Video Server with Tertiary Storage

Hojung Cha, Jongmin Lee, Jaehak Oh, †Rhan Ha
Dept. of Computer Science, Kwangwoon University, Seoul 139-701, Korea
†Dept. of Computer Engineering, Hong-Ik University, Seoul 121-791, Korea

Abstract

This paper discusses the design and implementation of a video server which uses tertiary device as a source of media archiving. In order to handle the tertiary device in the framework of disk-based stream service model, an effective disk caching mechanism is devised together with stream scheduling and admission control mechanisms. The proposed system has been implemented on a general-purpose operating system and its design principles are validated the with real experiment results. The results guide us how streaming servers with tertiary storage should be configured in real environments.

1 Introduction

Most of the video-on-demand servers currently in use or under development uses large capacity harddisks as its primary content repository. The harddisk-based VOD server provides a fast and reliable content access. However, VOD service usually requires a large amount of disk space, hundreds of gigabytes or even petabytes, for the contents and therefore the space requirement is in constant demand as multimedia contents are in widespread[1]. One approach to support the massive storage requirement of VOD server is to use tertiary storage - such as tape library or magneto-optical jukebox. A tertiary storage device typically consists of drives, cartridges and media exchanger. It has a very large storage capacity, but the media access time is significantly reduced due to the excessive media exchange time.

Supplementing a harddisk-based video server with tertiary storage devices requires many technical challenges[2, 3]. The handling of media contents should employ the hierarchical storage management concept. Not all contents can directly be streamed from the tertiary device and therefore all or parts of the frequently-accessed contents should be cached in harddisk. In this process, an efficient media staging mechanism which satisfies the timing constraints of continuous media is needed as well as the effective harddisk management mechanism for the cached media. The addition of tertiary device in a media storage hierarchy makes the scheduling and admission control policies of video server complicated. The scheduling mechanism for media staging should be integrated appropriately into the disk-based stream scheduling principle. The criteria of a new stream admission should also be

decided upon the operating and management principles of secondary and tertiary storage devices. Some research results are found in the literature regarding the use of tertiary device for media service. The works are mainly focused on media placement[4, 5, 6], media fetching[5, 6, 7], I/O scheduling[8, 9], and media caching[6, 10, 11] strategies. Most of these works are approached and validated by simulations and it is rare to find works based on real implementations.

This paper presents the design and implementation of a MO-jukebox based video streaming server. The server uses harddisk as a cache for staging media from the jukebox. We discuss the server structure, stream scheduling algorithm, disk caching mechanism and admission control algorithm. The proposed system has actually been implemented on Windows platform and its performance is analyzed with the experimental results. The paper is structured as follows. Section 2 describes the system structure and details the stream scheduling, disk caching and admission control policies. Section 3 discusses the experimental results. Section 4 concludes the paper.

2 System Structure

This section discusses the system structure of the proposed jukebox-based streaming server and details the stream staging and scheduling, disk caching and admission control policies.

2.1 Server Overview

There are a few design considerations when including tertiary device into the storage hierarchy for real-time streaming service. Due to the mechanical movement of media changer, the media access time is slower in several order than that of harddisk. A streaming server should service multiple streams periodically and concurrently. This requires the frequent exchange of media in tertiary device. Therefore, in order to support many concurrent streams efficiently, a systematic management of storage components is necessary and it includes an efficient I/O scheduling which reduces the media exchange time.

The storage hierarchy for video service consists of three layers: main memory, harddisk and tertiary storage. A conventional streaming server requires only main memory and harddisk with which media is periodically read from harddisk, buffered into main memory and then transmitted to network. In a video server with tertiary storage, however, the harddisk plays an important role of stream caching. As the access bandwidth of tertiary storage is lower than that of harddisk, the media stream should be pre-fetched from tertiary device and buffered into harddisk in a timely manner. In this process, the harddisk may store a complete or a part of media. Since the capacity of harddisk is limited, an efficient disk cache management mechanism is necessary and its implementation should consider the real-time characteristics of continuous media.

Figure 1 shows the structure of server components. The server consists of three main components: Service and Process Management Subsystem, I/O Management Subsystem and Resource Management Subsystem. The service and process management subsystem listens to end-user service requests, controls the service admission and schedules the system threads to deliver the requested media in real-time. The I/O management subsystem

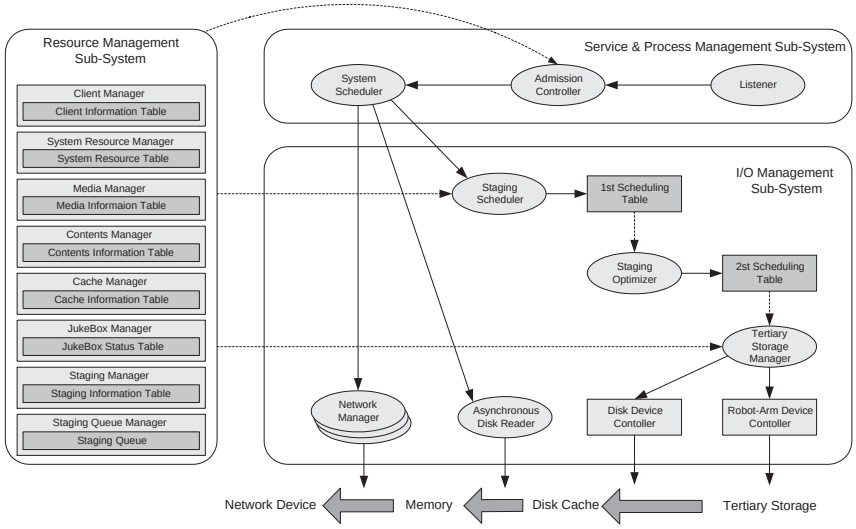

Figure 1: Server architecture

provides a device-driver level control of tertiary device. It is also in charge of I/O scheduling for the efficient data transfer between tertiary and harddisk. The staging scheduler and staging optimizer are the key components. Staging means the process of data movement from tertiary to harddisk cache. The staging scheduler decides the order of tertiary data access based on the parameters such as media rate, device bandwidth, amount of cached data. The staging optimizer enhances the utilization of tertiary device bandwidth. The resource management subsystem consists of several managers for system-wide resources and each manager maintains its own information or resource table.

A user's service request is processed as follows. The listener in the service and process management subsystem detects a new service request. The admission of this request is decided by the admission controller. The decision is based on the availability of system resources which is constantly maintained by the resource management subsystem. Once admitted, the request updates the resource usage and is ready for stream scheduling. The system scheduler which controls both the disk access and tertiary access periodically schedules the stream request to meet its timing constraints. The actual access of tertiary device is done by the tertiary storage manager which controls the media exchanger and disk drives. This way, media in tertiary device is read into disk cache by the scheduler and then transferred to network via system memory.

2.2 Staging Scheduling

The system scheduler is an important component for the reliable and guaranteed delivery of continuous media. It consists of two modules: process scheduler and staging scheduler. Both schedulers are running periodically based on the EDF principle. The process scheduler determines the execution order of disk and network accesses according to the degree of service emergency. The staging scheduler determines the priority of tertiary access based on media rate, device bandwidth, amount of cached data.

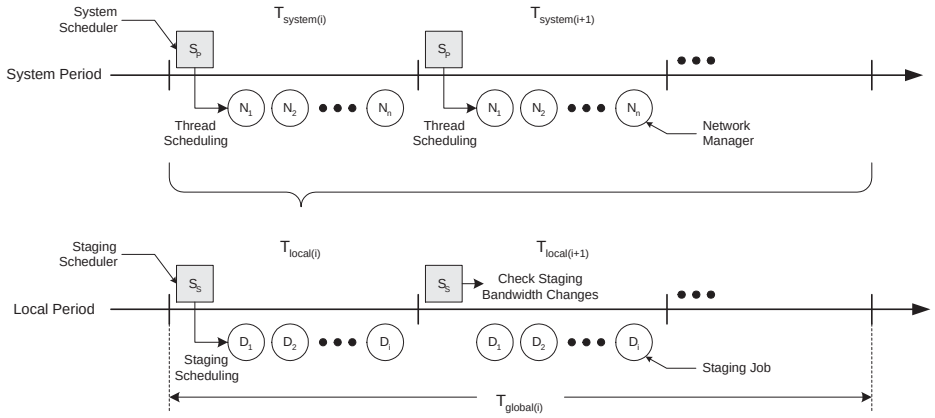

Figure 2: Concept of stream scheduling

The staging period (or local period), T_{local}, is set as a multiple of process scheduling period (or system period), T_{system} and it is calculated as in Equation 1. It is decided by the consuming time of a fixed-size cache block ($S_{cacheblock}$) by the most demanding stream rate ($\max(R_1, R_2, \cdots, R_n)$) allowed by the server.

$$T_{local} = \left\lfloor \frac{\frac{S_{cacheblock}}{\max(R_1,R_2,\cdots,R_n)}}{T_{system}} \right\rfloor \times T_{system} \quad (1)$$

The tertiary staging occurs when the media asked by the current service request is not cached on disk or the amount of cached media is not enough. The degree of staging emergency is decided by the amount of media already cached in disk - that is, by its playing time ($T_{cached(i)} = \frac{S_{cached(i)}}{R_i}$). The staging rate of each media access is determined by Equation 2 which considers both the media rate and the amount of data already cached in disk. The staging may act on each period or once in a multiple of staging periods. Here, the staging size is fixed.

$$R_{staging(i)} = \frac{S_{caching(i)} \times R_i}{S_{cached(i)} + S_{caching(i)}} \quad (2)$$

Figure 2 illustrates the concept of process scheduling and staging scheduling.

2.3 Disk Caching

The harddisk plays a key role in streaming media from tertiary device. It provides a buffer space to mediate the access rate of harddisk and tertiary device. Also, as the capacity of harddisk is limited, the harddisk should be used as a big cache and therefore an efficient management of disk cache is necessary. The cache management for continuous media is inherently different from that of conventional objects such as text or images. The conventional method focuses on increasing the cache-hit ratio whereas, for continuous media, the caching structure and its replacement policy should consider the sequentiality of cached media.

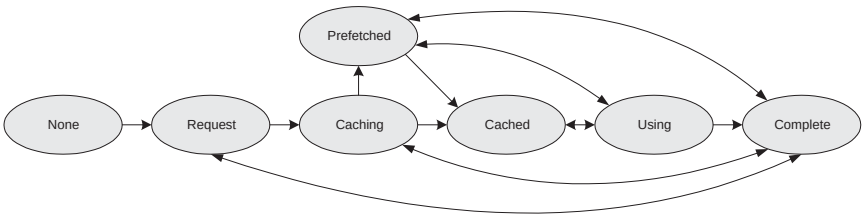

Figure 3: State diagram of cache block

The disk cache consists of fixed-size cache blocks and its size is determined by the media rate and the tertiary device bandwidth. When a stream requests media, the requested portion of media should be placed on disk cache. In order to prevent the initial cache miss, a certain amount of media is pre-loaded for each video in tertiary storage. The size of pre-loaded media is a multiple of cache block and it is never replaced by cache replacement policy. Figure 3 shows the state change of a cache block during its life cycle. A cache block is in one of the following seven states: NONE, REQUEST, CACHING, PREFETCHED, CACHED, USING, COMPLETE.

The cache replacement policy is based on several criteria. First, the cache blocks of active streams (the streams currently under service) are not replaced. That is, only the cache blocks occupied by inactive stream are the target for replacement. Second, a full-cached video is not replaced. This is to reflect the fact that a full-cached video has a high possibility of being requested again. Third, the replacement of cached blocks is done in reverse order. This is to guarantee the sequentiality of media even in disk cache. Based on theses principles, the victim block is decided by calculating the *cache distance* for each inactive video (Equation 3) in diskcache.

$$T_{cachedistance(i)} = \frac{S_{cached(i)}}{R_i} \tag{3}$$

By selecting the video whose cache distance is the biggest, the aggregate bandwidth requirement of tertiary device is minimized. Meanwhile, if there is no inactive blocks, the full-cached videos are then considered for replacement. Figure 4 details the cache replacement algorithm.

2.4 Admission Control

In order to provide a guaranteed service for each admitted stream, a new stream request should be evaluated for its admission. The admission control determines the admission based on the current resource availability of the server. There are multiple criteria for the resource availability: disk bandwidth, network bandwidth, amount of cached media and staging bandwidth.

$$M_{system} \geq \sum_{i=1}^{n} M_i + M_{(i+1)} \qquad \min(B_{disk}, B_{net}) \geq \sum_{i=1}^{n} R_i + R_{(i+1)} \tag{4}$$

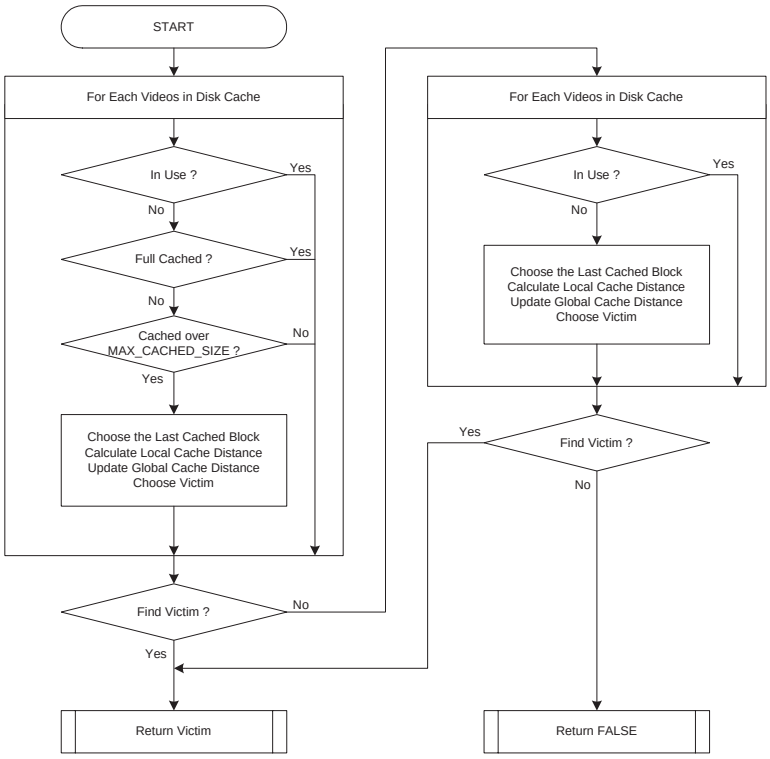

Figure 4: Cache replacement algorithm

$$N_{complete} \geq \sum_{i=1}^{n} N_{caching(i)} + N_{caching(i+1)} \tag{5}$$

$$B_{tertiary} \geq \sum_{i=1}^{n} R_{staging(i)} + R_{staging(i+1)} \tag{6}$$

Equation 4 limits that the admission of a new stream should not exceed the total system memory (M_{system}). It also regulates that the additional requirements of disk and network bandwidth should meet the streaming requirement ($min(B_{disk}, B_{net})$). Equation 5 checks the availability of disk cache space for a new stream. The number of cache blocks which can currently be replaced by the replacement policy ($N_{complete}$) should be equal or bigger than the sum of the number of cache blocks required for a new stream ($N_{caching(i+1)}$) and the number of cache blocks that the current streams will need to their completions ($\sum_{i=1}^{n} N_{caching(i)}$). Equation 6 checks the availability of tertiary bandwidth for a new stream. The sum of the staging bandwidth requirement of a new stream ($R_{staging(i+1)}$) and the aggregate bandwidth of currently active streams ($\sum_{i=1}^{n} R_{staging(i)}$) should be less than the maximum staging bandwidth ($B_{tertiary}$).

3 Experiments and Results

The server is implemented on a general-purpose operating system with an actual tertiary device. Based on the experimental results, this section discusses the performance and characteristics of the system components.

3.1 Experimental Setups

The server has been developed on a Windows 2000 platform which is equipped with a HP Magneto-Optical jukebox (HP-80fx). HP-80fx has 16 slots, one drive and one media exchanger. The jukebox control is implemented with the RSM (Removable Storage Management) API which is provided with Windows 2000. The video data used in the experiments are MPEG-1 encoded media.

The key objective of the experiments is to evaluate the performance characteristics of the implemented server experimentally. In particular, the server scheduling policy, the disk cache management policy and the admission control policy are validated with real system runs. We have setup an environment where a number of streams concurrently access the videos in tertiary storage and the video access pattern follows the well-known Zipf distribution[12]. There are 30 videos in tertiary storage and they are constantly requested in one minute interval. Three different θ values of Zipf distribution (0.0, 0.5, 1.0) are used to emulate the video access pattern. Figure 5(a) shows the access distribution used in the experiment. The access pattern is highly skewed by Zipf(0.0) whereas Zipf(0.0) induces an even distribution.

3.2 Results Analysis

Several aspects of the server performance have been evaluated. We have traced the number of active streams, the tertiary bandwidth, and the deadlines. With all these experiments two different size of disk cache (6GB and 12GB) are used to evaluate the effect of cache size,

Figure 5(b) shows the trace of the number of admitted streams for two access distributions and two cache sizes. It is apparent that for the highly skewed access pattern (Zipf(0.0)) the number of admitted streams is larger than Zipf(1.0). This is due to the fact there is high possibility of cache hit for the popular videos. The figure also shows that the number of admitted streams is highly dependent on the size of disk cache.

Table 1 details the performance statistics of the 3 hour experiment runs. It shows the admission ratio as well as the detailed reasons for the admit denials. The admission ratio increases both with the disk cache size and the skewness of access distribution. An interesting fact is the admission ratio is sensitive to the access distribution. For instance, the performance difference between Zipf(0.5) and Zipf(0.0) is trivial for both cache sizes, but the admission ratio sharply increases as θ approaches 0. In fact, the distribution of admission is similar to the video access distribution used in the experiment (Figure 5(a)). The table also shows the reason for admission denials. There are three factors for denial: lack of staging bandwidth (Equation 6), disk caching capacity (Equation 5) and lack of system resource (Equation 4). When the disk cache is 6GB, the admission denial is mainly caused the caching capacity and the lack of staging bandwidth. The lack of staging bandwidth

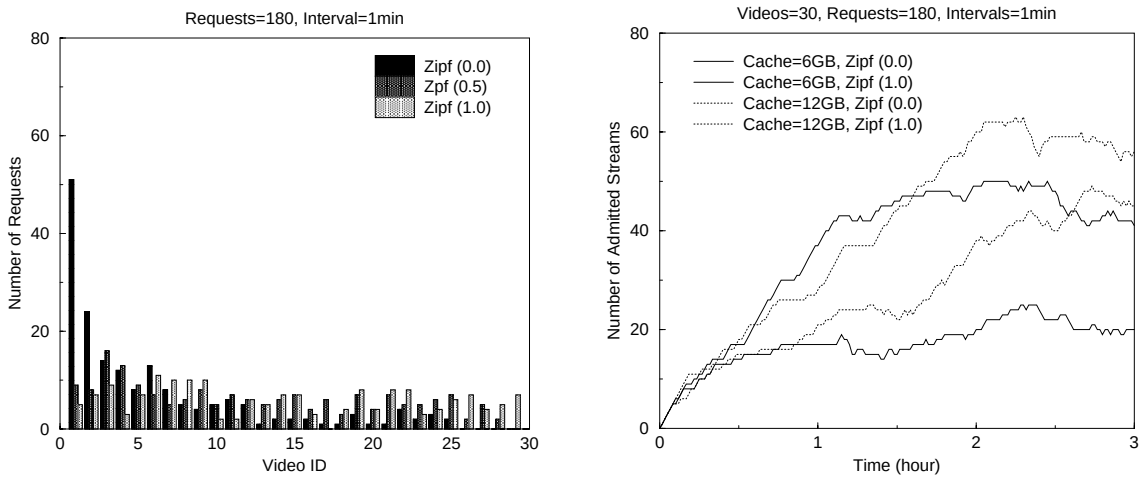

Figure 5: (a) Popularity vs. Video ID; (b) Admitted streams

Table 1: Performance statistics

Zipf (θ)	Cache Size	Total Requests	Total Admits	Admit Ratio	Total Denies	Staging B/W Limit	Cache Sp. Limit	System Res. Limit
0.0	6 GB	180	114	63.3 %	66	16	50	0
	12 GB	180	134	74.4 %	46	33	0	13
0.5	6 GB	180	60	33.3 %	120	13	107	0
	12 GB	180	96	53.3 %	84	82	2	0
1.0	6 GB	180	54	30.0 %	126	17	109	0
	12 GB	180	92	51.1 %	88	82	6	0

effects the admission in the initial stage of service. That is, since the pre-loaded media segments are relatively small, there is little time left for each media to access media from tertiary device and this consequently overloads the tertiary access. However, as the caching process continues the primary cause of admission denial moves to the limited size of disk cache. With 12GB disk cache, the main cause of admission denial is found to be the lack of staging bandwidth. In this case, the size of disk cache is big enough to satisfy Equation 5.

Figure 6 shows the traces of available staging bandwidth for Zipf(0.0) and Zipf(1.0). The graphs drop sharply in the beginning and rise appropriately according to other factors. Note that the reason for the initial decrease of the available staging bandwidth is explained above. With 6GB cache, the available staging bandwidths increase sharply for both Zipf(0.0) and Zipf(1.0). This is because the streams requesting new videos cannot be admitted to the system due to the lack of caching space (Equation 5). With 12GB, however, the available staging bandwidth increases gradually because of the high cache-hit ratio (Zipf(0.0)). For Zipf(1.0), the bandwidth availability stays low due to the low cache-hit ratio. These results strongly validate that the disk caching and admission control policies

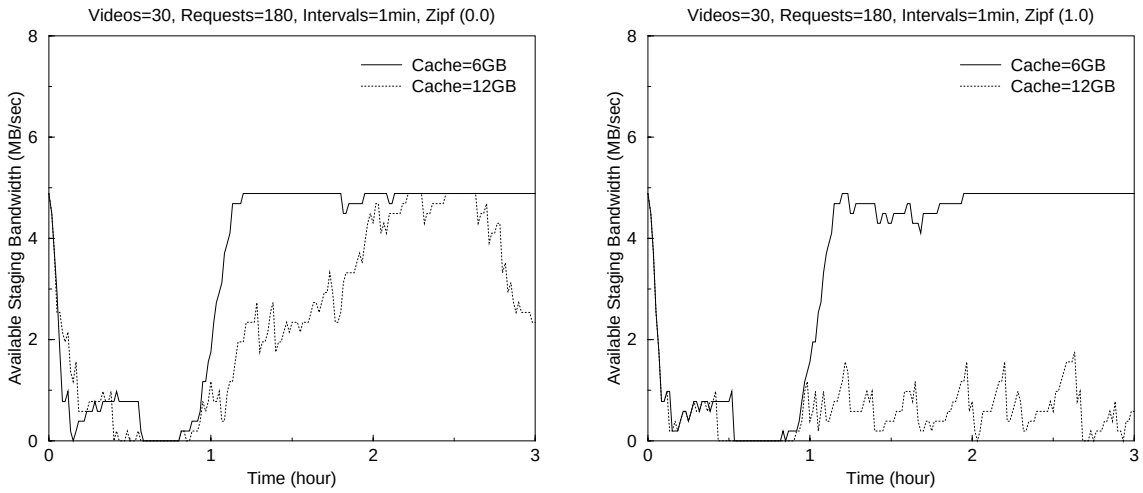

Figure 6: Available staging bandwidth; (a) Zipf(0.0), (b) Zipf(1.0)

work as expected.

4 Conclusion

This paper presented the design and implementation of a streaming server which uses tertiary device as a source of media archiving. Considering the excessive access delay of tertiary device, we have devised an effective disk caching system and related operating principles with which continuous media is directly streamed from tertiary storage and the system resources are highly utilized. We have actually implemented the system and validated the key design principles with real experiment results. The experiments show that there are many factors governing the performance of tertiary-based media server. The server performance is basically limited by the hardware-specific factors such as the media exchange time and access bandwidth of tertiary device. However, depending on the access distribution and the size of disk cache, in particular, the performance varies greatly. This fact guides us how streaming servers with tertiary storage should be configured in real environment and under what circumstances the servers are used effectively.

The implemented system is fully functional. The current server provides only a simple playback of media. A true streaming server should, however, support more complex way of media access such as random access, forward or reverse scan. These require a sophisticated method of staging scheduling, disk caching and admission control. This is part of our future research.

Acknowledgements

This work is supported by the Basic Research Program of the Korea Science and Engineering Foundation(grant no: 97-01-00-12-01-5) and by the Brain Korea 21 project.

References

[1] D.J. Gemmel, H.M. Vin, D.D. Kandlur, P.V. Rangan, L.A. Rowe, 'Multimedia Storage Server: A Tutorial', *IEEE Computer*, May 1995, pp. 40-49.

[2] Sung Bae Jun, Won Suk Lee, 'Video Allocation Methods in a Multi-level Server for Latge-scale VOD Services', *IEEE Transactions on Consumer Electronics*, Vol. 44, No. 4, November 1998, pp. 1309-1318.

[3] Kian Lee Tan, Beng Chin Ooi, Tat Seng Chua, 'On Video-On-Demand Servers with Hierarchical Storage', *The 5th International Conference on Database Systems for Advanced Applications*, Melbourne, Australia, April 1997, pp. 491-500.

[4] Tatsuo Mori, Hideharu Suzuki, Kazutoshi Nishimura, Hirotaka Nakano, 'Playback Techniques for a Video-on-Demand System Using an Optical Mass Storage System', *Japanese Journal of Applied Physics*, Vol. 35, Part. 1, No. 1B, January 1996, pp. 495-499.

[5] Peter Triantafillou, Thomas Papadakis, 'On-Demand Data Elevation in a Hierarchical Multimedia Storage Server', *The 23th International Conference on Very Large Data Bases*, Athens, Greece, August 1997, pp. 226-235.

[6] Siu-Wah Lau, John C.S. Lui, 'Designing a Hierarchical Multimedia Storage Server', *The Computer Journal*, Vol. 40, No. 9, November 1997, pp. 529-540.

[7] Shahram Ghandeharizadeh, Ali Dashti, Cyrus Shahabi, 'Pipelining Mechanism to Minimize The Latency Time in Hierarchical Multimedia Storage Manager', *Computer Communications*, Vol. 18, No. 3, March 1995, pp. 170-184.

[8] Makoto Mizukami, Shigetaro Iwatsu, Nobuyoshi Izawa, 'Real-Time Staging in Optical Disk Library', *Japanese Journal of Applied Physics*, Vol.36, Part. 1, No. 1B, January 1997, pp. 568-571.

[9] S. Prabhaker, D. Arrawal, A. El Abbadi, A. Singh, 'Efficient I/O Scheduling in Tertiary Libraries', *Technical Reports*, Computer Science Department, University of Columbia, TRCS96-26, October 1996.

[10] Shueng-Han Gary Chan, Fouad A. Tobagi, 'Hierarchical Storage Systems for On-Demand Video Servers', *Proceedings of the SPIE*, Vol. 2604, 1996, pp. 103-120.

[11] Shahram Ghandeharizadeh, Cyrus Shahabi, 'On Multimedia Repositories, Personal Computers, and Hierarchical Storage Systems', *ACM Multimedia*, San Francisco, California, October 1994, pp. 407-416.

[12] L. BresLau, P. Cao, L. Fan, Gl. Phillips, and S. Shenker, 'Web caching and zipf-like distributions: Evidence and Implications', *Proceedings of IEEE Infocom*, New York, March 1999, pp. 126-134.