Future Generation Computer Systems 12 (1997) 565–577

Fu-Ching Wang, Chun-Hung Wen, Chih-Yuan Cheng, Meng-Huang Lee, Tzu-How Lin, Szu-Chi Wang, Yen-Jen Oyang*

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC

Received July 1995, received in revised form February 1996; accepted December 1996

In this paper, we propose a new approach to exploit semiconductor memory to upgrade the service capacity of video-on-demand systems. The basic idea behind this so-called Gneralized Relay Mechanism is that if we can relay the data of frequently accessed programmes from one access request to another, then we can improve the service capacity of the system without requiring higher disk bandwidth. The design of the Generalized Relay Mechanism is aimed at optimizing allocation and trade-off of the memory resource and the disk bandwidth resource. A simulation-based study shows that the Generalized Relay Mechanism is a very competitive alternative in comparision with simply incorporating more hard disks. If compared with an intuitive approach of utilizing memory buffer that always fills up the memory with most frequently accessed programmes, the Generalized Relay Mechanism enjoys an advantage in terms of cost.

Keywords: Semiconductor memory; Video-on-demand; Service capacity

There are two major concerns in the design of video storage systems for on-demand playback applications. The first concern is how to provide sufficiently large storage capacity for archiving the video programmes [1]. The second concern is how to meet the real-time data retrieval bandwidth required by a large number of clients [2–5]. The bandwidth issue is the main concern of this paper.

The conventional measures that are adopted to meet the required high data retrieval bandwidth treat each playback request independently and resort entirely to disk bandwidth [2–4,6–8]. As a result, the number of disks required grows lineraly with the number of clients that the system is designed to support at one instant. In the real world, however, some programmes such as newly released movies or popular songs may be accessed much more frequently than the others. Hence, incorporation of a storage hierarchy is desired to reduce the overal costs while maintaining the same level of service capacity.

In recent years, several studies concerning exploiting semiconductor memory to upgrade the service capacity of video-on-demand (VOD) servers

* Corresponding author. Tel.: 8862 362 5336 × 431; fax: 8862 368 8675;
e-mail: yjoyang@csie.ntu.edu.tw

PII S0167-739X(97)00020-9

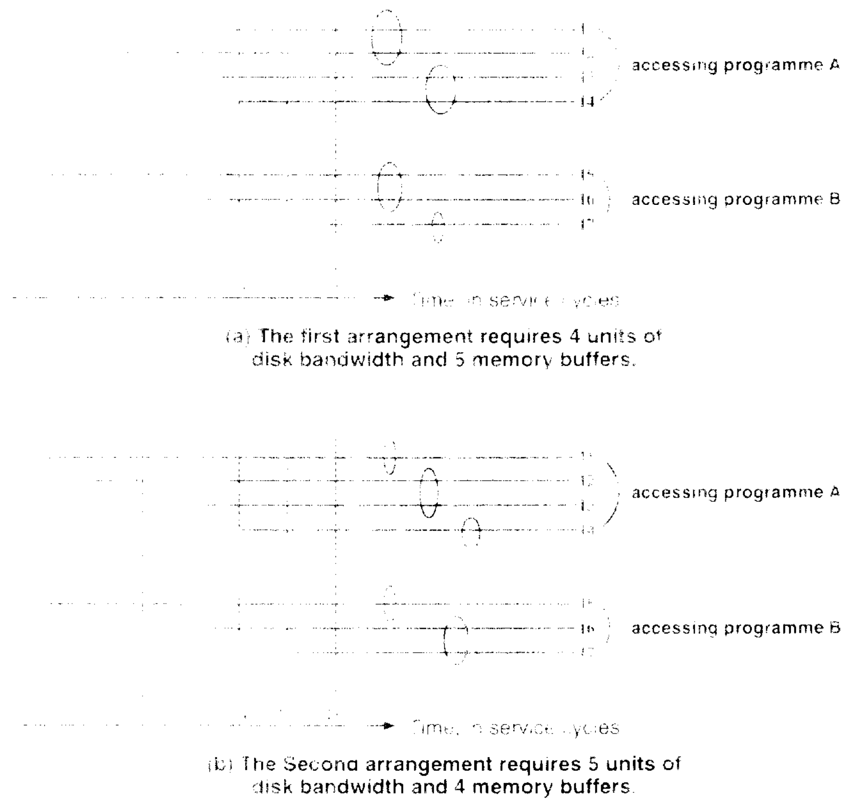


Fig. 1. Two resource allocation arrangements.

have been reported [9–12]. The studies were motivated by observing continuously a rapid increase of memory chip density. Actually, the basic techniques used in the interval caching policy [9], the generalized interval caching policy [10, 12], and the relay mechanism [11] are very similar. All these approaches use memory buffers to forward or relay the data from one request to another, if these requests access the same programme and start closely in time. Nevertheless, the interval caching policy and the generalized interval caching policy do not address run-time allocation and trade-off between the memory resource and the disk bandwidth resource. These two policies only deal with allocation of the memory resource among competing streams but do not address the situations in which disk bandwidth becomes limited. Fig.1 illustrates the trade-off between the memory resource and the disk bandwidth resource. Fig. 1(a) and (b) depict two different arrangements for the same pattern of events. In the figures, I_k denotes an access instance and the access instances grouped by a circle relay data from one to another. The number of circles determines the units of disk bandwidth used and the elapsed time between the access instances in the same group determines the number of memory buffers needed for relaying data. The arrangement shown in Fig. 1(a) requires four units of disk bandwidth and five memory buffers. The arrangement shown in Fig. 1(b) requires five units of disk bandwidth and four memory buffers. This example demonstrates the need to develop a resource allocation algorithm to optimize allocation and trade-off between the memory resource and the disk bandwidth resource.

The relay mechanism introduces a resource allocation algorithm [11] but there is room for further improvement. One of the main deficiencies of the relay mechanism is that it does not take into account the popularity characteristic of video programmes when carrying out run-time resource allocation. Our study reveals that this factor plays a significant role. Therefore, we propose a more comprehensive resource allocation algorithm called the Generalized Relay Mechanism. The Generalized Relay Mechanism not only takes into account the popularity characteristic of video pograms but also employs a practice that speculatively allocates memory buffers. According to a simulation-based study presented in Section 3, the Generalized Relay Mechanism is a very competitive alternative in comparision with simply incorporating more hard disks. If compared with an intutive approach of utilizing memory buffers that always fills up the memory with most frequently accessed programmes, the Generalized Relay Mechanism enjoys a significant advantage.

In the following discussion, an access instance denotes the event that one or more clients access a particular programme at a particular time. A relay chain is a group of access instances that relay data from one to another. A relay chain consists of a head and a number of followers. The head retrieves the desired data from the disk system and the followers are served by the data relayed through memory. A relay chain uses one unit of disk bandwidth and some amount of memory. The amount of memory used is determined by the elapsed time between the head and the tail of the chain. The elapsed time is measured in numbers of service cycles and the amount of memory is measured in numbers of memory buffers. A service cycle is a period of time during which the disk system retrieves one section of data for each playing stream. A memory buffer is of size needed to store one section of data retrieved from the disk system during one service cycle.

In attempting to optimize resource allocation, the Generalized Relay Mechanism dynamically splits and merges relay chains. If a relay chain is split into two, then one more unit of disk bandwidth is needed but a few memory buffers are released. On the other hand, if two relay chains are accessing the same programme are merged into one, then the system will release one unit of disk bandwidth with a delay but will take a few more memory buffers. Here, the release of disk bandwidth is delayed by a few service cycles waiting for the relayed data to reach the relay chain that will become the tail part of the merged relay chain. Fig. 2 demonstrates the splitting and merging operations. In the merging operation of

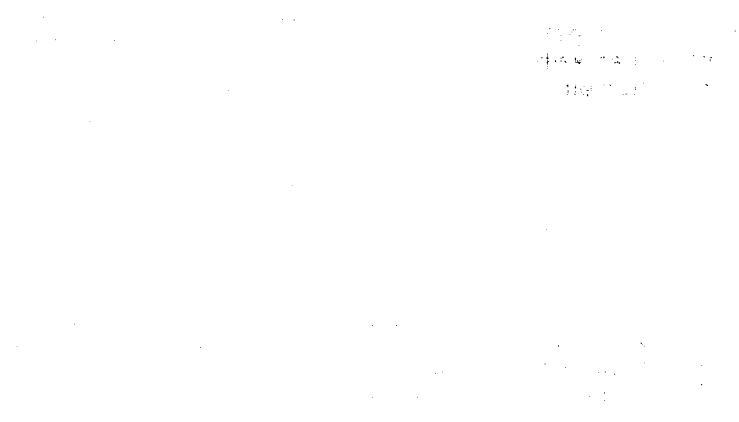


Fig. 2. Example of the merging and splitting operations.

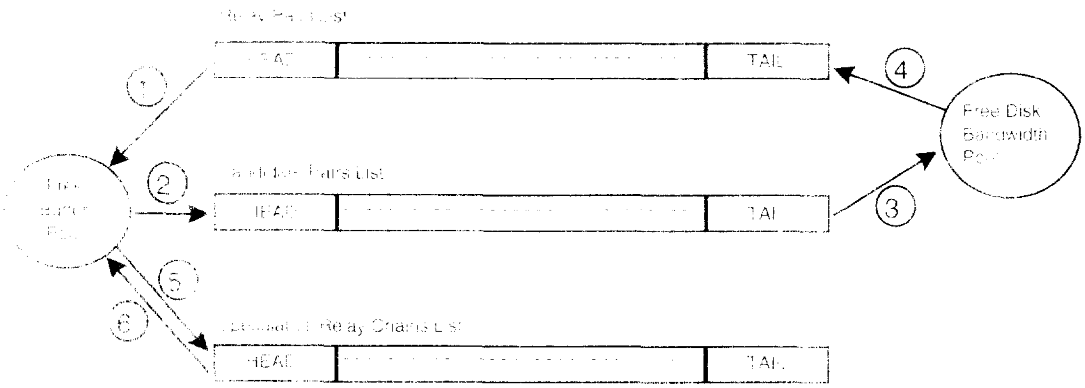


Fig. 3. The data structures used in the Generalized Relay Mechanism.

Fig. 2, the release of the disk bandwidth originally used by the relay chain containing I3 and I4 will be delayed by three service cycles.

Fig. 3 depicts the five data structures in the Generalized Relay Mechanism. The functions of these five data structures are described in the following:

1. The relay pairs list contains all the access instances pairs whose two access instances form two consecutive entities in one of the relay chains. A relay pair is a point where the splitting operation may be applied. The list is sorted in the descending order according to the number of memory buffers each pair occupies.
2. The candidate pairs list contains access instance pairs whose two access instances are not included in a relay chain but may be merged into a relay chain if memory buffers are availabe. The list is sorted in the ascending order according to the number of memory buffers each candidate pair requires in case merging is to be applied. The order defines which pair has a higher priority to be merged if memory is limited.
3. The speculative relay chains list contains all the speculative relay chains. A speculative relay chain is heated by a leading access instance and contains one or more memory buffers but no following access instance. A newly admitted access instance will head a speculative relay chain if free memory buffers are available. The data that the leading access instance retrieves from the disks are temporarily stored in the associated memory buffers with the hope that the data will soon be accessed by a second access instance that starts later.
This list is sorted out in the ascending order according to the buffer utilization factor of each speculative relay chain. The buffer utilization factor of a speculative relay chain is the popularity of the program that the leading access instance is accessing divided by the number of memory buffers associated with the chain.
4. The free buffer pool contains available memory buffers.
5. The free disk bandwidth pool keeps a record of available disk bandwidth.

Fig. 4 shows the pseudo-code of the resource allocation algorithm in the Generalized Relay Mechanism. This algorithm is executed at the beginning of every service cycle to optimize allocation of the memory resource and the disk bandwidth resource. The basic operation of the resource allocation algorithm is to merge the candidate pairs one by one according to the order of the candidate pairs list. Merging a candidate pair requires some memory buffers. Hence, if there are no sufficient memory buffers in the free buffers pool, then the resource allocation algorithm will try to reclaim the memory buffers allocated to the speculative relay chains and will try to split relay pairs subject to certain conditions. The resource allocation algorithm will break all the speculative relay chains whose buffer utilization factor is smaller than the candidate pair to be merged. The buffer utilization factor of a candidate pair is the popularity of the

```
while (Candidate pairs list is not empty)
begin
    if (free buffer pool is not empty)
    then begin
        allocate one buffer to each unmarked speculative relay chain using
        the buffer utilization factor as the priority function;
        /* i.e. The one with a higher buffer utilization factor has a higher priority */
        mark the speculative relay chains to which a buffer is just allocated
    end;
    if (the number of free buffers is not enough for merging the head of the candidate pairs [illegible])
    then begin
        release all buffers of the speculative relay chains whose buffer utilization factor [illegible]
        smaller than the buffer utilization factor of the head of the candidate pairs [illegible]
        /* This step is applied as [illegible] in Figure [illegible] */
        while (free disk bandwidth is available and the number of buffers [illegible]
        list is greater than two times the number of buffers required by the head of [illegible])
        begin
            split [illegible]        /* This step will [illegible]
                                     [illegible] take one unit of disk bandwidth [illegible] */
        end
        if (the number of free buffers [illegible] for merging the head [illegible])
        then break from the while loop
    end;
    else begin
        merge the head of the candidate pairs list    /* This step will release [illegible]
                                                      Figure 3, and take a few buffers [illegible] as [illegible] */
    end
end;

while (the number of free buffers available is less than the number [illegible])
begin
    remove the head of the speculative relay chain and release the buffer [illegible]
    /* This step will put [illegible] buffers [illegible] the free buffer pool [illegible] */
end
allocate a new buffer [illegible] speculative relay chain and remove the mark
```

Fig. 4. Pseudo-code of the resource allocation algorithm.

program that the access instances are accessing divided by the number of memory buffers required to carry out the merging. A relay pair is subject to being split if the number of memory buffers that it uses is larger than two times the number of memory buffers required by the candiate pair to be merged. Nevertheless, splitting a relay pair required one more unit of disk bandwidth. If there is no disk bandwidth available, then the splitting operation cannot proceed.

The resource allocation algorithm tries to merge the candidate pairs one by one according to the order of the candidate pairs list. If the merging of one candidate pair cannot proceed due to insufficient resources, then the resource allocation algorithm terminates.

The above discussion deals with normal playback. In the real world, users may want to pause and resume at a later time. When such a request is issued, the access instance to be paused will be removed from the containing relay chain if it is in a relay chain.The system then reserves a unit of disk bandwidth from the free disk bandwidth pool, which will be used by the access instance when it resumes. Since the issuing of a pause request is not predictable, the system may maintain a low mark in the free disk bandwidth pool in order to handle pause requests in a satisfactory way. When free disk bandwidth runs below the low mark, the system should stop admitting new users and the remaining free disk bandwidth will be used solely to support pause requests. The level of the low mark should be determined by

Fig. 5. Disk array configuration and operations.

how frequently a user may pause the playback and by the total number of users that the system needs to support.

This section reports an evaluation of the Generalized Relay Mechanism based on simulation runs. The system is assumed to have the 2-level disk array configuration as shown in Fig. 5(a) [4,5]. The system comprises a number of disks divided into M groups. These M groups of disks form a coarse-grain disk array [13] and each group of disks is actually a fine-grain disk array [13]. The data from a video programme are stored in the system in an interleaved manner as shown in Fig. 5(a). In this figure, A_i and B_i represent sections of data from programmes A and B, respectively. All the data sections are of the same size. Since playback operations invoke no writes to the disks, we purposely omit the parity data in the disk arrays. Without the implementation of the Generalized Relay Mechanism, the access instances in the system are divided into M access teams according to their starting times. As shown in Fig. 5(a), access instances in the same access team always retrieve data from the same disk group at the same time and access instances in different access teams never access the same disk group at the same time. These M access teams access the M groups of disks in a round robin and synchronized manner. During each service cycle, the system retrieves one section of data from the disks for each access instance. During the next service cycle, these M access team rotate their positions to access next sections of data. Note here that two access instances in the same team may have different starting times but the elapsed time between their starting times must be equal to a multiple of M service cycles. Since the system must not violate the real time constraints, the number of access instances in a team cannot exceed a predetermined ceiling. Fig. 5(b) shows the operation of the system with additional memory to implement the Generalized Relay Mechanism. The main difference is that each access team now contains a number of relay chains rather than individual access instances. In this figure, the lists enclosed by braces represent relay chains.

In the simulation, the video programmes are assumed to be compressed by the MPEG-1 standard [14]. The rejection rate during a simulation run, i.e. the rate that the requests are rejected due to limited capacity, is used to measure the service capacity of the system. The simulation is conducted as follows. In every service cycle, each idle client, a client that is not accessing a programme, independently decides whether an access request is to be issued based on a pre-determined issuing probability. If an access request is to be issued, the client, independently of the other clients, selects a programme to access based on a popularity distribution function. The popularity distribution function specifies how likely each of the programmes in the system will be selected by a client. In this simulation, we used the distribution defined in the following as the popularity function:

$$p_k = \frac{(1-\alpha)\alpha^k}{\alpha(1-\alpha^M)} \quad \text{or} \quad p_k = \alpha p_{k-1},$$

where M is the total number of programmes, $k = 1, 2, 3, \ldots, M$, and α is a parameter to the formula with $0 < \alpha < 1$.

The issuing probability defined above can actually be converted to the well-known average arrival rate in a queueing model. By Little's result [15], when the system enters a steady state, the average arrival rate times program length should be equal to the total number of active clients. Therefore,

$$L*(S*N)*P = (1-S)*N, \tag{1}$$

where L denotes the length of a programme in numbers of service cycles, P the issuing probability of an idle client, N the total number of clients in the system, and S denotes the ratio of number of idle clients over total number of clients when the system enters a steady state. Based on (1),

$$S = \frac{1}{1+L*P},$$

and the average arrival rate of the system is equal to

$$S*N*P = \frac{N*P}{1+L*P}.$$

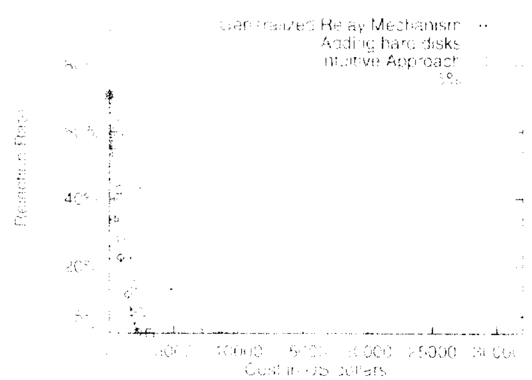
Table 1
The stimulation parameters for the Karaoke system

No. of clients	**2000**
No. of programmes	**5000**
Duration of one service cycle	**1.8 s**
Duration of each programme	**3 min**
No. of disk groups	**4**
Capacity of hard disk	**2 Gbytes**
I/O bandwidth of a hard disk	**4 MBytes per second**
Memory cost per MBytes	**US$4**
Hard disk unit cost	**US$600**
Simulation time	**2 h**
α value of distribution	**0.99**

We studied the effects of the Generalized Relay Mechanism in two applications. The first application is a Karaoke system, which is a prevailing entertainment in many Asia countries. The typical length of a Karaoke programme is about 3 min. The second application is a movie-on-demand system. While the Karaoke system represents applications with short video programmes, the movie-on-demand system represents applications with long video programmes. We compared the effect of the Generalized Relay Mechanism with that of an intuitive approach for caching video programmes and that of simply incorporating more hard disks. The intuitive approach always fills up the memory with the most frequently accessed programmes.

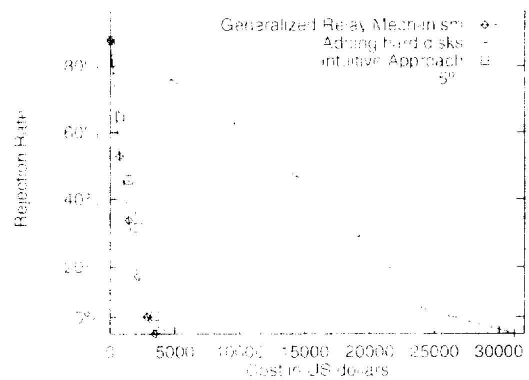
3.1 The effects in a Karaoke system

Table 1 gives the simulation parameters for the Karaoke system. The base system is assumed to have four hard disks along with an archiving storage system for storing all 5000 Karaoke programmes. The hard disks store the most popular programmes to its capacity limit. The number of programmes and the α parameter of the popularity distribution function are derived from the statistical data collected in a Karaoke store. In the simulation, the Generalized Relay Mechanism is applied to only the most popular 30 programmes. This practice is rea-



	Added Hardware Resources and Costs		
Rejection rate	Generalize Relay Mech.	Adding hard disk	Intuitive approach
5%	468 MBytes of memory US$ 1872	12 HD US$7200	648 MBytes of memory US$2592
0.1%	612 MBytes of memory US$ 2448	16 HD US$9600	828 MBytes of memory US$[illegible]

(a) Issuing rate = [illegible]



	Added Hardware Resources and Costs		
Rejection rate	Generalize Relay Mech	Adding hard disk	Intuitive approach
5%	720 MBytes of memory US$ 2880	44 HD US$26400	864 MBytes of memory US$3456
0.1%	864 MBytes of memory US$ 3456	52 HD US$31200	972 MBytes of memory US$3888

(b) Issuing rate = 10%

Fig. 6. The simulation results of the Karaoke system.

Table 2
The performance improvements of Generalized Relay Mechanism over other approaches

Issuing rate (%)	Rejection rate (%)	Cost comparison	
		Excluding the cost of base system (%)	Including the cost of base system(%)
(a) Over intuitive approach (cost comparision (IA-GRM)/IA)			
1	5	28	14
	0.1	26	15
10	5	17	9
	0.1	11	7
(b) Over adding hard disk approach (cost comparision (AHD-GRM)/AHD)			
1	5	74	56
	0.1	75	60
10	5	89	82
	0.1	89	83

GRM – Generalized Relay Mechanism; IA – intuitive approach; AHD – adding hard disk approach.

sonable because the remaining programmes have a very low probability of being selected by a newly started access instance, which implies that the temporal locality of the data in such a programme is quite low.

Fig. 6 compares the effects achieved with the Generalized Relay Mechanism, the intuitive approach, and the approach of simply adding more hard disks in two simulation runs. The horizontal axis of Fig. 6 represents the costs of additional hardware added to the base system. In the first run, the issuing probability of an idle client is 1%. In the second run, the issuing probability of an idle client is 10%. In both cases, the Generalized Relay Mechanism yields better effect than the intuitive approach in terms of costs. It is shown in Table 2(a). If we use 0.1% as the acceptable level of the rejection rate, then the improvements of the Generalized Relay Mechanism over the intuitive approach in terms of costs are 26% and 11% for the first case and the second case, respectively.

Similarly, as shown in Table 2(b), if we use 0.1% as the acceptable level of the rejection rate, then the improvements of the Generalized Relay Mechanism over the approach of simply adding hard disks in terms of costs are 75% and 89% for the first and the second cases, respectively.

From Table 2(b), the results confirm that the merit of the Generalized Relay Mechanism largely depends on the temporal locality of the buffered data. Higher issuing probability means that more clients will be accessing programmes at a time, which implies the temporal locality of the buffered data is higher. Another factor that also has influence on the effectiveness of the Generalized Relay Mechanism is the total number of clients. With the same level of issuing probability, more clients means that the number of clients that will be accessing programmes at a time is larger. However, the same effect can be achieved by changing the issuing

Table 3
The simulation parameters for the Karaoke system

No. of clients	4000
No. of programmes	100
Duration of one service cycle	1.8 s
Duration of each programme	2 h
No. of disk groups	4
Capacity of hard disk	2 GBytes
I/O bandwidth of a hard disk	4 MBytes per second
Memory cost per MBytes	US$4
Hard disk unit cost	US$600
Simulation time	6 h
α value of distribution	0.75

Fig. 7. The simulation results of the movie-on-demand system.

Table 4
The performance improvements of GRM over adding hard disks approach

Issuing rate (%)	Rejection rate (%)	Cost comparision(AHD – GRM)/AHD	
		Excluding the cost of base system (%)	Including the cost of base system(%)
0.01	5	−43	−34
	0.1	−65	−51
0.025	5	5	4.6
	0.1	5	4.4
0.1	5	43	39
	0.1	38	34
1	5	85	75
	0.1	84	75

GRM–Generalized Relay Mechanism; AHD–adding hard disk approach.

probability as was done in these two simulation runs.

3.2. *The effects in a movie-on-demand system*

Table 3 gives the simulation parameters for the movie-on-demand system. The base system is assumed to have 16 hard disks, divided into four disk groups, along with an archiving storage system for storing all 100 movies. The hard disks store the most popular movies to its capacity limit. The number of programmes and the α value that defines the popularity distribution are based on the statistical data collected in a movie-on-demand application [12]. In the simulation, the Generalized Relay Mechanism is applied to only the most popular 15 movies.

Fig. 7 compares the effects achieved with the Generalized Relay Mechanism and the approach of simply adding more hard disks in four simulation runs. The intuitive approach of caching video programmes is not feasible in this case because the size of a movie is too large. In these four simulation runs, the issuing probability of an idle client is set to be 0.01%, 0.025%, 0.1%, and 1%, respectively. For the movie-on-demand application, lower issuing probabilities are used because a movie is much longer than a Karaoke song and thus a client may take longer breaks between successive service requests. Table 4 compares the costs of the system incurred by adopting the Generalized Relay Mechanism and the approach of adding hard disks. Again, the results confirm that the merit of the Generalized Relay Mechanism largely depends on the temporal locality of the buffered data. When the issuing probability is low, the Generalized Relay Mechanism fails to beat the approach of adding hard disks due to low temporal locality.

4. Conclusion

In this paper, we propose the Generalized Relay Mechanism to exploit semiconductor memory to upgrade the service capacity of video-on-demand systems. The design of the Generalized Relay Mechanism is aimed at optimizing allocation and tradeoff of the memory resource and the disk bandwidth resource. A simulation-based study shows that the Generalized Relay Mechanism enjoys a significant advantage over the intuitive approach of always filling up the memory with the most frequently accessed programmes. The simulation results also show that implementing the Generalized Relay Mechanism is a competitive alternative in comparison with simply incorporating more hard disks. The design decision really depends on the specification of the system and the clients' behavior. It is expected that exploiting semiconductor memory to upgrade the service capacity of video-on-demand servers will become a favorite approach in future as advances in semiconductor technologies continue to increase the density of memory chips and cut the costs at a fast pace.

Finally, the work presented in this paper is just a start of the effort to optimize allocation and trade-off of the memory resource and the disk bandwidth resource in a video-on-demand server. The Generalized Relay Mechanism basically employs a heuristic resource allocation algorithm. Further studies are needed to determine the nature of the problem, whether it is an NP-complete problem or not. Should the search be for an NP-complete problem, then better heuristic algorithms would be of interest.

[1] C. Federighi and L.A. Rowe, A distributed hierarchical storage manager for a video-on-demand system, in: *Proc. IS&T/SPIE Symp. on Electronics Imaging Science and Technology* (1994).

[2] P. Venkat Rangam and H.M. Vin, Designing file systems for digital video and audio, In: *Proc. 13th ACM Symp. on Operating System Principles* (1991) 81–94.

[3] P.S. Yu, M.S. Chen and D.D. Kandlur, Design and analysis of a grouped sweeping scheme for multimedia storage management, in: *Proc. 3rd Int. Workshop on Network and Operating System Support for Audio and Video* (1992) 44–55.

[4] Y.-J. Oyang, M.-H. Lee, C.-H. Wen and C.-Y. Cheng, Design of multimedia storage systems for on-demand playback, in: *Proc. 11th Int. Conf. on Data Engineering* (1995) 457–465.

[5] Y.-J. Oyang, C.-H. Wen, C.-Y. Cheng, M.-H. Lee and J.-T. Li, A multimedia storage system for on-demand playback, *IEEE Trans. Consumer Electron.* 41 (1) (1995)53–63.

[6] P. Lougher and D. Shepherd, The design of a storage server for continuous media, *Comput. J.* 36 (1) (1993) 32–42.

[7] J. Gemmell and S. Christodoulakis, Principles of delay-sensitive multimedia data storage and retrieval, *ACM Trans. Inform. Systems* 10 (1) (1992) 51–90.

[8] F.A. Tobagi, J. Pang, R. Baird and M. Gang, Streaming RAID – a disk array management system for video files, in: *Proc. 1st ACM Int. Conf. on Multimedia* (1993).

[9] A. Dan and D. Sitaram, Buffer management policy for an on-demand video server, in: *IBM Research Report*, RC 19347, Yorktown Heights, NY, 1993.

[10] A. Dan and D. Sitaram, A generalized interval caching policy for long video workloads, in: *IBM Research Report*, RC 20206, Yorktown Heights, NY, 1995.

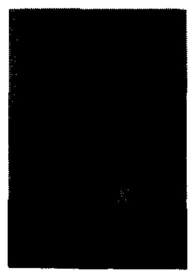
[11] C.H. Wen, C.Y. Cheng, M.H. Lee, F.C. Wang and Y.J. Oyang, A study of buffer cache management in video-on-demand system, in: *Proc. Int. Workshop on HDTV and the Evolution of Television* (Taipei, Taiwan, 1995).

[12] A. Dan and D. Sitaram, Generalized interval cacheing policy for mixed interactive and long video workloads in: *Proc. IS&T/SPIE Symp. on Electronic Imaging Science and Technology (Conference: Multimedia Computing and Networking 1996)* (1996).

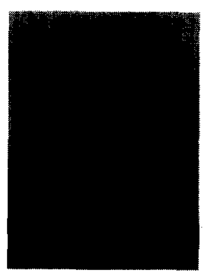
[13] G.R. Ganger, B.L. Worthington, R.Y. Hou and Y.N. Patt, Disk arrays: High-performance, high reliability storage subsystems, *IEEE Comput.* 27 (3) (1994) 30–37.

[14] International standard ISO/IEC 11172-1, The International Organization for Standardization and the International Electronical Commission, August 1993.

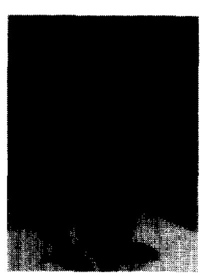
[15] L. Kleinrock, *Queueing Systems*, Vol. I (Wiley-Interscience, New York, 1975).



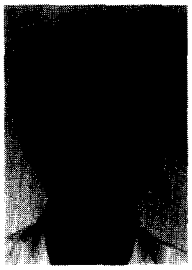
is currently a Ph.D. student in the Department of Computer Science and Information Engineering of National Taiwan University. He received his M.S. degree in Computer science from Polytechnic Institute of New York in 1977. He has been working with Computer and Communication Research Lab. of Industrial Technology Research Institute since 1978. His major interests include multi-media servers, set-top box, video-on-demand systems and expert systems.



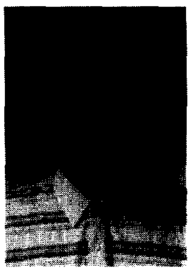
received the Ph.D. degree in Computer Science and Information Engineering from National Taiwan University in 1995, and the B.S. and M.S. degrees in Computer Engineering from National Chiao Tung University in 1987 and 1989, respectively. His research interests include microprocessor architecture, and multimedia computing systems. He is currently on the military service. He can be reached at chwen@solar.csie.ntu.edu.tw.



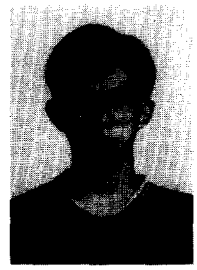
is currently a Ph.D. student in the Department of Computer Science and Information Engineering of National Taiwan University. He received the B.S. degree in Computer Science and Information Engineering from the same University in 1991, and the M.S. degree in 1993. His major research interests include multimedia storage systems, and operating systems.



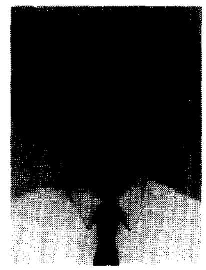
received the B.S. degree in Electrical Engineering from National Cheng Kung University in 1987, the M.S. degree in 1989, and the Ph.D. degree in Computer Science and Information Engineering from National Taiwan University in 1996. He is currently an Associate Professor in the Department of Information Management, Shih Chien College. His research interests include multimedia storage systems, operating systems, and expert systems. He can be reached at meng@iis.sinica.edu.tw.



is currently a graduated student at the Dept. of Computer Science and Information Engineering of National Taiwan University, Taipei, Taiwan. He received the B.S. degree in Computer Science and Information Engineering from National Chiao Tung University in 1995. His research interests are video- on-demand and image processing. He can be reached at how@solar.csie.ntu.edu.tw.



is currently on the military service. He received the B.S. degree at the Dept. of Computer Science and Information Engineering of National Taiwan University in 1995, Taiwan. His research interests are video on demand. He can be reached at wsc@solar.csie.ntu.edu.tw.



received the B.S. degree in Information Engineering from National Taiwan University in 1982, the M.S. degree in Computer Science from the California Institute of Technology in 1984, and the Ph.D. degree in Electrical Engineering from Stanford University in 1988. He is currently a Professor in the Department of Computer Science and Information Engineering, National Taiwan University. From 1989 to 1996, he was an Associate Professor in the same department. His research interests include design of video server systems and digital libraries. He can be reached at yjoyang@csie.ntu.edu.tw.