ABSTRACT

MPEG-aware Disk Storage System. (August 2000)

Qian Ren, B.S., Tsinghua University

Chair of Advisory Committee: Dr. A.L. Narasimha Reddy

In the past decades the demand for systems that can process and deliver massive amount of storage has increased. Multimedia applications such as streaming audio and video require large amounts of data to be read from disk and processed for timely delivery to the client. Large decision support data bases not only need to read large amounts of data from storage media but also require scalable processing power as the data set size grows.

Increasing performance and decreasing cost of microprocessors are making it feasible to move more processing power to the data source as proposed in active disks or multiported storage devices. This allows us to investigate new methods of multimedia disk storage system that were not plausible in the past.

MPEG-aware disk storage system is a disk storage system with intelligence about the MPEG formatted multimedia data. MPEG-aware disk has a powerful CPU in the disk controller. In a multimedia server with an MPEG-aware disk storage system, common functions of preprocessing MPEG data from the disk is done by the disk CPU instead of the server CPU, resulting in considerable system-level performance improvements. This allows more multimedia files to be serviced by one server. The

presented prototype implementation of MPEG-aware disk storage system is built on Linux operating system.

ACKNOWLEDGMENTS

I would like to give thanks to my advisor, Dr. Narasimha Reddy, for his help and guidance throughout my academic career. For the last year he has worked with me and given me opportunities to grow and progress under his leadership. His genuine interest in the success of his students has been a major factor in my success, both academic and professional.

I would like to thank my parents, for their guidance and encouragement throughout my life and graduate study. I could not finish the graduate study without their encouragement even though they've been thousands of miles away.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

In the past decades the demand for multimedia services like Video On Demand has dramatically increased. Multimedia applications such as streaming audio and video require large amounts of data to be read from disk and processed for timely delivery to the client. Large decision support databases not only need to read large amounts of data from storage media but also remain scalable processing power as the data set size grows.

Traditionally, multimedia server has the same architecture with ordinary servers with several disks connected to the I/O bus. Multimedia data may be preprocessed in server's Central Processing Unit before it is sent out to clients. One potential problem of this configuration is that the server CPU could be the bottleneck, as all the services need to preprocess data in the server CPU. If a large amount of processing is required, server CPU power may not be scalable to meet the needs. If preprocessing results in reduced data sets, precious I/O interconnect bandwidth is wasted in moving larger data sets to the server's memory. Caching large multimedia data sets in the server's memory may also result in reduced cache hits as multimedia data is typically accessed sequentially.

Several trends are making it feasible to move more processing power to the data source. Disk array controllers today have CPUs that are at most one generation behind

This thesis follows the style and format of *IEEE Transactions on Multimedia.*

state-of-the-art workstations. Individual disk controllers are further behind, but are catching up fast. Chip technology is to the point where it is feasible to combine a CPU with the specialized ASIC responsible for drive's processing into a single chip. This makes it possible to integrate a 200 MHz RISC core in the same die space as the ASIC with room to spare [1].

In this thesis, we present an MPEG-aware disk storage system that is endowed with processing power and intelligence about the MPEG formatted multimedia data on its physical media. MPEG-aware disk has a CPU in the disk controller, which facilitates data processing carried out by the CPU of the multimedia server. It also has intelligence about the MPEG formats, which is critical in preprocessing multimedia data. We call this system an MPEG-aware disk storage system.

The rest of this thesis is organized as follows. Chapter II investigates the present status of research in enhanced storage devices and multimedia services. Chapter III explains the MPEG-aware disk storage architecture and the multiple functions supported by the MPEG-aware disks. Chapter IV presents background information necessary for understanding the implementation details. Chapter V discusses the implementation details. Chapter VI outlines the performance experiments for the prototype MPEG-aware disks. Finally Chapter VII presents summary and conclusions.

CHAPTER II

BACKGROUND AND RELATED WORK

2.1 Video On Demand Service and Its Requirements

Video On Demand service providers are now providing much wider services and opportunities than traditional cable television. Many hotel rooms offer such services as movies on demand, games, and Internet access through Video On Demand server. These services have a competitive price comparing to the video rental, and customers do not need to travel for the services.

User interactivity is another potential advantage of VOD services. Usual user interactive functions involve movie fast-forwarding or rewinding. User interactivity requires preprocessing of MPEG data before it is delivered to the user [2].

Various researchers have suggested adapting the quality of the MPEG stream to fit the available resources. A high quality video stream may be modified into a low bandwidth stream to fit the available bandwidth. Such adaptation requires processing power at the server. Such preprocessing of data is best done at the server so as to utilize the network resources efficiently. This may be the option when customer does not have enough resources.

2.2 Active Disks

It has become feasible, due to the increasing performance and falling price of microprocessors, to move system intelligence into peripherals from the CPU [3]. Decreasing costs of CPUs have resulted in proposals for Active Disks – next generation disk drives that provide an environment for executing application code directly at individual drives. Partitioning processing across clients, servers and storage devices can reduce the amount of data crossing the network and exploit the processing cycles available at the storage device [1, 4]. Specifically, applications that apply simple filters or statistics to stored data or that make disk or network scheduling decisions based on current information at the storage device can improve the efficiency of an application's use of network and host resources, resulting in more scalable and higher performance storage system.

2.3 Digital Video Standards

Moving Picture Experts Group (MPEG) has developed a serious of digital video, audio standards. MPEG I and MPEG II are widely used in network multimedia applications. Besides MPEG, QuickTime from Apple, Media Player from Microsoft, Real Player from Real Networks use different standards and are common in today's multimedia networks.

Digital Video Standards define a standard for both multimedia encoder and decoder.

MPEG-I is a widely used standard, and later versions of MPEG are aiming to improve performance. MPEG II is widely used in DVD industry today. This thesis will focus on MPEG-I formatted multimedia data because of its wide usage in Video On Demand services [5].

Normally multimedia standards involve both video and audio and must deal with three main parts: system standard, video standard and audio standard. The role of system is to synchronize video and audio playback so that video and audio are presented to customers at the time they should be. Video standard deals with how the video is encoded and compressed, and audio standard deals with the audio encoding and compressing.

2.4 Multi-ported Disk Storage System

In the previous work by this research group, a multiported disk storage system was proposed and developed. Traditionally, large disk farms have been deployed by connecting several disks to a single server, Several servers, each with several disks, can be deployed to create a distributed storage system. One problem with this configuration is that a given storage server can become a bottleneck, as all disk activity for a particular server has to be processed by that server. If lots of processing is required on the retrieved data, this can exacerbate the problem. If the data processing occurs at the server, the server becomes even more of a bottleneck. If the data is processed at the client, all of the data has to be sent to the client from the server, even if the processing causes some of the retrieved data to be discarded, as in a SELECT operation on the database. This results in wasted interconnection bandwidth.

In the previous work, an enhanced storage device is developed which is endowed with more processing power and intelligence than the traditional block storage device. Data retrieved from disks are processed at the disk controller lever before sending them to the operating system. It is called *Multiported Storage Device* [6] because it provides several functions at the device controller level. Operating system contacts different ports of the disk controller and asks for different service from the disk controller. For example, data encryption might be port A for multiported disk, and port B might be for compression. When operating system contacts port A with data, the disk controller encrypts the data before writing to the disk and decrypts the data on a need before returning it to the operating system.

The multiported disk device can be efficiently used in a Multimedia disk storage system. Usually multimedia files are very large and occupy large amount of disk space. With different services, sometimes the video server has to store different formatted multimedia data in the disk. Like for different user preference, Real Player's formatted video sometimes has to be coexistence with MPEG formatted video, resulting in tremendous disk capacity overhead. If only one formatted file stored in disk, and be reformatted to other standards at disk controller level, then only one standard file exist in the disk, and the operating system could contact different port of the multiported storage device and get different standard multimedia files.

More over, in order to provide Quality of Services for different customers because of their bandwidth limitations, video server has to store several files for one multimedia purpose, like “T1” format, “56k” format. This results in disk capacity overheads also. Multiported storage device provide video servers with a new method to create different bandwidth fitted files. With one file on disk, different port of the multiported device could provide different bandwidth fitted data to the operating system.

CHAPTER III

MPEG-AWARE DISK STORAGE SYSTEM

We propose an enhanced disk storage system called MPEG-aware disk storage system, which has disk intelligence about MPEG standards and disk processing power that can facilitate in preprocessing of data being transmitted to the client. We follow the idea of multiported storage system and focus on providing pluggable function modules for MPEG-aware storage system. For example, when the interactive user requests fast-forwarding play of the movie, fast-forwarding module is plugged into MPEG-aware storage system, and the operating system will get fast-forwarded data from the disk. Operating system does nothing but transmit the data from the disk to the clients. Different function modules are plugged into a muliported device to provide different services. The operating system contacts the disk at the designed port to get designed service.

Video servers need to preprocess data from disk before sending them to the client for several reasons reasons. Common preprocessing requirements are listed below.

Upon a fast forward request, video servers need to send data at a faster rate to the client. This requires preprocessing of data in the server to change the bit rate and frame rate of the multimedia data.

Customers have different bandwidths for accessing the Internet. Video servers should provide different services to customers based on their available bandwidth. This requires preprocessing of data for propose of adjusting data quality to match the existing bandwidth.

During the play period of the video, the bandwidth to the customer may very due to traffic burstiness in the network. Providing the customers with a fixed data rate determined at the beginning of the service may result in waste of bandwidth or network congestion. Providing adaptive transmission bandwidth service requires preprocessing.

With services to different countries, native language is preferred in each country. In order to satisfy customers from anywhere, video servers have multiple audio components for different languages, and before sending data to client, the preferred language could be determined by the source address. The preferred language combined with video components then can be sent to the client. This requires preprocessing since audio components need to be combined with other components.

Video server could provide different channels to customers, which most standards allow. Customers may want to watch one channel while keeping an eye on another channel. Picture-in-Picture is then necessary for this service. Preprocessing is required since video component need to be reorganized before sending the data to the customer.

Like cable TV, weather reporting or important events could be inserted into video, or the title of the movie. This again requires preprocessing of the data at the server.

CHAPTER IV

IMPLEMENTATION BACKGROUND

4.1 ISO 11172 Standard

ISO stands for International Standard Organization. ISO 11172 is the official name of the MPEG-I standard that is composed of ISO 11172-system, ISO 11172-video and ISO 11172-audio.

4.2 ISO 11172 System

4.2.1 Overview

MPEG system layer has the basic task of combining/decoding one or more audio and video compressed bitstreams into/out of one single bitstream as shown in Figure 1. It defines the data stream syntax that provides for timing control and the interleaving and synchronization of audio and video bitstream [7, 8].

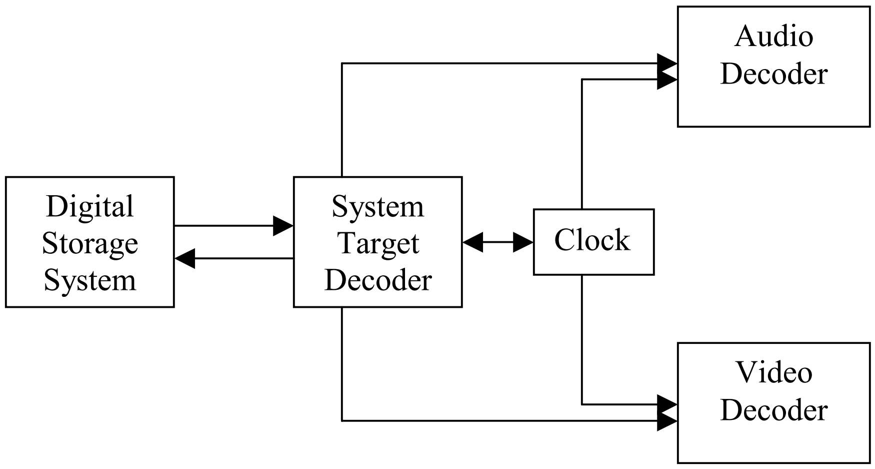
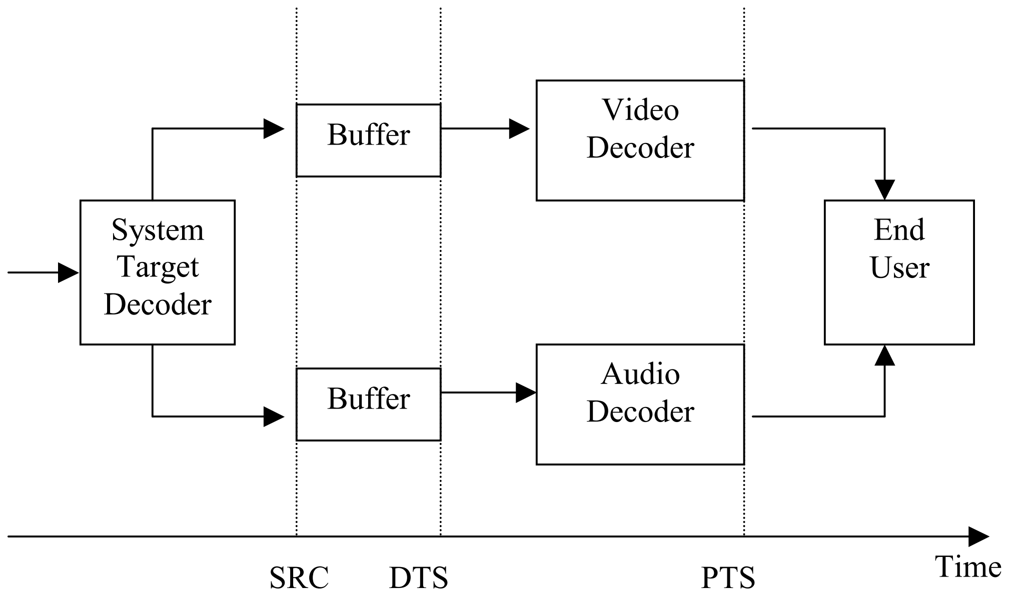


Figure 1. MPEG System Structure

In addition to providing demultiplexing bitstreams, system target decoder also prevents buffer over flow or under flow and provides bitstream synchronization. System_Clock_Reference (SRC), Decoding_Time_Stamp (DTS) and Presentation_Time_stamp (PTS) are timing information embedded in bitstream for system target decoder and video or audio decoders to synchronize streams and manage buffers [7]. There are two buffers involved in each video or audio stream. One buffer is used before the bitstream enters the stream specific decoder, and the other is used inside the stream decoder. Relationship among the time stamps and buffers are illustrated in Figure 2.



SRC: System_Clock_Reference
DTS: Decoding_Time_Stamp
PTS: Presentation_Time_Stamp

Figure 2. Relationship Among Time Stamps

4.2.2 ***Start Code***

The system and video streams contain unique byte aligned 32-bits called *Start Codes* as showed in Table 1 [7]. These start codes all have a 23 zero bits followed by one '1' bit, and the final byte then identifies the particular start code. The byte-aligned property is maintained in video and system streams, and this property is essential for data decoders to correctly decode MPEG bit stream.

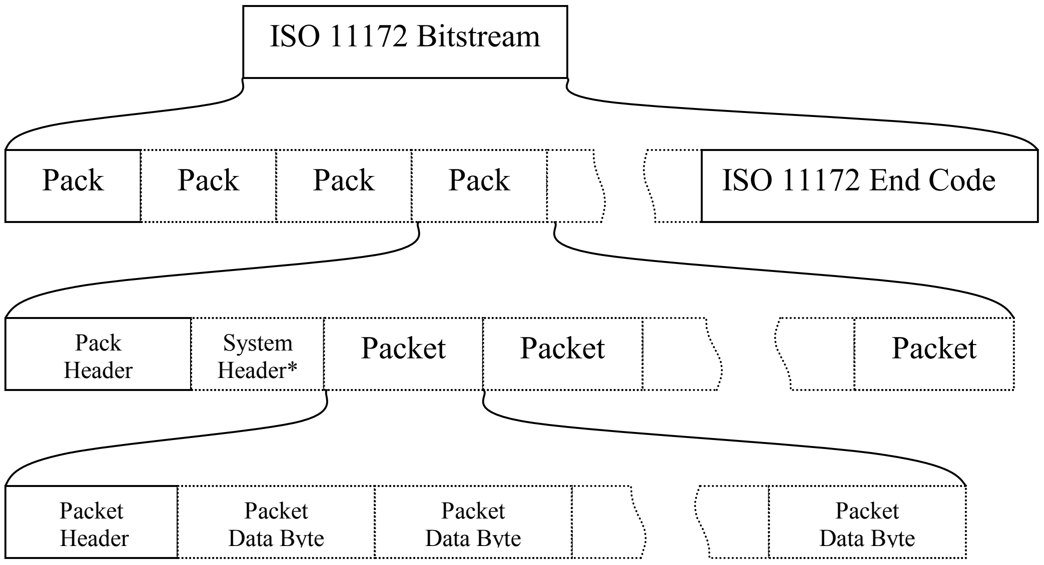
Table 1: MPEG Start Codes in Numeric Order

Start Code Name	Hexa-decimal	Binary
Video start codes:		
Picture start code Slice	00000100	00000000 00000000 00000001 00000000
start code	00000101	00000000 00000000 00000001 00000001
….	….	….
Slice_start_code	000001AF	00000000 00000000 00000001 10101111
Reserved	000001B0	00000000 00000000 00000001 10110000
Reserved	000001B1	00000000 00000000 00000001 10110001
User_data_start_code	000001B2	00000000 00000000 00000001 10110010
Sequence_header_code	000001B3	00000000 00000000 00000001 10110011
Sequence_error_code	000001B4	00000000 00000000 00000001 10110100
Extension_start_code	000001B5	00000000 00000000 00000001 10110101
Reserved	000001B6	00000000 00000000 00000001 10110110
Sequence_end_code	000001B7	00000000 00000000 00000001 10110111
Group_start_code	000001B8	00000000 00000000 00000001 10111000
System start codes:		
Iso_11172_end_code	000001B9	00000000 00000000 00000001 10111001
Pack_start_code	000001BA	00000000 00000000 00000001 10111010
System_header_start_ Code	000001BB	00000000 00000000 00000001 10111011
Packet start codes:		
Reserved stream	000001BC	00000000 00000000 00000001 10111100
Private_stream_1	000001BD	00000000 00000000 00000001 10111101
Padding stream	000001BE	00000000 00000000 00000001 10111110
Private_stream_2	000001BF	00000000 00000000 00000001 10111111
Audio stream 0	000001C0	00000000 00000000 00000001 11000000
….	….	….
Audio stream 31	000001DF	00000000 00000000 00000001 11011111
Video stream 0	000001E0	00000000 00000000 00000001 11100000
….	….	….
Video stream 15	000001EF	00000000 00000000 00000001 11101111
Reserved stream 0	000001F0	00000000 00000000 00000001 11110000
….	….	….
Reserved stream 15	000001FF	00000000 00000000 00000001 11111111

Video start codes identifies a bit stream, and can only be found in a video bit stream. System start codes are used for system syntax. Most system start codes in system syntax are packet start codes that specify the packet.

4.2.3 System Bitstream Overview

Figure 3 shows an overview of ISO 11172 system bitstream [7, 8].



*: System header required in 1st pack

Figure 3. System Bitstream Structure Overview

Pack starts with pack_start_code (0x000001BA). Pack header provides system_clock_reference and mux_rate fields that determine when the stream bytes should enter the system target decoder (STD) and the byte rate at which the bytes should arrive at STD respectively.

System header is required in the first Pack. The system headers in one stream are required to be identical. The system header provides legal constraints on the bit stream. It specifies how many audio and video streams are encoded in the stream. System decoders use data elements in system header to determine system buffer sizes. If a new stream is going to be encoded in the bit stream, an iso_11172_end_code is required before the start of the new bit stream.

A packet starts with packet_start_code. Packet header provides both presentation_time_stamp and decoding_time_stamp. Decoding_time_stamp determines when the packet should leave the system target decoder and enter the stream specific decoder. The presentation_time_stamp determines when the decoded packet should be presented to the end user.

System level data elements are mainly used to synchronize encoded streams and provide enough information for decoders to setup the buffer environment.

4.3 ISO 11172 Video

Figure 4 shows an overview of ISO 11172 video bitstream [7, 9].

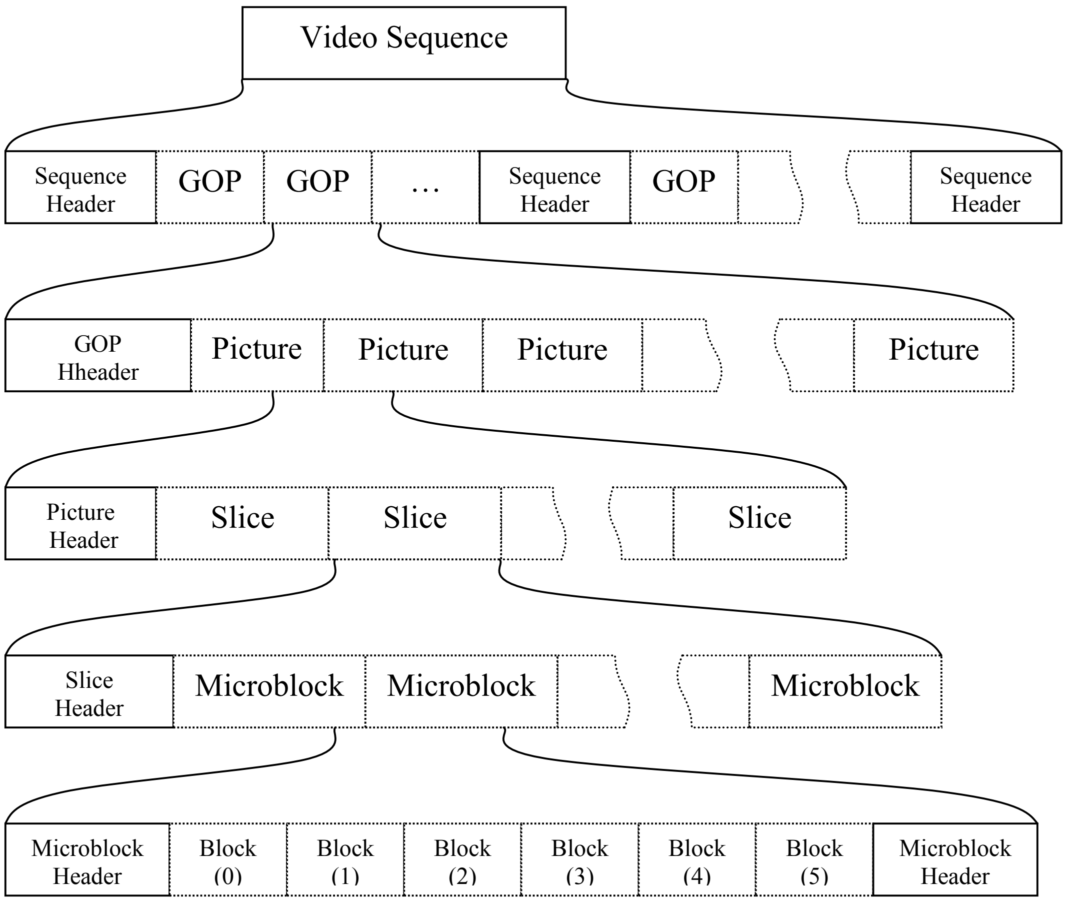


Figure 4. Video Bitstream Structure Overview

A video stream starts with a sequence header. Sequence header has data elements essential for video decoding like picture size, frame rate and quantizer matrix.

Group of Picture (GOP) layer provides timing information for video decoders. GOP header has time code elements specifying the hours-minutes-seconds time interval from the start of the sequence.

Picture header has important data elements like temporal reference and picture_coding_type. Temperal_reference is used to determine the display order of pictures. Picture_coding_type tells the decoder which decoding type should be used to decode this picture (I, P, B or D frame).

Slice layer is the last layer that has byte-aligned start codes. The header of slices has data elements for quantizer_scale field that initialize the quantizer scale factor.

CHAPTER V

IMPLEMEMTATION DETAIL

5.1 Software Structure

MPEG-aware disk storage system should provide the following features to be universally usable by different operating systems.

First of all, the returned block size has to be the default return size of the operating system, otherwise it will be considered an illegal block by most operating systems. Linux requires a 1K return block size with disk reads. This requirement complicates the disk processing that modifies the data length. Because of MPEG standard, and any modification of length of data requires future and past knowledge of data and modification of packet header fields. This means that the disk may have to retrieve more data than default block size, process it and return data of default block size.

Second requirement is fast response. Disk should return data as fast as possible to meet service requirements. In the case of MPEG-aware disk storage system, because of the interactive and video on demand features, data also should be returned to customers as soon as possible. This requirement means that the disk CPU cannot retrieve large amount of data from the disk, process it and return it to the operating system. Instead, disk CPU should return processed data as soon as possible.

Third requirement is the multi-task feature. Disks should be able to handle multiple requests from operating system generated by different applications. There might be several MPEG files on one MPEG-aware disk, and there could be more than one file that needs to be processed at the same time. The MPEG-aware disk CPU has to be shared among the competing requests.

Based on the above requirements, the MPEG-aware storage system is designed as shown in Figure 5.

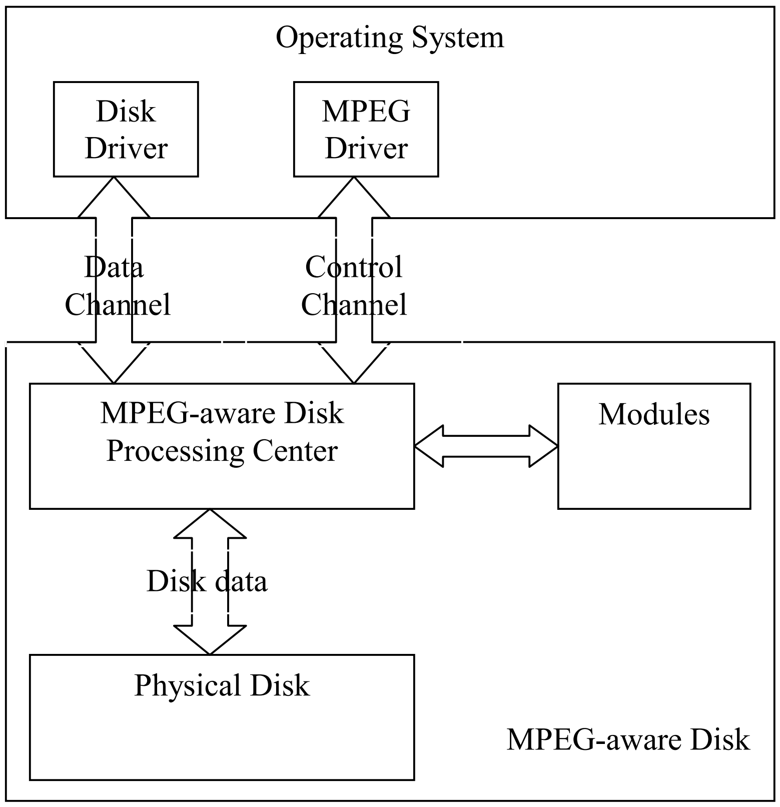


Figure 5. Software Structure of
MPEG-aware Disk Storage System

Operating system needs to load the MPEG-aware module into a multiported disk drive. A control message is issued by the operating system each time there is a data request from the disk driver. Control messages are different when different modules are loaded into the system. When fast-forwarding module is loaded, control message includes the desired frame rate. When Quality of Service module is loaded, common control messages include slice numbers to be deleted, packet length field location and value, and any information for the module to work correctly. In general, control messages include what ever is needed by the module to correctly process the data for this data request. After raw data from the physical disk is processed in processing center, the processed data is serviced to operating system. At the mean time, control messages about this data are provided to operating system, which will be issued back to MPEG-aware disk system by operating system the next time the same application issues data request.

In this design, multiple requests from different applications could be serviced concurrently since the operating system actually holds control messages, which will be sufficient for the next process and will be provided to MPEG-aware disk system when the data request is issued. MPEG-aware disk driver in the operating system is responsible to provide correct control messages to the processing center regarding the different applications.

The MPEG-aware disk-processing center provides timely response and returns constant return size. In addition, it also has a common interface with different function modules to possibly accept third party developed function modules in the future.

Function modules are actually where the raw data from disk is processed. They are designed for each specific need, and will be loaded into the MPEG-aware disk system at the beginning of an application with control messages from the operating system.

5.2 Processing Center

The processing center communicates with the operating system, function modules and the physical disk. It is the heart of the MPEG-aware disk storage system.

In order to minimize the modification of existing operating system, the processing center needs to provide OS with constant size blocks. In addition, to provide timely service, the processing center needs to reply to OS within minimum time after the request is received.

Complex buffer management occurs in the processing center. Two basic buffers holding raw data from physical disk and processed data from function modules are utilized.

After the processing center received a new application request from OS, it reads raw data into its raw data buffer and loads function modules into the MPEG-aware disk system based on the control messages sent from OS. Control messages from OS with the raw data buffer address and the processed data buffer address are sent to function modules. Function modules process the data with its specific function, and fill the processed data buffer with the processed data. Since processing MPEG formatted data may need more or less raw data than the returning size, a fixed size larger than returning size is read into the raw data buffer at the beginning of the application. Following requests from the same application will first invoke the function module plugged in to determine whether the existing data in the raw buffer is enough for this process. If yes, no physical disk access is made for this request, data in the raw data buffers is processed and written into processed data buffer. If no, the processing center retrieves maximum amount of data from disk that the raw data buffer could hold. Since retrieving data from physical disk is a block level operation instead of byte level operation, the raw data buffer is filled in multiples of disk blocks. This is analog of read ahead commonly used in today's storage system and operating system. The function module finishes when the processed data length exceeds the constant return size and when the returning data won't be changed due to future data in the raw data buffer. The function should not return when future processing will modify the existing processed data. The function modules return modified control messages and pointers to raw data buffer and processed data buffer to the processing center when both above requirements are meet. The processing center then provides the processed size data to the operating system through the data channel. At the same time, leftovers in the raw data buffer and processed data buffer from function modules are sent to the operating system through the control channel for future use.

Application requests beyond the first request will invoke the processing center to test if enough processed data is available. If yes, nothing is done except providing the OS with the processed data through data channel. If the processed data is not enough to return this request, a new cycle begins by testing the raw data buffer as described in the above paragraph. This cycle repeats each time a request is issued by the operating system to MPEG-aware disk storage system.

MPEG-aware disk processing center provides timely response and constant size blocks to operating system in order to minimize changes to the operating system. Common interface with function modules establishes the standard for future implementation of new function modules.

5.3 Implemented Modules

5.3.1 Fast-forwarding

The implementation of Fast Forwarding relies on the Frame Rate field of Video Sequence Header.

The Frame Rate field of sequence header is a 4 bits field. This field tells the decoder how fast the pictures should be presented. Because the audio cannot be synchronized with fast-forwarded video, audio packets are dropped.

Table 2 shows typical MPEG frame rates [7, 9].

Table 2. Typical Frame Rates in MPEG Standard

Frame rate	Normal frame rate	Typical applications
0000		Forbidden
0001	23.976	Movies on NTSC broadcast monitors
0010	24	Movies, commercial clips, animation
0011	25	PAL, SECAM, generic 625/50HZ component video
0100	29.97	Broadcast rate NTSC
0101	30	NTSC profession studio, 525/60HZ
0110	50	Noninterlaced PAL/SECAM/625 video
0111	59.94	Noninterlaced broadcast NTSC
1000	60	Noninterlaced studio 525 NTSC rate
1001		
….		Reserved
1111		

There are two levels of frame rates as shown in Table 2 with one level doubling the speed of the other level. In the data processing level, disk CPU scans retrieved raw data for sequence header, and then locates the Frame Rate field, and then doubles the frame rate by modifying the fields. Most commercial MPEG files have an original frame rate of 30 frames per second, and it will be changed to 60 frames per second in the data processing level.

5.3.2 Quality of Service

This module provides video at different level of QoS. We have implemented this by actively changing the number of slices in a video stream.

A picture of frame is composed of multiple slices with each slice going from left to right at a fixed width.

Slices in I frames will never be dropped since other frames depend on I frames to decode the data. Slices in P or B frames are partially dropped. The percentage of slices dropped P, B frames depends on the client’s available bandwidth. When the complete MPEG file could not be sent to client in time, one slice of the B frame is dropped. The position of the dropped slice is rotated in frames in a round robin fashion to minimize the visual QoS impairment. When dropping one slice is not enough to meet the bandwidth requirement, additional slices will be dropped, and so on until all the slices in B frame are dropped. For lower levels of QoS, slices from P frames are dropped similar to B frames.

Audio and Video should also be synchronized when slices are dropped. This requires the modification of audio packets in the stream, and the system level packets. This requirement makes the dropping of slices very complex since there are two levels of packets with one encapsulating the other one. This is like changing the TCP headers provided within IP packets. A straightforward way is to decode the system level packets, change the necessary fields in the video part, and then reencode the system level packets. In other chapters, we show the operation of encoding and reencoding is too expensive to be supported by MPEG-aware disks. Such operations involve CPU-intensive processing and require large amounts of memory.

5.3.3 Implementation Based on Decoding and Reencoding

Several other function modules were implemented based on decoding the data in the raw data buffer, make necessary modifications and then reencode to processed data buffers.

Function modules implemented with this method provides more powerful features. We realized Picture in Picture, Quality of Service based on modifying quatilization table in MPEG encoding and other nice features with this method.

These types of modules require about 1000 times more CPU power than the simple header-modification modules as shown in following the chapters. Existing processors are not powerful enough to encode MPEG in real time. This is true with existing MPEG encoders.

These types of modules are aborted at the test stage due to above reason.

CHAPTER VI

EXPERIMENTATION AND MEASUREMENT

6.1 System Configuration

In order to test the implemented MPEG-aware disk storage system, we tested it on a simulated environment. Linux box with Petium a 166 processor is used to act as the MPEG-aware system. Its CPU works as the MPEG-aware disk CPU and it's hard disk works as the physical disk of MPEG-aware disk. Another linux machine is used as the server. An Ethernet 10 Base T network between the two machines simulates I/O interconnect. Throughputs of simulated MPEG-aware disk are measured at the disk machine to eliminate the possibility of bus being a bottleneck. A third Linux machine simulates the client requesting service from the server.

6.2 Disk Memory Size Effect on Efficiency

The disk has to have certain memory to hold the disk CPU program and data for MPEG processing. Different function modules ma have different demands on memory. Fast forwarding requires less memory than QoS module due to the nature of the processing.

As observed in the results below, memory size has no obvious effect on the disk efficiency. As low as the disk memory is large enough to hold the function module code and the necessary data for processing, maximum processing speed is close to constant with different memory sizes.

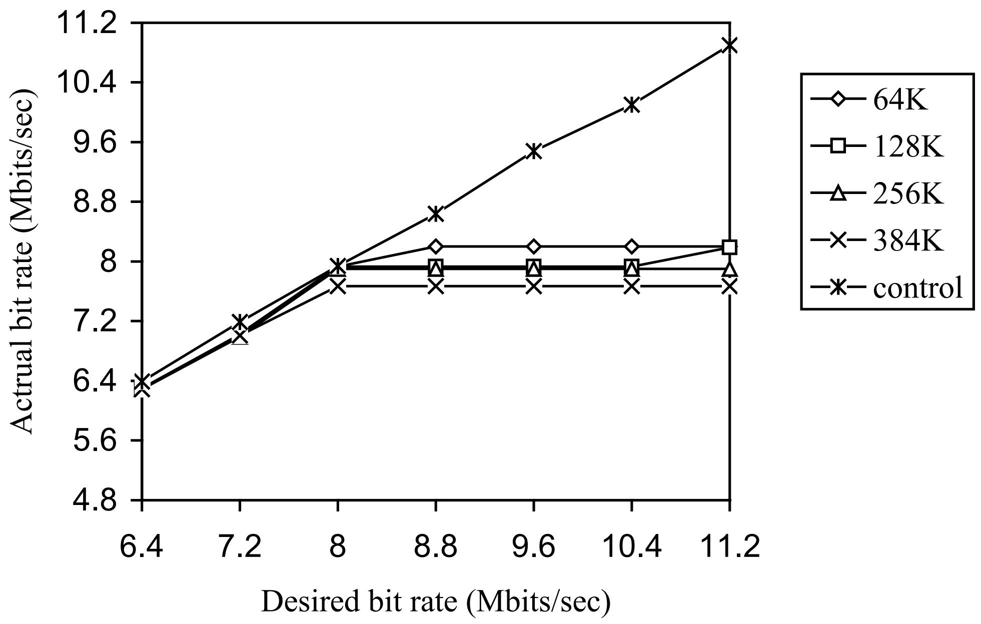


Figure 6. Memory Size Effect On Efficiency With Fast-forwarding Module

In Figure 6, no obvious difference is observed with different memory sizes on the efficiency of MPEG-aware storage system. The maximum throughput of fast-forwarding function is around 8 Mbits/sec per second. Standard bit rate of MPEG files is around 1.6 Mbits/sec, which varies with different screen sizes and other factors. The "ideal" curve is obtained from the MPEG-aware disk storage system with no function module plugged in. For the "ideal" curve, the CPU of the disk does nothing except receives data from disk and provides data to the operating system. When we compare the results to the ideal curve, disk CPU is obviously the bottleneck of MPEG-aware disk storage system.

Disk memory similarly has no obvious effect on efficiency when Quality of Service module is plugged into the MPEG-aware system as show in Figure 7. Memory size has no effect on Quality of Service modules efficiency either. The maximum throughput the disk could handle is much less than fast-forwarding module because the quality of service module involves much more complex processing than fast forwarding.

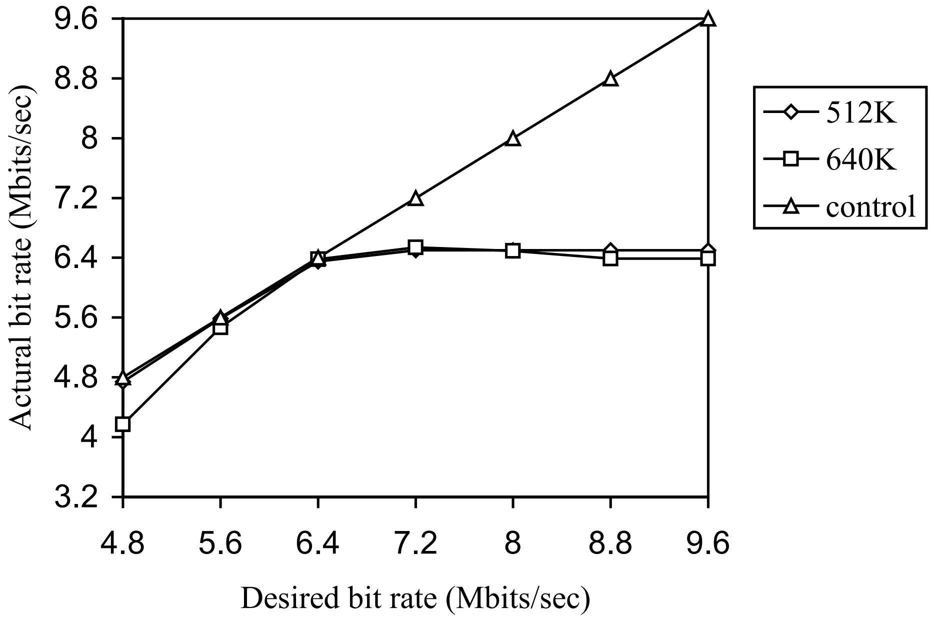


Figure 7. Memory Size Effect On Efficency With QoS Module

6.3 System Throughput

The maximum throughput and standard bit rate of MPEG movies could indicate number of services one MPEG-aware disk could handle at one time.

N = (max throughput)/(standard bit rate)

When fast-forwarding module is plugged in, max throughput is 8 Mbits/sec. Let’s take 1.6Mbits/sec as standard bit rate. When the stream bit rate is 1.6 Mb/sec, this implies that only 5 streams can be supported in parallel. To test this, another experiment is conducted to find the actual number of streams the MPEG-aware disk can handle concurrently.

The throughput of each application is set to 1.6 Mbits/sec. When a small number of applications access the MPEG-aware disk and the disk CPU is not the bottleneck, the throughput of each application is satisfied. When the number of applications exceeds a threshold, disk CPU starts becoming the bottleneck and desired throughputs of applications are evenly impaired.

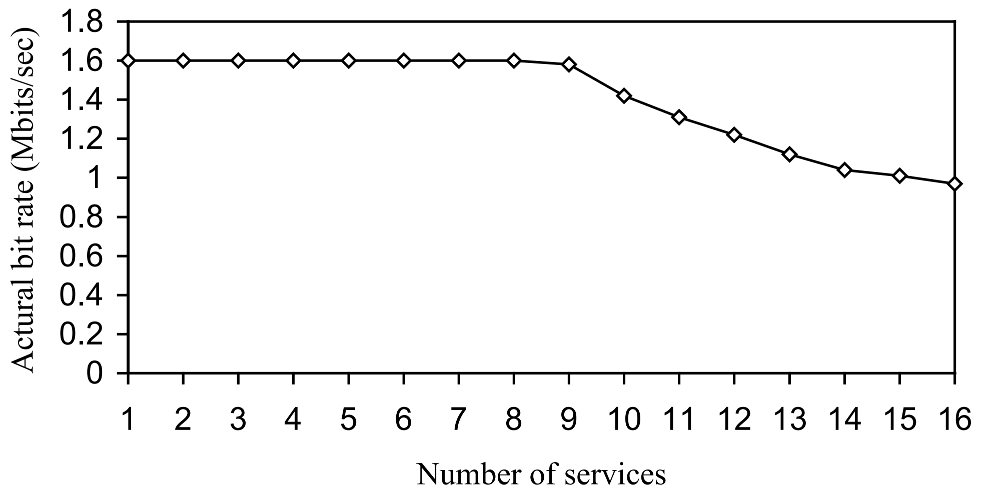


Figure 8. Number of Services With Fast-forwarding Module

Figure 8 shows the realized throughput of each application number of streams is increased. Based on the result, up to 8 streams can be serviced by fast-forwarding module. The difference between the result and theory might be the consequence of interaction between applications in the linux machine that simulates the MPEG-aware disk. After the completion of one request from an application, the processed disk data remains in simulation Linux machine memory. The simulation machine many return the data in memory when next application requests the same block without accessing MPEG-aware disk. This results the increased number of services in the server attached MPEG-aware disk module. In network attached disk module, MPEG-aware disk storage does not as much memory as the simulation machine, so the increased throughput won't be obvious in network attached MPEG-aware disk system.

When the Quality of Service module is used, up to 4 streams could be handled by one MPEG-aware disk at the same time as shown in Figure 9. This result is consistent with the result of maximum throughput simulation throughput obtained in Figure 7. Since the QoS module requires more processing power, smaller number of streams could be concurrently supported than by fast-forwarding module.

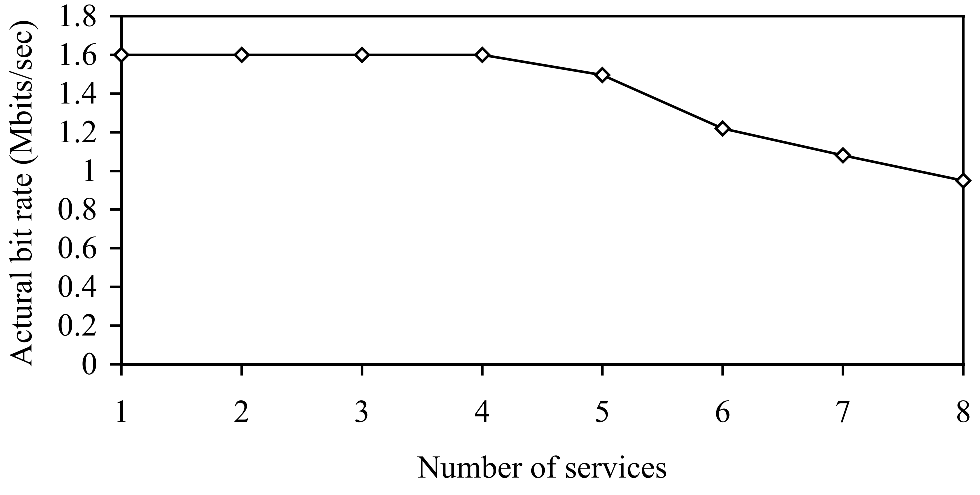


Figure 9. Number of Services With QoS Module

6.4 Modules With Decoding and Reencoding

Maximum throughput was measured with modules that decode and reencode MPEG data. The maximum throughput of those modules falls between 1Kbits/sec and 5Kbits/sec, which is far from acceptable range of MPEG display bit rate.

MPEG decoding and encoding requires high processing power, especially MPEG encoding which is very processing intensive. No existing software MPEG encoder can work in real-time.

CHAPTER VII

SUMMARY

The MPEG-aware disk storage system provides the video server many benefits compared to traditional disks. From performance point of view, it saves server I/O interconnect bandwidth, network bandwidth, video server memory. It also increases the throughput of services the video server could handle and decreases the service delay. From functionality point of view, the MPEG-aware disk enables the video server real-time modification ability of broad casting multimedia data. It provides QoS to end users based on network bandwidth availability, and it provides an interface for future interactive function module implementation.

The limitation of processing power is the bottleneck of MPEG-aware disk storage system because MPEG involves extensive computing. Increasing the processing power in MPEG-aware disk will head to a better performance of the MPEG-aware disk storage system.

Future work on MPEG-aware disk storage system focuses on providing handy function modules for different requires which is easily pluggable into MPEG-aware disk. Improving software structure efficiency is beneficial but not critical because structure involved processing time is negligible when compared to MPEG involved processing time.

REFERENCES

[1] Erik Riedel and Garth Gibson, “Active disks – remote execution for network-attached storage”, *Technical Report CMU-CS-970198*, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, December 1997.

[2] Angela Stratton, “Implementation of video on demand using an MPEG client-server model and ATM networks”, *Kevta Computer Science Department Online Megazine,* vol. 4, no. 4, Department of Computer Science, Brigham Young University, Provo, UT, May 1995. Available: http://lal.cs.byu.edu/ketav/issue_2.5/vod/vod.html

[3] Gang Ma and A. L. Narasimha Reddy, “An evaluation of storage systems based on network-attached disks”, *Technical Report TAMU-ECE-1998-02*, Department of Electrical Engineering, Texas A&M University, College Station, TX, August1998.

[4] Garth A. Gibson, David F. Nagle, Khalil Amiri, Fay W. Chang, Eugene Feinberg, Howard Gobioff, Chen Lee, Berend Ozeceri, Erik Riedel, David Rochberg and Jim Zelenka, “File server scaling with network-attached secure disks”, in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmetrics ’97),* Seattle, WA, June 15-18, 1997. Available: http://www.pdl.cs.cmu.edu/Publications/publications.html

[5] Shanawaz Basith, “MPEG: standards, technology and application”, *Surprice,* Department of Computing, Imperial College of Science, Exhibition Road, London, UK, June 1996. Available: http://www-dse.doc.ic.ac.uk/~nd/surprise_96/journal/vol2/sab/article2.html

[6] Marcus Grande and A.L. Narasimha Reddy, “Multi-ported disk storage”, *Technical Report TAMU-ECE-2000-01*, Department of Electrical Engineering, Texas A&M University, College Station, TX, January 2000.

[7] Joan Mitchell, William Pennebaker, Chad Fogg and Dider Legall, *MPEG video compression standard,* Chapman & Hall, New York, NY, 1997.

[8] ISO/IEC JTC1/SC29/WG11, “Generic coding of moving pictures and associated audio:systems”, *ISO/IEC Draft International Standard 13818 – 1: Systems*, International Standard Organization, Case Postale, CH Geneve, Switzerland, November, 1994.

[9] ISO/IEC JTC1/SC29/WG11, “Generic coding of moving pictures and associated audio:systems”, *ISO/IEC Draft International Standard 13818 – 2: Video*,

International Standard Organization, Case Postale, CH Geneve, Switzerland, November, 1994.