1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Supporting Fast-Forward/Backward in MPEG based VOD Server

*Jin-Uok Kim, *Se-Jin Hwang, *Myong-Soon Park, and **Sung-Kn Jin

*Computer Science & Engineering Dept.
Korea University
Seoul, Korea

** Network Computing Dept.
Electronics Telecommunications Research Institute
Deajon, Korea

Abstract

One of the well known challenges in a Video-on-Demand(VOD) system is to support high-level VCR-like functions, FF/FB(Fast-forward/Backward). Decoding only I-frames within MPEG has been widely used to implement FF/FB.

The requirement of large storage space to separately store I-frames is the main disadvantage of this mechanism. Also, since it ignores the characteristics of MPEG, there is inefficiency in accessing I-frames stored in disks. To alleviate these problems on supporting FF/FB, we introduce new method using *index table*. By using this mechanism, we reduce additional storage space and get better performance on servicing FF/FB

1. Introduction

The recent evolution in multimedia technologies, in terms of computing, storage, and communication, has made it possible to accomplish multimedia service, VOD(Video. on Demand) service, interactive TV, on-line tutorial, and electronic library [2][3][4][8]. Among them, VOD service is attracting increasing attention. VOD service is a technology that sends MPEG(Moving Pictures Expert Group) or other video format data to many users in real time.

One of the key components in the above application is the VOD server that is responsible for the storage and transmission of videos. Depending on the application, a VOD server may be required to store hundreds of videos, and to concurrently transmit data for a few hundred videos to clients. Furthermore, the data for every video must be transmitted at a fixed rate depending on the compression technique employed (for MPEG-1 the rate is about 1.5 Mbps). MPEG is a widely used multimedia stream in VOD server, because inter-frame compression techniques of MPEG provide significant advantages in both storage and transmission. Consequently, it is welcomely accepted for VOD service [10][11].

The main characteristics of VOD service is to provide interaction with contacted users. For providing interactivity in VOD service, it is necessary for users to provide VCR operations, such as FF/FB(Fast-Forward/Backward), normal play, pause, stop and so on.

To support FF/FB, VOD server must service the independent unit of MPEG, I-frame. Sakamoto[5] introduce a method of decoding I-frame in MPEG. In this method, the server services only I-frame when client requests FF/FB operation. It is widely used to support FF/FB in MPEG based VOD server.

Sakamoto's method needs additional storage space to store I-frames and does not consider the size of the I-frame. Henceforth, it has inefficiency on servicing I-frames spanning over more than one request block. For this case, two separate requests should be made to retrieve the I-frame. Thus, we introduce a new method both considering the size of I-frame and using fairly small storage space rather than Sakamoto's. It uses *index table* for support FF/FB operations in VOD server. We lessen the inefficiency in accessing disk by eliminating the two requests to retrieve adjacent two disk blocks. With *index table*, we can also reduce the additional storage space to store I-frames in the disk.

This paper is organized as follows. In section 2, related works to support FF/FB are described. Section 3 addresses the configuration of system model and the method of using *index table* for FF/FB operations. Section 4 compares and evaluates the performance of proposed method. We make a conclusion in section 5

2. Motivation

2.1 FF/FB with MPEG

The MPEG[9] video stream consists of three frame types. The first type is the inter-frame(I). This frame is only encoded with reference to the current frame. The second frame type, the predictive frame(P), is different from the I-frame in that it is encoded with reference to the previous I or P frame. The third frame type, the bi-directional frame(B), is even more complex because it is coded with reference to a past or a future I or P frame. The interrelationships among these frames are shown in Fig 1.

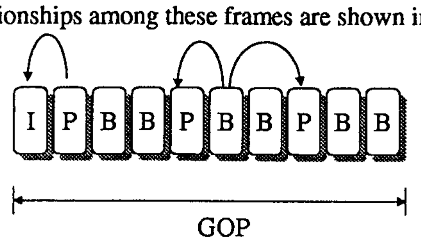

Fig 1 The relationship of MPEG stream, I frame, P frame and B frame

The set of these frames is defined as GOP (Group of Pictures). A GOP must start with an I-frame and may be followed by any number of I, P, or B frames in any order, and begins with a group of picture header, and ends at the next group of pictures header. Therefore, we find that independent of decoding unit of MPEG is both I-frame and GOP.

The inter-frame reference implies that it is not possible to decode a P frame without the proceeding I or P frame. Similarly, it is not possible to decode and playout B frames without the corresponding I and /or P frames.

This is in turn means that it is not possible to playout every third frame of MPEG stream to achieve 3 times the playout when client station requests FF/FB, because this subset would include B frames without the corresponding I or P frames. Therefore, to support FF/FB with MPEG stream, we can consider the reference among three frames, I frame, P frame, and B frame. Sakamoto's approach has been the most renowned mechanism exploiting the structure mentioned above.

2.2 Sakamoto's Approach

Sakamoto *et al.*[5] proposed "Decoding only I frame" to service FF/FB request. This method needs two kinds of storage space per one movie. One is used to store a MPEG stream for normal playout mode, and the other is only I-frame for FF/FB mode. It is widely used approach since Sakamoto's method is inherent simplicity and it is free of the inter-frame dependency of MPEG. There are two problems in this method. First, VOD server sends client the I-frames of MPEG stream when client station requests FF/FB. In the client station, the client decodes only I-frame in real time, and the volume of data (I-frames) sent to client is larger than usual, normal play since the size of I-frames is larger than P and B frames. As a result, more network bandwidth is required during FF/FB period. Second, the VOD server has to store additionally another MPEG stream consisted with I-frames only. The amount of additional space reaches about 15 percent to 30 percent of original MPEG stream. Third, Sakamoto's method sends two requests to retrieve adjacent blocks when it accesses an I-frame spanning over the blocks. However, in general, the size of I-frame is larger than the block of disk, therefore, we had better access an I-frame by its origin size. By doing it, we can reduce both the number of request, and the overhead of retrieval.

The former problem, excessive network bandwidth required, is solved by Park[6] with the cooperation of client's node. The client decodes I-frame repeatedly in several times. Accordingly, I-frame as much as the number of repetition is not needed. However, the rests are still pending problems.

In this paper, we propose the method of using *index-table* for solving in the second and the third problem, Using *index-table*, the server accurately locates the I-frame in disk. The additional storage space is alleviated by indication of accurate position within disks. The third problem is solved with the management of I-frames stored in the disks by not the unit of disk block but the unit of I-frame using *index-table*.

3. Proposed Approach

3.1 System Environment

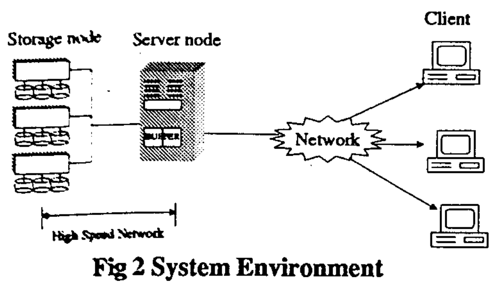

Fig 2 System Environment

There are two major types of parallel VOD servers: shared multiprocessors and distribute memory cluster architectures. In a Multiprocessor system, there is a set of storage nodes, a set of computing nodes, and a shared memory. MPEG stream is sent to the memory buffer through a high-speed network or bus, and then to clients. A mass storage system has presented the capacity of supporting hundreds of requests. However, it is not yet clear that a multiprocessor VOD server can be scalable. A clustered architecture is easy to scale to thousands of server nodes. The network between server node and storage node is high-speed link. In this system, the data transmission time is less than disk read time. Data is retrieved from the storage nodes and sent to the server nodes.

In this paper, we assume a clustered architecture. Such configuration is diagrammed as Fig. 2. The storage nodes, each with a local disk array, are responsible for storing MPEG stream in some storage medium, such as disks. The server nodes are responsible for client's requests. On receiving a request from a client, a server node will schedule it to a time interval. It delivers appropriate data blocks within some time deadline to the client during the playout of a video.

3.2. Stream Placement in Multiple Disks

In this section, we describe how to distribute MPEG streams to multiple disks to support both FF/FB and normal playout. To distribute MPEG stream to several disk is a technique that is normally used to provide very high data throughput through interleaving sequential material across multiple disks. VOD server supports the need that many people view movies concurrently.

We assume that the multimedia stream is divided into blocks. It is called "GOP-block". Each block begins with an I-frame and consecutive B or P frames. The block is the elementary unit of storage and retrieval at normal play mode, and I-frame is elementary unit of retrieval at FF/FB mode.

The stream placement method assumes that blocks are stored in a round-robin manner on disks. We consider a VOD server that contains n disk-set with blocks distributed in a round-robin manner across the disks. Formally, we can represent it as follows:

$$F(b,n) = (b \bmod n)$$

Where b is a block and n is the number of disk-set.

An example of the block placement for $n = 4$ is given in Fig. 3. As shown in Fig.3, the fifth block of stored MPEG stream is located in disk-set1.

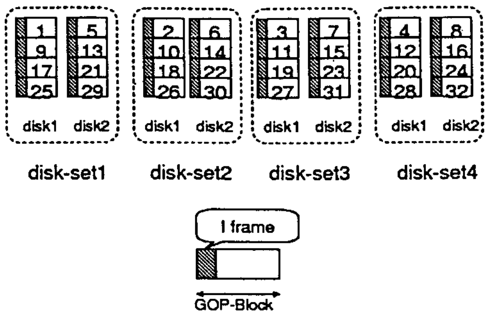

Fig 3 GOP block placement layout

In the previous section, we refer to the system environment and it has storage node for storing MPEG stream and retrieving requested MPEG stream. Each storage node has a disk-set. In normal playout mode, client requests a GOP-block for playout that data. After server node receives client's request, it requests the requested GOP-block to target storage node. The storage node knows the next request GOP-block of client since MPEG stream is continuous and client requests GOP-block one by one. Therefore the storage node prefetchs next GOP-block in memory and, at the same time, send requested GOP-block from storage mode's memory to server node through high speed network.

In FF mode, the retrieval process is same as normal playout mode. But, in this mode, requested blocks are only I-frames. This I-frame access method is explained in next section.

The advantage of this prefetching scheme in storage node is to overlap the time between sending time of one block to server node and retrieval time of block from disk.

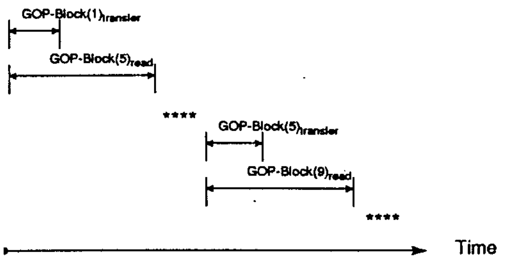

Fig4 The flow between sending time and retrieval time in Disk-set1

Figure 4 shows that, in a disk-set1, transfer time of requested I-frame, GOP-Block(1), of FF mode and disk read time of next I-frame, GOP-Block(5), is overlapped. Transfer time of i-th is less than disk read time of $i+1$-th I-frame because transfer time of one block is faster than disk read time of one block with high-speed network.

3.3. I-frame Access Method

In this section, we describe how to access I-frame to support FF mode and how to support FF/FB functions based on the stream placement method.

3.3.1 Index table

When a client requests a FF/FB operation, the server sends I-frame of each GOP block in Fig 2. To access each I-frame, we use a set of location information in disks. This information is called *"index table"*. This *index table* supports both the interactive serviceability of FF/FB functions, and the prompt switching between FF/FB and normal playout.

The *index table* has two sub-indices. One is an index for playout mode, and the other for FF/FB mode. The sub-index for play mode consists of two fields, Block-offset and Block-size. Block-offset is the location of selected block in disks, and the block-size is the size of the block. The other index for FF mode also consists of two fields both I-offset and I-size. I-offset is the location of I-frame in selected block from disks, and I-size is the size of that I-frame. In the *index table*, the value of Block-offset and I-offset is not equal. The reason is as follows: One GOP block has the headers of GOP (Group of Pictures), and one block has headers in this order, GOP header , I-frame header, and other frame headers such as P frame headers or B frame headers.

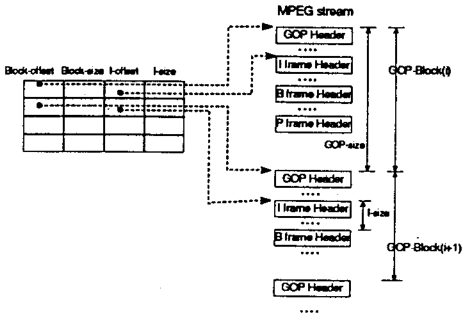

Fig 5 Mapping relation between *index table* and MPEG stream

Fig 5 illustrates the mapping relation between *index table* and MPEG stream in disks. We assumed that MPEG stream consists of consecutive GOP blocks. Each GOP has one I-frame and other frames such as P frames and B frames. Block–offset indicates the GOP header of the block. Meanwhile, I-offset indicates the header of I-frame of the block.

3.3.2 Supporting FF/FB functions

Using the *index table* in I-frame access method, the VOD server can service FF/FB request. We explain how to do it.

- Fast-Forward(FF):

In the FF mode, the VOD server retrieves I-frames using *index table*. The procedure of servicing FF is as follows,

1) When clients request FF, the VOD server looks up the location of current block to be played in the *index table*.
2) Then, the server selects the corresponding location of I-frame closest to the block.
3) Through the selected information both I-offset and I size, VOD server request I-frame to disks
4) The VOD server requests the I-frames in ascending order in *index table*.
5) The VOD server aggregate I-frames, and send the I-frames to clients

- Fast-Backward(FB):

In the FB mode, the procedure of servicing FB is as follows,

1) When VOD server catches up FB request, the procedure of looking up *index table* is the same that of FF.
2) Through the selected information both I-offset and I-size, VOD server requests I-frames to disks.
3) VOD server requests the I-frames in counter direction to the case of FF
4) Then, VOD server aggregates the received I-frames, and sends the I-frames to clients.

4. Evaluation

In this section, we evaluate the performance of both the proposed method and Sakamoto's. For examining the practicality of our methodology, we evaluated the jitter delay of the two methods in the case of bursty disk status based on client-server model.

4.1 Environment

4.1.1 Video selection

Using the proposed method, we simulated the arrival of video request by a uniform process. Assume that there are M different videos available in the VOD server. Upon arrival in the VOD server, a client selects in order to watch the *i*th video with probability $P_i = 1/M$, $1 \leq i \leq M$, we also assume that the request probabilities are steady through the simulation.

4.1.2 Disk model

We assume that VOD server has 6 disks and disk parameters used in the simulation is Atlas XP34300[7]. Atlas is fabricated by Quantum, 5Gbyte SCSI hard disk, widely used for multimedia applications. Table 1 shows the disk parameters of the Atlas. We assume that all disks' capability is same and disk queue works in FIFO(first In First Out) manner, with FCFS(First Come First Serve) disk scheduling.

Table 1 Disk parameters

Average random seek(read)	8 ms
Maxmun seek time	19 ms
Rotaional delay	8.3 ms
Heads / Drive	20
Tracks / Surface	3820
Sectors / Track	137
A sector size	512 Byte

4.1.3 Simulator

To compare proposed method with Sakamoto's method in same environment, we implemented simulator. The simulator is written in SMPL[1], and consists of request generator, event control loop, SCSI interface, and disk drive module. The composition of simulator is shown in Fig.6. The simulator periodically generates request in request generator, and the request enters event control loop. All event generated by simulator have transition to next event and is controlled by event control loop. The request generated by request generator is transferred to disk drive module. Through this mechanism, request response time is evaluated. We assume that the number of total requests is 10000 in this simulator.

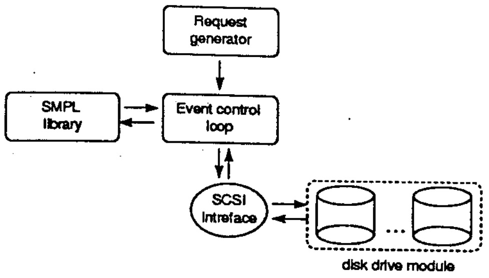

Fig 6 The configuration of simulator

4.2 Result

Our methodology moves an disk arm to the specific location within the disk depending on the value of I-size field of the *index table*, therefore the substantial amount of extra time should be invested to manipulate the disk arm. While, Sakamotos' can exploit the linear disk scanning on serving FF/FB by retrieving separate consecutive I-frames. Here, we can find the difference between two methods. Therefore, we evaluated the jitter delay of the two methods in the case of bursty disk status based on client-server model.

In Fig. 7~9, The X-axis is number of users. The Y-axis indicates the number of packets responded beyond the required time deadline.

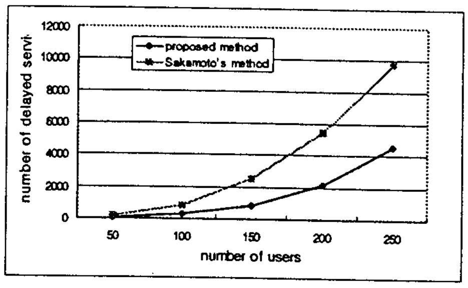

Fig.7 Delayed service vs number of users(FF:50%, normal play:50%)

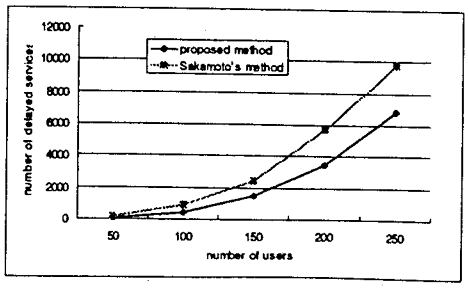

Fig.8 Delayed service vs number of users(FF:30%, normal play:70%)

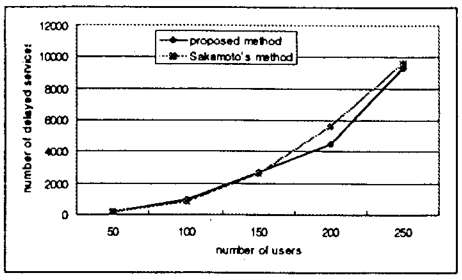

Fig.9 Delayed service vs number of users(FF:0%, normal play:100%)

The general trends of client preferences are skewed to normal playout rather than FF/FB. Therefore, the ratio of normal FF mode to playout mode from 50-50 to 0-100 is usual cases in commercial VOD server. In Fig 7, it shows that the ratio of FF/FB mode to play mode is 50 -50. According as users increase number of delayed services increase at the Fig 7. And it shows that proposed method is less number of delayed services.

In Fig 8, it shows that the ratio of FF/FB mode to play mode is 30-70. And in Fig 9, it shows that the ratio of FF/FB mode to play mode is 0 -100. According as the ratio of normal play mode is increased, the difference of delayed services between proposed method and Sakamoto's method is all much the same.

We compared the size of additional storage space needed by proposed method as well as Sakamoto's method above mentioned in section 2. Because proposed method is equal to Sakamoto's method in that it is playout I-frame when client requests FF/FB functions. However, proposed method is to resolve the disadvantages of Sakamoto's method in respect of additional storage space.

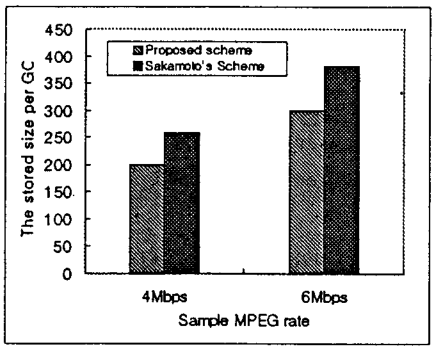

Fig.10 The stored size per GOP with sample MPEG rate

Fig.10 shows the space of storage resource needed to store one GOP. The proposed scheme store only one GOP, but Sakamoto's scheme store both one GOP to support normal playout and one I-frame to support FF. In 4Mbps encoding rate with MPEG-2, proposed scheme need 199.8 Kbytes, but Sakamoto's scheme need 258.1 Kbytes of storage space per GOP. Therefore, our strategy is highly superior to the Sakamoto's scheme since it does not need extra storage space to separately store I-frames.

5. Conclusion

In this paper, we have explored the stream placement method and I-frame access method to provide interactive VCR-like functions for MPEG stream.

Our stream placement method for all disks stored MPEG stream to service FF/FB divide original MPEG stream into each blocks, and I-frame access method service client's request interactively. The proposed method has *index table* to support rapidly accessing the location of I-frames when client requests FF/FB function. Through this mechanism, the proposed method takes a great advantage in respect of storage space rather than Sakamoto's method. Together with it, The access time of our mechanism to support FF/FB has been also evaluated and compared with Sakamoto's way. Better performance was shown in the case of mixed both normal playout requests and FF/FB requests.

Reference

[1] M.H. MaxDougall, "Simulating Computer Systems Techniques and Tools," The MIT Press, Cambridge, Massachusetts, 1987.

[2] P. K. Andleigh, K. Thakrar, "Multimedia System Design," Prentice Hal PTR, Upper Saddle River, 1996.

[3] D. Deloddere, W. Verviest, "Interactive Video on Demand," IEEE Communication Magazine, May 1194.

[4] P.Rangan, H.Vin, and S. Ramanathan, "Designing an On-Demand Multimedia Service," IEEE Communication Magazine, Vol.30, July 1992.

[5] H. Sakamoto, A. Uemori, H. Sugiyama, and K. Nishimura "Video Server Architecture Supporting Real-Time Input and Immediate Playback," In the Proc. of the Multimedia Japan 96, pp.224-231, March 1996.

[6] H. K. Park, and H.J. Cha, "An Efficient Implementation of Fast Forward/Reverse Functions for VOD Service," In the Proc. of the Transactions of the Korea Information Processing Society, pp.280-284, 1996.

[7] Atlas XP 34300 SCSI Hard Disk Drive Product Brief, Oct., 1994.

[8] R.S, Rangan PV, "Architecture for Personalized Multimedia," IEEE Multimedia, 1:37-46, 1994.

[9] ISO, "Coding of moving pictures and associated audio- for digital storage media up to 1.5 Mbit/s," ISO standard IS 11172.

[10] L.A. Rowe, D.A. Berger, and J.E. Baldeschwieler, "The Berkeley Distributed Video-on-Demand System," Multimedia Computing Proc. of the Sixth NEC Research Symposium, Ed. T. Ishiguro, SIAM, 1996.

[11] H. Tezuka, T. Nakajima, "Simple Continuous Media Storage Server on Real-Time Mach," USENIX Technical Conf., San Diego, CA, 1996.