階層式貯存系統之模擬
A Simulation Model of Hierarchical Storage System[1]

黃胤傅　　　　　黃景茂
Yin-Fu Huang and Jiing-Maw Huang

雲林科技大學電子與資訊工程所
Institute of Electronics and Information Engineering
National Yunlin University of Science and Technology
huangyf@el.yuntech.edu.tw

摘要

為了說明階層式貯存系統(HSS)的性能，我們提出一個模擬模式，來執行並觀察當接收來自遠端隨從者的讀取需求時，伺服器會有何行為反應。在模擬當中，我們建立一個一般且開放式的貯列模式來模擬 HSS。同時為觀察在不同情況下貯存系統的行為反應，我們也訂出模擬環境的工作量、成本和評估參數。

關鍵字:階層式貯存系統, 多媒體貯存伺服器, 模擬, 佇列模式

Abstract

To show the performance of Hierarchical Storage System (HSS) when receiving the requests from the clients in the network, a simulation model is proposed and a simulation is undertaken. In the simulation model, we build a general and open queuing model to simulate HSS. To observe the behaviors of the storage system under different situations, some workload parameters, cost parameters, and measure parameters are also given.

Keywords: Hierarchical storage system, Multimedia storage server, Simulation. Queuing model

1 Introduction

Recent advances in computing and communication have made on-line access to multimedia information both possible and cost-effective. The environment to support these services consists of multimedia storage servers and numerous client sites connected with high-speed networks. Clients can retrieve multimedia information from the servers for real-time playback. Furthermore, client can issue different requests such as stop, pause, resume, even fast-forward and fast-backward in an interactive way. The media can be classified into two types, namely non-continuous media such as conventional text and numerical data, and continuous one which could be audio and video.

The continuous media (CM) has two fundamental characteristics [4]. One is real-time storage and retrieval - CM recording devices generate a contiguous stream of media quanta that must be stored in real-time; in reverse, the media quanta must also be presented using the same time sequence in real-time. Any derivation from this time sequence will incur the jerkiness in video presentation, or pops in audio play. Another is high data transfer rate and large storage space - Digital video and audio playback demand a high data transfer rate. Moreover video and audio media usually require a large storage space provided by multimedia storage servers.

To support these two fundamental characteristics of CM, an efficient large-scale storage server must be designed. Large-scale storage servers will need to combine the cost-effectiveness of archive storage devices with the high performance of magnetic disks, RAIDs [1] [3], and CD-ROMs. The storage system is organized as a hierarchy where magnetic disks and RAIDs are used as a cache for the archive storage devices [5]. To make the storage system as efficient as possible in processing client requests, the powerful and efficient **Hierarchical Storage System** (HSS) must be analyzed first and then designed later.

To analyze the performance of HSS when receiving the requests from the clients in the network, a simulation model is proposed and a simulation is undertaken in this paper. The reason why an analytic model is not adopted here is the complexity and intractability of HSS. The simulation work proceeds in two stages. In the first stage, due to no decision about the configuration of HSS, we try to tailor a general simulation model to fit all different configurations of HSS. In the second stage, the workloads resulted from clients will be investigated and analyzed deliberately. Feeding these workloads into the simulation model, we

[1]This study was supported, in part, by the National Science Council, Republic of China, under the contract number: NSC 86-2213-E-224-003.

can figure out how the performance tuning is carried out in HSS according to different workloads.

The remainder of the paper is organized as follows. Section 2 describes the environment of hierarchical storage system. Then a simulation based on the environment is proposed in Section 3. In Section 4, we conduct a series of experiments and show how to construct the hierarchical storage system. Finally brief conclusions and future research directions are given in Section 5.

2 The environment of hierarchical storage system

The environment of the hierarchical storage system shown in Fig. 1 is a client/server model [9]. At one end, a client such as a video application, medical one, or shopping one can issue an access request to a storage server at the other end through a high-speed wide area network. After the storage server's processing, the data transmitted back to the client could be text, still image, graphic, audio, and even video, and they may be displayed on the client's screen in multiple data streams. Multiple data streams not only are synchronized with each other, but must be transmitted in a real-time performance requirement [2] [6] [7]; that is, during playback, no output devices suffer from starvation or jitter. Currently, no editing is supported in our model. In other words, the applications at the client end are based on **Media on Demand** (MOD) [5] technology to retrieve their target data.

The storage server at the other end of the wide area network can be viewed as a logical site which consists of several physical nodes connected with an Ethernet-based **Local Area Network** (LAN). These nodes can be classified into three types; that is filters, media servers, and archive servers. The request from a client is first received by the filter, and the filter then retrieves its local structured text data if necessary or dispatches the request to appropriate media servers according to the dictionary stored in the filter. Thus the filter plays both roles of the dispatcher and the server storing structured text data. In order not to occupy the bandwidth of a high-speed channel, the request and the reply of structured text data are planned to be transmitted in a low-bandwidth channel. As for the media servers, when receiving the forwarded request, they retrieve the variety of data from different facilities, such as hard disks, CD-ROMs, and even high-performance RAIDs. If the target data are currently not available on the media servers, they may request an archive server to download the target data to their local devices through LAN. In general, the archive server is facilitated with jukeboxes whose components could be CD-ROMs or tapes. The multimedia data stored at media servers and archive ones, in general, occupy large storage space, and require a specified storage server to process themselves. Furthermore, the multimedia data retrieved from media servers are transmitted back to the client through another high-bandwidth channel in order to meet the real-time requirements.

3 The simulation model

In this section, a general queuing model which can actually reflect the behaviors of the storage server when receiving a request from a remote client, is proposed. There are several features involved in this queuing model. First, it can simulate the behaviors of the storage server under different traffic loads. Second, given the variety of the maximum data stream and cost parameters, the capability of the storage server could be scalable. Third, from the model, we can determine how large the buffer size should be such that it is the most efficient for a media server. Fourth, it can reflect the performance under different hit ratios of hard disks and RAIDs. Fifth, it can also show the situations under different load biases of facilities.

As the workloads, related parameters could be categorized as follows. They are 1) request pattern, 2) system environment, and 3) media granule. In order to study the effect of the server resulting from the requests, a set of cost parameters about CPU time, I/O time, and transmission time are given. Finally, two measure parameters such as throughput and start-up latency are used to show the strengths and weaknesses of performance under different workloads.

3.1 The queuing model

To make the simulation as close to a real situation as possible, a general and open queuing model shown in Fig. 2 is proposed. This model simulates the storage server mentioned in Section 2. The requests from clients arrive at the storage server at a specified random interval. The filter of the storage server first catches the requests, and then processes them as long as the number of data streams currently being active in the storage server is below an upper limit. If the upper limit is exceeded, the storage server would not handle the requests and put them in **Data Stream Control Queue** till some active requests are consumed in the storage server. As the requests go through **Filter CPU Queue** and **Server**, the filter determines appropriate servers where they should be dispatched, according to the dictionary stored in the filter. If the target data are text files, they will be fetched directly from the local filter and sent back to the clients through a low-bandwidth channel. **Filter I/O Queue/Server** and **LBC Queue/Server** shown in Fig. 2 depict this situation. If the target data are stored in remote media servers, the requests will be forwarded to appropriate servers through LAN.

There could be several media servers connected with LAN, though only one media server is shown in Fig.

2. The media server is a general one equipped with a few different facilities, such as hard disks, CD-ROMs, and even high-performance RAIDs. When the media server receives a forwarded request, the server processes it immediately as long as the system's free buffers are enough. If the available buffers are not enough, the request will be suspended in **Buffer Control Queue** till some occupied buffers are released. As the request goes through **Media CPU Queue** and **Server**, the server determines the facilities where the target data are stored, according to the request pattern. For different facilities, each one has its own queue and server. The target data will be retrieved from appropriate facilities and sent back to the clients through a high-bandwidth channel. The different **I/O Queues/Servers** and **HBC Queue/Server** shown in Fig. 2 depict this situation. If the target data are too large to be fetched at one time, the same procedure will be executed once more. Furthermore, based on the concepts of hierarchical storage system, hard disks or RAIDs act as a cache storage of archive servers in the queuing model. In other words, when a client issuing an initial play request, if the target data are not available on the media server, they must be downloaded from the archive server first, and then retrieve them from hard disks or RAIDs. The related **Archive CPU Queue/Server** and **Archive I/O Queue/Server** are shown in Fig. 2.

The requests issued from clients could be classified into two types. One is control requests such as stop, pause, and resume which themselves do not fetch data. Another one is retrieval requests such as conventional access, play, fast forward, and fast backward. As mentioned before, the retrieval requests fetch their target data from appropriate servers. Correspondingly, the control requests will be processed in a different way. Stop and pause requests record the demanded action of a client, and resume requests wake up the retrieval request suspended in **Block Queue**. The demanded status will be referenced when a retrieval request has more data to fetch. If the demanded status is stop, the retrieval request needn't fetch data again and completes. While the demanded status is pause, the retrieval request will be put into **Block Queue** till a resume request is issued later.

3.2 Workload parameters

In the simulation model, the workload parameters as shown in Table I can be categorized into three types. They are 1) request pattern, 2) system environment, and 3) media granule. According to different workloads, the queuing model can reflect the behaviors of the storage server when receiving a request from a remote client. Moreover, by giving these parameters appropriate values, the features involved in the queuing model could be shown up in the simulation.

The parameter **req_arr_time** is the mean inter-arrival time among requests from all clients in the network, and can be used to simulate different traffic loads. The next three parameters are related to each other. The parameter **fewer_data_stm_prob** is the probability of requests with fewer data streams. The parameter **fewer_data_stm_no** is the mean number of data streams for requests with fewer data streams. The parameter **more_data_stm_no** is for requests with more data streams. These three parameters have a great effect on the data stream control in the storage server. The next three parameters are also related to each other. The parameter **small_req_prob** is the probability of small requests occurring in the simulation. The parameter **small_req_size** indicates the mean size of data retrieved by a small request. The parameter **large_req_size** is for requests with a large amount of data. These three parameters have an influence on the frequency of device I/Os executed in the storage server. Then, the parameter **dist_dev_type** describes the ratio of target data distributed on devices such as the filter's hard disks, the media servers' hard disks, RAIDs, and CD-ROMs. Last, the parameter **play_prob** is the probability of play request among all requests.

The parameter **max_data_stm** is the maximum number of data streams being processed in the storage server in order to guarantee the quality of services (Qos). In cooperation with the cost parameters discussed in the next section, the maximum data stream can be adjusted to show the capability of the storage server. The parameter **max_buf_size** is the maximum size of system buffers allocated in the media server. It aids us to tailor the system buffer and makes a media server most efficient. Another two system parameters are related to the hit ratio of device-based caches; that is the parameter **hd_hit_ratio** for hard disks and the parameter **raid_hit_ratio** for RAIDs. They are used to reflect the performance under different hit ratios.

The last three parameters are related to the granule size of different devices [5]; that is hard disks, RAIDs, and CD-ROMs. In general, a block (granule) size of hard disks and CD-ROMs is fixed, whereas a granule size of RAIDs could be different when RAIDs are configured as level 5. Different granule sizes are associated with the number of device I/Os executed in the media server.

3.3 Cost parameters and measure parameters

To evaluate the performance of HSS, several cost parameters must be given to the servers described in the queuing model. These cost parameters shown in Table II could be classified according to 1) server types, 2) request types, and 3) cost types. There are totally ten servers in the queuing model. In spite of what requests are, they all are processed in **Filter CPU Server** first, but only local requests flow through **Filter I/O Server**. Only one cost parameter is required for each filter server. For remote requests dispatched to media servers, different cost parameters are given for **Media CPU Server**, based on control or retrieval requests. For retrieval requests, when they must download the target data from archive servers, the cost parameters about CPU and I/O are specified for **Archive CPU** and **Archive I/O Servers**, respectively. For each device server, different cost parameters are given according to the request types such as play, fast forward, and fast backward. Finally, we also consider the transmission cost on a low bandwidth channel and a high bandwidth channel, respectively.

Finally two measure parameters are used to show the performance of HSS; one is **throughput** and another one is **start-up latency** [8]. The throughput measures the number of completed requests per unit of time. The start-up latency measures the delay when the first target data are received from the storage servers. These both measure parameters can truly reflect strengths and weaknesses of different configurations for the same workloads. A good configuration should have a higher throughput and a shorter start-up latency.

4 The experimental results

4.1 The experiments

As for the experiments, the simulation model is implemented with the language GPSS World which is the third-party product run on OS/2 in Pentium PC. Each experiment is simulated for 10^7 time units.

Experiment 1: Traffic load

This experiment simulates the behaviors of the storage server under different traffic loads. The throughput and start-up latency under different inter-arrival time are shown in Fig. 3. From this figure, we find that the throughput is almost in inverse proportion to the inter-arrival time, except the case of 10^3 ms. The reason is that the system is too busy to handle the incoming requests, so the throughput in the case of 10^3 ms is not as we expect. However, in general, the more heavy load, the more throughput produced. As for the start-up latency, there is no great difference among these traffic loads, except the heaviest load (i.e. the case of 10^3 ms). We have the same reason to explain why there is a enormous start-up latency occurred in the case of 10^3 ms. Besides, regardless of traffic loads, any play request must be processed at least 2 seconds to get its first target data.

Experiment 2: Scalability

Given the variety of the maximum data stream and cost parameters, this experiment simulates the capability of the storage server. The throughput and start-up latency under different cost parameters and 10 data streams permitted are shown in Fig. 4. From this figure, we find that the throughput is almost the same under different cost parameters. As for the start-up latency, in general, the more cost, the more start-up latency produced. Fig. 5 shows the throughput and start-up latency under different maximum data streams permitted and fixed filter CPU cost. From this figure, we find that the throughput is also almost the same under different maximum data streams. As for the start-up latency, we find that the start-up latency is almost the same, except the case of 10 data streams. This fact explains that at least 30 data streams are required to support in the system to get better performance.

Experiment 3: Buffer size

This experiment simulates and shows how large the buffer size should be such that it is the most efficient for a media server. The throughput and start-up latency under different buffer sizes and fixed probability 0.1 for small requests (i.e. most requests are large) are shown in Fig. 6. From this figure, we find that the throughput is almost the same, except the case of 1 MB. As for the start-up latency, we have great improvement when 8 MB system buffers are allocated in the system, and no more benefits are got when buffer size is 16 or 32 MB. Fig. 7 shows the throughput and start-up latency under different probabilities of small requests and 1 MB buffer allocated. We find that the more small requests, the more throughput produced, unless the probability of small requests is more than 0.5. As for start-up latency, we have the best performance when the probability of small requests is 0.9. Besides, not shown in both figures, we also observe when most requests are small, they will have less start-up latency if more system buffers size are allocated in the storage server.

Experiment 4: Hit ratio

This experiment simulates and reflects the performance under different hit ratios of hard disks and RAIDs. The throughput and start-up latency under different hit ratios for RAIDs are shown in Fig. 8. From this figure, we find that the throughput is the same for all cases. As for the start-up latency, we have better performance when the hit ratio is higher. Fig. 9 shows the throughput and start-up latency under different hit ratios for the media server's hard disks. We find that the same results occur in the media server's hard disks. Thus we know if the hit ratios for storage devices are high, it will have little start-up latency.

Experiment 5: Load bias

This experiment simulates and shows the situations under different load biases of facilities. The throughput and start-up latency under different load biases of facilities are shown in Fig. 10. From this figure, we find that the throughput is almost the same under different load biases of facilities, except the case - more loads on the filter's hard disks. As for the start-up latency, it is related to what device the target data is stored bias. The start-up latency increases with the filter's hard disks, CD-ROMs, RAIDs, and the media servers' hard disks. Although CD-ROMs have largest cost parameters, they still perform better than RAIDs and the media servers' hard disks, because there is no overhead to download the target data from the archive storage.

4.2 Summary

After analyzing the results through these experiments, we have some observations and suggestions to build up a hierarchical storage system. First, in general, the more heavy load, the more throughput and start-up latency produced. However under extremely heavy load, the throughput is not increased as we expected, and the start-up latency will be very serious. Second, the cost parameter specified in the filter CPU and maximum data streams permitted in the storage server has little influence on the throughput. However that the data streams permitted in the storage server is too small will result in worse start-up latency. Here we suggest at least 30 data streams are required to support in the system to get better performance. Third, when most requests are large, only 8 MB buffers are enough to get better start-up latency. In general, the more small requests, the better start-up latency, specially when more buffers are allocated. Fourth, the hit ratio has little influence on the throughput, but if the hit ratios for storage devices are high, it will have little start-up latency. Thus an efficient device-cache management is desirable in a hierarchical storage system. Last, surprisingly, CD-ROMs has better start-up latency than RAIDs and the media servers' hard disks, because there is no overhead to download the target data from the archive storage. Therefore it is critically important to design an efficient device-cache management to eliminate the download overhead from the archive storage.

5 Conclusions

In this paper, a general simulation model is proposed and a simulation is conducted to evaluate a hierarchical storage system. We expect the simulation model can measure any different configurations of HSS, and then provide us a guideline to find the most efficient configuration for different workloads. Feeding these workloads into the simulation model, we can figure out how the performance tuning is carried out in HSS. Besides after observing these experiment results, we consider an efficient device-cache management is the most urgent in a hierarchical storage system. Finally, based on the proposed model, we can also explore more sophisticated one to analyze the scheduling algorithms implemented in different devices attached at media servers and the inter-process communication between processes running at media servers.

References

[1] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," ACM Computing Surveys, Vol. 26, No. 2, pp. 145-185, June 1994.

[2] B. Furht, "Multimedia Systems: An Overview," IEEE Multimedia, pp. 47-59, Spring 1994.

[3] G. R. Ganger, B. L. Worthington, R. Y. Hou, and Y. N. Patt, "Disk Arrays: High-Performance, High-Reliability Storage Subsystems," IEEE Computer, pp. 30-36, Mar. 1994.

[4] D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe, "Multimedia Storage Servers: A Tutorial," IEEE Computer, pp. 40-49, May 1995.

[5] D. Jadav and A. Choudhary, "Designing and Implementing High-Performance Media-on-Demand Servers," IEEE Parallel and Distributed Technology, pp. 29-39, Summer 1995.

[6] Y. J. Oyng, C. H. Wen, C. Y. Cheng, M. H. Lee, and J. T. Li, "A Multimedia Storage System for On-demand Playback," IEEE Trans. Consumer Electronics, Vol. 41, No. 1, pp. 53-64, Feb. 1995.

[7] A. A. Rodriguez and L. A. Rowe, "Multimedia Systems and Applications," IEEE Computer, pp. 20-22, May 1995.

[8] R. Steinmetz, "Analyzing the Multimedia Operating System," IEEE Multimedia, pp. 47-59, Spring 1995.

[9] Y. Y. Yao etc., "Media Server Version 1.0: Specification of Software Requirements," CCL ~~ITRI,~~ 1994.

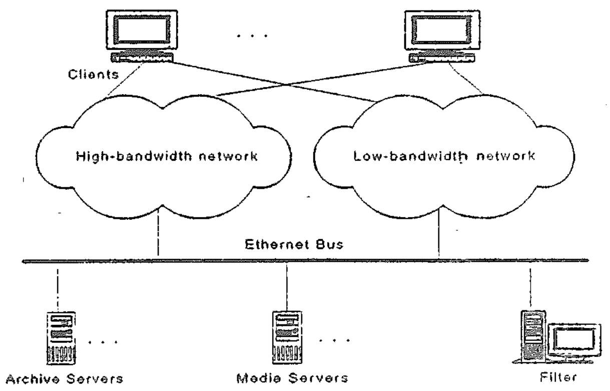

Fig. 1 The Environment of Hierarchical Storage System

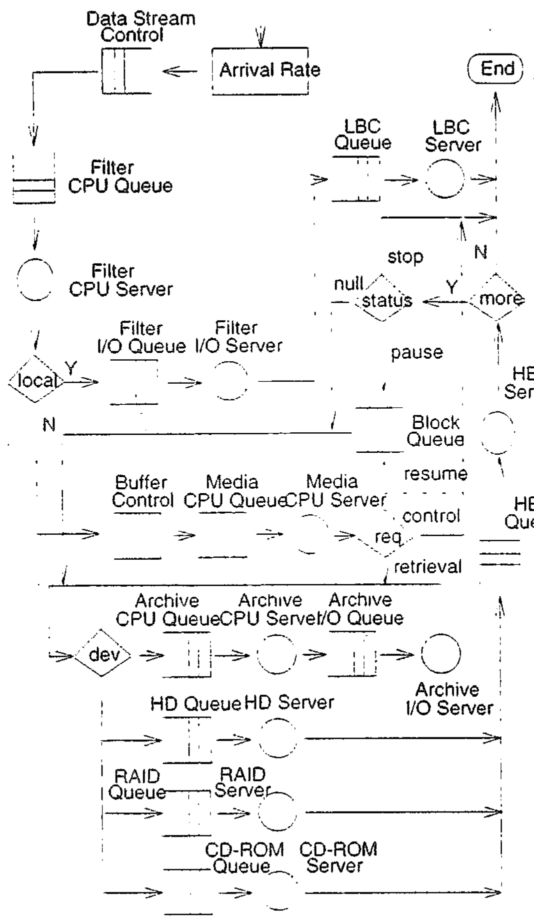

Fig. 2 The Queuing Model

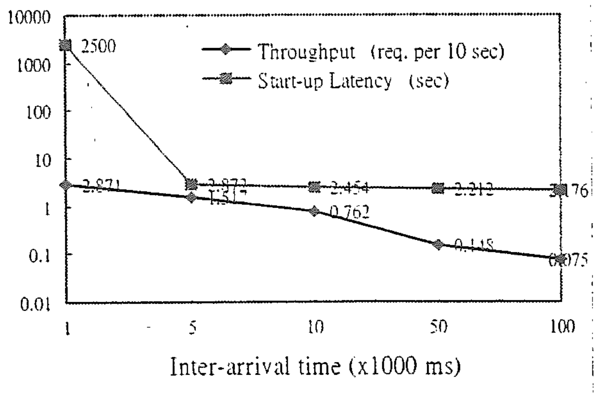

Fig. 3 Throughput & Start-up Latency vs. Traffic Load

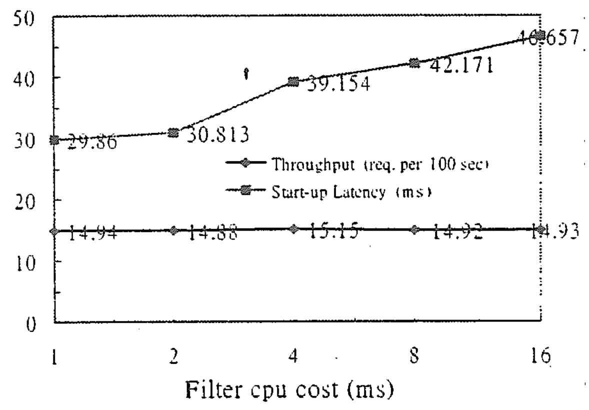

Fig. 4 Throughput & Start-up Latency vs. Cost(Max. Data Stream = 10)

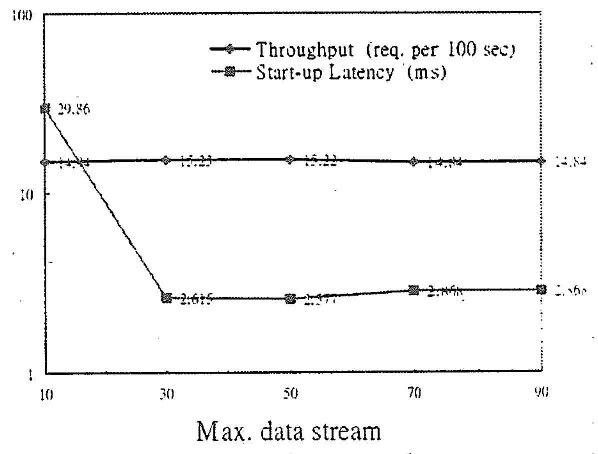

Fig. 5 Throughput & Start-up Latency vs. Max. Data Stream (Filter CPU Cost = 1 ms)

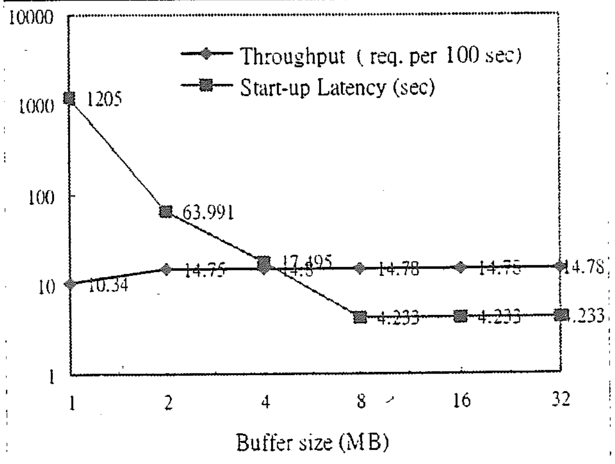

Fig. 6 Throughput & Start-up Latency vs. Buffer Size (Small request Probability = 0.1)

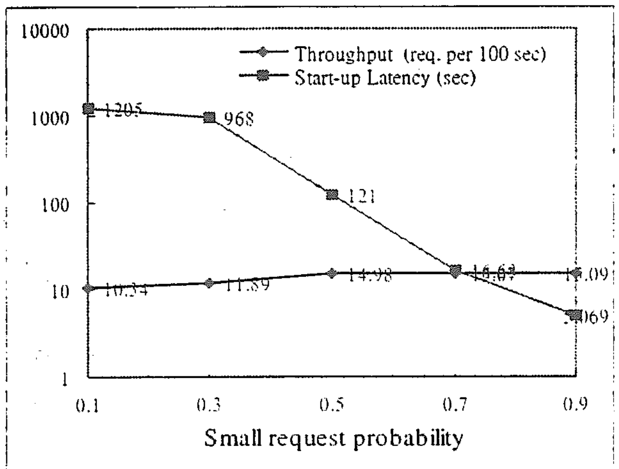

Fig. 7 Throughput & Start-up Latency vs. Small Request Probability (Buffer Size = 1 MB)

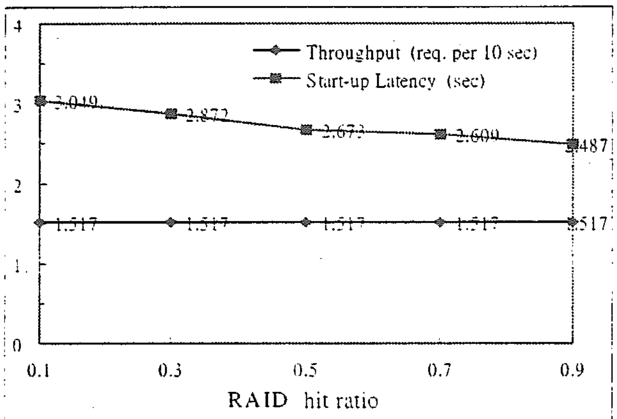

Fig. 8 Throughput & Start-up Latency vs. RAID Hit Ratio

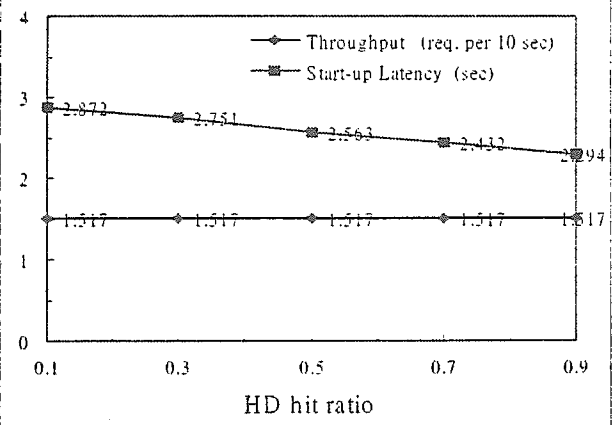

Fig. 9 Throughput & Start-up Latency vs. HD Hit Ratio

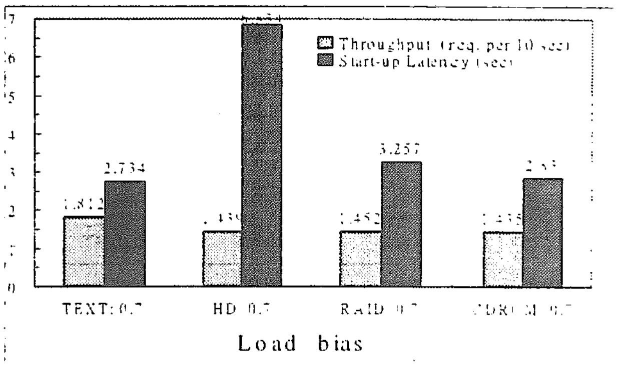

Fig. 10 Throughput & Start-up Latency vs. Load Bias

Table I Workload parameters

Request pattern:	
req_arr_time	10^3, 5×10^3, 10^4, 5×10^4, 10^5 (ms)
fewer_data_stm_prob	0.5
fewer_data_stm_no	2
more_data_stm_no	5
small_req_prob	0.1, 0.3, 0.5, 0.7, 0.9
small_req_size	500 KB
large_req_size	10 MB
distr_dev_type	0.2, 0.3, 0.3, 0.2 0.7, 0.1, 0.1, 0.1 0.1, 0.7, 0.1, 0.1 0.1, 0.1, 0.7, 0.1 0.1, 0.1, 0.1, 0.7
play_prob	0.5
System environment:	
max_data_stm	10, 30, 50, 70, 90
max_buf_size	1, 2, 4, 8, 10, 16, 32 MB
hd_hit_ratio	0.1, 0.3, 0.5, 07, 0.9
raid_hit_ratio	0.1, 0.3, 0.5, 07, 0.9
Media granule:	
hd_gran_size	5 KB
raid_gran_size	10 KB
cd-rom_gran_size	5 KB

Table II Cost parameters

filter_cpu	1, 2, 4, 8, 16 (ms)
filter_local_io	20+total granular no.*2
media_control_cpu	1
media_retrieval_cpu	1
archive_retrieval_cpu	1
archive_retrieval_io	1000
hd_play_io	20+granular no.*2
hd_fast_forward_io	10
hd_fast_backward_io	10
raid_play_io	6+granular no.*0.6
raid_fast_forward_io	3
raid_fast_backward_io	3
cd-rom_play_io	60+granular no.*6
cd-rom_ fast_forward_io	30
cd-rom_ fast_backward_io	30
lbc_trans	500
hbc_trans	50