage system used for storing and retrieving digital video.

Cache Hit Rate and Performance

For most systems caching data on faster and more expensive storage media, the purpose is to increase the throughput and reduce the average response time. For this to have any effect, it is necessary that some of the videos are accessed more frequently than other videos. We have studied how access distributions and the size of the disk cache influence the cache hit rate and the performance and cost of the video archive. The main conclusions from this part of the study are:

- The performance of the archive becomes highly dependent on the skewness of the access distribution. The more skewed the access distribution is, the higher throughput or lower average response time can be achieved. We have shown examples where using a disk cache capable of storing five percent of the video sequences increases the throughput by 1500 percent *or* reduces the average response time by approximately 90 percent when the videos are accessed with a 90/9/1 access distribution.

- The more skewed the access distributions is, the smaller the disk cache can be and still result in a large performance increase.

- For access distributions that are close to be uniform, the effect of using a disk cache is small. We have shown examples where including a disk cache increases the average cost per retrieved video sequence.

In Chapter 12, we studied how the performance of a tertiary storage system was influenced by different access distributions. The main conclusion was that the throughput and response times were mainly unaffected by the access distribution. The reason for this was that the videos were stored on the same type of storage medium, and thus had approximately the same access costs and access times. The reason for the access distribution to have a high impact on the performance when using a cache is that the storage media used in the cache and in the tertiary storage system have very different access costs and access times.

Disk Cache Size versus Tertiary Bandwidth

By including a disk cache in the video archive, the size of the cache becomes a new configuration parameter, which must be selected in order to optimize the performance and cost of the archive. The performance of the video archive is improved by increasing the size of the disk cache. Alternatively, the performance may also be improved by increasing the transfer bandwidth of the tertiary storage system. We have evaluated how the size of the disk cache and the number of drives in the tertiary storage system must be configured to get an optimal performance and a lowest possible cost. The main conclusions are:

- The cost of a video archive with a given performance can be reduced much by selecting the optimal combination of disk cache size and number of tertiary drives in the library units.
- The ratio between the size of the disk cache and the number of drives in the tertiary storage system is highly dependent on the throughput and cost of the tertiary drives.
- Our simulation results show that the optimal size of the disk cache is more or less independent of the throughput of the system. To adjust for a higher or lower throughput requirement, the number of tertiary drives should be adjusted. The reason for this is that the disk cache should mainly hold the most popular video sequences, while the tertiary storage system should handle the less popular video sequences. When increasing the load on the system, the number of popular video sequences does not increase, but the number of requests which goes to less popular video sequences increase. It should be noted that our experiments were performed with one fixed access distribution. As shown in Section 14.2, the access distribution has a high degree of influence on the throughput that can be achieved for different cache sizes.

In this part of the study, we have also shown how the video archive simulator can be used for determining the size of the disk cache and which technology and configuration that should be used in the tertiary storage system in order to achieve a given performance to a lowest possible storage cost.

Video Allocation Strategies

In Chapter 13 we studied allocation strategies for allocating video sequences to storage media and library units based on the access probability of the video sequences. In this chapter we have performed a similar evaluation of the same allocation strategies to determine if the use a disk cache has any influence of the the conclusions from Chapter 13.

- Even when the most popular video sequences are cached on disk, the allocation strategy for video sequences to tertiary storage media and library units may have a large impact on the performance of the video archive.
- Caching the most popular video in a disk cache reduces the skewness of the access distribution for the videos that have to be served by the tertiary storage system. This may reduce the negative effect of a bad allocation of videos to storage media and library units.
- As the size of the video cache increases, the difference between the allocation strategies decreases.
- In order to make a good allocation of the video sequences to storage media and library units, you have to have both information about the access distribution and the a model of the behavior of the storage system.

The evaluation of the different allocation strategies shows that the relative ordering of the allocation strategies with regards to throughput and response time is mostly unchanged compared to the results found in Chapter 13.

Chapter 15

Technology Development

Software is slowing faster than hardware is accelerating.

Wirth's Law

In this thesis, we have studied properties and performance of using different tertiary storage technologies for storage and retrieval of digital video. Within data engineering, there is hardly anything that changes so rapidly as processing and storage capacity. This makes it harder to do research that involves the performance of processors and storage devices since performance is improving so fast. Most of the research done in this thesis is based on specifications of technology that were available in 1998. In this chapter, we give a short overview of how the performance of storage technologies used in this thesis have changed during the last six years and we briefly discuss how this influences the results presented in the thesis.

15.1 Technology Development

One of the best known rules of thumb in data engineering is *Moore's Law*. Originally, Moore's Law applied to circuit densities, but it is also used for microprocessors, main memory (RAM), and disk storage capacity. According to Moore's Law, storage capacity should be doubled every 18th months, or become four times larger every three years. Based on this, the storage capacity should today (2004) be about eight times higher than in 1998. This section presents an overview of the development of the storage technologies used in this thesis during the last fire years. Table 15.1(a) contains an overview of the storage capacity, bandwidth, and storage and bandwidth cost for some of the technologies used in the research. In Table 15.1(b), we have included performance data of corresponding storage devices that are available in 2004.

	Storage (Gbytes)	Bandwidth (Mbytes/s)	Media cost ($/GB)	Bandwidth cost ($/(Mbytes/s))
Seagate Barracuda 18 LP	18	18.8	46.6	42.5
DVD (writable)	4.7	2.77	8.9	108
Tandberg MLR1	13	2.2	4.0	682

(a) 1998

	Storage (Gbytes)	Bandwidth (Mbytes/s)	Media cost ($/GB)	Bandwidth cost ($/(Mbytes/s))
Seagate Cheetah 10K.6	146.8	59.9[a]	3.6	8.7
DVD (writable)	4.7	8.3[b]	0.3	4.8
Tandberg SLR140	70	6.0	1.1	180

(b) 2004

[a]The transfer rate varies between 43 and 78 Mbytes/s with an average of 59.9 Mbytes/s.
[b]For reading DVD-R disks, the maximum speed for most drives is 6X. For reading DVD-ROM disks, 16X drives with a maximum sustained transfer rate of 22.1 Mbytes/s are available.

Table 15.1 Comparison of storage capacity, transfer rate, and cost of storage and bandwidth for representative storage technologies in 1998 and 2004.

Magnetic Disks

As seen from the examples in Table 15.1, during the last six years, the storage capacity of magnetic disks, or hard disks, is approximately eight times higher in 2004 compared to 1998, which is consistent with Moore's Law. This is not the case for the transfer rate, which has only increased by a factor of approximately three. This development in storage capacity and transfer rate is consistent with development trends. During the last 15 years, the storage capacity of hard disks has been improved by a factor 1000, while the transfer rate has only improved 40 times in the same time period (Gray and Shenoy, 2000).

During the last six years there has been a large reduction in the cost of hard disks. The storage cost today is only about eight percent of what it was six years ago. It should be noted that the two drives used in this comparison are high-end SCSI and Fibre channel drives. If we had included IDE/ATA drives, which support even higher storage capacities at a lower cost, the reduction in storage cost would have been even larger.

DVD

The performance characteristics for DVD has developed quite differently than for hard disks. For recordable DVDs, the storage capacity has been mostly unchanged since 1998. The reason is that 4.7 Gbytes is the storage capacity defined by the DVD standard for single sided, single layer DVDs. Most available recordable DVD media are still single sided, although there exists double-sided DVD-R disks storing 9.4 Gbytes.

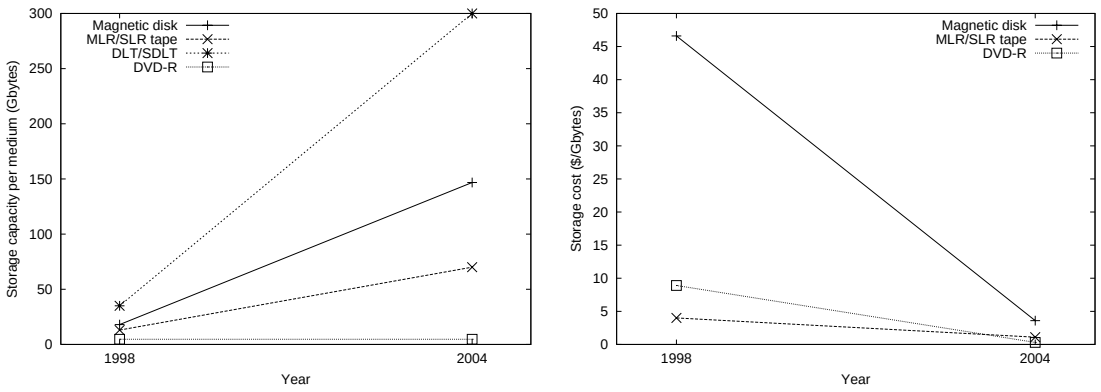
In 1998, DVD was a relatively new technology and most drives were based on existing CD-ROM technology. Thus, DVD drives had great potential for being improved. And so they have. In our study, we have used a 2X DVD drive, which was high-end in 1998. Today (2004), high-end DVD-ROM drives are 16X drives, but reading DVD-R media is usually only supported up to a speed of 6X. Although the maximum transfer rate of a DVD-ROM drive today is eight times higher, the average transfer rate has improved less. Depending on the position on the disk, the transfer rate typically varies between 8 MB/s and 21 MB/s for a 16X drive. The reason for the variation in transfer rate is that many high-end drives operates as CAV or ZCLV (Zone Constant Linear Velocity) drives instead of as CLV drives (Sadashige, 2000).

The cost of using DVD has seen even larger reductions. The cost of a DVD drive is today (2004) only one tenth of what it cost in 1998. In 1998, the first recordable 4.7 Gbytes DVD-R disks had just been made available. Today, there is a large market of recordable DVDs and the media cost is today approximately three percent compared to the cost of a DVD-R in 1998.

Since 1998 there has been a huge development in the recordable DVD technology. In 1998 only DVD-R disks were available. Since then, DVD-RW, DVD+R, DVD+RW, and DVD-RAM have become widely available. All of these are available in single sided versions storing 4.7 Gbytes, and some are available in double-sided versions storing 9.4 Gbytes. Currently, there is work on the next generation of optical disks which are expected to be able to store somewhere between 30 and 50 Gbytes of data per disk and have a transfer rate of approximately 36 Mbytes/s. Two of the main candidates to become the next generation "DVD" are HD-DVD and Blu-ray Disc, both based on using blue laser instead of red laser for reading and writing the disk (Sadashige, 2003).

Magnetic Tape

Magnetic tape has had a slower development in the performance and cost than hard disks. We have used performance data for the Tandberg MLR1 and the IBM Magstar as representative drives in our study of using magnetic tape in video storage systems. The latest drive in the Tandberg MLR/SLR series stores 70 Gbytes per tape and has a transfer rate of 6 Mbytes/s. Compared to the Tandberg MLR1 drive used in most of our experiments, this is a five times increase in storage capacity and three times in bandwidth. The Magstar MP drive series has



(a) Storage capacity

(b) Storage cost

Figure 15.1 Storage capacity and storage cost for different storage technologies in 1998 and 2004.

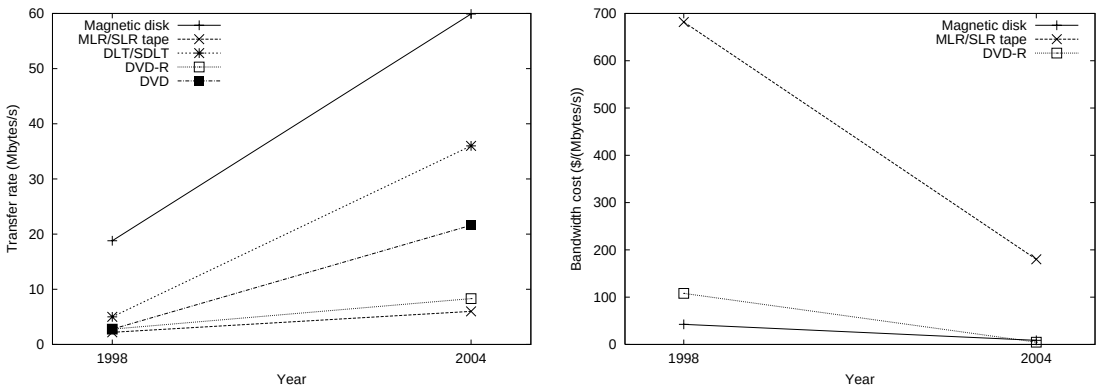
been discontinued by IBM. The storage cost of using magnetic tape is reduced by approximately 75 percent during the last six years.

If storage capacity or storage cost is the main issue, SDLT 600 (Super DLT) tapes storing 300 Gbytes, LTO Ultrium 2 tapes storing 200 Gbytes, and Sony SAIT-1 tape storing 500 Gbytes are available today. With these high storage capacities, the storage cost is reduced to approximately $ 0.3 per GB, which is comparable to the storage cost of using recordable DVD media.

Compared to hard disks, magnetic tapes use moderate areal densities and it should be possible to increase the storage capacity further to several terabytes per tape (Dee, 2002). The major vendors of magnetic tape predict the storage capacity per medium to double approximately every second year. SDLT media storing more than a terabyte and with a transfer rate of more than 100 MB/s are expected to be available in 2007-2008 (Sadashige, 2003). SAIT media storing a terabyte and having a transfer rate of 60 MB/s should become available in 2005 and four terabyte per medium and a transfer rate of 240 MB/s should become available in 2009 (Sony, 2003).

15.1.1 Comparing Storage Technologies

This far, we have presented the development of the storage technologies we have studied in this thesis. In this section, we compare the storage technologies with regards to *storage capacity*, *transfer rate*, and *latency*.



(a) Transfer rate

(b) Cost of bandwidth

Figure 15.2 Transfer rate and the cost of bandwidth for different storage technologies in 1998 and 2004.

Storage Capacity and Storage Cost

To illustrate how the storage capacity and cost have developed during the last years, Figure 15.1 presents the storage capacity and storage cost for some representative storage technologies in 1998 and 2004. For both magnetic disk and magnetic tape, the storage capacity per medium is today between five and ten times higher than in 1998. During the same period, the storage cost has been reduced by between 75 and 90 percent. Thus, for the same amount of money, you can today buy between four and ten times more storage capacity than you could six years ago. A more interesting observation, is that for the price you paid for high-end tape storage six years ago, you can today get high-end hard disk storage. The cost decline for hard disks and magnetic tape is not due to decrease in cost of the media, but in increased capacity per medium, made possible by the increase in areal density (Sadashige, 2003).

The cost ratio between hard disk and tape is decreasing. In 1998, the cost of using hard disks was more than ten times higher than using magnetic tape. Today, for the tape technologies used in this thesis, the cost ratio between a high-end hard disk and magnetic tape is reduced to only four. Thus, the cost of hard disk storage is approaching the cost of magnetic tape. Only for tape media with very high storage capacities like the SDLT 600, LTO Ultrium 2, SAIT, or low-cost tape technologies, the ratio between hard disk and tape is still more then ten. If we had included lower cost IDE/ATA drives, which today have a storage cost of less than a dollar per Gbytes, the difference between hard disk and tape would have been even smaller.

In 1998, the cost of using recordable DVD disks was higher than using tape

for storing video. Although the storage capacity of a DVD disk has not been increased, the price has been reduced greatly. The cost of using recordable DVDs is today lower than most tape technologies. Although recordable DVDs are about to outperform magnetic tape on media cost, the storage density is still higher for magnetic tape, and is likely to increase further as the storage capacity of magnetic tape increases. Compared to hard disks, using DVD-R disks costs about one tenth of using high-end hard disks.

One important observation about how the storage capacity for these storage technologies develop is that magnetic disk and magnetic tape improves in incremental steps as new improvements in technology are made, while the storage capacity of optical media, at least CD and DVD, is based on agreed standards. Thus, the storage capacity of the optical media increases in larger steps each time a new standard is agreed on. The CD, which is about 20 years old, still stores approximately the same amount of data as twenty years ago. The DVD is likely to keep the current storage capacities defined today until the next generation DVD becomes available.

Transfer Rate and Cost of Bandwidth

In Figure 15.2, the transfer rate and cost of bandwidth is presented for some representative technologies in 1998 and 2004. The main conclusion is that the transfer rate improves more slowly than the storage capacity. While the storage capacity of the hard disks and the tape technologies used in this study has improved by a factor of five to ten, the transfer rate has only improved by a factor of approximately three. The cost of the bandwidth has had the same development, with a reduction of approximately 80 percent compared to 1998. The cost of bandwidth for tape drives is approximately 20 times higher than the bandwidth of hard disk.

For recordable DVD disks, the improvement in transfer rate is approximately the same as for hard disks and tape. But the cost of the bandwidth for using DVDs has been reduced dramatically, and is today comparable with hard disks. The reason for this large reduction in price is that in 1998, DVD was a relatively new storage technology while today it is a commodity and almost every computer is shipped with a DVD drive.

Latency

We have shown that the transfer rate is increasing at a lower rate than the storage capacity. The latency or response time is improving even more slowly. For hard disks, the main reason for improved response time is the higher number of rotations of the disks. In 1998, high-end disks rotated with 7200 or 10.000 revolutions per minute. Today, the high-end disks rotate with 10.000 or 15.000 revolutions per minute. If we compare the average access time for the two disks used as example in Table 15.1, the average access time has been reduced from 6.9 ms to 4.7 ms, or by approximately 30 percent.

The response time of DVD drives has been improved by faster drives spinning with a higher speed. By using alternative technologies like operating the drive as a CAV or ZCLV drive, accessing data on the disk has become faster than using standard CLV drive technology. Today's DVD drives have an average access time of about 100 ms.

For magnetic tape, the reduction in latency has been even lower. Most manufacturers have focused on storage capacity and bandwidth, and for many drive technologies the average access time is even higher today than six years ago. For instance, for the tape drives included in Table 15.1, the average access time has increased from 45 seconds for the MLR1 drive to 99 seconds for the SLR140 drive. The Magstar MP series, which focused on providing a low response time for applications needing random access to the tape, has been discontinued by IBM. There are still no drives available utilizing the LTO Accelis technology (Linear Tape-Open Technology, 1998).

Other Factors

In addition to the performance related factors mentioned above, there are also other technical factors that differentiate the storage technologies.

- **Power consumption.** Making the hard disks faster by increasing the number of revolutions per minute, also increases energy consumption. In systems with a high number of hard disks, the amount of produced heat might be come a problem. Thus, using many hard disks can become costly both in power consumption and for getting rid of the produced heat.

- **Media durability and reliability.** Although the reliability of high-end hard disks is very high, it still can not compete with the durability and reliability of magnetic tapes and optical disks. Given correct storage conditions, magnetic tape is estimated to have a life time of about 50 to 100 years (Judge, Schmidt, Weiss and Miller, 2003), while recordable DVDs have an expected storage life of at least 60 years (Sadashige, 2000).

15.2 Consequences for our Work

As shown in the previous section, the performance of the storage technologies used in this study has improved greatly during the last six years. The performance improvement has followed the trends from the last twenty years. The storage capacity has been improved mostly as predicted by Moore's Law. The transfer rate has had a smaller improvement, but still all the technologies today have a transfer rate which is about three times higher than in 1998. The latency has improved the least.

During this period, there has not been any paradigm shifts in the storage technology. The largest change has been for the DVD, which has gone from being a new storage technology for digital video, to become a widely used storage media for data in general. In this section, we discuss how the improvements in the storage technology during the last six years influence the results presented in this thesis.

15.2.1 Optimizing Access Times for Serpentine Tape

In Chapter 6, 7, and 8, we studied how to optimize the use of serpentine tape for retrieval of multiple data objects from a single tape. We showed that the average access time and the effective bandwidth of a tape drive could be considerably improved by use of scheduling. The model and scheduling algorithm were developed and evaluated using Tandberg MLR1 and Quantum DLT 2000 drives. As shown in this chapter, both the storage capacity and the transfer rate of today's serpentine tape drives are much higher than for the tape drives we have used. This improvement in performance does not make an access time model and scheduling algorithms less important. On the contrary, there are several technical factors that make it more important today to be able to efficiently schedule concurrent requests for data stored on serpentine tape.

- **Higher storage capacities.** With the increase in storage capacity, each tape will be able to store a higher number of data objects like X-ray images or video sequences. A higher number of data objects per tape, increases the likelihood of having more concurrent requests to the same tape. As we have shown, the more objects that should be retrieved, the lower the average access time per object becomes. Thus, higher storage capacities might improve the effect of using a scheduler.

- **Higher seek times.** The higher storage capacity of the tapes has mainly been achieved by the following three factors: more parallel tracks on each tape, higher linear storage density, and longer tapes. The longer tapes have increased the average access time for retrieving data on a tape. For instance, the length of the MLR1 tapes used in our study is 366 meters (Tandberg Data, 1996), while the length of an SLR140 tape, which is the newest member of the MLR/SLR tape family, is 506 meters (Tandberg Data, 2003). As a consequence, the average access time is increased from 45 seconds to 99 seconds. Thus, today, each access for an object stored on tape may be even more costly than six years ago. Thus, the savings by use of a scheduler may be even higher today.

- **Higher transfer rates.** The higher transfer rates of today's tape drives reduce the transfer time of a data object stored on tape. Thus, the seek time will constitute an even higher fraction of the total time for retrieving data

stored on tape. In order to utilize the higher transfer rate and improve the effective transfer rate, it is even more important to reduce the amount of time used for seeking.

These three factors are due to the technical development in serpentine tape technology. There are also other factors not directly related to the improvement in tape technology that determine the usefulness of being able to model access times and schedule concurrent accesses to a serpentine tape:

- **The importance of tape as a storage medium decreases.** Due to competition from cheaper hard disks and optical storage media like recordable DVDs, tape is slowly losing its status as the storage medium with the lowest cost and highest storage density. This makes magnetic tape become a less likely choice for applications that need to store huge amounts of data.
- **Online access to *all* data.** With the steady improvements in technology, more organizations want to have online access to data stored within the organization. This might be data that already is stored on magnetic tape, or data with such high storage requirements and low access probabilities that magnetic tape becomes the most economical choice. Examples of such organizations might be hospitals that want online access to e.g., X-ray images and MR-scans, oil companies that want access to seismic data, or television producers that have large film and video archives. Providing online access to these kinds of large data collections may require optimal use of the transfer rate of the tape drives, and thus benefit from being able to schedule concurrent accesses. To be able to optimize the use of hierarchical storage systems requires a detailed model for the access time for data objects stored on tertiary media (Van Meter and Gao, 2000; Lijding, 2003).

The importance of being able to model access times and schedule concurrent requests for data stored on serpentine tape depends on the future applications that might use tape for storing large amount of data and the access pattern these applications have to the stored data.

15.2.2 Comparing Tertiary Storage Technologies

In Chapter 10 and 11, we studied the properties and the performance of two tape technologies and one optical disk technology for use in library units used for storing and retrieving digital video. The main conclusions are still valid, but due to the large reduction in cost of using recordable DVD compared to tape, the trends in favor of recordable DVD have become even stronger during the last years:

- **Improved transfer rate.** The transfer rate of recordable DVD and magnetic tape has increased by approximately the same rate during the last six years.

As shown in Section 11.2, due to the much shorter seek time of a DVD disk compared to tape, the effective bandwidth using DVD has improved much more than for tape drives for retrieval of short video sequences. Also for retrieval of longer video sequences will the improvement in transfer rate favor DVD more than tape.

- **Seek time.** For DVD, the seek time has been improved, but as shown in Section 11.3, this has little effect on the performance. For some tape technologies, the seek time has increased due to longer tapes. This has a negative effect on the both response time and throughput particularly for retrieval of short video sequences.

- **Cost.** As shown in this thesis, there are several factors which influence the cost of storing and retrieving digital video. The large reduction in cost of the DVD-R medium and DVD drives have reduced the cost of a library unit. The large improvement in storage capacity for tape media increases the amount video that can be stored in a library unit. Both of these factors reduce the storage cost per GB of video. The increased transfer rate of the drives reduces the cost per stream, but due to the large reduction in cost of DVD drives, the reduction in cost per stream will be higher for DVD compared to tape.

In Section 10.4, we showed that if throughput, response time, and cost per stream were the most important criteria for selecting a storage technology, then DVD would provide the lowest cost per stream. If the main factor for deciding on a storage technology was the storage cost, tape-based systems would provide the lowest cost per stored video sequence. With the large reduction in both media cost and drive cost for DVD, this has changed more in favor of DVD. There will still be systems where it is cheaper to use magnetic tape, mainly due to the much higher storage capacity per medium.

15.2.3 Access Distributions and Allocation Strategies

In Chapter 12, we studied the effect non-uniform access distributions have on the performance of a tertiary storage system. We showed that for retrieval of short video sequences, the access distribution had very little effect on the performance, while for retrieval of long video sequences, skewed access distribution could make some of the storage media *hotter* than other media. The improved transfer rate of today's media drives will reduce the effect of media that are too popular since the drive will be able to serve many more requests. On the other hand, the increased storage capacity of tapes increases the likelihood of having multiple hot video sequences on a tape.

In Chapter 13, we studied different strategies for allocating video sequences to tertiary storage media and library units. The allocation strategies determine

where on a storage medium a given video should be placed and in which library unit the medium should be stored. The placement of the video sequence on the medium determines the seek time for retrieving a given video sequence. It does not influence on the transfer time. Thus, the best allocation strategy is the one that minimizes the seek time. The technology improvement has improved the transfer time, while the seek time has not been improved. Thus, in order to utilize the improved transfer rate, the importance of using a good allocation strategy is even more important today than six years ago. The improved transfer rate increases the difference in performance of the allocation strategies, but the relative ordering of the allocation strategies should be the same as shown in Chapter 13.

15.2.4 Caching Videos on Hard Disks

The effect of caching the most frequently used videos in a disk cache was studied in Chapter 14. The main technology changes during the last six years that influence the results presented in this chapter are the increased transfer rate of the storage devices and the reduced cost ratio between hard disks and magnetic tape. In the first part, we studied how the size of the disk cache and the access distribution affects the performance of the video archive. With the exception of even higher performance numbers and lower cost, the main results should still be valid. In the second part, we studied the relationship between the size of the disk cache and the number of tertiary drives. The results in this section are influenced by the following factors:

- **Transfer rate increases.** All storage technologies have had approximately the same increase in the transfer rate during the last six year. As shown in Section 14.3, a higher *tertiary transfer rate* can be traded with a lower amount of *secondary storage*. Since the transfer rate of the tertiary storage devices have been increased, the number of tertiary drives relative to the size of the disk cache can be reduced.

- **Reduced cost ratio between hard disk and magnetic tape.** The larger reduction in cost of hard disks compared to magnetic tape has made it less costly to increase the size of the disk cache. As a consequence, the number of tape drives can be reduced. This is not the case for DVD, since the cost of DVD drives have been reduced even more than the cost of hard disk drives.

Thus, with today's tertiary drives, the experiments presented in Section 14.3 would have concluded with a slightly larger disk cache and a lower number of tertiary drives as the least costly archive configuration. The evaluation of the allocation strategies would be largely uninfluenced by the development in the storage technologies.

The increased storage capacity per hard disk and per tape also affects the video archive. For the tertiary storage system, the higher storage capacity per

tape makes it possible to store more video data in each library unit. Thus, the number of library units can be reduced. As a consequence, the number of robot mechanisms will be reduced, while at the same time the number of drives in each library unit might be increased. For video archives that mainly retrieves short video sequences, we have shown that the robot mechanism might become the bottleneck.

The large reduction in cost of hard disks may reduce the cost of the disk cache. Unfortunately, this requires us to use larger disk drives in the cache. As shown earlier in this chapter, the storage capacity increases much faster than the transfer rate. A consequence of this is that using the most cost-optimal disk size with regards to storage cost will reduce the transfer rate of a disk cache. In order to achieve the required bandwidth of the disk cache, it might be necessary to increase the number of disks, by for instance using smaller disks in the disk cache, or use disk drives with even higher transfer rates.

15.3 Conclusions

Based on the above, the technology improvement during the last six years has the following consequences on the results presented in this thesis:

1. Bandwidth becomes more important. The storage capacity is increasing more rapidly than the transfer rate of the storage devices. Thus, it will become even more important to utilize the bandwidth efficiently.

2. The relative cost of secondary storage (hard disk) versus tertiary storage is decreasing. The media cost of hard disk might become equal to the media cost for tape. Thus, disks will replace tape in many applications. Still, other factors like durability of more than 50 years, cost of connecting the disks (cables and cabinets) to the computer system, and power consumption will continue to be in favor of tertiary storage.

3. DVD outperforms magnetic tape on price. When starting this study, recordable DVD disks were more costly. This has changed during the last six years. This has made recordable DVD even more favorable compared to tape as a tertiary storage medium for video. The main problem for DVD compared to tape is the lower storage capacity per medium.

With these large improvement in the storage technology, it could be assumed that the challenges for storing digital video should be reduced correspondingly. This is probably not the case. Processor and network technologies have improved similarly, which has made it possible to process and deliver digital video with even higher quality. Thus, the performance improvements in the technology are likely outweighed by a steadily increasing demand for storage and delivery of

more video with a higher quality. Based on this, the challenges that storage systems for digital video faced in 1998 are still challenging for today's storage systems. The reasearch issues studied in this thesis are still relevant in the context of digital video archives.

Chapter 16

Conclusions and Further Work

To be sure of hitting the target, shoot first, and, whatever you hit, call it the target.

Ashleigh Brilliant – UC Berkeley 'street' philosopher

The objective of this thesis has been to achieve a better understanding of the challenges of storing large amounts of digital video and to evaluate different storage technologies for use in digital video archive systems. In this last chapter, we conclude this work by stating the main conclusions and the main contributions. Finally, we outline some directions for further research.

16.1 Main Conclusions

The research presented in this thesis is divided into two main parts. The first part is a study of the use of magnetic tape as a storage media for digital video, while the second part is an evaluation of tertiary storage technologies and storage architectures for use in digital video archives.

The focus of the first part was to optimize the utilization of the transfer rate for serpentine tape. The strategy for achieving this was to develop a detailed model of the access times for data stored on the tape and to develop scheduling algorithms for concurrent requests to the tape. The main conclusion is that it is possible to establish an accurate access time model for serpentine tape drives that is more generic and has a lower cost with regards to instrumentation than existing access time models for serpentine tape. Further, by using this access time model and the proposed scheduling algorithm, it is possible to achieve significant improvements in initial latency, average access time, and the number of requests that can be served by a single tape drive in cases where there are multiple video sequences to be retrieved from the same tape.

In the second part, we evaluated three tertiary storage technologies for use in a large video archive and several strategies for how to organize the data within an archive. The main conclusion from this part of the study is that there are no

single storage technology or storage architecture that is the optimal solution for use in a video archive. The choice of storage technology and storage organization depends on the size of the requested video sequences, the load on the archive, and the access probability for the videos. It also depends on whether the main goal is to maximize throughput, minimize response time, or minimize the cost of the archive. In order to implement a cost-efficient digital video archive, it is necessary to have detailed knowledge about the available storage technologies as well as the anticipated use of the archive.

When it comes to comparing the three tertiary storage technologies, the main conclusion is that the DVD technology outperforms the tape technologies in most of the investigated cases. The main reason for DVD performing better than tape technologies is the much lower seek times. The difference in seek time is due to the random access supported by DVD drives versus the sequential access supported by tape based storage technologies. Only in systems where the response time is not an important criterion and the requests are for long video sequences, the tape based systems are able to perform comparable with a DVD-based system. At the time this study was performed (1998-2003), tape still provided the cheapest storage.

The study also shows that when allocating videos to storage media, it is necessary to have knowledge about access probabilities in order to optimize the performance of the tertiary storage devices. In a video archive where the access distribution is sufficiently skewed, the best method to improve the performance is to include a disk-based cache.

16.2 Contributions

16.2.1 Using Serpentine Tape for Storing Digital Video

The research performed on optimizing the use of *serpentine* tape as a storage medium for digital video has focused on reducing the access time for video sequences stored on tape. The main contributions from this work are:

- A novel *access-time model* for serpentine tape. Compared to other access-time models for serpentine tape, this model is more generic and has a lower cost for instrumentation of the individual tapes. The model has been evaluated using two different serpentine tape drives. (Chapter 6)

- A novel scheduling algorithm for optimizing the utilization of serpentine tape. This algorithm is evaluated using simulations and measurements on two different serpentine tape drives. (Chapter 7 and Appendix C)

- An evaluation of using serpentine tape for storage of video sequences and images. (Chapter 8).

These results have also been presented in the following papers: "Low-cost access time model for a serpentine tape drive" (Sandstå and Midtstraum, 1999b), "Improving the access time performance of serpentine tape drives" (Sandstå and Midtstraum, 1999a), and "Analysis of retrieval of multimedia data stored on magnetic tape" (Sandstå and Midtstraum, 1998).

16.2.2 Large Scale Storage of Digital Video

A large video archive must support both storage and retrieval of the videos. The last part of the thesis evaluated use of tertiary storage as the main storage in a digital video archive. The main contributions from this part of the research are:

- An architecture and simulator for a video archive. This simulator can be used for evaluating the properties and the performance and cost of using different tertiary storage technologies in a video archive. It is also suitable for evaluating improvements of the storage technologies. In Chapter 11, it is shown that the simulator can be used for evaluating the effect improvements in tertiary storage devices has on the performance of the storage system. (Chapter 9)

- An evaluation of three tertiary storage technologies. The storage technologies are evaluated for use in a system storing and delivering digital video. The main criteria used for evaluating the storage technologies have been response time, throughput, and the cost of using the different storage technologies. (Chapter 10)

- An evaluation of how technological improvements of the storage devices will effect the overall performance of a video archive system. (Chapter 11)

- An evaluation of how the access distribution for the stored videos influences the performance. The performance of the tertiary storage system is evaluated for three access distributions. (Chapter 12)

- Several allocation strategies of videos to storage media are presented and evaluated. (Chapter 13 and 14.4)

- An evaluation of the effect of using a disk-based cache for improving the performance of the video archive. In addition to throughput and response time, an evaluation of the cost of using disk for caching videos is performed. The trade-off between the size of the cache and the number of tertiary drives has been investigated. (Chapter 14)

Although most of the work presented in this thesis has been related to storing and retrieving digital video, the results are also valid outside this context, and should be useful for systems that utilize tertiary storage for storing data.

16.2.3 Related Work

During this doctoral study, we have also done other related work, which is not included in this thesis. This includes the following work:

- **VideoSTAR.** The results from the work on the VideoSTAR project has been published in the following papers: "Searching and browsing a shared video database" (Hjelsvold et al., 1995a)[1], "A temporal foundation of video databases" (Hjelsvold et al., 1995b), and "Integrated video archive tools" (Hjelsvold et al., 1995).

- **Elvira video servers.** A performance evaluation of the first Elvira video server has been performed. This evaluation is presented in the paper "Video server on an ATM connected cluster of workstations" (Sandstå et al., 1997).

  The Elvira II video archive server has been designed and implemented. The main feature that distinguishes this server from most other video servers is the integrated support for tertiary storage making it suitable for use in large video archives. An overview of the architecture of Elvira II is included in Appendix A.

16.3 Criticism

The research presented in this thesis is based on performing evaluations of several strategies for optimizing the use of different storage technologies. The experiments performed range from modeling a specific serpentine tape drive to performance evaluations of storage systems for digital video archives. In this section, we present some criticism of some of the research presented in this thesis:

- **Workload.** The investigated systems have not been evaluated with real-world workloads or been used in real world applications. In both the research on optimizing the performance of serpentine tape and on technologies for large scale video archives, we have only used synthetic workloads. To compensate for this, we have used several different workloads during the investigations, but still a more realistic workload would have been valuable in order to validate the results.

- **Choice of technology.** In this study, we have used specific models of storage devices for representing the different storage technologies. The danger of using specifications of current storage devices, is that the results might be influenced by a particular behavior or performance of one of these storage devices. Another drawback of using specifications of current storage

[1] A version of this paper has also been published as a chapter in the book "Multimedia Database Systems: Design and Implementation Strategies" (Nwosu, Thuraisingham and Berra, 1996)

devices is that the results might quickly become outdated due to the fast development of new and improved technology.

- **Validation of simulation results.** The study of storage systems for digital video archives have been performed using a simulator. Some of the results have been evaluated by comparing them with known analytical models or by careful inspection of the simulation results. To improve the confidence of the simulation results, more of the results could have been validated by use of analytical or statistical methods.

- **Evaluation criteria.** Another issue with the simulation results is the choice of performance metrics when studying the performance of the video archive. In most of the experiments, we have used the *average* response time as the main criterion for usability. In many of the experiments, the response time is not uniformly distributed. This is particularly the case when introducing a disk-based cache. The majority of the requests are fulfilled with a very short delay, while the requests that are served by the tertiary storage system experience a response time that is much higher than the average response time. In some of these experiments, the average response time presented as the main result could have been accompanied by the variance of the response time.

16.4 Further Work

Although we are about to finish the last chapter of this thesis, there are still plenty of research that can be done within the topic of supporting storage and delivery in large video archives. In this section we give some suggestions for continued work related to this thesis:

- The experiments done for evaluating the performance of serpentine tape and video archives could be conducted based on real-world workloads. This would validate the results given in this thesis and possibly produce new and interesting results.

- The results from the performance study of video archives could be compared against the performance of a real video archive. This could either be done by building a working prototype based on the architecture presented in this thesis or by trying to compare the performance of an existing video archive to the results we have presented in this thesis.

- A study could be performed to determine the requirements organizations with large amounts of analog or digital video have for a digital video archive. This should also determine how the archive would be used, the access pattern, and the required performance.

- The implemented prototype for the Elvira II video archive server could have been used more for conducting performance and usability studies with larger amounts of digital video.

References

Aggarwal, C. C., Wolf, J. L. and Yu, P. S. (1996a). On optimal batching policies for video-on-demand storage servers, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, pp. 253–258.

Aggarwal, C. C., Wolf, J. L. and Yu, P. S. (1996b). A permutation-based pyramid broadcast scheme for video-on-demand systems, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, pp. 118–126.

Allen, A. D. (2001). Optimal delivery of multi-media content over networks, *Proceedings of ACM Multimedia '01*, Ottawa, Canada, pp. 79–88.

ATSC (2001a). ATSC standard A/52A, Digital Audio Compression (AC-3), Advanced Television Systems Committee.

ATSC (2001b). ATSC standard A/53B, Digital Television Standard, revision B, Advanced Television Systems Committee.

Balabanian, V., Casey, L., Greene, N. and Adams, C. (1996). An introduction to Digital Storage Media — Command and Control, *IEEE Communications Magazine* **34**(11): 122–127.

Bell, A. E. (1999). The dynamic digital disk, *IEEE Spectrum* **36**(10): 28–35.

Berson, S., Ghandeharizadeh, S., Muntz, R. and Ju, X. (1994). Staggered striping in multimedia information systems, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Minneapolis, Minnesota, pp. 79–90.

Berson, S., Golubchik, L. and Muntz, R. R. (1995). Fault tolerant design of multimedia servers, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, California, pp. 364–375.

Bhatt, B., Birks, D. and Hermreck, D. (1997). Digital television: making it work, *IEEE Spectrum* **34**(10): 19–28.

Bhide, A. K., Dan, A. and Dias, D. M. (1993). A simple analysis of the LRU buffer policy and its relationship to buffer warm-up transient, *Proceedings of the 9th International Conference on Data Engineering*, IEEE Computer Society, Vienna, Austria, pp. 125–133.

Birk, Y. (1995). Track-pairing: a novel data layout for VOD servers with multi-zone-recording disks, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Washington D.C., pp. 248–255.

Birk, Y. (1997). Random RAIDs with selective exploitation of redundancy for high performance video servers, *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV'97*, St. Louis, Missouri, pp. 13–23.

Bitton, D. and Gray, J. (1988). Disk shadowing, *Proceedings of 14th International Conference on Very Large Data Bases*, Los Angeles, California, pp. 331–338.

Boll, S., Heinlein, C., Klas, W. and Wandel, J. (2000). MPEG–L/MRP: Adaptive streaming of MPEG videos for interactive internet applications, *Proceedings of 6th International Workshop on Multimedia Information Systems*, Chicago, Illinois, pp. 104–113.

Bolosky, W. J., Barrera III, J. S., Draves, R. P., Fitzgerald, R. P., Gibson, G. A., Jones, M. B., Levi, S. P., Myhrvold, N. P. and Rashid, R. F. (1996). The Tiger video fileserver, *Proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV'96*, Zushi, Japan, pp. 97–104.

Braden, R., Zhang, L., Berson, S., Herzog, S. and Jamin, S. (1997). *Resource ReSerVation Protocol (RSVP)*, RFC 2205, IETF.

Bradshaw, M. K., Wang, B., Sen, S., Gao, L., Kurose, J., Shenoy, P. and Towsley, D. (2001). Periodic broadcast and patching services — implementation, measurement, and analysis in an internet streaming video testbed, *Proceedings of ACM Multimedia '01*, Ottawa, Canada, pp. 280–290.

Brataas, G., Crespo, S. P., Hughes, P. H. and Miethe, K. (1998a). Evaluation of high performance databases for interactive services, Deliverable 5, Documentation of performance test results, *Technical Report P617*, Eurescom.

Brataas, G., Klovning, E., Sandstå, O. and Torbjørnsen, Ø. (1998b). Evaluation of high performance databases for interactive services, Deliverable 5 Annex, Detailed measurements of the cluster system, *Technical Report P617*, Eurescom.

Bratbergsengen, K. (1997). *Filsystemer. Lagring og behandling av store datamengder*, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. In Norwegian.

Brubeck, D. W. and Rowe, L. A. (1996). Hierarchical storage management in a distributed VOD system, *IEEE Multimedia* **3**(3): 37–47.

Bryhni, H., Lovett, H., Maartmann-Moe, E., Solvoll, D. and Sørensen, T. (1996). On-demand regional television over the Internet, *Proceedings of ACM Multimedia '96*, Boston, Massachusetts, pp. 99–107.

Carey, M. J., Haas, L. M. and Livny, M. (1993). Tapes hold data, too: Challenges of tuples on tertiary store, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC, pp. 413–417.

Cha, H., Lee, J. and Oh, J. (2002). Constructing a video server with tertiary storage: Practice and experience, *Multimedia Systems* **8**(5): 380–394.

Cha, H., Lee, J., Oh, J. and Ha, R. (2001). Video server with tertiary storage, *Proceedings of the 18th IEEE Symposium on Mass Storage Systems/9th NASA Goddard Conference on Mass Storage Systems and Technologies*, San Diego, California, pp. 303–312.

Chan, S.-H. G. and Tobagi, F. (2001). Distributed servers architecture for networked video services, *IEEE/ACM Transactions on Networking* **9**(2): 125–136.

Chan, S.-H. G. and Tobagi, F. (2003). Modeling and dimensioning hierarchical storage systems for low-delay video services, *IEEE Transactions on Computers* **52**(7): 907–919.

Chan, S.-H. G. and Tobagi, F. A. (1999). Designing hierarchical storage systems for interactive on-demand video services, *Proceedings of IEEE Multimedia Applications, Services, and Technologies*, Vancouver, Canada.

Chang, E. and Garcia-Molina, H. (1997a). Effective memory use in a media server, *Proceedings of 23rd International Conference on Very Large Data Bases*, Athens, Greece, pp. 496–505.

Chang, E. and Garcia-Molina, H. (1997b). Reducing initial latency in media servers, *IEEE Multimedia* **4**(3): 50–61.

Chen, M.-S. and Kandlur, D. D. (1996). Stream conversion to support interactive video playout, *IEEE Multimedia* pp. 51–58.

Chen, P. M., Lee, E. K., Gibson, G. G., Katz, R. H. and Patterson, D. A. (1994). RAID: High-performance, reliable secondary storage, *ACM Computing Surveys* **26**(2): 145–185.

Chervenak, A. L. (1994). *Tertiary Storage: An Evaluation of New Applications*, PhD thesis, University of California at Berkeley.

Chervenak, A. L. (1998). Challenges for tertiary storage in multimedia servers, *Parallel Computing* **24**(1): 157–176.

Chervenak, A. L., Patterson, D. A. and Katz, R. H. (1995). Choosing the best storage system for video service, *Proceedings of ACM Multimedia '95*, San Francisco, California, pp. 109–119.

Chou, H.-T. and DeWitt, D. J. (1985). An evaluation of buffer management systems, *Proceedings of 11th International Conference on Very Large Data Bases*, Stockholm, Sweden, pp. 127–141.

Christodoulakis, S. and Ford, D. A. (1988). Performance analysis and fundamental performance trade offs for CLV optical disks, *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, Chicago, Illinois, pp. 286–294.

Christodoulakis, S., Triantafillou, P. and Zioga, F. A. (1997). Principles of optimally placing data in tertiary storage libraries, *Proceedings of the 23rd VLDB Conference*, Athens, Greece, pp. 236–245.

Cruyssen, P. V. D. and Rijckaert, M. J. (1978). Heuristic for the asymmetric travelling salesman problem, *The Journal of the Operational Research Society* **29**(7): 697–701.

Dan, A. and Sitaram, D. (1995). An online video placement policy based on bandwidth to space ratio (BSR), *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, California, pp. 376–385.

Dan, A. and Sitaram, D. (1997). Multimedia caching strategies for heterogeneous application and server environments, *Multimedia Tools and Applications* **4**(3): 279–312.

Dan, A., Dias, D. M., Mukherjee, R., Sitaram, D. and Tewari, R. (1995a). Buffering and caching in large-scale video servers, *Proceedings of COMPCON '95: Technologies for the Information Superhighway*, San Francisco, California, pp. 217–224.

Dan, A., Kienzle, M. and Sitaram, D. (1995b). A dynamic policy of segment replication for load-balancing in video-on-demand servers, *Multimedia Systems* **3**(3): 93–103.

Dan, A., Shahabuddin, P., Sitaram, D. and Towsley, D. (1995c). Channel allocation under batching and VCR control in video-on-demand systems, *Journal of Parallel and Distributed Computing* **30**(2): 168–179.

Dan, A., Sitaram, D. and Shahabuddin, P. (1994). Scheduling policies for an on-demand video server with batching, *Proceedings of the ACM Multimedia 94*, San Francisco, California, pp. 15–23.

Dan, A., Sitaram, D. and Shahabuddin, P. (1996). Dynamic batching policies for an on-demand video server, *Multimedia Systems* **4**(3): 112–121.

Dashti, A. E. and Shahabi, C. (2000). Data placement techniques for serpentine tapes, *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Hawaii, pp. 3208–3217.

Dee, R. H. (2002). The challenges of magnetic recording on tape for data storage (the one terabyte cartridge and beyond), *Proceedings of the 10th Goddard Conference on Mass Storage Systems and Technologies/19th IEEE Symposium on Mass Storage Systems*, College Park, Maryland, pp. 109–119.

Digital (1992). *Digital DLT2000/DLT2700 Cartridge Tape Subsystem Product Manual*, Digital Equipment Corporation.

Ding, J.-W. and Huang, Y.-M. (2003). Resource-based striping: An efficient striping strategy for video servers using heterogeneous disk-subsystems, *Multimedia Tools and Applications* **19**(1): 29–51.

Douceur, J. R. and Bolosky, W. J. (1999). Improving responsiveness of a stripe-scheduled media server, *Multimedia Computing and Networking 1999*, Vol. 3654 of *Proceedings of SPIE*, San Jose, California, pp. 192–203.

Doğanata, Y. N. and Tantawi, A. N. (1994). Making a cost-effective video server, *IEEE Multimedia* **1**(4): 22–30.

Doğanata, Y. N. and Tantawi, A. N. (1996). Storage hierarchy in multimedia servers, *in* S. M. Chung (ed.), *Multimedia Information Storage and Management*, Kluwer Academic Publishers, chapter 3, pp. 61–94.

Drapeau, A. L. and Katz, R. H. (1993a). Striped tape arrays, *Proceedings of the 12th IEEE Symposium on Mass Storage Systems*, Monterey, California, USA, pp. 257–265.

Drapeau, A. L. and Katz, R. H. (1993b). Striping in large tape libraries, *Proceedings of the conference on Supercomputing '93*, Portland, Oregon, USA, pp. 378–387.

Dybvik, G. A. (1997). *Editing of MPEG-1 system streams*, Master's thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. In Norwegian.

Effelsberg, W. and Haerder, T. (1984). Principles of database buffer management, *ACM Transactions on Database Systems* **9**(4): 560–595.

ETSI (2000). Digital video broadcasting (DVB); implementation guidelines for the use of MPEG-2 systems, video and audio in satellite, cable and terrestrial broadcasting applications, *Technical Report ETSI TR 101 154 V.1.4.1*, ETSI, Sophia Antipolis Cedex, France.

Federighi, C. and Rowe, L. A. (1994). A distributed hierarchical storage manager for a video-on-demand system, *Proceedings of Storage and Retrieval for Image and Video Databases II, IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, San Jose, California, pp. 185–197.

Ferrari, D. (1990). Client requirements for real-time communication services, *IEEE Communications Magazine* **28**(11): 65–72.

Ford, D. A. and Myllymaki, J. (1996). A log-structured organization for tertiary storage, *Proceedings of the 12th International Conference on Data Engineering*, New Orleans, Louisiana, pp. 20–27.

Ford, D. A., Morris, R. J. T. and Bell, A. E. (1996). Redundant arrays of independent libraries (RAIL): A tertiary storage system, *Proceedings of COMPCON '96: Technologies for the Information Superhighway*, Santa Clara, California, pp. 280–285.

Fox, E. A. (1991). Advances in interactive digital multimedia systems, *IEEE Computer* **24**(10): 9–21.

Fransman, M. (1996). Information regarding the information superhighway and interpretive ambiguity, *IEEE Communications* **34**(7): 76–80.

Furht, B. (1994). Multimedia systems: An overview, *IEEE Multimedia* **1**(1): 47–59.

Furht, B., Westwater, R. and Ice, J. (1998). Multimedia broadcasting over the internet: Part I, *IEEE Multimedia* **5**(4): 78–82.

Gallefoss, E. (1997). *Elvira management system*, Master's thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. In Norwegian.

Garcia-Martinez, A., Fernandez-Conde, J. and Vina, A. (2000). Efficient memory management in video on demand servers, *Computer Communications Journal* **23**(3): 253–266.

Gelman, A. D., Kobrinski, H., Smoot, L. S. and Weinstein, S. B. (1991). A store-and-forward architecture for video-on-demand service, *Proceedings of the 1991 IEEE International Conference on Communications*, Vol. 2, Denver, Colorado, pp. 842–846.

Gemmell, D. J. (1996). Disk scheduling for continuous media, *in* S. M. Chung (ed.), *Multimedia Information Storage and Management*, Kluwer Academic Publishers, chapter 1, pp. 1–21.

Gemmell, D. J., Vin, H. M., Kandlur, D. D., Rangan, P. V. and Rowe, L. A. (1995). Multimedia storage servers: A tutorial, *IEEE Computer* **28**(5): 40–49.

Georgiadis, C., Triantafillou, P. and Faloutsos, C. (2002). Fundamentals of scheduling and performance of video tape libraries, *Multimedia Tools and Applications* **18**(2): 137–158.

Ghandeharizadeh, S. and Shahabi, C. (1994). On multimedia repositories, personal computers, and hierarchical storage systems, *Proceedings of the ACM Multimedia 94*, San Francisco, California, pp. 407–416.

Ghandeharizadeh, S., Dashti, A. and Shahabi, C. (1995). A pipelining mechanism to minimize the latency time in hierarchical multimedia storage managers, *Computer Communications* **18**(3): 170–184.

Ghandeharizadeh, S., Kim, S. H., Shahabi, C. and Zimmermann, R. (1996). Placement of continuous media in multi-zone disks, *in* S. M. Chung (ed.), *Multimedia Information Storage and Management*, Kluwer Academic Publishers, chapter 2, pp. 23–59.

Golubchik, L. and Rajendran, R. K. (1998). A study of the use of tertiary storage in multimedia systems, *Proceedings of the Sixth NASA Goddard Conference on Mass Storage Systems and Technologies/Fifteenth IEEE Symposium on Mass Storage Systems*, College Park, Maryland, pp. 229–247.

Golubchik, L., Lui, J. C. S. and Muntz, R. R. (1996). Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers, *Multimedia Systems* **4**(3): 140–155.

Golubchik, L., Muntz, R. R. and Watson, R. W. (1995). Analysis of striping techniques in robotic storage libraries, *Proceedings of the 14th IEEE Symposium on Mass Storage Systems*, Monterey, California, pp. 225–238.

Gonzalez, R. C. and Woods, R. E. (1992). *Digital Image Processing*, Addison-Wesley.

Gray, J. and Shenoy, P. (2000). Rules of thumb in data engineering, *Proceedings of the 16th International Conference on Data Engineering*, San Diego, California, pp. 3–12.

Gray, J., Horst, B. and Walker, M. (1990). Parity striping of disc arrays: Low-cost reliable storage with acceptable throughput, *Proceedings of 16th International Conference on Very Large Data Bases*, Brisbane, Australia, pp. 148–161.

Griwodz, C., Bär, M. and Wolf, L. C. (1997). Long-term movie popularity models in video-on-demand systems or The life of an on-demand movie, *Proceedings of the ACM Multimedia 97*, Seattle, Washington, pp. 349–357.

Griwodz, C., Zink, M., Liepert, M., On, G. and Steinmetz, R. (2000). Multicast for savings in cache-based video distribution, *Multimedia Communication and Networks*, Vol. 3969 of *Proceedings of SPIE*, San Jose, California, pp. 24–35.

Halvorsen, P., Goebel, V. and Plagemann, T. (1998). Q-L/MRP: A buffer management mechanism for QoS support in a multimedia DBMS, *Proceedings of the IEEE International Workshop on Multi-Media Database Management Systems*, Dayton, Ohio, pp. 162–171.

Halvorsen, P., Griwodz, C., Lund, K., Goebel, V. and Plagemann, T. (2003a). Storage systems support for multimedia applications, *Research Report No. 307*, Department of Informatics, University of Oslo, Norway.

Halvorsen, P., Plagemann, T. and Goebel, V. (2003b). Improving the I/O performance of intermediate multimedia storage nodes, *Multimedia Systems* **9**(1): 56–67.

Haskell, B. G., Puri, A. and Netravali, A. N. (1997). *Digital Video: An Introduction to MPEG-2*, Digital Multimedia Standards Series, Chapman & Hall.

Hillyer, B. K. and Silberschatz, A. (1996a). On the modeling and performance characteristics of a serpentine tape drive, *Proceedings of the 1996 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Philadelphia, Pennsylvania, pp. 170–179.

Hillyer, B. K. and Silberschatz, A. (1996b). Random I/O scheduling in online tertiary storage, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, pp. 195–204.

Hillyer, B. K. and Silberschatz, A. (1996c). Storage technology: Status, issues, and opportunities, *Technical report*, AT&T Bell Laboratories, Murray Hill, New Jersey.

Hillyer, B. K. and Silberschatz, A. (1998). Scheduling non-contiguous tape retrievals, *Proceedings of the Sixth NASA Goddard Conference on Mass Storage Systems and Technologies/Fifteenth IEEE Symposium on Mass Storage Systems*, College Park, Maryland, pp. 113–123.

Hillyer, B. K., Rastogi, R. and Silberschatz, A. (1999). Scheduling and data replication to improve tape jukebox performance, *Proceedings of the 15th International Conference on Data Engineering*, Sydney, Australia, pp. 532–541.

Hilton, M. L., Jawerth, B. D. and Sengupta, A. (1994). Compressing still and moving images with wavelets, *Multimedia Systems* **2**(5): 218–227.

Hjelsvold, R., Langørgen, S., Midtstraum, R. and Sandstå, O. (1995). Integrated video archive tools, *Proceedings of the ACM Multimedia 95*, San Francisco, California, pp. 283–293.

Hjelsvold, R. (1995). *VideoSTAR – A Database for Video Information Sharing*, Dr.ing. thesis, Norwegian Institute of Technology, Trondheim, Norway.

Hjelsvold, R. and Midtstraum, R. (1994). Modelling and querying video data, *Proceedings of 20th International Conference on Very Large Data Bases (VLDB'94)*, Santiago de Chile, Chile, pp. 686–694.

Hjelsvold, R., Midtstraum, R. and Sandstå, O. (1995a). Searching and browsing a shared video database, *Proceedings from the First International Workshop on Multi-Media Database Management Systems*, Blue Mountain Lake, New York, pp. 90–98.

Hjelsvold, R., Midtstraum, R. and Sandstå, O. (1995b). A temporal foundation of video databases, *in* J. Clifford and A. Tuzhilin (eds), *Recent Advances in Temporal Databases, Proceedings of the International Workshop on Temporal Databases*, Workshops in Computing, Springer, Zurich, Switzerland, pp. 295–314.

Hjelsvold, R., Midtstraum, R. and Sandstå, O. (1996). Searching and browsing a shared video database, *in* K. C. Nwosu, B. Thuraisingham and P. B. Berra (eds), *Multimedia Database Systems: Design and Implementation Strategies*, Kluwer Academic Publishers, chapter 4, pp. 89–122.

Hughes, P. H. and Brataas, G. (2000). Scalability of a workstation cluster architecture for video-on-demand applications, *Proceedings of Computer Performance Evaluation/TOOLS 2000*, Vol. 1786 of *Lecture Notes in Computer Science*, Springer, Schaumburg, Illinois, pp. 277–293.

ITU-R (1995). *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, ITU-R Recommendation BT.601–5*, International Telecommunication Union – Radiocommunication Bureau, Geneva, Switzerland.

ITU-R (2002). *Parameter values for the HDTV standards for production and international programme exchange, ITU-R Recommendation BT.709–5*, International Telecommunication Union – Radiocommunication Bureau, Geneva, Switzerland.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*, Wiley.

Jain, R. and Hampapur, A. (1994). Metadata in video databases, *SIGMOD RECORD* **23**(4): 27–33.

Jansen, P. G., Hanssen, F. and Lijding, M. (2003). Early quantum task scheduling, *Proceedings of 15th Euromicro Conference on Real-Time Systems*, Porto, Portugal, pp. 203–210.

Johnson, T. and Miller, E. L. (1998a). Benchmarking tape system performance, *Proceedings of the Sixth NASA Goddard Conference on Mass Storage Systems and Technologies/Fifteenth IEEE Symposium on Mass Storage Systems*, College Park, Maryland, pp. 95–112.

Johnson, T. and Miller, E. L. (1998b). Performance measurements of tertiary storage devices, *Proceedings of the 24th VLDB Conference*, New York, USA, pp. 50–61.

Jones, M. B. (1997). The Microsoft interactive TV system: An experience report, *Technical Report MSR-TR-97-18*, Microsoft Research, Redmond, Washington.

Judge, J. S., Schmidt, R. G., Weiss, R. D. and Miller, G. (2003). Media stability and life expectancies of magnetic tape for use with IBM 3590 and Digital Linear Tape systems, *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, San Diego, California, pp. 97–100.

Kienzle, M. G., Dan, A., Sitaram, D. and Tetzlaff, W. (1995). Using tertiary storage in video-on-demand servers, *Proceedings of COMPCON '95: Technologies for the Information Superhighway*, San Francisco, California, pp. 225–233.

Koenen, R. (1999). MPEG–4 Multimedia for our time, *IEEE Spectrum* **36**(2): 26–33.

Korst, J. (1997). Random duplicated assignment: An alternative to striping in video servers, *Proceedings of the ACM Multimedia 97*, Seattle, Washington, pp. 219–226.

Koteng, R. (1996). *Video server with VCR-functionality for MPEG-1*, Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway. In Norwegian.

Kozuch, M., Wolf, W. and Wolfe, A. (2000). An experimental analysis of digital video library servers, *Multimedia Systems* **8**(2): 135–145.

Krishnan, R. and Little, T. D. C. (1997). Service aggregation through rate adaptation using a single storage format, *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV'97*, St. Louis, Missouri, pp. 209–218.

Langørgen, S. (1994). *Elvira – Experimental video server for ATM*, Master's thesis, Norwegian Institute of Technology, Trondheim, Norway. In Norwegian.

Lau, S.-W. and Lui, J. C. S. (1996). Scheduling and replacement policies for a hierarchical multimedia storage server, *Proceedings of Multimedia Japan 96, International Symposium on Multimedia Systems*, Yokohama, Japan, pp. 68–75.

Lau, S.-W. and Lui, J. C. S. (1997). Scheduling and data layout polices for a near-line multimedia storage architecture, *Multimedia Systems* **5**(5): 310–323.

Laursen, A., Olkin, J. and Porter, M. (1994). Oracle Media Server: Providing consumer based interactive access to multimedia data, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Minneapolis, Minnesota, pp. 470–477.

Le Gall, D. (1991). MPEG: A video compression standard for multimedia applications, *Communications of the ACM* **34**(4): 46–58.

Li, J. and Orji, C. (1996). I/O scheduling in tape-based tertiary systems, *Journal of Mathematical Modelling and Scientific Computing*.

Liao, W. and Li, V. O. K. (1997). The split and merge protocol for interactive video-on-demand, *IEEE Multimedia* **4**(4): 51–62.

Lignos, D. (1995). Digital linear tape (DLT) Technology and product family overview, *Proceedings of Fourth NASA Goddard Conference on Mass Storage Systems and Technologies*, Maryland, USA.

Lijding, M. E., Hanssen, F. and Jansen, P. G. (2002a). A case against periodic jukebox scheduling, *Technical Report TR-CTIT–02–47*, Centre for Telematics and Information Technology, University of Twente, The Netherlands.

Lijding, M. E., Jansen, P. and Mullender, S. (2002b). A flexible real-time hierarchical multimedia archive, *Proceedings of Joint International Workshop on Interactive Distributed Multimedia Systems / Protocols for Multimedia Systems (IDMS/PROMS)*, Vol. 2515 of *Lecture Notes in Computer Science*, Springer, Coimbra, Portugal, pp. 229–240.

Lijding, M. E. M. (2003). *Real-Time Scheduling of Tertiary Storage*, PhD thesis, University of Twente, Enchede, The Netherlands.

Lijding, M. E., Mullender, S. and Jansen, P. (2002c). A comprehensive model of tertiary-storage jukeboxes, *Technical Report TR-CTIT–02–41*, Centre for Telematics and Information Technology, University of Twente, The Netherlands.

Linear Tape-Open Technology (1998). Linear Tape–Open (LTO) Technology, White paper.

Liou, M. (1991). Overview of the px64 kbit/s video coding standard, *Communications of the ACM* **34**(4): 59–63.

Little, T. D. C. and Venkatesh, D. (1995). Popularity-based assignment of movies to storage devices in a video-on-demand system, *Multimedia Systems* **2**(6): 280–287.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment, *Journal of the ACM* **20**(1): 46–61.

LoBue, M. T. (2002). Surveying today's most popular storage interfaces, *IEEE Computer* **35**(12): 48–55.

Lund, K. and Goebel, V. (2003). Adaptive disk scheduling in multimedia DBMS, *Proceedings of ACM Multimedia '03*, Berkeley, California, pp. 65–74.

Mackay, W. E. and Davenport, G. (1989). Virtual video editing in interactive multimedia applications, *Communications of the ACM* **32**(7): 802–810.

Martin, C., Narayanan, P. S., Özden, B., Rastogi, R. and Silberschatz, A. (1996). The Fellini multimedia storage server, *in* S. M. Chung (ed.), *Multimedia Information Storage and Management*, Kluwer Academic Publishers, chapter 5, pp. 117–146.

McDysan, D. E. and Spohn, D. L. (1995). *ATM: Theory and Application*, McGraw-Hill.

Mesquite Software, Inc. (1994). *Users' Guide CSIM18 Simulation Engine (C++ Version)*, Mesquite Software, Inc., Austin, Texas.

Milenkovic, M. (1998). Delivering interactive services via a digital TV infrastructure, *IEEE Multimedia* **5**(4): 34–43.

Moon, C. and Kang, H. (2001). Heuristic algorithms for I/O scheduling for efficient retrieval of large objects from tertiary storage, *Proceedings from the 12th Australasian Database Conference*, Gold Coast, Queensland, Australia, pp. 145–152.

Moore, R., Prince, T. A. and Ellisman, M. (1998). Data-intensive computing and digital libraries, *Communications of The ACM* **41**(11): 56–62.

More, S. and Choudhary, A. (2000). Scheduling queries on tape-resident data, *Proceedings of European Conference on Parallel Computing*, Vol. 1900 of *Lecture Notes in Computer Science*, Springer, Munich, Germany, pp. 1292–1301.

Moser, F., Kraiß, A. and Klas, W. (1995). L/MRP: A buffer management strategy for interactive continuous data flows in a multimedia DBMS, *Proceedings of the 21st VLDB Conference*, Zurich, Switzerland, pp. 275–286.

MPEG-1 (1992). *Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s*, ISO 11172, International Organization for Standardization.

MPEG-2 (1996). *ISO 13818-6 MPEG-2, Part 6, DSM-CC – Digital Storage Media Command & Control*, ISO 13818, International Organization for Standardization.

MPEG-2 (2000). *Generic coding of moving pictures and associated audio information*, ISO 13818, International Organization for Standardization.

Nemoto, T. and Kitsuregawa, M. (1999). Scalable tape archiver for satellite image database and its performance analysis with access logs – hot declustering and hot replication –, *Proceedings of the 16th IEEE Symposium on Mass Storage Systems/7th NASA Goddard Conference on Mass Storage Systems and Technologies*, San Diego, California, pp. 59–71.

Nussbaumer, J.-P., Patel, B. V., Schaffa, F. and Sterbenz, J. P. G. (1995). Networking requirements for interactive video on demand, *IEEE Journal on Selected Areas in Communications* **13**(5): 779–787.

Nwosu, K. C., Thuraisingham, B. and Berra, P. B. (eds) (1996). *Multimedia Database Systems: Design and Implementation Strategies*, Kluwer Academic Publishers.

Özden, B., Rastogi, R. and Silberschatz, A. (1996a). Buffer replacement algorithms for multimedia databases, *in* S. M. Chung (ed.), *Multimedia Information Storage and Management*, Kluwer Academic Publishers, chapter 7, pp. 163–182.

Özden, B., Rastogi, R. and Silberschatz, A. (1996b). Disk striping in video server environments, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, pp. 580–589.

Pang, H. (1997). Tertiary storage in multimedia systems: staging or direct access?, *Multimedia Systems* **5**(6): 386–399.

Pappas, N. and Christodoulakis, S. (2000). Design & development of a stream service in a heterogeneous client environment, *Proceedings of 26th International Conference on Very Large Data Bases*, Cairo, Egypt, pp. 578–589.

Patterson, D. A. (1997). How to have a bad career in research/academia, Talk given at Academic Careers Workshop.

Patterson, D. A., Gibson, G. and Katz, R. H. (1988). A case for redundant arrays of inexpensive disks (RAID), *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, Chicago, Illinois, pp. 109–116.

Plagemann, T., Goebel, V., Halvorsen, P. and Anshus, O. (2000). Operating system support for multimedia systems, *Computer Communications Journal* **23**(3): 267–289.

Prabhakar, S., Agrawal, D., Abbadi, A. E. and Singh, A. (1996). Tertiary storage: Current status and future trends, *Technical Report 1996–21*, Department of Computer Science, University of California, Santa Barbara, California.

Prabhakar, S., Agrawal, D., Abbadi, A. E. and Singh, A. (1997). Scheduling tertiary I/O in database applications, *Proceedings of Eight International Workshop on Database and Expert Systems Applications*, Toulouse. France, pp. 722–727.

Prabhakar, S., Agrawal, D. and Abbadi, A. E. (2003). Optimal scheduling algorithms for tertiary storage, *Distributed and Parallel Databases* **14**(3): 255–282.

QIC Development Standard (1994). *QIC-5010-DC, Serial Recorded Magnetic Tape Cartridge For Information Interchange, Revision E*, Quarter-Inch Cartridge Drive Standards, Inc.

Reddy, A. L. N. and Wyllie, J. (1993). Disk scheduling in a multimedia I/O system, *Proceedings of the First ACM International Conference on Multimedia '93*, Anaheim, California, pp. 225–233.

Reddy, A. L. N. and Wyllie, J. C. (1994). I/O issues in a multimedia system, *IEEE Computer* **27**(3): 69–74.

Remmem, D. (1997). *VCR functionality for MPEG2 video delivered by a video server*, Master's thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. In Norwegian.

Rompogiannakis, Y., Nerjes, G., Muth, P., Paterakis, M., Triantafillou, P. and Weikum, G. (1998). Disk scheduling for mixed-media workloads in a multimedia server, *Proceeding of ACM Multimedia '98*, Bristol, England, pp. 297–302.

Ruemmler, C. and Wilkes, J. (1994). An introduction to disk drive modeling, *IEEE Computer* **27**(3): 17–29.

Sadashige, K. (2000). Optical recording and recordable DVD overview, *Proceedings of the 17th IEEE Symposium on Mass Storage Systems/8th NASA Goddard Conference on Mass Storage Systems and Technologies*, College Park, Maryland, pp. 295–311.

Sadashige, K. (2003). Data storage technology assessment – 2002 — Projections through 2010, *Technical report*, National Media Laboratory, USA.

Salem, K. and Garcia-Molina, H. (1986). Disk striping, *Proceedings of the 2nd International Conference on Data Engineering*, Los Angeles, California, pp. 336–342.

Sandstå, O. and Midtstraum, R. (1998). Analysis of retrieval of multimedia data stored on magnetic tape, *Proceedings of the IEEE International Workshop on Multi-Media Database Management Systems*, Dayton, Ohio, pp. 54–63.

Sandstå, O. and Midtstraum, R. (1999a). Improving the access time performance of serpentine tape drives, *Proceedings of the 15th International Conference on Data Engineering*, Sydney, Australia, pp. 542–551.

Sandstå, O. and Midtstraum, R. (1999b). Low-cost access time model for a serpentine tape drive, *Proceedings of the 16th IEEE Symposium on Mass Storage Systems/7th NASA Goddard Conference on Mass Storage Systems and Technologies*, San Diego, California, pp. 116–127.

Sandstå, O., Langørgen, S. and Midtstraum, R. (1996). Design and implementation of the Elvira video server, *Norsk Informatikkonferanse 1996*, Alta, Norway, pp. 259–270.

Sandstå, O., Langørgen, S. and Midtstraum, R. (1997). Video server on an ATM connected cluster of workstations, *Proceedings of XVII International Conference of the Chilean Computer Science Society*, Valparaíso, Chile, pp. 207–217.

Santos, J. R. and Muntz, R. (1998). Performance analysis of the RIO multimedia storage system with heterogeneous disk configurations, *Proceedings of ACM Multimedia 98*, Bristol, England, pp. 303–308.

Santos, J. R., Muntz, R. R. and Ribeiro-Neto, B. (2000). Comparing random data allocation and data striping in multimedia servers, *2000 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Santa Clara, California, pp. 44–55.

Schuett, A., Katz, R. and Chervenak, A. (1997). Helical scan tape reliability: Myths and realities, Technical report.

Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V. (2003). *RTP: A Transport Protocol for Real-Time Applications*, RFC 3550, IETF.

Schulzrinne, H., Rao, A. and Lanphier, R. (1998). *Real Time Streaming Protocol (RTSP)*, RFC 2326, IETF.

Schwetman, H. (1996). CSIM18 – the simulation engine, *Proceedings of the 1996 Winter Simulation Conference*, Coronado, California, pp. 517–521.

Schwetman, H. and Brumfield, J. (1997). Data analysis and automatic run-length control in CSIM18, *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, Georgia, pp. 687–692.

Sen, S., Rexford, J. and Towsley, D. (1999). Proxy prefix caching for multimedia streams, *Proceedings of INFOCOM '99: Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, pp. 1310–1319.

Shastri, V., Rajaraman, V., Jamadagni, H., Rangan, P. V. and Sampath-Kumar, S. (1996a). Design issues and caching strategies for CD-ROM based multimedia storage, *Proceedings of Multimedia Computing and Networking 1996*, Vol. 2667 of *Spie Proceedings*, San Jose, California, pp. 30–47.

Shastri, V., Rangan, P. V. and Sampath-Kumar, S. (1997). DVDs: much needed "shot in the arm" for video servers, *Multimedia Tools and Applications* **5**(1): 33–63.

Shastri, V., Rangan, P. V., Rajaraman, V., Pittet, A. and Kumar, S. S. (1996b). Performance issues in CD-ROM-based storage systems for multimedia, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, pp. 618–621.

Shenoy, P. and Vin, H. M. (2002). Cello: A disk scheduling framework for next generation operating systems, *Real Time Systems Journal* **22**(1): 9–47.

Shenoy, P. J., Goyal, P. and Vin, H. M. (1995). Issues in multimedia server design, *ACM Computing Surveys* **27**(4): 636–639.

Shriver, E. (1997). *Performance modeling for realistic storage devices*, PhD thesis, New York University.

Shriver, E., Merchant, A. and Wilkes, J. (1998). An analytic behavior model for disk drives with readahead caches and request reordering, *Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, Madison, Wisconsin, pp. 182–191.

Sitaram, D. and Dan, A. (2000). *Multimedia Servers: Applications, Environments, and Design*, Morgan Kaufmann Publishers, San Francisco, California.

SMPTE (1998). *Television – 1920 × 1080 Scanning and Analog and Parallel Digital Interfaces for Multiple Picture Rates, SMPTE 274M*, Society of Motion Picture and Television Engineers.

Sony (2003). Sony SAIT: A strategic choice for managing the digital content explosion, *Technical report*, Sony Electronics Inc.

Sørensen, G. (1997). *Server for digital video archives*, Master's thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. In Norwegian.

Sørensen, G. and Sandstå, O. (1996). ECM – Elvira Catalog Manager, *Technical report*, Department of Computer Systems and Telematics, Norwegian University of Science and Technology, Trondheim, Norway.

Stallings, W. (1996). IPv6: The new internet protocol, *IEEE Communications* **34**(7): 96–108.

Steinmetz, R. (1995). Analyzing the multimedia operating system, *IEEE Multimedia* **2**(1): 68–84.

Steinmetz, R. and Nahrstedt, K. (1996). *Multimedia: Computing, Communications and Applications*, Prentice Hall, chapter 7 Optical Storage Media.

Strategic Research (1996). Demystifying tape performance, *Technical report*, Strategic Research Corporation, Santa Barbara, California.

Sun Microsystems (1999). Sun StorEdge A5200 fibre channel array and Sun Enterprise 10000 server set new milestones for I/O speed and performance, *White paper*, Sun Microsystems, Palo Alto, California.

Tandberg Data (1996). *Tandberg MLR1 Series Streaming Tape Cartridge Drives Reference Manual*, 1st edn, Tandberg Data, Oslo, Norway.

Tandberg Data (1997). *Tandberg MLR Library Series Reference Manual*, Tandberg Data, Oslo, Norway.

Tandberg Data (2003). *Tandberg SLR7, SLR50, SLR60, SLR75, SLR100, SLR140 Reference Manual*, 11th edn, Tandberg Data, Oslo, Norway.

Tantaoui, M. A., Hua, K. A. and Sheu, S. (2002). Interaction with broadcast video, *Proceedings of ACM Multimedia '02*, Juan-les-Pins, France, pp. 29–38.

Taylor, J. (1999). DVD-Video: Multimedia for the masses, *IEEE Multimedia* **6**(3): 86–92.

Tekalp, A. M. (1995). *Digital Video Processing*, Prentice Hall Signal Processing Series, Prentice Hall.

Tewari, R., Mukherjee, R., Dias, D. M. and Vin, H. M. (1996). Design and performance tradeoffs in clustered video servers, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, pp. 144–150.

Triantafillou, P. and Georgiadis, I. (1999). Hierarchical scheduling algorithms for near-line tape libraries, *Proceedings of 10th International Workshop on Database and Expert Systems Applications*, Florence, Italy, pp. 50–54.

Triantafillou, P. and Papadakis, T. (1997). On-demand data elevation in a hierarchical multimedia storage server, *Proceedings of the 23rd VLDB Conference*, Athens, Greece, pp. 226–235.

Triantafillou, P. and Papadakis, T. (2001). Continuous data block placement in and elevation from tertiary storage in hierarchical storage servers, *Cluster Computing* **4**(2): 157–172.

Triantafillou, P., Christodoulakis, S. and Georgiadis, C. A. (2002). A comprehensive analytical performance model for disk devices under random workloads, *IEEE Transactions on Knowledge and Data Engineering* **14**(1): 140–155.

Tsao, S.-L. (2001). A low cost optical storage server for near video-on-demand systems, *IEEE Transactions on Broadcasting* **47**(1): 56–65.

Tsao, S.-L., Huang, Y.-M., Lin, C.-C., Liou, S.-C. and Huang, C.-W. (1997). A novel data placement scheme on optical discs for near-VOD servers, *Proceedings of the 4th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS '97)*, Vol. 1309 of *Lecture Notes in Computer Science*, Springer, Darmstadt, Germany, pp. 133–142.

Tsiolis, A. K. and Vernon, M. K. (1997). Group-guaranteed channel capacity in multimedia storage servers, *1997 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Seattle, Washington, pp. 285–297.

Van Meter, R. (1997). Observing the effects of multi-zone disks, *Proceedings of USENIX 1997 Annual Technical Conference*, USENIX, Anaheim, California, pp. 19–30.

Van Meter, R. and Gao, M. (2000). Latency management in storage systems, *Proceedings of the 4th USENIX Symposium on Operating System Design & Implementation*, San Diego, California, pp. 103–118.

Vin, H. M., Goyal, P., Goyal, A. and Goyal, A. (1994a). An observation-based approach for designing multimedia servers, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Boston, Massachusetts, pp. 234–243.

Vin, H. M., Goyal, P., Goyal, A. and Goyal, A. (1994b). A statistical admission control algorithm for multimedia servers, *Proceedings of the ACM Multimedia 94*, San Francisco, California, pp. 33–40.

Vin, H. M., Rao, S. S. and Goyal, P. (1995). Optimizing the placement of multimedia objects on disk arrays, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Washington D.C., pp. 158–165.

Viswanathan, S. and Imielinski, T. (1996). Metropolitan area video-on-demand service using pyramid broadcasting, *Multimedia Systems* **4**(4): 197–208.

Vogt, C. (1995). Quality-of-service management for multimedia streams with fixed arrival periods and variable frame sizes, *Multimedia Systems* **3**(2): 66–75.

Wallace, G. K. (1991). The JPEG still picture compression standard, *Communications of the ACM* **34**(4): 30–44.

Wang, Y. and Du, D. H. C. (1997). Weighted striping in multimedia servers, *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Ottawa, Ontario, Canada, pp. 102–109.

Wijayaratne, R. and Reddy, A. L. N. (2000). Providing QOS guarantees for disk I/O, *Multimedia Systems* **8**(1): 57–68.

Wilkes, J., Golding, R., Staelin, C. and Sullivan, T. (1995). The HP AutoRAID hierarchical storage system, *In proceeding from the fifteenth ACM Symposium on Operating Systems Principles*, Cooper Mountain, Colorado, pp. 96–108.

Worthington, B. L., Ganger, G. R. and Patt, Y. N. (1994). Scheduling algorithms for modern disk drives, *Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, Nashville, Tennessee, pp. 241–251.

Worthington, B. L., Ganger, G. R., Patt, Y. N. and Wilkes, J. (1995). On-line extraction of SCSI disk drive parameters, *Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, Ottawa, Canada, pp. 146–156.

Xu, L. (2001). Efficient and scalable on-demand data streaming using UEP codes, *Proceedings of ACM Multimedia '01*, Ottawa, Canada, pp. 70–78.

Yu, P. S., Chen, M.-S. and Kandlur, D. D. (1993). Grouped sweeping scheduling for DASD-based multimedia storage management, *Multimedia Systems* **1**(3): 99–109.

Zhang, L., Deering, S., Estrin, D., Shenker, S. and Zappala, D. (1993). RSVP: A new resource ReSerVation protocol, *IEEE Network* **7**(5): 8–18.

Zimmermann, R. and Ghandeharizadeh, S. (1997). Continuous display using heterogeneous disk-subsystems, *Proceedings of the ACM Multimedia 97*, Seattle, Washington, pp. 227–238.

Appendix A

Elvira II Video Archive Server

This appendix gives an introduction to the Elvira II video archive server. The Elvira II video archive server is a parallel video server that supports efficient storage and delivery of digital video to users. In addition to using hard disks for storing the most frequently used videos, it supports mass storage of digital video using tertiary storage.

The Elvira II video archive server was designed and implemented as part of the initial work on studying storage and delivery of digital video. Working on the practical issues of designing and implementing a working prototype for a digital video archive server has given valuable experiences into the central problems of storing and delivering digital video. The Elvira II video archive server has been used as a framework and for establishing the context for the research areas covered in this thesis. The research on digital video archives is based on the architecture of the Elvira II video archive server. This appendix gives an introduction to how such a video archive server can be designed and implemented.

A.1 Background

The work on storage and delivery of digital video using a video server was initiated by the VideoSTAR project (Hjelsvold, 1995). In the VideoSTAR project a data model for storing meta-information about digital video (Hjelsvold and Midtstraum, 1994) and a model for querying and browsing of digital video (Hjelsvold et al., 1996) were created. Based on this model the VideoSTAR prototype for a video database management system was implemented (Hjelsvold et al., 1995).

VideoSTAR stores both meta-data and the digital video in ordinary Unix files. To achieve high quality playback of the digital video, it required the video files to be stored on the user's workstation. This was a solution that neither scaled well nor promoted sharing of video between users. To efficiently share the digital video among multiple users it was necessary to store the video files on a central server. With the aim of being able to store and efficiently play back the video from a server, the first version of the Elvira video server was designed and im-

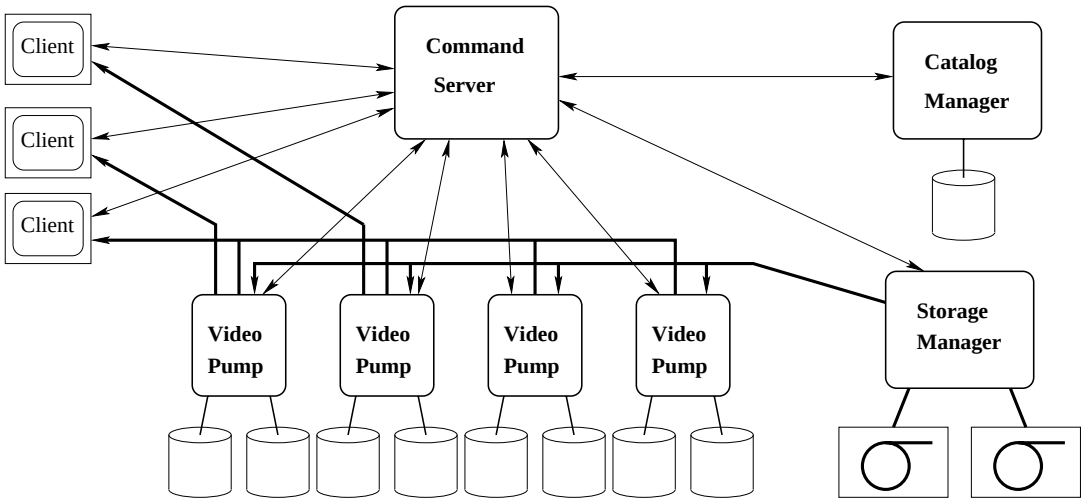


Figure A.1 The architecture for the Elvira II video archive server. Thick lines show how video data is transported while thin lines show how commands and control information are exchanged.

plemented (Langørgen, 1994). The first version supported storage and playback of Motion JPEG compressed video. This was later extended to support MPEG-1 video (Sandstå, Langørgen and Midtstraum, 1996). The Elvira video server was mainly used for storing and delivering digital video for the VideoSTAR prototype. It was also used by the LAVA project for delivering television news on the Internet (Bryhni, Lovett, Maartmann-Moe, Solvoll and Sørensen, 1996).

To continue the research on storage and delivery of digital video, we decided to design and implement a new experimental video server. The design of the new video server called Elvira II video archive server was based on our experiences from using the first Elvira prototype (Sandstå et al., 1997). We wanted the new video server to be even more efficient in the delivery of digital video and to be able to scale better, both in terms of the number of nodes that could efficiently be utilized in video delivery, and in the amount of video that could be stored on the server. In addition to use hard disks for storing the video, we wanted the new server to be able to use tertiary storage for supporting storage of large amounts of digital video.

A.2 Elvira II Video Archive Server

The Elvira II video archive server consists of two main parts. The first part is the *video server*, which is responsible for delivering the video to the users. The video server consists of a *Command Server* and multiple *Video Pumps*. The capacity of the

video server can be scaled by adding more video pumps as they are independent of each other. The second part is the *video archive*, responsible for providing mass storage for digital video. In addition, the system contains a *video catalog* that is responsible for storing meta information about the content stored in the video server and in the video archive.

The architecture of Elvira II video archive server is presented in Figure A.1. This figure contains the main modules in the video archive server. The clients connect to the video server and request a certain movie. The video server functions as a cache for the video archive. If the requested video sequence is stored on the video server's disks, it is delivered directly from disk. If the video is not found on the disks, the video is retrieved from the video archive into the video server's disks and then delivered to the user. The video catalog is used both by the users of the video archive for browsing and searching the content of the archive and by the video server for finding storage information about each video sequence it has to deliver.

A.2.1 Elvira II Command Server

The video archive server consists of multiple server processes. One of them is the *Command Server* and the rest are *Video Pumps*. Remote clients connect to the command server and submit commands to it. If a video sequence is to be delivered, the command server allocates delivery capacity on one or more of the Video Pumps, and instructs them to start the delivery of the requested video.

The command server normally runs on a separate node in the video server, but can also run on one of the video pump nodes. The command server uses the video catalog to get meta-data about the video sequences stored in the video pumps. The main responsibilities of the command server are:

- **Respond to requests from clients.** The clients connect to the Command Server using a TCP/IP connection. The implemented client interface is based on the MPEG-2 DSM-CC standard (MPEG-2, 1996). This defines commands that clients can use for controlling the playback of video.

- **Resource administration.** The command server keeps track of the current usage of disk and network bandwidths to make sure that the system is not overloaded, in order to avoid disruption of the video delivery. The is done by performing admission control before a client is allowed to start playback of a new video. It is also the responsibility of the command server to keep the pump nodes evenly loaded.

- **Control the video pumps.** This is done by sending commands for allocating resources in the video pumps and commands for controlling the video delivery.

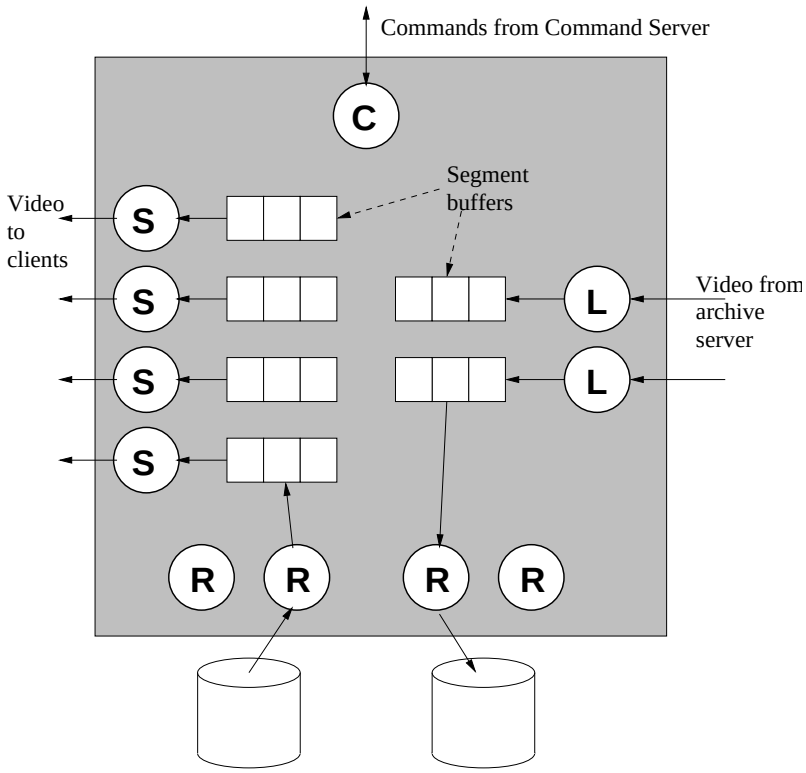


Figure A.2 The design of the Elvira II video pump. The threads are specified as: C = Command Thread, S = Delivery Thread, L = Loading Thread, and R = Disk Thread.

All communication between the video clients (video players) and Elvira II goes through the command server. Each time the client issues a new command, the command server updates the DSM-CC state machine for the video and sends the necessary commands to the video pumps that are responsible for delivering the video to the client.

A.2.2 Elvira II Video Pumps

When a video sequence is loaded into the video server, manually or from the video archive, it is stored either in one of the video pumps or striped across multiple video pumps. To increase the delivery capacity of the server, popular video sequences may also be stored on multiple video pumps. It is the command server that is responsible for assigning clients to video pumps with spare capacity.

The main responsibility of the video pump is to deliver the video sequences to the clients as isochronous data streams. The video is delivered to the client using UDP on top of IP or ATM. In addition to delivering data to the clients, the

video pumps are also taking part in transferring video sequences from the video archive to the video pumps' disks.

Figure A.2 gives an overview of the architecture of the Elvira II Video Pump. A video pump process consists of the following threads:

- A *Command thread* which is responsible for communication with the command server. This thread receives commands containing information about the video sequences to deliver, the client address the video should be delivered to, when to start the delivery and which speed the playback should have. It also answers requests about the status of the video pump like CPU usage and status for delivery of the video sequences.
- For each disk used by the video pump, there are two *Disk threads* that are responsible for reading video data (segments) from the disk into the main memory. These threads are also responsible for writing video that is received from the video archive to disk.
- A set of *Delivery threads* that are responsible for delivering video data from main memory to the network.
- A set of *Loading threads* that are responsible for receiving video from the video archive, storing the video segments in main memory, and insert them into the disk threads' write queues.

To ensure that the clients receive the video in time for the presentation, all operations performed by the *delivery threads* and the *disk threads* are given a deadline for when they have to be completed. If an operation can not be performed within its deadline, that operation is skipped. Skipped operations leads to interruptions in the video playback.

Elvira II File System and Storage format

Elvira has its own file system. This can either be placed on top of a standard Unix file system or use raw disk devices. The main reason for having a special file system is to give the video server full control of where to place the different video sequences on the disk and to be able to optimize reading (and writing) of video from (to) the disks.

When a file containing a video is stored in the video server it has to be converted to the Elvira file format. Figure A.3 shows an overview of the storage format used by Elvira. The video is broken into *containers* which are stored in *segments*. The rationale for using these two storage units are:

- A *container* is the storage unit used by Elvira for sending the video to the client. The size of a container is selected to achieve good utilization of the network and to keep the cost of *transferring* the video as low as possible. In

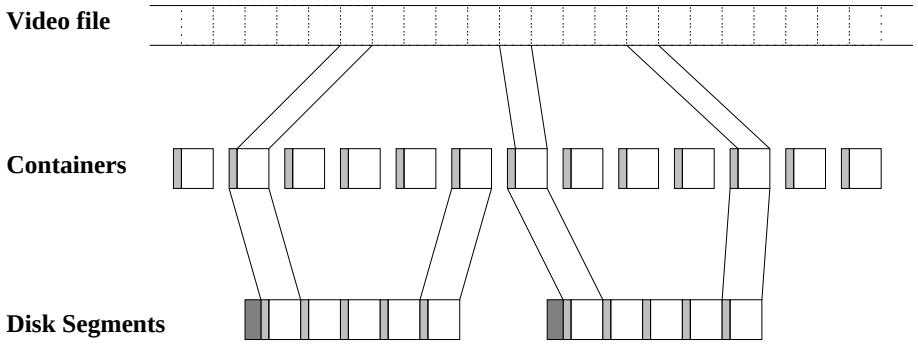


Figure A.3 The storage format used by Elvira. The original video file is first divided into containers suitable for sending on the network. The containers are then stored in segments suitable for storing on disk.

addition to video data, each container has a *container header*. This header contains information about the point in time the container should be received by the client (show time), and about the time it should be presented on the screen (movie time). The *show time* is used by the video pump to determine when to send the container to the client.

- A *segment* is the storage unit used for storing video on the disk. A *segment* contains multiple *containers*. The segment size is selected to give a high utilization of the disk bandwidth.

For a typical configuration using UDP (on top of IP or ATM) the container size will be 8 KB and the segment size will be in the range from 64 KB to 512 KB.

For each video sequence being delivered, a few (typically three) video segments are resident in main memory. As soon as all containers in one of these are sent to the client, a request for the next video segment is entered into the disk scheduler's queue. This request contains information about which segment to retrieve and the deadline for when the segment has to be present in main memory.

As shown in Figure A.2, the delivery threads are responsible for sending the video to the client. The delivery thread uses the show time field from the video container to determine the time when the container has to be sent to the client. If it is not able to send the container within the deadline given by the show time, it skips the container and continues with the next container.

The main advantage of having this storage format is that it breaks the video into storage/transfer units where the size of the container and segment can be independently optimized for transfer from disk to main memory and for transfer on the network. Another advantage of this storage format is that it is independent of the video format. By including the necessary timing information about which time each container should be sent from the server, it is not necessary for the

video pumps to do any processing of the video during delivery of the video. All the necessary processing of the video can be done once when storing the video in the Elvira file system. In our experiments, we have stored both MPEG-2 and Motion-JPEG video in Elvira II without having to do any changes in the video pumps.

The drawback of using a container format that is independent of the video compression format, is that the video client must be able to understand the Elvira container format in order to recreate the video stream. Since the container header consists of just two timestamps this is a very easy job, still this makes it difficult to use a standard video client for displaying the video.

Support for VCR operations

In addition to play back video sequences in the same speed and sequence that the original video had, a video server should also support the main functionality provided by modern VCR and DVD players: This includes playback of the video using a different speed than the originally recorded speed (mostly fast forward and fast rewind, but also slow motion) and jump to a random position within the video.

Elvira II uses *index files* for supporting random access within a video sequence. For each video file stored, there is an index file that contains a list of positions in the film where it is possible to re-start playback. For video formats that compress the individual video frames independently of the surrounding frames, each frame is a possible position for continuing the playback. For more complex compression formats that use interframe encoding, it is not possible to jump to every position within the video. For example, using MPEG-2 it is only possible to jump to positions that are encoded as an I frame.[1] Each entry in the index file contains the timestamp and storage information for the video container that this time stamp corresponds to. When a user requests a jump to a given position in the film, the video pump finds the entry in the index file closest to this position and starts to deliver the video from that point.

There are several strategies for implementing support for fast forward and fast rewind in a video server. In Elvira II, we decided to use separate files containing special fast forward and fast rewind versions of the video. These files contain only a subset of the frames of the original video, for example every fifth frame. Since the quality of the displayed video can be reduced during operations like fast forward, the size of the fast forward file can be further reduced. The main advantage of using separate video files for fast forward and fast rewind is that it does not increase the use of resources during these operations above what is used during normal playback. By having lower quality versions of the video that contain less frames, the video server is able to deliver a fast forward version

[1] For MPEG-2, this requires that the encoder has not changed the encoding strategy between the start position and the position where the user has requested the playback to resume from.

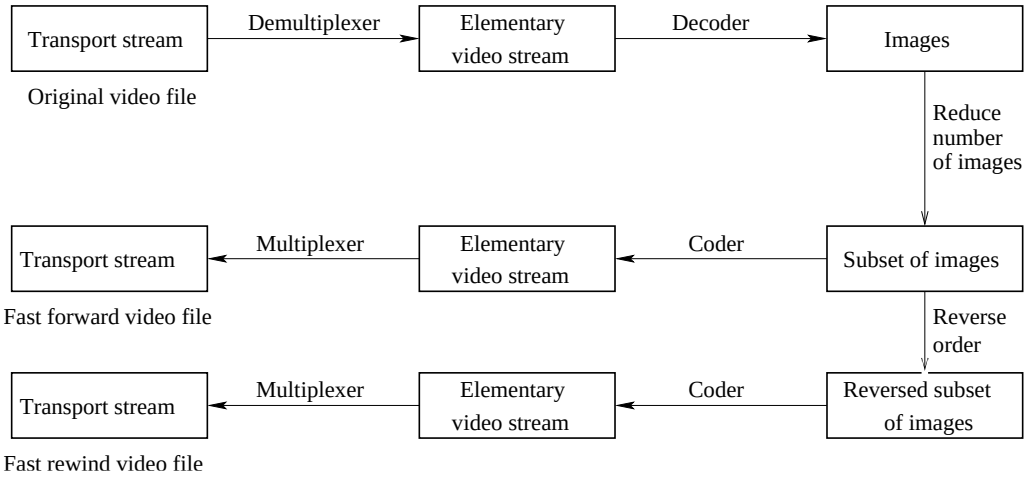


Figure A.4 The process of generating files for supporting fast forward and fast rewind of MPEG-2 video in Elvira II.

or a fast rewind version of the video without increasing disk, memory, CPU, or network resource usage.

When the user issues a command for doing fast forward, the video pump uses the index file for the fast forward file to find the closest position where it can change from delivering the video at normal speed to delivering the video at the requested speed. If the user requests the video to be delivered at a speed which is lower than the speed used for creating the fast forward file, the video pump uses the fast forward file, but delivers the video containers to the network at a slower speed.

The process of creating these extra fast forward/rewind files can be done once as a preprocessing stage when loading the video into the video archive. The main drawback of using separate files for fast forward/rewind is that this requires more storage space. Since these extra files only contains a small fraction of the frames, and can be compressed with a lower quality, the storage space only increases by more approximately 20-30 percent.

Support for MPEG-2

The implemented version of Elvira II supports video compressed using Motion JPEG and MPEG-2. To illustrate the processing of the video that is necessary for including a video in Elvira II, we briefly show how MPEG-2 compressed video can be stored and delivered to a user. A more detailed description of how the support for MPEG-2 video is implemented can be found in (Remmem, 1997).

Elvira II requires that the MPEG-2 video is compressed as a *transport stream* when it is stored in the video server. If it is compressed as either an *elementary*

stream or a *program stream*, it must be converted to a transport stream.

When a MPEG-2 video is loaded into Elvira it has to be stored as Elvira II video containers. In order to enable the video pump to know at which time to deliver each of the containers, the show time has to be stored in the container header. A MPEG-2 transport stream contains multiple time stamps. In our implementation we use the *Program Clock Reference* (PCR) to get information about which time the video pump should deliver each of the containers. Thus, to store a transport stream in the Elvira file system, we mainly do the following operations on the video file:

1. Decodes the transport stream to get the PCR values. Based on these we compute the delivery times to store in the video container's header.

2. The transport stream is built up from transport packages, each of 188 byte. These transport packages are stored as payload in the video containers.

Many MPEG-2 videos have very few access points for random access. In order to provide better support for random access in the video we insert an Random Access Indicator (RAI) and a sequence header in front of each Group of Pictures (GOP) when converting the transport stream to video containers.

To produce files for fast forward and rewind from a MPEG-2 file is a complex task. Figure A.4 shows the main tasks that have to be performed in order to produce these files. Based on a MPEG-2 transport stream, the following steps are performed when producing the fast forward and rewind files:

1. The transport stream has to be demultiplexed in order to get the elementary stream containing the video.

2. By decoding the video stream we get the individual frames.

3. To get frames for the fast forward and rewind files we select every *Nth* frame from the original video. For example, to produce a version of the video that is suitable for fast forward up to five times the speed of the original video, we select every fifth frame.

4. The selected frames are used as basis for producing the fast forward version of the video. In order to make a fast rewind version of the film, we sort the selected frames in opposite order.

5. The frames are encoded using an MPEG-2 coder to produce the Elementary video streams for the fast forward and fast rewind versions of the video.

6. The Elementary video streams are run through an MPEG-2 multiplexer in order to produce transport streams.

The resulting transport streams are stored in the Elvira II file system in the same way as the original version of the video.

The stored MPEG-2 video contains multiple time stamps that can be used by the video client to decode and present the video to the user. The most important of these are Program Clock Reference (PCR), Decoding Time Stamp (DTS), and Presentation Time Stamp (PTS). In our implementation of the video server we do not change any of these when doing a change of speed or when doing a reposition within the video. If the video pumps should do this it would require that it "patched" the content of the video containers that contained any of these time stamps before sending them to the client. Our experience is that these time stamps are not very important during decoding. The MPEG-2 decoder we used decoded the received frames correctly without bothering about that these time stamp where not adjusted correctly when the playback changed from normal speed to fast forward/fast rewind or when the playback jumped from one position to another position in the video.

A.2.3 Elvira II Video Catalog Manager

To store necessary meta-information about the video stored in the video archive, a *Video Catalog* has been implemented (Sørensen and Sandstå, 1996). The purpose of the Elvira II Video Catalog Manager is to function as an information central, where the other Elvira processes and users of the video server and archive can access and store information about the videos stored in the archive. The data model used by the Video Catalog is based on the VideoSTAR data model (Hjelsvold and Midtstraum, 1994). The VideoSTAR data model has been extended to contain information about the compression format used for the video and storage information. This information is used by Elvira II to determine where in the video server a given video is stored and to determine the resources needed for delivering the video to the user.

For each video, the Catalog Manager has information about compression format, bit rate and where the video is currently stored within the server (which video pump it resides in and where in the video archive it is stored). It also has information about the fast forward and fast rewind files that have been generated for each of the videos.

In addition to contain physical storage information, the video catalog supports some of the operations defined in (Hjelsvold et al., 1995b). The most important of these are support for virtual video documents (Mackay and Davenport, 1989) that can be composed by multiple physical video segments and that a video document can have multiple physical representations. For example, the same video can be stored in the server using different compression technologies. This makes it possible for the user to specify which quality she wants on the delivered video, depending on the capacity of the network. It also provides opportunities for the server to change from a high quality version of the video to a lower quality ver-

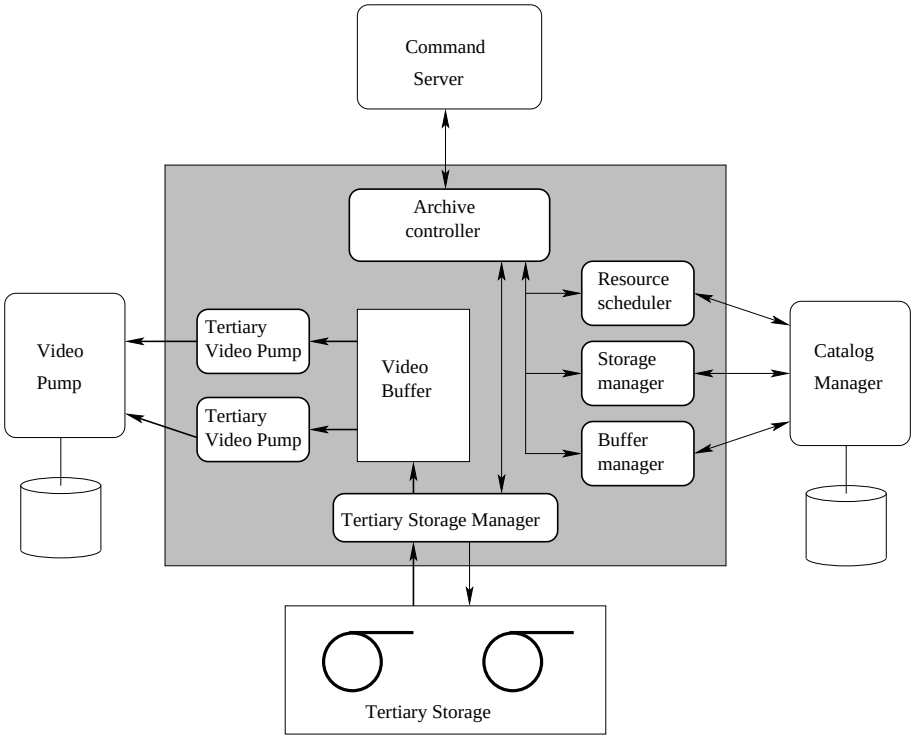


Figure A.5 Overview of the main modules in Elvira II Storage Manager.

sion of the video during playback if it is about to get overloaded.

A Web interface for searching and browsing of the Elvira Video Catalog has been implemented (Gallefoss, 1997).

A.2.4 Elvira II Storage Manager

The Elvira II Storage Manager is responsible for managing and providing storage for the videos stored in the video archive. On request from users, the command server will instruct the Storage Manager to retrieve a given video from its storage system. The video is first transferred from the storage manager to one (or multiple) of the video pumps where it is stored on hard disks. Then it is the responsibility of the video pump(s) to deliver the video to the user. The video pumps act as a *cache* for the storage manager.

Figure A.5 gives an overview of the architecture of the Elvira II Storage Manager. The main operations performed by the Storage Manager are as follows:

- Storage administration. The storage manager is responsible for administration of the videos stored in the archive. It also administrates the storage media that are used for storing the videos and the free storage space that is available for storing new videos.

- Buffer administration. The video pumps contain a copy of the most popular videos and the videos currently being delivered. Since the video pumps in general will have less storage space than the video archive, they function as a *cache* for the video archive. The Storage Manager is responsible for deciding which videos that should reside in the video pumps. Each time a new video is loaded from the video archive to the video pumps, the storage manager has to be sure that the video pumps have enough free storage space. The Storage Manager does this by instructing the video pumps to release the space occupied by one or more of the currently cached videos. To manage the video pumps' buffer space the storage manager uses an LRU strategy.

- Resource scheduler. The Storage Manager receives requests from the command server to load videos from the archive to the video pumps. To optimize the use of the storage devices, the Storage Manager uses a resource scheduler. This can reorder the sequence of when the requests for a video is issued to the storage system.

- Video transfer. When the storage manager has decided to load a video from the archive and into the video pumps, it instructs the storage system to start reading the video and delivering it to the video pump. On the video pump, one of the *Loading threads* will receive the video and make sure it is stored on the video pump's disks.

More details about the design and implementation of the Elvira Storage Manager can be found in (Sørensen, 1997). Since this project did not have access to any media robots, the implemented version of the storage manager uses multiple Tandberg MLR1 tape streamers (Tandberg Data, 1996). In the experiments performed with the Elvira Storage Manager, tapes were loaded and unloaded by a human "tape robot".

A.3 Use of Elvira II Video Archive Server

The implemented version of the Elvira II Video Archive Server runs on clusters of Sparc workstations. In the configuration we have used for running experiments we have used four workstations connected by an ATM switch as video pumps and a separate workstation running the command server and the video catalog. As the Storage Manager we used a workstation that had two MLR1 tape drives connected.

The design and implementation of the video server were done as the initial part of the work behind this thesis. During the process of implementing the video server, it was used as the framework for multiple master theses (Dybvik, 1997; Gallefoss, 1997; Koteng, 1996; Remmem, 1997; Sørensen, 1997).

The Elvira II video archive server has also been used by a Eurescom project that evaluated and compared advantages of different video server architectures (Hughes and Brataas, 2000; Brataas, Crespo, Hughes and Miethe, 1998a; Brataas et al., 1998b). The Elvira II server was used as the basis of evaluating the performance of a video server based on independent workstations. This architecture was compared against a video server based on one parallel machine (Oracle Media Server on the nCube machine (Laursen, Olkin and Porter, 1994)).

A.4 Summary

In this appendix, we have given a short presentation of the Elvira II video archive server. The architecture of the Elvira II video archive server is based on using a parallel video server as a cache for much larger video archive system. The archive stores most of the videos on tertiary storage. The video server stores the most frequently used videos and is responsible for delivering the video to its users. The architecture for the Elvira II video archive server is used when studying storage systems for digital video archives.

Appendix B

Analysis of Model Error Rates for a MLR1 Tape

The access time model for serpentine tape drives was presented in Chapter 6. The purpose of this appendix is to provide a detailed example of how well the access time model estimates seek times. The model partitions all possible seeks into eight seek classes as shown in Figure 6.5. For each seek class, we provide a function which estimates seek times. To demonstrate how accurately the model estimates seek times, we present statistical data and plots of the difference between estimated and measured seek times for tape accesses for each of the eight seek classes.

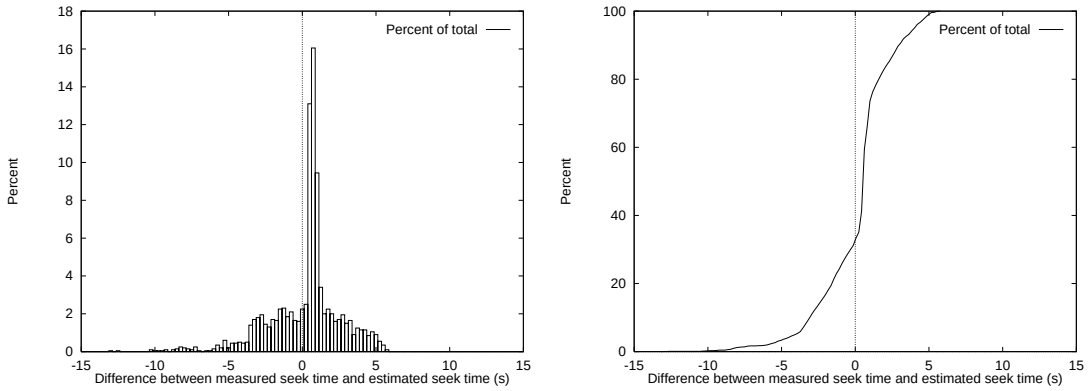
In the measurements presented here, we have used *one* MLR1 tape filled with 32 KB blocks and characterized by use of the *Write-Turn* strategy. Using this tape, we performed 2000 tape accesses, each for one data block. The accesses were selected to give approximately the same number of seeks within each seek class. Figure B.1 shows the distribution of the difference between measured and estimated seek times for all the tape accesses. In Table B.1 we present the *Average absolute deviation* between measured and estimated seek times for all seeks, and for seeks within each of the eight seek classes. The average absolute deviation is computed using the following definition:

$$Average\ absolute\ deviation = \frac{1}{n}\sum_{i=1}^{n} \mid measured_i - estimated_i \mid$$

This is the average difference an application would experience between measured and estimated seek times for tape accesses. We have also included the *Standard deviation* between measured and estimated seek times in the table.

To give a measurement of how close the model models a *specific* tape, we define the *Model closeness* as:

$$Model\ closeness = \frac{1}{n}\sum_{i=1}^{n}(measured_i - estimated_i)$$



(a) Error distribution (b) Cumulative error distribution

Figure B.1 a) Distribution of the difference between measured and estimated seek times for all seeks. **b)** Cumulative distribution of the difference between measured and estimated seek times for all seeks.

Seek class	Samples	Average absolute deviation (s)	Standard deviation	Model closeness (s)
1	393	1,117	1,386	0,027
2	243	2,588	2,044	0,250
3	212	1,887	1,426	0,045
4	216	1,072	1,550	–0,086
5	267	2,188	1,489	0,205
6	213	1,829	1,368	0,195
7	213	1,070	1,581	–0,215
8	243	2,600	1,591	0,514
All	2000	1,767	1,673	0,119

Table B.1 Differences between measured and estimated seek times for 2000 tape accesses grouped by seek class. All times are in seconds.

The closer the *Model closeness* comes to zero, the better does the model estimate seek times for a particular tape. The last column of Table B.1 shows this for each of the seek classes for the used tape.

B.1 Model Deviation Rates for each Seek Class

The rest of this appendix contains plots for each seek class showing the deviation between measured and estimated seek times. For each seek class we provide four plots:

a) **The seek profile.** To get an impression of how well the model estimates the measured seek times within each seek class, we plot the seek time as a function of the physical seek distance. All seeks are ordered by increasing seek distance, and the measured seek time for each seek is used to draw a line. We also include a plot of the function used by the model to produce seek time estimates.

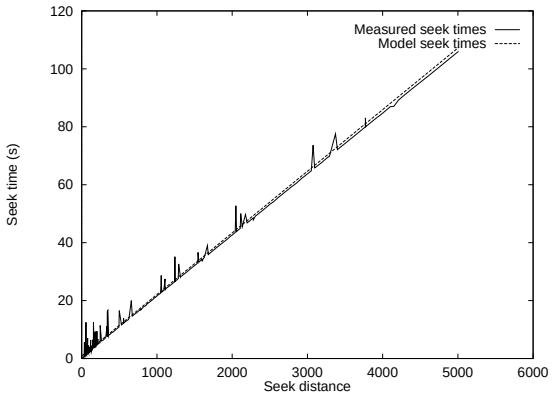
b) **Difference for each seek.** To visualize how much the seek time of individual seek operations differ from the estimated seek time, this plot shows the difference between measured and estimated seek times for each seek. The seeks are ordered along the x-axis in the order they were performed.

c) **Error distribution.** This plot shows the distribution of the difference between measured and estimated seek times.

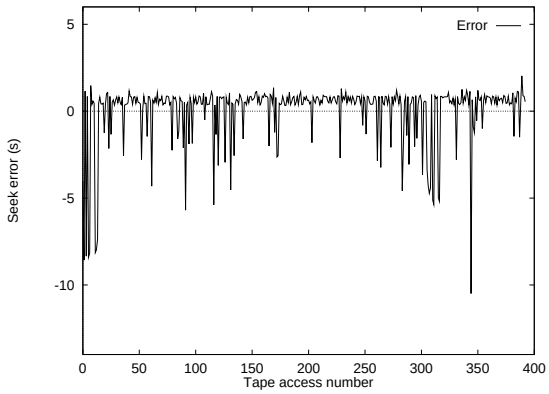
d) **Cumulative error distribution.** This plot shows the cumulative distribution of the difference between measured and estimated seek times.

In all plots, the value shown is negative if the measured seek time is larger than the estimated seek time.

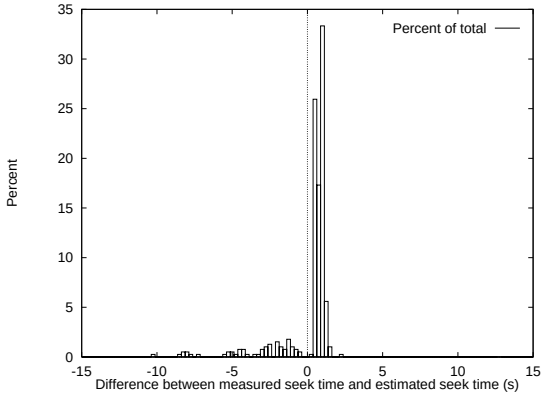
Seek class 1



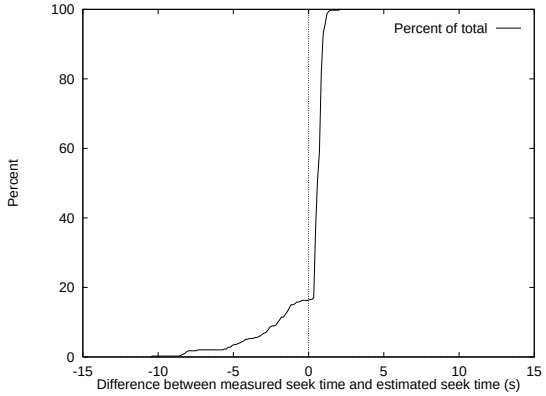
(a) Seek profile



(b) Error for each seek



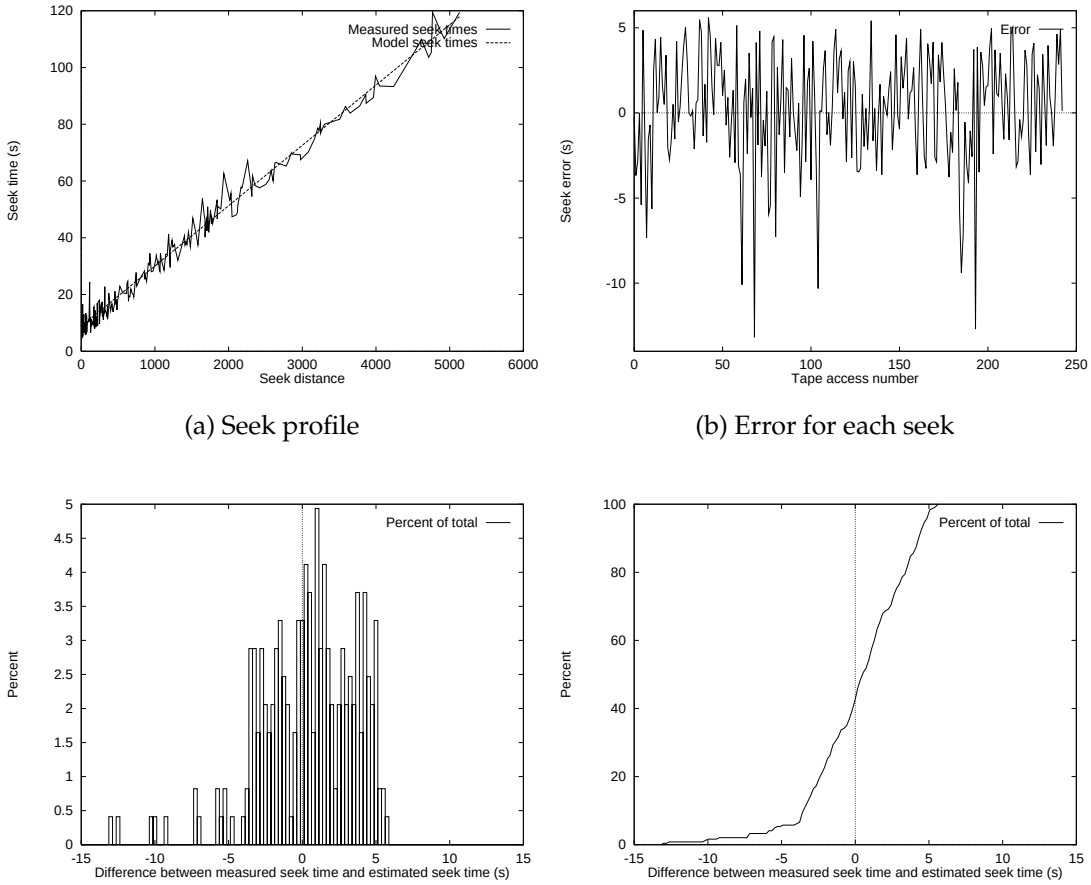
(c) Error distribution



(d) Cumulative error distribution

Figure B.2 Seek class 1.

Seek class 2



(a) Seek profile

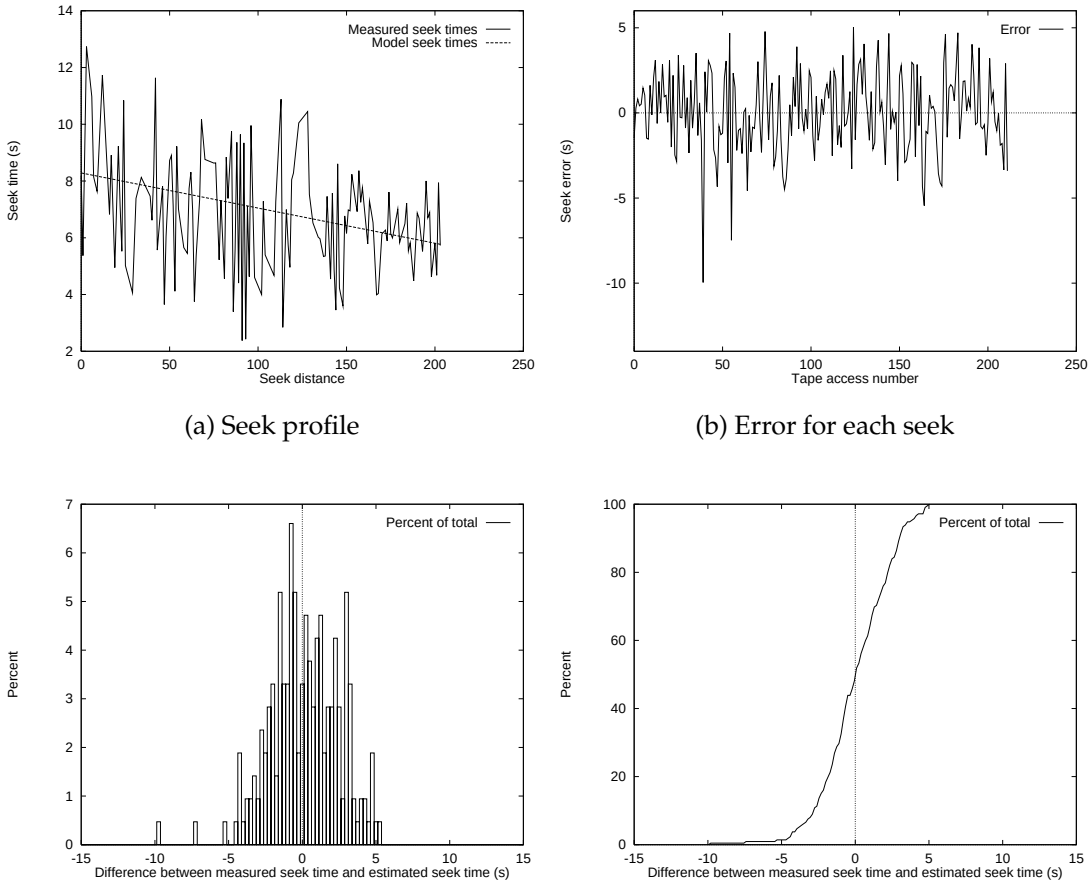
(b) Error for each seek

(c) Error distribution

(d) Cumulative error distribution

Figure B.3 Seek class 2.

Seek class 3



(a) Seek profile

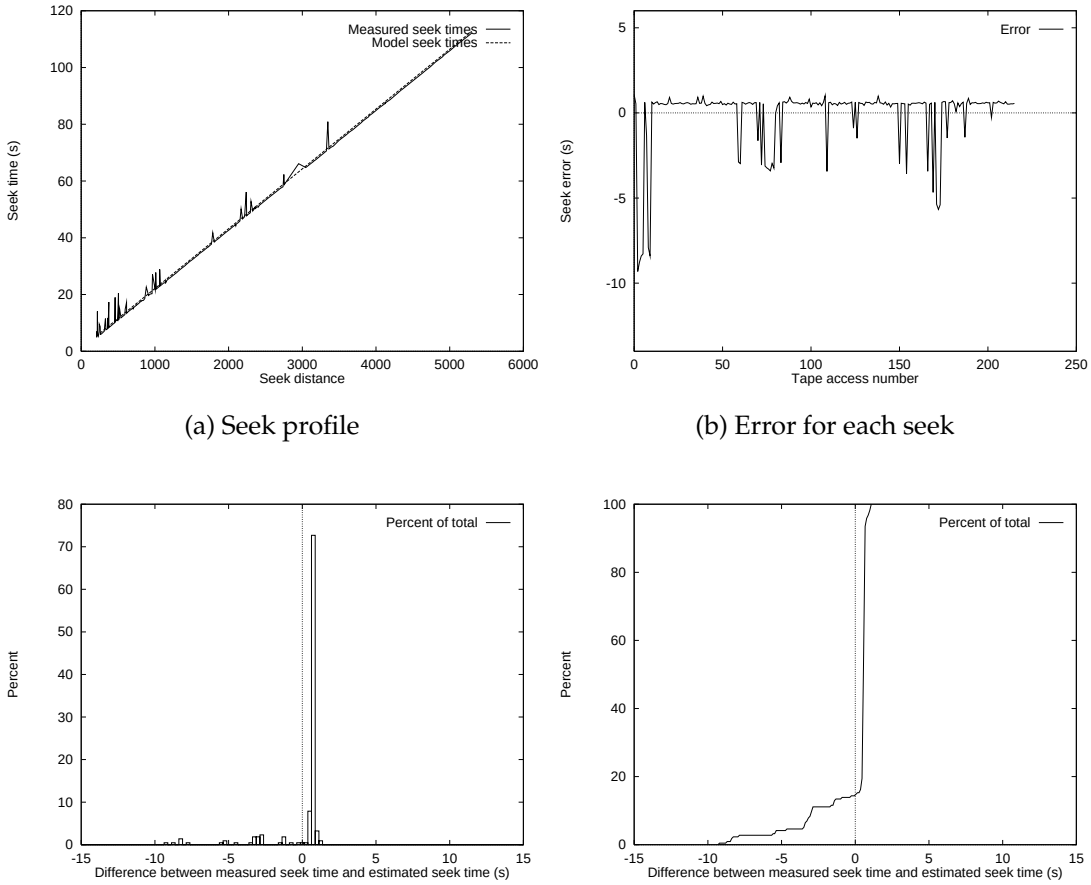
(b) Error for each seek

(c) Error distribution

(d) Cumulative error distribution

Figure B.4 Seek class 3.

Seek class 4



(a) Seek profile

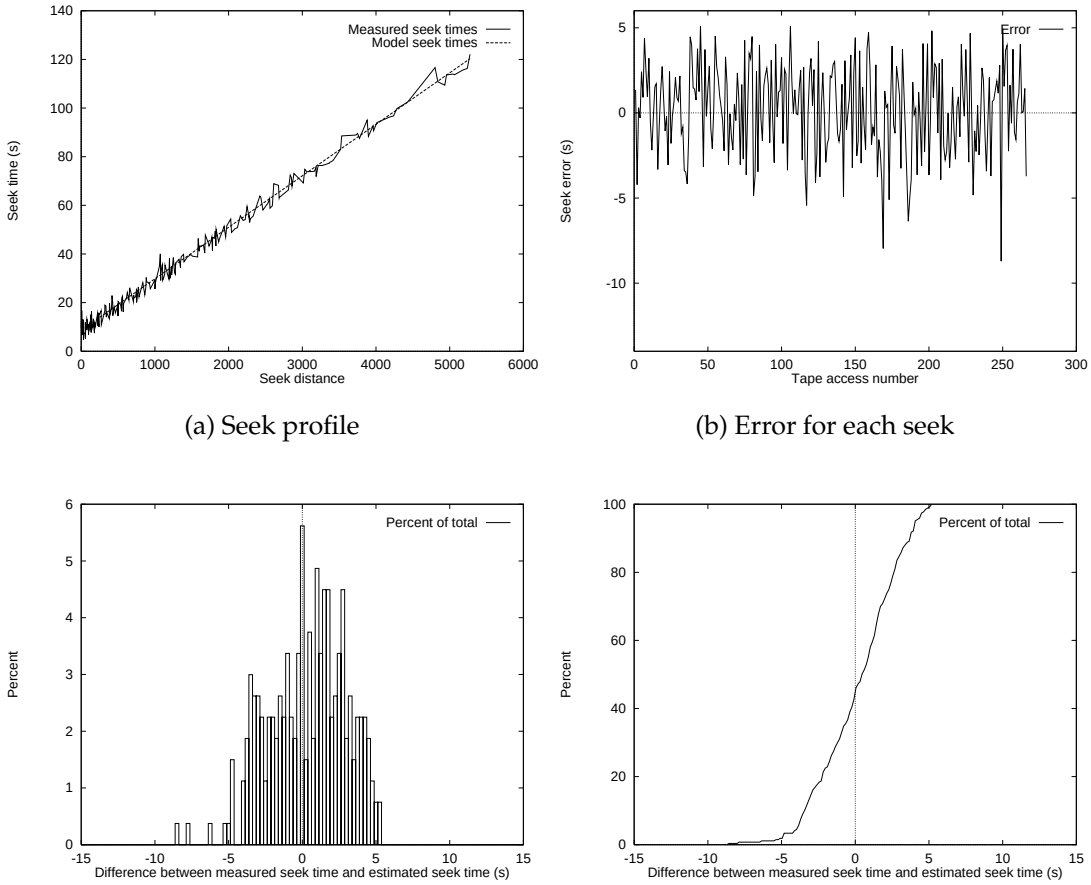
(b) Error for each seek

(c) Error distribution

(d) Cumulative error distribution

Figure B.5 Seek class 4.

Seek class 5



(a) Seek profile

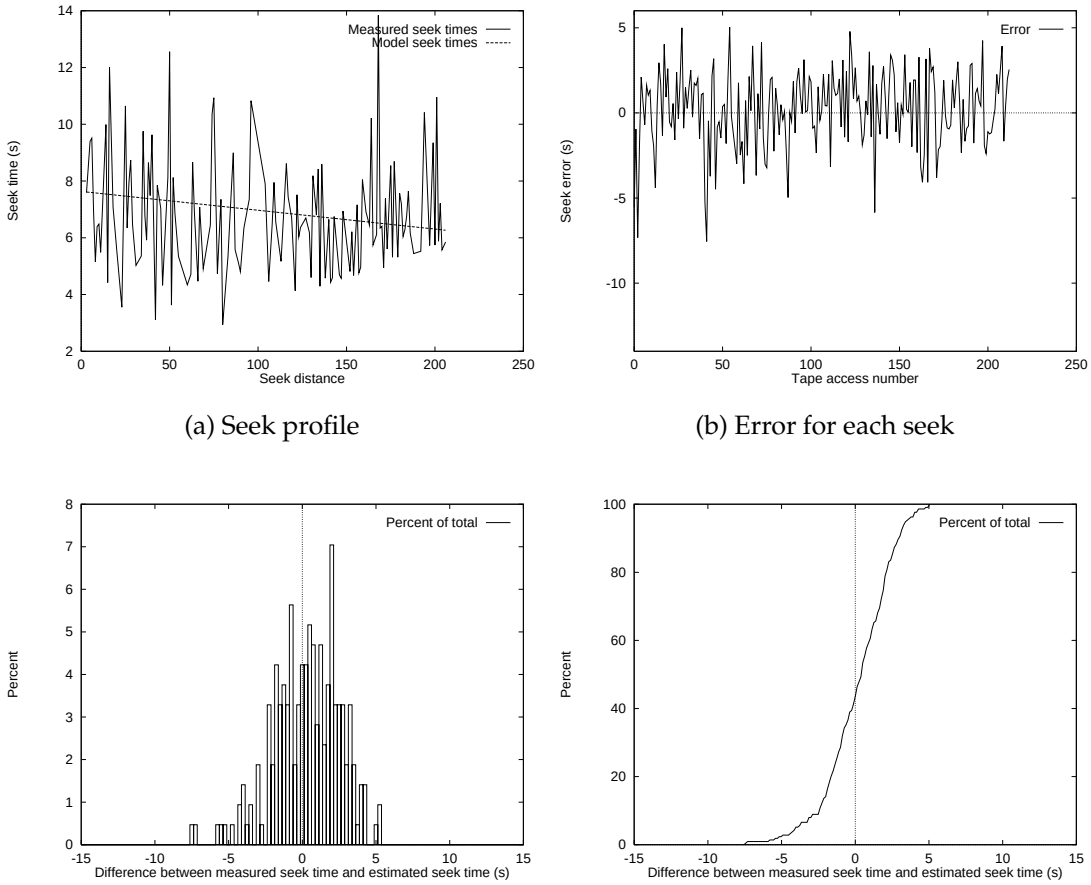
(b) Error for each seek

(c) Error distribution

(d) Cumulative error distribution

Figure B.6 Seek class 5.

Seek class 6



(a) Seek profile

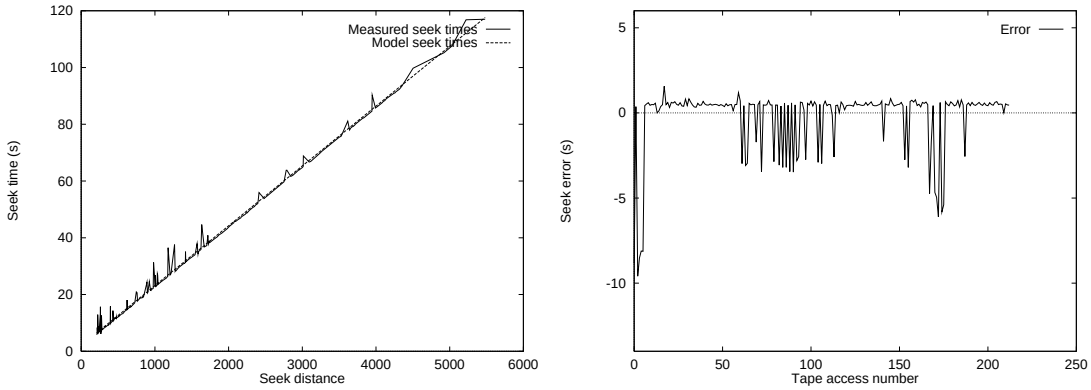
(b) Error for each seek

(c) Error distribution

(d) Cumulative error distribution

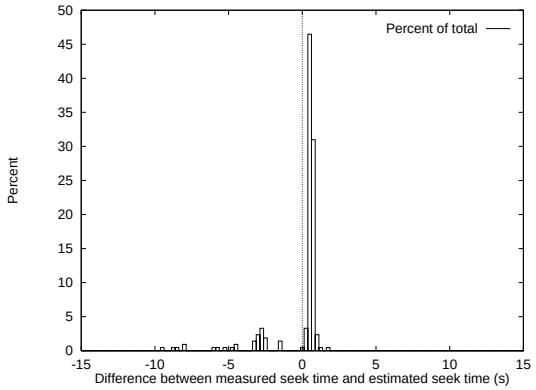
Figure B.7 Seek class 6.

Seek class 7

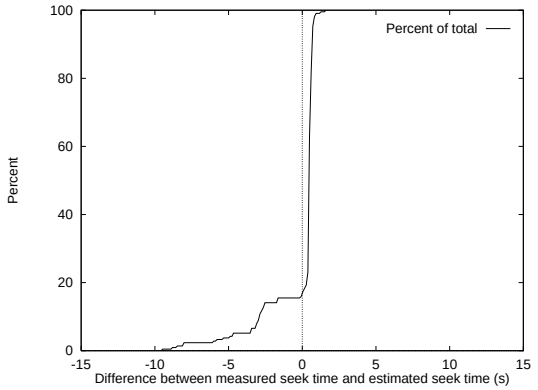


(a) Seek profile

(b) Error for each seek



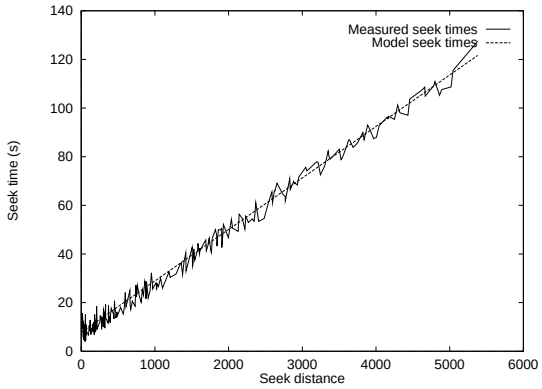
(c) Error distribution



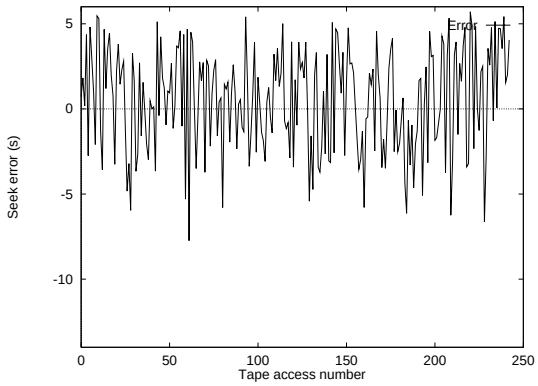
(d) Cumulative error distribution

Figure B.8 Seek class 7.

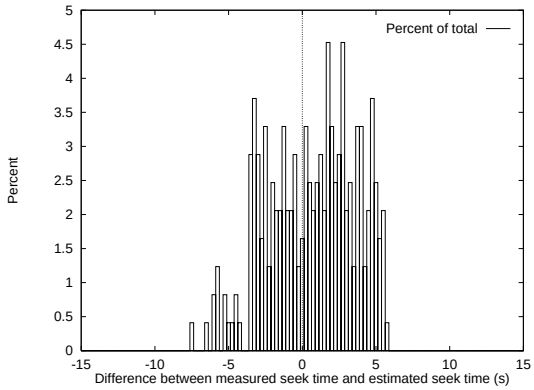
Seek class 8



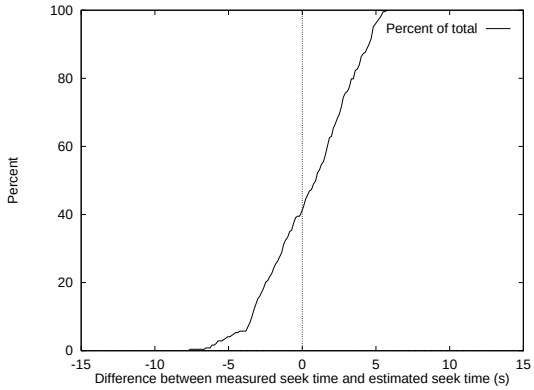
(a) Seek profile



(b) Error for each seek



(c) Error distribution



(d) Cumulative error distribution

Figure B.9 Seek class 8.

Appendix C

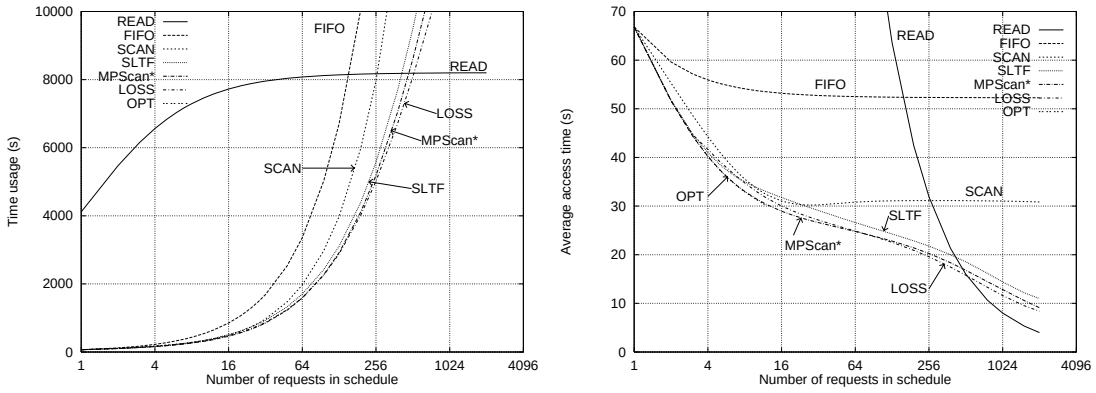
Quantum DLT 2000

The access time model for serpentine tape presented in Chapter 6 was mainly developed using the Tandberg MLR1 drive. To test the usability and accuracy of the tape model on another tape drive, we have used the Quantum DLT 2000 drive (Digital, 1992). In this appendix, we present results from using the scheduling algorithms presented in Chapter 7 for scheduling requests on the Quantum DLT 2000 drive.

The Quantum DLT 2000 drive is one of the first of the high capacity drives of the DLT series. One cartridge is able to store 10 GB of data (without use of compression). The transfer rate of the drive is 1.25 MB/s (with the compression turned off). The physical data layout on the tape is similar to the Tandberg MLR1 drive. Both drives use tapes that are 366 meters long. The width of the tape used by the DLT drive is twice the width of a MLR1 tape (a half inch compared to a quarter inch). The number of physical data tracks is 128, where two physical tracks are grouped into one logical track, giving 64 logical data tracks. The corresponding numbers for the MLR1 drive are 144 physical tracks grouped as 72 logical data tracks.

C.1 Scheduling of Random I/O Requests

In Chapter 7 we presented nine scheduling algorithms for scheduling requests for serpentine tape drives. From simulations and practical measurements using the Tandberg MLR1 drive, we concluded that our new algorithm, Multi-Pass Scan Star (MPScan*), performed equal or better than any of the other algorithms for schedule sizes from two to about thousand requests. In this section we present similar experiments performed using the Quantum DLT 2000 drive. Some of the results presented here are also included in Chapter 7.



(a) Total time

(b) Average access time per I/O request

Figure C.1 Total and average access times using different scheduling algorithms for different problem sizes. Results from the SORT and MPScan schedulers are not included.

C.1.1 Simulations of the Scheduling Algorithms

To compare the relative performance of the algorithms, we have simulated the scheduling algorithms using the access time model for the DLT 2000 drive presented in Section 6.4.2. All simulations were performed on sets of request lists containing from one to 2048 requests. Each request was for one random 32 KB block on the tape. For schedules with less that 256 requests, we used 100,000 different request lists. For longer schedules, the number of request lists were gradually reduced to 500 request lists for schedules of 2048 requests due to CPU limitations. For the OPT algorithm, the largest request lists contained 12 requests, and was run only 100 times due to the high CPU usage. All simulations started with the tape drive positioned at the beginning of the tape.

Figure C.1(a) and C.1(b) show total execution times and average access time per request using the different scheduling algorithms. The results are very similar to what we got using the Tandberg MLR1 drive. With only one request, the average access time will be 67 seconds for all scheduling strategies, except for the READ strategy. Without any scheduling, the average access time will stabilize at about 52 seconds as we increase the number of requests in the schedule. For schedules with less than 12 requests, the curves for MPScan* and OPT overlap, and any of them can be used. As long as the number of requests in a schedule is less than about 80 requests, MPScan* produces the best schedules. For schedules with lengths from 80 to about 500 requests, LOSS is the preferred scheduler. For even longer schedules, the best way to retrieve the requested data is to read the entire tape, i.e., use the READ strategy.

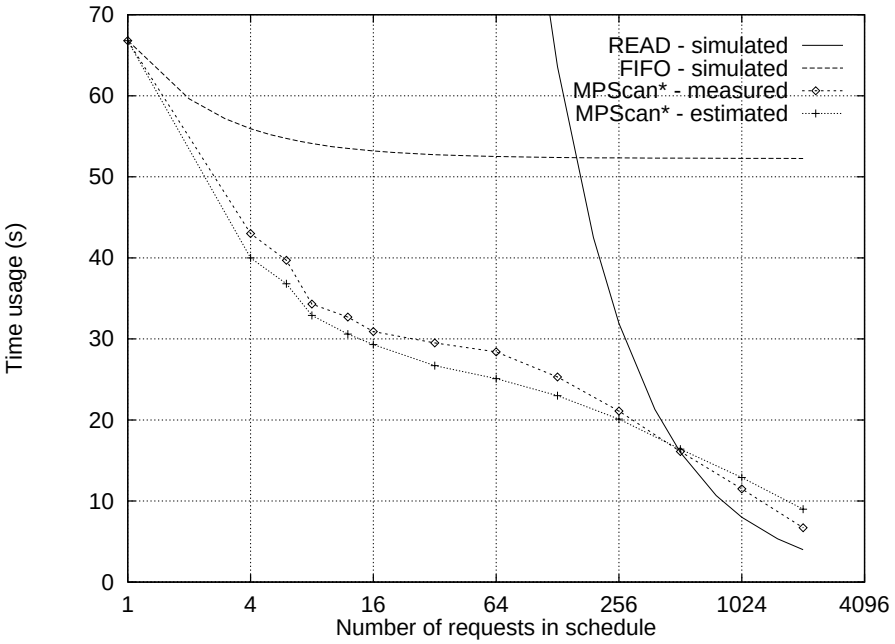


Figure C.2 Measured average access times for requests scheduled by MPScan*. Estimated average access times for the same requests are included to show the average deviation between estimated and measured access times. Simulated access times for FIFO and READ are included for reference.

For the Tandberg MLR1 drive, MPScan* was best for schedules containing up to about 1000 requests. The reason MPScan* has to give up to LOSS for smaller schedule sizes for the DLT 2000 drive, is due to the fewer key points on each track. The fewer key points makes the initial MPScan schedule contain longer seek distances each time the drive has to change track.

C.1.2 Validation of MPScan* on a Quantum DLT 2000 Drive

To validate the correspondence between estimated access times and actual access times when executing the requests of a schedule on a drive, we have run schedules produced by the MPScan* scheduler on the DLT 2000 drive. As for the simulations, each request was for one random 32 KB block, and the tape drive was positioned at the beginning of the tape when we issued the first request. For schedules containing up to 12 requests, 20 schedules were executed on the DLT 2000 drive. For schedules containing 16 and 32 requests, ten schedules were executed. For larger schedules, only five schedules were executed on the drive.

The access time per request was measured. Based on these measurements, the average access times were computed. These are presented in Figure C.2. To visualize the accuracy of the estimated access times produced by the model, we

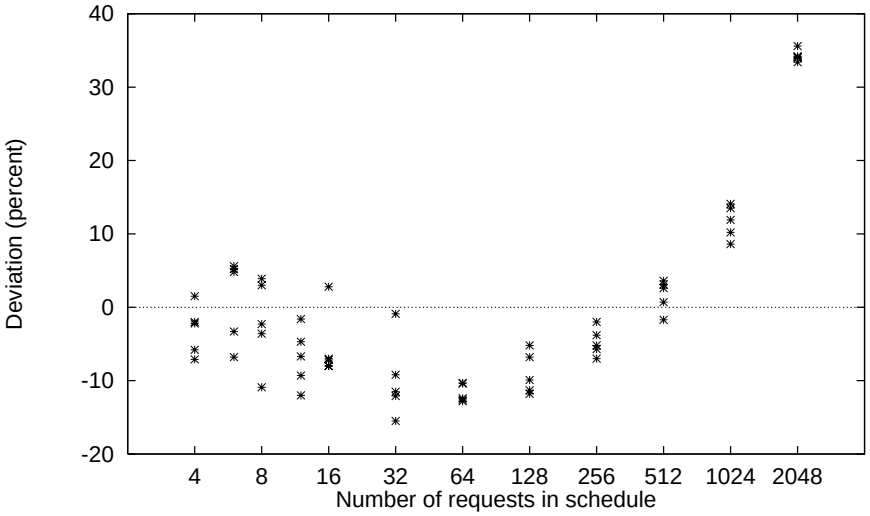


Figure C.3 Percent deviation in estimated execution times for schedules produced by MPScan* and run on the DLT 2000 drive. For each problem size, the results from five schedules are included. A positive value indicates that the estimated execution time is a number of percent higher than the actual measured execution time.

have included the estimated average access times for the same requests in the figure. From the figure we see that for schedules of less than about 500 requests, the model underestimates the cost of executing the schedule on the drive, for larger schedules the model overestimates the cost. To provide a better comprehension of the accuracy of the estimated execution times, Figure C.3 shows the deviation in percent between estimated and measured execution times for a set of schedules produced by the MPScan* scheduler, and run on the DLT 2000 drive. The main reason why the model underestimates the execution times for the schedules with less than 500 requests and overestimates the execution times for larger schedules, is the fact that the model does not take the two speed behavior of the DLT 2000 drive into account. The drive will perform short seek distances using the lower read speed only. Due to the use of linear regression for estimating the cost functions, the seek time for short seeks will be overestimated. As the number of requests in a schedule increases, more of the requests in the final schedule will belong to seek class 1. E.g., for a schedule of 2048 requests, about 1900 of the requests will be in seek class 1, and most of these will have a rather short seek distance. Figure C.4 contains an actual plot of the measured seek times for a schedule of 2048 request. The plot contains only the requests falling into seek class 1. From this figure, it can be seen that the shorter the seek distance is, the more do the model overestimate the actual access time. To only way to avoid this, would be to include the two speed behavior of the DLT 2000 drive into the model.

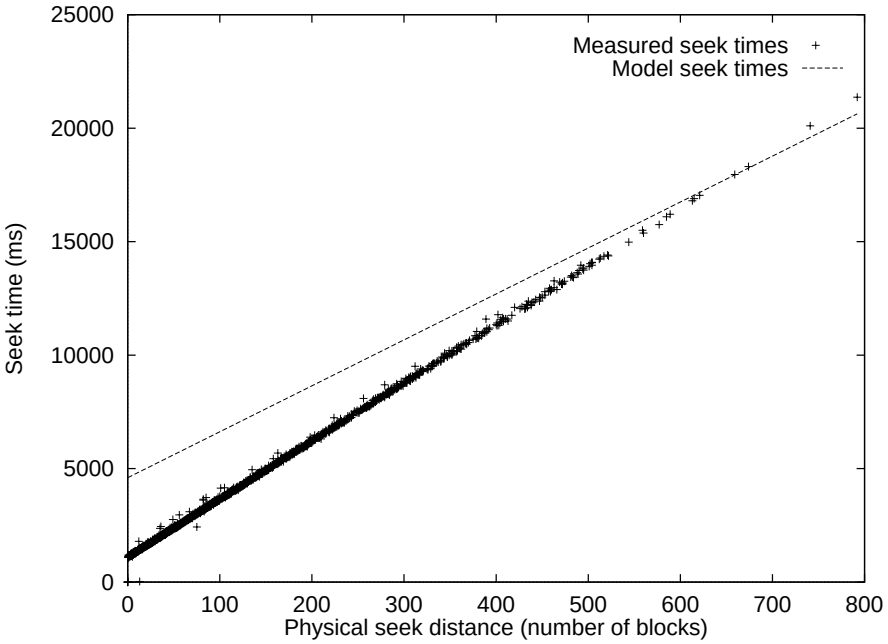


Figure C.4 Plot of actual seek times for the 1880 requests in seek class 1 of a schedule consisting of 2048 requests. All seeks are shorter than 800 blocks. The total length of a track is approximately 4600 blocks. Seek times estimated by the model are included to show how the model overestimates seek times for short seeks.

Despite the fact that the estimated time to perform a schedule on a tape drive is much larger than the actual time for large schedules, the model is good enough for the scheduling algorithms to produce a good ordering of the requests. And by studying Figure C.1, for schedules of this size (more than 500 requests), we should not use any of the complex scheduling strategies. The best way to read this many simultaneous requests is to read the entire tape, i.e., use the model independent READ strategy. The execution time for the READ strategy is not based on the seek time model, but rather on the transfer time. The transfer time is easier to estimate, and thus will be more accurate.