

An Efficient Disk-Array-Based Server Design for a Multicast Video Streaming System

Patton, P. H. Chan Jack, Y. B. Lee
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{phchan1, yblee}@ie.cuhk.edu.hk

Abstract

Recently, a number of researchers have started to investigate new video-on-demand (VoD) architectures using batching, patching and periodic broadcasting. These architectures, compared to traditional unicast VoD systems, are much more scalable and can serve thousands or even millions of clients concurrently. Nevertheless, existing studies are usually focused on architectural issues. The problem of designing an efficient server to implement these new multicast VoD architectures has received little attention. While existing server designs using round-based schedulers can still be used, results show that such designs are sub-optimal as they do not exploit the characteristics of fixed-schedule periodic broadcasting channels. This study addresses this challenge by presenting an efficient server design for a recent multicast VoD architecture called Super-Scalar Video-on-Demand (SS-VoD). Results show that the efficient server design can increase the system capacity by 60% compared to traditional video server designs. This paper presents details of this new server design, derives a performance model, and analyzes it using numerical results.

1. Introduction

Conventional video streaming services are commonly provided by true-video-on-demand (TVoD) systems where a dedicated channel is reserved for each and every client. While TVoD

possesses a simple architecture and allows interactive playback control, the scalability of the system is ultimately limited by the server and the network bandwidth. As every client in the system requires a dedicated channel for the whole video duration, the resources needed to serve thousands or even millions of clients are tremendous.

One promising solution to this scalability challenge is through the intelligent use of network multicast. Network multicast enables a server to send a few streams of video data for reception by a large number of clients, thereby significantly reducing the amount of resources required. A number of pioneering studies have investigated such architectures, such as patching [1-2], batching [3-6], and periodic broadcasting [2-4].

Recently, Lee and Lee [2] proposed a Super-Scalar video-on-demand (SS-VoD) architecture combining the virtues of batching, patching, and periodic broadcasting for implementing scalable and efficient VoD services. In a SS-VoD system, multicast channels are divided into two types - static channels and dynamic channels. Each channel transmits video data at the video playback rate using network multicast. Static channels are organized in a time-staggered manner to stream the whole video repeatedly and periodically. Dynamic channels are scheduled with batching and patching to enable clients to begin playback quickly. By simultaneously caching data from a static channel, the client can eventually merge back to an existing static channel and release the

dynamic channel for reuse by other clients. Results show that with the same resources, SS-VoD can achieve significantly shorter startup delay compared to conventional true-video-on-demand (TVoD) and near-video-on-demand (NVoD) architectures [2].

In this paper, we present an efficient disk-array-based server design for implementing the video server in a SS-VoD system. The video server in a SS-VoD system is unique in that there are both statically scheduled and dynamically scheduled video channels. Existing video servers in general [7-11], and disk schedulers in particular [12-15] are designed either for systems with statically scheduled video channels (e.g. NVoD) [3-6], or for systems with dynamically scheduled video channels (e.g. TVoD). The former simply cannot be applied to a SS-VoD system as the video placement policy and I/O scheduler typically do not allow random data retrievals. The latter, on the other hand, can still be applied to a SS-VoD system but the efficiency will be sub-optimal as the static channels' periodic retrieval patterns are not exploited to increase retrieval efficiency. In general, the problem of designing a server supporting *both* static batching channels and dynamic patching channels has not been addressed.

This study has two main contributions. First, we investigate the problem of designing an efficient server supporting both random and periodic data retrievals. In particular, we present a new video placement policy and retrieval algorithm that can support random data retrievals, while still be able to exploit the periodic data retrieval pattern to increase disk retrieval efficiency.

Second, we tackle the disk-zoning problem by incorporating a new Weighted Group Segment Pairing Scheme (WSGP) to the video placement policy. By pairing an outer zone with an inner zone and allocating video data according to the zone's storage capacity, we can achieve increased disk utilization without sacrificing disk storage capacity.

Compared to conventional server designs using round-based schedulers, this new efficient server design can increase the system capacity by as much as 60% with the same buffer requirement. This paper presents details of this new server design, derives a performance model, and analyzes it using numerical results.

2. Related Works

In this section, we review a number of previous studies on video server design and on disk zoning, and then compare them with this study.

Video server design has been studied extensively in the literature. Gemmell *et al.* [8] provides an excellent overview of the area explaining the key challenges and reviewing existing solutions. Most of the existing server designs for TVoD systems are centered around round-based algorithms, such as the CSCAN scheduler [13] and the Grouped Sweeping Scheme (GSS) scheduler [12]. Common among these round-based schedulers are the assumption that there is no correlation between the active video streams, i.e., the video streams playback video independently at arbitrary schedules. While this assumption is valid and necessary for TVoD systems, it is sub-optimal for a SS-VoD system where some of the multicast video streams are prescheduled and thus have a fixed temporal relation with one another.

On the other extreme of the spectrum is NVoD systems where all video streams are broadcast repeatedly and periodically in a fixed schedule [3-6]. Armed with complete knowledge of the broadcasting schedules, one can then design an optimized video placement and disk retrieval scheme to increase disk efficiency. The principle is to take advantage of the fixed temporal relation between broadcast video streams and place video data interleaved manner so that the server can retrieve video data continuously with minimal disk seeking.

For example, Chen and Manu [14] proposed a video placement policy called Segment Group Pairing (SGP) to allocate video data in zone-bit-recording (ZBR) disk for NVoD servers.

With SGP, data blocks that are to be retrieved in the same round are divided evenly into two groups. The first group is stored continuously in the outer zone while the second group continuously in the inner zone. This continuous placement reduces seeking overhead in data retrieval. In a service round, the disk head will first retrieve the group of blocks located in the outer zone and then seek to the inner zone to retrieve the remaining data blocks. This scheme enables the data rates of both zones to be averaged and thus results in a higher deterministic disk throughput.

Nevertheless, this algorithm did not account for capacity differences among different zones. In particular, inner zones usually have lower capacity compared to outer zones. As a result, SGP will likely fill up the inner zone before the outer zone is fully utilized and thus the remaining storage capacity in the outer zones become unused. To tackle this limitation, we extend the SGP policy to account for zone capacity differences. This new policy, called Weighted Segment Group Pairing (WSGP) allocates data blocks to the inner and outer zones proportional to the zone capacities, thus eliminating the above-mentioned limitation.

Nevertheless, the WSGP policy still does not address the problem of supporting both random and periodic video streams. In our server design, we develop a new placement policy and I/O scheduler incorporating the virtues of CSCAN/GSS for random video streams, and data interleaving for periodic video streams. In the next section, we first review the SS-VoD architecture. To facilitate comparison, we present a GSS-based server design in Section 4 and then introduce our new server design in Section 5. We then compare their performances using numerical results in Section 6 and summarize the study in Section 7.

3. The Super-Scalar Architecture

In this section, we briefly review the Super-Scalar video-on-demand architecture (SS-VoD) investigated in this study. Interested readers are referred to [2] for detailed design, analysis, and implementation of the architecture. SS-VoD is built on top of a multicast-enabled network with servers streaming video data using network multicast. Let M be the number of video titles served by the video server and let C be the total number of multicast channels available to a video server. For simplicity, we assume C is divisible by M and hence each video title is served by the same number of multicast channels, denoted by $N_M = C/M$. These multicast channels are then divided into two groups of N_S static multicast channels and $N_D = N_M N_S$ dynamic multicast channels.

A video is repeatedly multicast over all N_S static multicast channels in a time-staggered manner. Adjacent channels are offset by

$$T_R = \frac{L}{N_S} \quad (1)$$

seconds, where L is the length of the video title in seconds. Transmissions are repeated continuously, i.e. transmission will restart immediately from the beginning of the video when reaching the end of the video. These static multicast channels are used as the main channel for delivering video data to the clients. A client may start out with a dynamic multicast channel but it will be merged back to one of these static multicast channels to continue the video session until completion. To reduce the response time while still leveraging the bandwidth efficiency of multicast, SS-VoD reserves a portion of the multicast channels and schedules them dynamically to enable clients to begin playback quickly. When a new client arrives, it either waits for the next upcoming multicast transmission from a static multicast channel, or

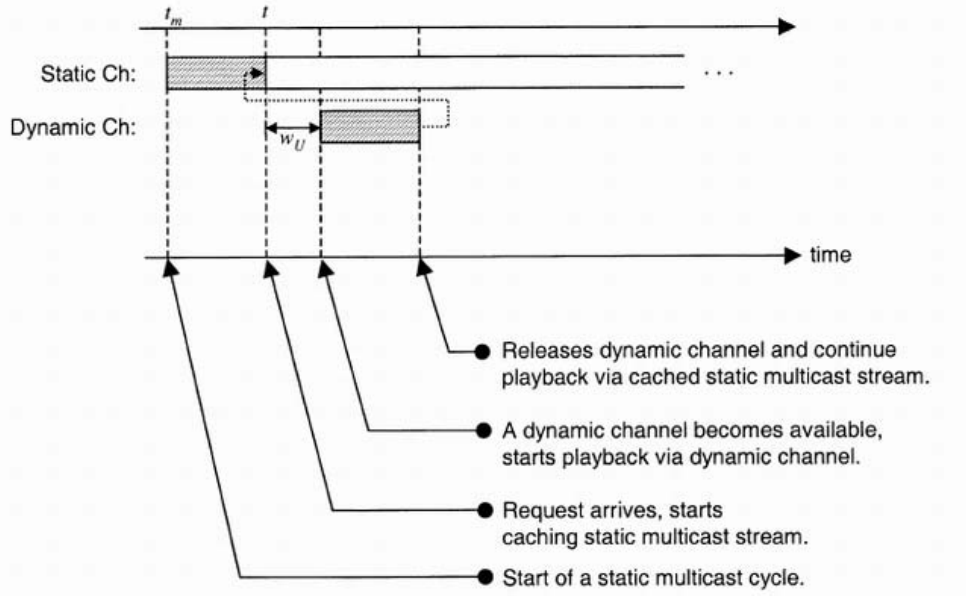


Fig. 1. Admission of client with channel merging using dynamic channels.

begins playback with a dynamic multicast channel. Specifically, assume the client enters the system at time t_0 , which is between the start time of the previous multicast cycle, denoted by t_m , and the start time of the next multicast cycle, denoted by t_{m+1} . The client will be assigned to wait for the next multicast cycle if the waiting time, denoted by w_i , is equal to or smaller than a predefined admission threshold δ . We call these requests *statically admitted*.

On the other hand, if the waiting time is longer than the threshold δ , then the client will request a dynamic multicast channel to begin playback (*dynamically admitted*), while at the same time caches video data from the static multicast channel with the multicast cycle started at time t_m as shown in Fig. 1. Note that the client may need to queue up and wait for a dynamic multicast channel to become available. If additional clients requesting the same video arrive during the wait, they will be batched together and served by the same dynamic channel. As a client caches video data from the static multicast channel during video playback, its playback will eventually reach the point where the cached data begin and from that point onwards the client can simply continue playback using data received from the static multicast channel and release the dynamic

multicast channel. This admission process increases the system's capacity significantly as the dynamic channels are occupied only for a short duration compared to the whole video session duration in TVoD. The study by Lee and Lee [2] showed that this SS-VoD architecture not only outperforms TVoD significantly (e.g. resource reduction over 90% in large-scale systems), but also can be scaled up to user population of any size.

4. A GSS-based Server Design

In this section, we apply the well-known Grouped Sweeping Scheme (GSS) [12] scheduler for use in a SS-VoD server. This design will serve as a baseline to compare the efficient server design to be presented in Section 5.

Let N be the number of disks in the system, assuming the disks are homogeneous. The disk's storage is divided into fixed-size blocks of Q bytes each, and a service group is defined to consist of all the data blocks at the same location from each of the N disks. Video data are striped across the N disks as shown in Fig. 2, effectively forming a RAID-4 [16] disk array without parity. Denote the j^{th} data block of video i by $b_{i,j}$. Then, the first N blocks of video i , $[b_{i,1}, b_{i,2}, b_{i,3}, \dots, b_{i,N}]$

are allocated to the first service group. This storage allocation scheme ensures load balance among all N disks.

Fig. 3 depicts the Grouped Sweeping Scheme (GSS) proposed by Yu *et al.* [12]. In GSS, a macro round is divided evenly into G micro rounds, with each micro round serving a separate group of video streams. Assuming all the videos are encoded using constant-bit-rate (CBR) encoding method with the same bit-rate R_V , then in each micro round the server retrieves one data block from each disk for each channel, and this data block is then multicast over the next G micro rounds (i.e. one macro round). Reducing G we can pack more video streams in a group and this results in increased disk efficiency, albeit at the expense of increased buffer requirement and scheduling delay and vice versa. In the extreme case with $G=1$, GSS reduces to SCAN; and in the other extreme case with $G=n$, where n is the maximum number of streams that the system can support, GSS reduces to first-come-first-serve.

In SS-VoD, static channel starts every T_R seconds. However, transmission in GSS can only start at the beginning of a micro round and hence does not necessary match the transmission schedule of the static channels. In particular, when T_R is not divisible by the micro round time, some of the static channels will not be able to transmit precisely at the scheduled time. To avoid this problem, we can use additional buffers to perform read-ahead to absorb the time differences. However, this will increase the buffer requirement up to 50% and is thus not desirable. Alternatively, we can choose the value of G and Q such that T_R is an integer multiple of the duration of a micro round to avoid the additional buffer requirement. For each video stream, N data blocks, one from each disk, are retrieved in a micro round for transmission over the next G micro rounds. Thus, the service round length

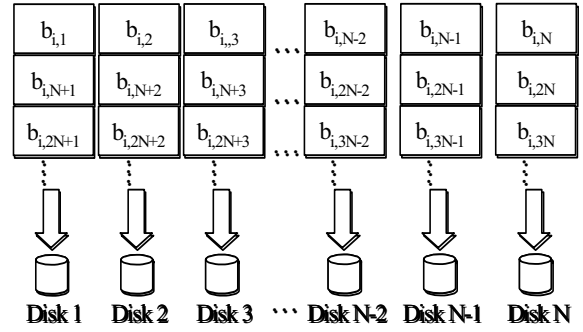


Fig. 2. Allocation of video blocks among disks for video i .

T_r is given by

$$T_r = \frac{NQ}{R_V}, \quad (2)$$

and the total server buffer requirement [12] is given by

$$B_{Server} = CNQ\left(1 + \frac{1}{G}\right), \quad (3)$$

where C is the total number of multicast channels in the server.

As the static channels' offset T_R is integer multiples of the micro round length, it is easy to see that the static channels will be equally distributed to all G groups. The remaining disk capacity is then used to support dynamic channels. Due to space limitation, we omit the derivations of the performance model for this GSS-based server design.

5. An Efficient Server Design

In SS-VoD, there are two types of multicast channels – static and dynamic multicast channel. Static multicast channels stream the whole video while dynamic channels serve clients with up to the first T_R seconds of the video only. In terms of data access pattern, static channels retrieve data in a fixed schedule. By contrast, the data access pattern of dynamic channels is random. In this section, we present an efficient design for the SS-VoD server. Specifically, this efficient server design has

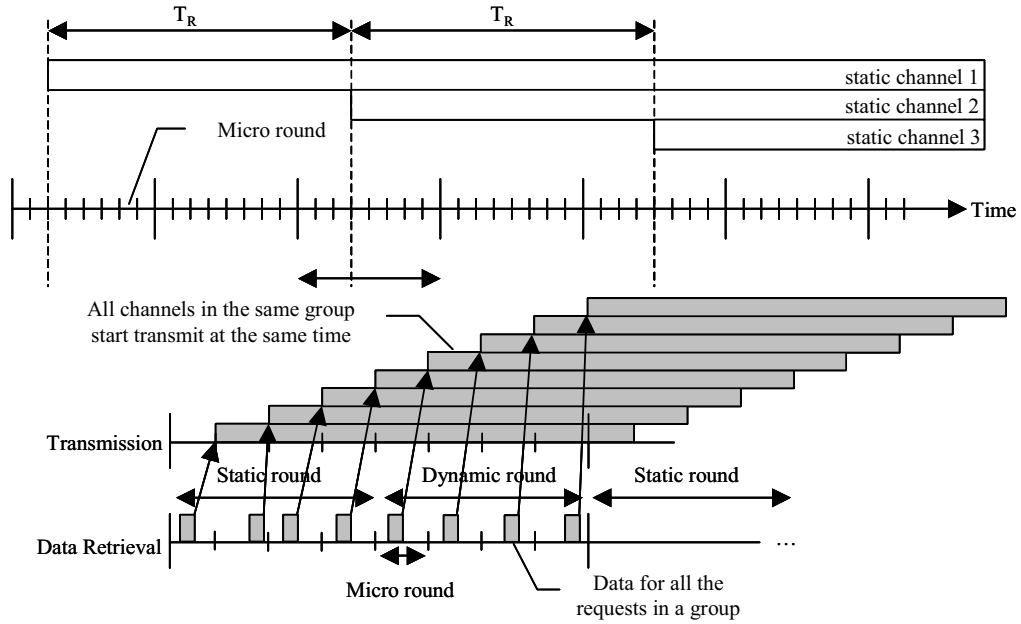


Fig. 3. Transmission and retrieval schedule of GSS.

three distinctive features. First, the disk storage is organized using an improved Weighted Segment Group Pairing (WSGP) scheme to exploit disk zoning to increase disk throughput and storage utilization. Second, an interleaving data placement policy is used to store video data to be transmitted over the static channels to exploit the static channels' periodicity. Third, the first T_R seconds of the videos are replicated for placement in the outermost zones to increase disk throughput for serving the dynamic channels. We design a new scheduler to schedule the data retrievals for both static and dynamic channels and quantify its performance. These are presented in details in the following sections.

A. The Weighted Segment Group Pairing (WSGP) Scheme

Today's hard disks commonly employ zone-bit-recording (ZBR) technique to increase disk capacity. In zoning, outer tracks are equipped with more sectors than inner tracks to exploit the increased disk surface area available. With a constant rotation speed, the data transfer rate of the outer zone is also higher than the inner zones. Traditional deterministic performance

analysis limits one to use the lowest data transfer rate in the innermost zone for system dimensioning and thus waste the higher transfer rate available in the outer zones.

To improve disk throughput, we devise a Weighted Segment Group Pairing (WSGP) scheme based on the Segment Group Pairing (SGP) proposed by Chen *et al.* [14]. In WSGP, the group of data blocks to be retrieved in the same round (i.e. a $G_{j,k}$) is divided into two sub-groups, denoted by $G_{j,k}^1$ and $G_{j,k}^2$. The first group is placed in an outer zone with higher transfer rate and the second group is placed in an inner zone with lower transfer rate. For a disk with Z zones, zone h and zone $(Z - h + 1)$ are paired together and the two sub-groups, $G_{j,k}^1$ and $G_{j,k}^2$, are allocated to these two zones respectively. In the original SGP [14] the groups are of equal size. This is undesirable as the zones often have different capacities and the extra capacity in the larger zone will be wasted. Thus we extend SGP to WSGP by dividing the group into sub-groups of sizes proportional to the zone capacities. Results show that with WSGP, we can achieve

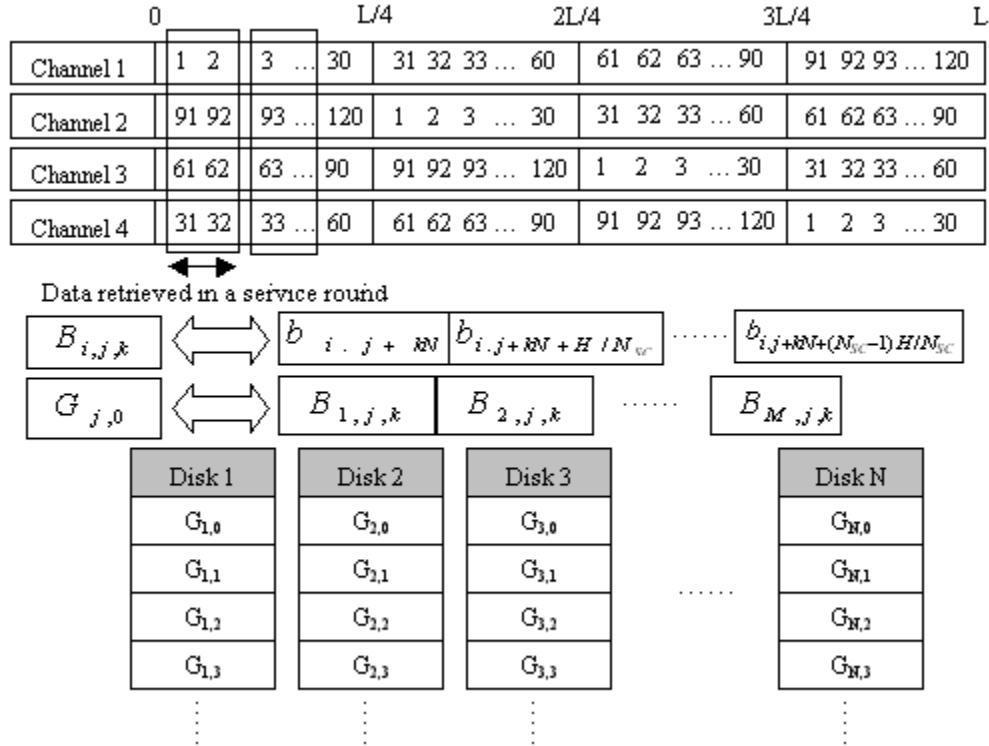


Fig. 4. Data access pattern and placement of static channels.

100% disk storage utilization while achieving same throughput as SGP.

B. Interleaving of Data Blocks

Next we consider the interleaving data placement policy for serving the static channels. Consider a system with $N=2$, $N_S=4$ and $k=120$ as shown in Fig. 4, illustrating the data access pattern for the static channels. We observe that data blocks $b_{i,j}, b_{i,j+30}, b_{i,j+60}, b_{i,j+90}$ of video i stored in disk 1 and data blocks $b_{i,j+1}, b_{i,j+31}, b_{i,j+61}, b_{i,j+91}$ stored in disk 2 are always retrieved together in the same service round, with $j \in [1, 3, 5, \dots, 29]$. Thus by placing these data blocks in a continuous portion of the disk surface, we can effectively eliminate the disk seeks required in conventional round-based schedulers. However, this placement policy only works for static channels where the transmission schedules are known and fixed. We address data retrievals for dynamic channels using replication in the next section.

C. First T_R Seconds Replication

Data retrievals for dynamic channels are more random in nature and hence the interleaving data placement policy offers no advantage. Moreover, with the WSGP policy in place, serving the dynamic channels using the interleaving data placement policy will result in the disk constantly seeking between outer tracks and inner tracks, further degrading disk throughput. To tackle this problem, we note that dynamic channels have one crucial property - it only serves up to the first T_R seconds of a video. Therefore we propose replicating the first T_R seconds of each video in the outermost zones of the disk, thereby taking advantage of the higher transfer rate of the outermost zones. Assume the first Z_{dyn} zones are used to store the replicated video data, then the WSGP algorithm will begin pairing zone $(Z_{dyn}+1)$ and Z . Fig. 5 illustrates the overall data layout.

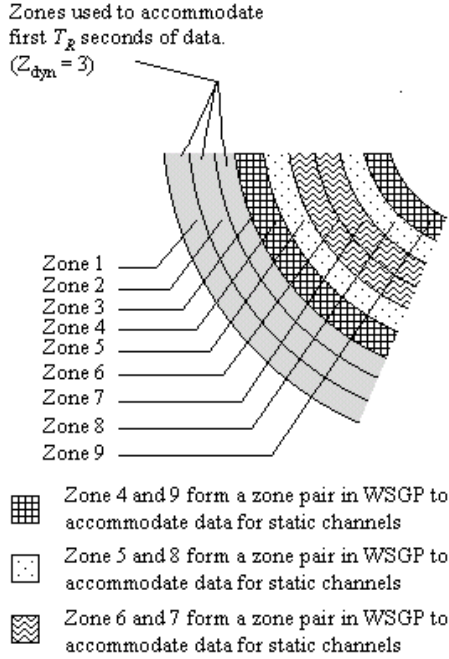


Fig. 5. Data layout for a disk with $Z = 9$ and $Z_{dyn} = 3$.

D. An Integrated Scheduler

To support data retrievals for both static and dynamic channels, we devise a new integrated scheduler based on the three previously-discussed design features. The integrated scheduler is still round based but each round is divided into two parts – a *static round* and a *dynamic round* as shown in Fig. 6. In a static round, two continuous data retrievals are performed, one for an outer-track block and one for an inner-track block. The retrieved data will then be used for transmission over the static channels. The dynamic round is further sub-divided into G_D dynamic micro-rounds and the dynamic channels are then assigned to these G_D dynamic micro-rounds as in the GSS case. Retrievals within a dynamic micro-round will be executed using SCAN and the retrieved data will be transmitted over the dynamic channels. The buffer requirement of this scheduler is thus given by

$$NQ(2N_S + N_D) \quad (4)$$

as illustrated in Fig. 7.

This scheduler, however, has a subtle problem. We found that the server buffer requirement is thereby dominated by the memory used to cache data for the static channels, which involved continuous retrievals for two large data blocks, increases the buffer requirement significantly. To reduce the buffer requirement, we sub-divide the static round into G_S micro-rounds of equal durations, where G_S equals to the number of videos. In each static micro-round, static channels belonging to the same video are scheduled and data are transmitted at the end of the static micro-round. Since different group retrieves data from different zone of the disk, the static micro-round will be of different durations. However, as we have shown in Fig. 8a and 8b, the buffer requirement of the server can still be obtained in the same way as in GSS.

With this modification, it can be shown that the total buffer requirement is reduced to

$$NQ(N_S(1 + \frac{1}{G_S}) + N_D(1 + \frac{1}{G_D})). \quad (5)$$

In practice, we can reduce the buffer requirement of static channels by 40%. Again we omit derivations of the performance model due to space limitation. We evaluate and compare performance of the proposed server design in the next section.

6. Performance Evaluation

In this section, we evaluate the performance of the presented efficient server design and compare it against the GSS-based server design. Table 1 lists the key system parameters used in the numerical calculations and Table 2 gives the specification of the disks used in performance evaluation.

A. Server Capacity

Fig. 9 plots the server capacity versus the server buffer size constraint for our efficient

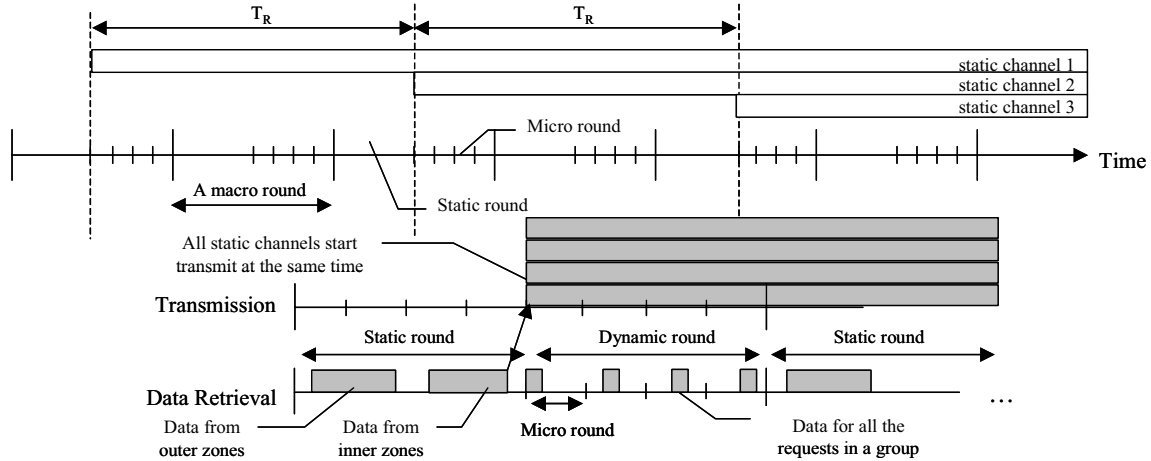


Fig. 6. Scheduling disk retrieval and Network transmission of WSGP

server design and the GSS-based server design for four different disks. Compared to GSS-based design, our design can increase the server capacity by up to 60%. Moreover, the performance gain increases to about 40% on average when the buffer available is more than 1GB as evident in Fig. 10. This is because the interleaved data placement policy reduces disk seeks substantially and thus the gain in I/O efficiency due to larger block size becomes more significant.

B. Utilization of Disk Capacity

The major advantage of using WSGP is that while achieving full utilization of disk storage capacity, it provides similar performance improvement as SGP. Results show that with the same system configuration, SGP can only utilize 91% storage capacity of the disk while WSGP can achieve 100% utilization.

7. Conclusions

In this paper, we presented an efficient disk-array-based server design for the Super-Scalar VoD system. By exploiting disk zoning, the periodicity of the static channels, and the shortened service duration of dynamic channels, using WSGP, interleaving data placement, and first T_R seconds replication respectively, we were able to increase the server capacity by as much as 60% compared to the conventional GSS-based server design.

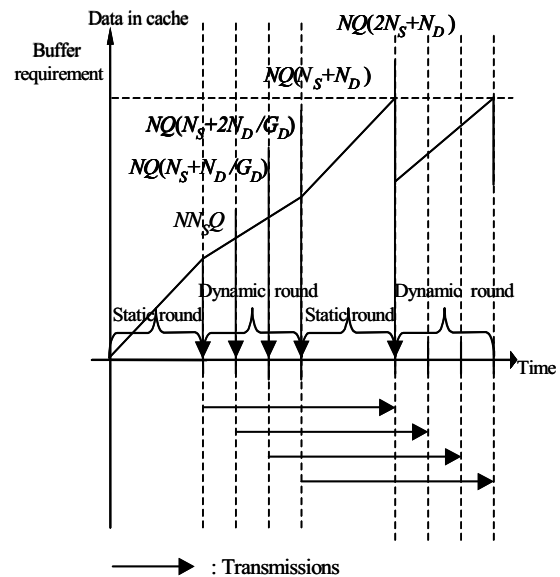


Fig. 7. Data in cache at different temporal positions in a service round.

While the server design presented in this study is specifically targeted for use in a SS-VoD system, the design principles are general and thus can be applied to other multicast video streaming architectures with both periodic and aperiodic multicast streaming channels.

Acknowledgements

This research is funded by a Direct Grant, and Earmarked Grants (CUHK 4209/01E and 4328/02E) from the HKSAR Research Grant

Council and the AoE-IT, a research grant from the HKSAR University Grants Council.

References

[1] Jack Y. B. Lee, "UVoD - A Unified Architecture for Video-on-Demand Services," *IEEE Communications Letters*, vol.3 (9), September 1999, pp.277-279.

[2] Jack. Y. B. Lee and C. H. Lee, "Design, Performance Analysis, and Implementation of a Super-Scalar Video-on-Demand System," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12(11), November 2002, pp.983-997.

[3] S. Viswanathan and T. Imielinski, "Metropolitan Area Video- on-Demand Service Using Pyramid Broadcasting," *IEEE Multimedia Systems*, vol. 4, pp.197-208, 1996.

[4] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," *Proceedings of the ACM SIG-COMM '97*, Cannes, France, Sep 1997, pp.89-100.

[5] C.C. Aggarwal, J.L. Wolf, and P.S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," *IEEE Proceedings of the International Conference on Multimedia Computing and Systems*, pp. 118-126, Jun 1996.

[6] L. S. Juhn, and L. M. Tseng, "Harmonic Broadcasting for Video-on-Demand Service," *IEEE Transactions on Broadcasting*, vol. 43(3), Sep 1997, Page(s): 268-271.

[7] Klaus Breidler et al., "A Comparative Study of Selected Parallel Video Servers," *Proceedings of IEEE 2000 11th International Workshop on Database and Expert Systems Applications*.

[8] D.J. Gemmell, H.M. Vin, D.D. Kandlur, P.V. Rangan, and L.A. Rowe. "Multimedia Storage Servers: A Tutorial". *IEEE Computer*, 28(5):40-49, May 1995

[9] W.J. Bolosky et al., "The Tiger Video Fileserver," *Proceedings of Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*, IEEE Computer Society Press, Los Alamitos, Calif., 1996.

[10] Zheng-Ru Lin and Ming-Syan Chen, "Design and performance study of scalable video storage in a disk-array-based video server," *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference*, vol.3, 2000 pp. 1341-1344

[11] Jin B. Kwon and Heon Y. Yeom, "Generalized Data Placement For Periodic Broadcast of Videos," School

of Computer Science and Engineering Seoul National University Seoul, South Korea 151-742.

[12] P.S. Yu, M.S. Chen and D.D. Kandlur. "Grouped sweeping scheduling for DASD-based multimedia storage management," *ACM Multimedia Systems*, 1(3): 99-109, 1993.

[13] Narasimha Reddy and A.L.; Wyllie, "I/O issues in a multimedia system," *J.C. Computer, Volume: 27 Issue: 3*, March 1994 Page(s): 69 -74.

[14] S. Chen and T. Manu, "A Novel Video Layout Strategy for Near Video-on-Demand Servers," *Proceedings of IEEE International Conference on Multimedia Computing and Systems' 97*, Ottawa, Ont., Canada, pp. 37-45, June 1997.

[15] Shiao-Li Tsao; Yueh-Min Huang, "An efficient storage server in near video-on-demand systems," *Consumer Electronics, IEEE Transactions*, Volume: 44 Issue: 1, Feb. 1998.

[16] Garth A. Gibson, "Redundant Disk Arrays: Reliable, Parallel Secondary Storage," *The MIT Press*, 1992.

System Parameter	Symbol	Value
Video data rate	R_V	4Mb/s (Mpeg 2)
Number of disk	N	8
Length of video	L	7200s
Static channel per video	N_S	20
Dynamic channel per video	N_D	20

Table 1. System parameters used in performance evaluation.

Disk Parameter	Value			
Disk Model	Atlas 10K	Barracuda	Cheetah 9LP	IBM 18es
Disk rotation speed (rpm)	10025	7200	10045	7200
Full strobe seek time (ms)	10.828	16.679	10.627	12.742
Track to track seek time (ms)	1.245	1.943	0.831	1.086
Head switching time (ms)	0.176	0.100	0.030	0.062
Number of data surfaces	6	5	12	5
Blocks per disk	17938986	4110000	17783240	17916240
Total number of tracks	10022	5172	6962	11474
Number of zones	24	11	11	55

Table 2. Specification of different disks used in performance evaluation.

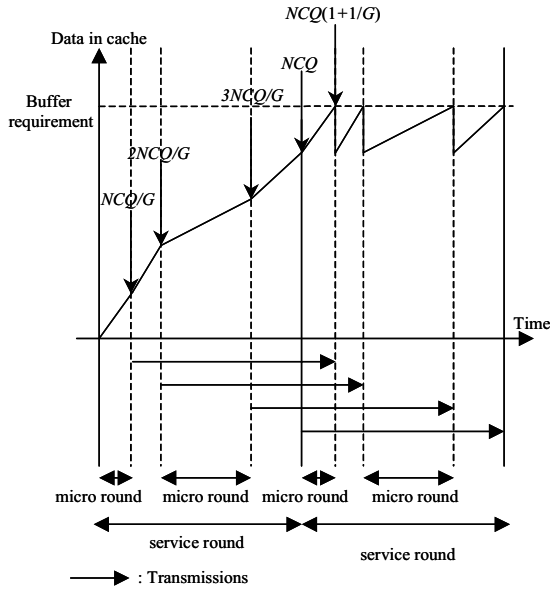


Fig. 8a. Server buffer requirement when micro rounds are of different durations. ($G=4$)

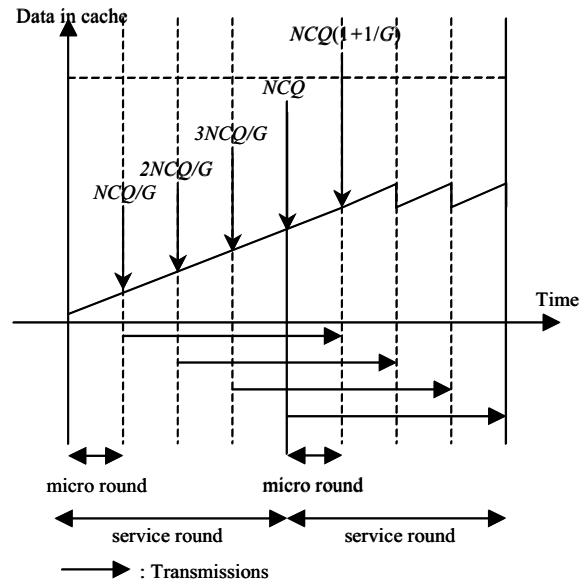


Fig. 8b. Server buffer requirement when micro rounds are of same durations. ($G=4$)

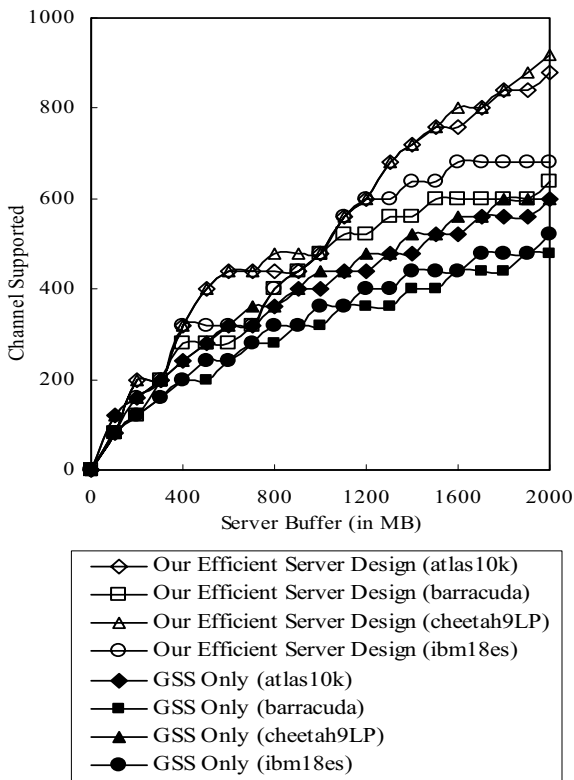


Fig. 9. Server capacity versus server buffer constraint for four hard disk models.

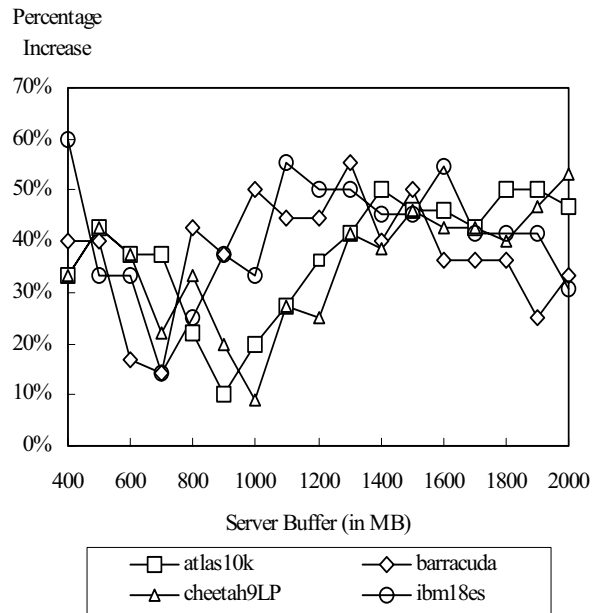


Fig. 10. Percentage increase in number of channels supported compared with GSS.