An Efficient Periodic-Request-Grouping Technique for Reduced Seek Time in Disk Array-based Video-on-Demand Server

Woonseok Kim
School of Computer Science and Engineering
Seoul National University
Seoul, Korea, 151-742

Jihong Kim
School of Computer Science and Engineering
Seoul National University
Seoul, Korea, 151-742

Sam H. Noh
Department of Computer Engineering
Hong-Ik University
Seoul, Korea, 121-791

Sang Lyul Min
School of Computer Science and Engineering
Seoul National University
Seoul, Korea, 151-742

ABSTRACT

Disk throughput is significantly affected by scheduling of disk I/O requests. In the case of VoD (Video-on-Demand) servers, disk throughput is directly related to the number of user requests that can be served simultaneously. In this paper, we propose an efficient periodic request grouping scheme for disk array-based VoD servers to improve the disk throughput. To reduce the disk seek time that are needed in processing the periodic I/O requests, this scheme groups the periodic requests that access adjacent regions into one, and arranges the groups in a pre-determined order. We show that by using this scheme, we can reduce the average disk bandwidth required by a stream and serve more periodic requests simultaneously than existing schemes. We also propose an adaptation technique that conforms the proposed scheme to access pattern changes of user requests. We performed simulation studies to evaluate the performance of the proposed scheme.

Keywords: Multimedia Server, RAID(Redundant Arrays of Independent Disks, Request Grouping.

1. INTRODUCTION

In recent years, we have witnessed significant advances in both networking technology, and computation technologies involving the digitization and compression of video. As a result, a growing number of applications need access to video data stored in digital form on secondary storage devices (e.g., video-on-demand, multimedia messaging). Especially, the advent of internet makes such multimedia applications easy to use the data distributed geographically distributed.

Video data such as MPEG-1 format data has several characteristics that are different from common data: voluminous and time-critical. Generally, a movie file requires 0.5 GB to 20 GB of storage. If we use a set of chips such as DRAM or flash memory to store a number of video data, the cost of the server system would be prohibitively expensive. Hence, it is natural to use disks as secondary storage in this systems. However, since disks have limited capacity and relatively high latency for data access, disk arrays are commonly used in the server to achieve huge capacity and higher server throughput.

In the case of VoD(Video-on-Demand) servers, due to the timing constraint in data transmission, only a limited number of users are allowed to be serviced. Such restriction usually comes from the data access latency in disks, and the disk throughput is significantly affected by scheduling of disk I/O

requests. In disk arrays such as RAID(Redundant Arrays of Independent Disks) [1], an I/O request is divided into several sub-requests that access different disks. In particular, if there are I/O requests for multimedia data, the I/O request will be divided into a number of sub-requests, and the sub-requests will access the disk arrays periodically for a long time [2, 3].

Generally, to increase the disk throughput, some optimization algorithms such as SCAN or CSCAN are used in disk scheduling. However, since such algorithms optimize the disk latency only for a single disk, and do not consider the effect of periodic accesses, the throughput cannot be fully utilized as in [4]. That is, in disk arrays, when I/O requests are scheduled, the interference between adjacent disks should be considered.

In this paper, we propose an efficient periodic request grouping scheme for disk array-based VoD servers to improve the disk throughput. To reduce the disk seek time that are needed in processing the periodic I/O requests, this scheme groups the periodic requests that access adjacent regions into one, and arranges the groups in a pre-determined order(e.g., in left-symmetric or right-symmetric fashion). We show that by using this scheme, 1) we can reduce the average disk bandwidth required by a stream, and 2) the server can serve more periodic requests simultaneously than existing schemes. Experimental results show that the number of streams, with marginal increase in memory requirement, were increased by 9.7% where the seek time occupies 31% of a block access delay. We also propose an adaptation technique that conforms the proposed scheme to access pattern changes of user requests. We performed simulation studies to evaluate the performance of the proposed scheme and to confirm that the server sustain the change of user access pattern.

2. SYSTEM MODEL AND ASSUMPTIONS

To achieve fault tolerance and high throughput, striping technique is generally used in disk array-based systems such as RAID. The storage model used in this paper is based on a coarse-grained striping model described by Ozden, Rastogi and Silberchatz [4] as shown in Figure 1.

For discussion purpose, we assume a disk array that consists of eight disks and contains 12 video files as shown in

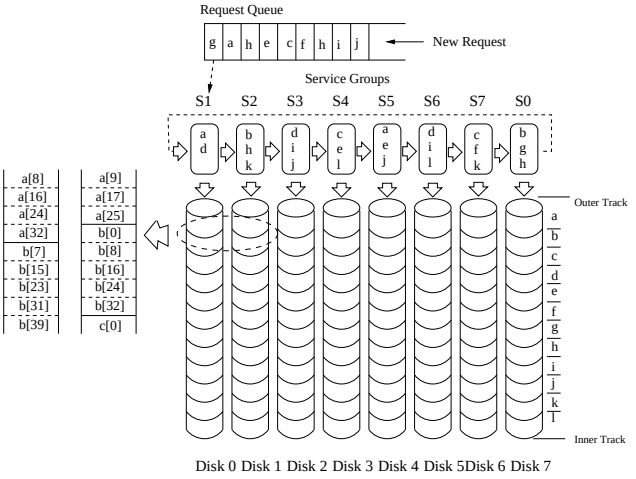

Figure 1. System Model

Figure 1. In Figure 1, a video file consist of several *data blocks* which are distributed over the disks in round-robin fashion. For example, if the first data block of a video file is stored in k-th disk, then the second data block is stored in $k+1$-th disk, and so on. Thus, if the server read a data block from k-th disk for a user request, then it will read a next data block from $k+1$-th disk. The video file names are denoted $a \sim l$, where a[5] refers to the sixth data block of video file named 'a'. If a client request for video b arrives, the server retrieves the first data block of video file b from the disk containing b[0]. Then, the server will retrieve the subsequent data blocks from other disks in round-robin fashion. As the client receives the data block, it will decode the data and display the result at a pre-specified rate r_{disp}[1]. As long as the server can transmit the subsequent data block before the client consumes the prior one, the display behavior will not be effected. This implies that the server can retrieve data blocks from the disk array at fixed intervals, and the client request can be assumed to be periodic requests. The interval, then, can be used as the period length, L_{period} (in seconds), and calculated as follows where b is the size of a data block (in bits):

$$L_{period}\ (\text{sec}) = \frac{b\ (\text{bits})}{r_{disp}}, \tag{1}$$

where r_{disp} is the rate of display (in bits per second).

For example, if the data block size is 192 KB and the video data was encoded in 1.5 mbps MPEG-1 format, the

[1]In this discussion, for simplicity, we assume the Constant-Bit-Rates (CBR) scheme

Table 1. Disk parameters and other notations

q	the number of periodic requests in a service group
$f_{seek}(dist)$	disk seek time for the track distance $dist$
$T_{rotation}$	average rotational latency
$R_{transfer}$	inner track transfer rate (bps)
T_{settle}	disk head settle time
$LS_{i,j}$	track number of data block needed by jth scheduled request in ith service group
$LE_{i,j}$	track number under disk head after retrieving data block needed by jth scheduled request in ith service group

server can only transmit one block to the client roughly every second. If we assume that the server can retrieve a data block from a disk in 0.3 seconds, this implies that the server could retrieve at least three data blocks from the same disk every period. We define a set of periodic requests that want to access the same disk in any period as a *service group*. In this example, since there are eight disks, eight service groups are available and each of which use one different disk at any given time. Thus, overall, the server can serve 24 periodic requests concurrently during a period.

By increasing the block size, we increase the L_{period}, further increasing the number of periodic requests that can be served simultaneously. But, since, in general, data stored in the disk cannot be transmitted directly to the client via network and must be loaded into memory first, the larger block sizes require larger amount of memory resulting in increased system cost [5]. The cost effective optimal block size can be calculated by using the method described by Ozden et al. [4].

When accessing data stored in a disk, we may undergo some delay caused by the known mechanical behavior(e.g. seek, rotation, and transfer delays). In disk arrays, the disk access delay for serving a set of periodic requests can be described as follow by using notations described in Table 1.

$$\begin{aligned} T_{access} &= f_{seek}(|LE_{i-1,last} - LS_{i,0}|) \\ &+ \sum_{j=0}^{q-2} f_{seek}(|LS_{i,j} - LE_{i,j-1}|) \\ &+ q(T_{rotation} + \frac{b}{R_{transfer}} + T_{settle}) \quad (2) \end{aligned}$$

As shown in this equation, seek time can be divided into two parts: $f_{seek}(|LE_{i-1,last} - LS_{i,0}|)$ and $\sum_{j=0}^{q-2} f_{seek}(|LS_{i,j} - LE_{i,j-1}|)$. The former term is the seek delay caused when a disk serves the first scheduled request of service group i. This is affected by the last scheduled request of the service group that used the disk during the previous period. The latter term is the seek delay caused when a service group retrieves the data blocks for its own periodic requests within a period. This depends on the disk scheduling policy that the system adopts. In general, to minimize this value, CSCAN or SCAN algorithms are used.

To guarantee that clients receive its subsequent data on time, the number of periodic requests q should be restricted. That is, the amount of time should not exceed the length of the period (i.e. $L_{period} \geq T_{access}$).

Without increasing the memory requirement, our goal is to maximize the q by reducing seek delays that are caused when each service group accesses a disk for its periodic requests within a period.

3. PERIODIC REQUEST GROUPING TECHNIQUE

Note from Eq. (2), that the seek delay can be reduced in the following two cases.

- The initial seek time, that is, the first term $f_{seek}(|LE_{i-1,last} - LS_{i,0}|)$ is reduced, when at the beginning of a period, the disk head position is closer to the track on which the data block needed by the first scheduled periodic request is stored.
- Within a period, the second term $\sum_{j=0}^{q-2} f_{seek}(|LS_{i,j} - LE_{i,j-1}|)$ is reduced, as the periodic requests included in each service group require physically adjacent data block.

To make the system serve periodic requests in this way, we set the system as follows :

- Divide each disk area into P service areas. Let each service group to be dedicated to one of P service areas. (For fair partitioning of service area, P should be the exact divisor of the number of disks [2]). In Figure 2(a),

[2] If P equal to 1, it means the normal case that uses the SCAN policy.

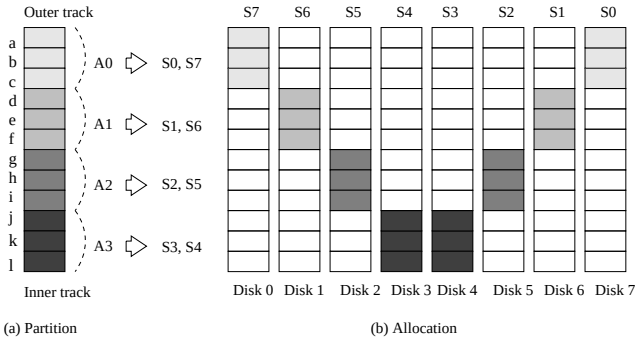

Figure 2. Disk Partition and Allocation

service groups $S0$ and $S7$ are dedicated to service area $A0$, service groups $S1$ and $S6$ to service area $A1$, and so on.

- Arrange service areas in rotational fashion among the disks. (When the movement of the disk head is toward the outer track, service areas are arranged in a left-symmetric fashion. Otherwise, service areas are arranged in a right-symmetric fashion.) This is shown in the Figure 2(b).

In our scheme, each service group only includes periodic requests that access the data block stored on its service area. Consequently, the first scheduled periodic request in a service group and the last scheduled periodic request in the previous group access adjacent data blocks. Ideally, the whole disk area is scanned only once during P periods, and the seek delay is minimized. For example, when a client request that requires the video e, it first queued in the request-queue. The server checks the service group $S1$ and $S6$ to see whether they are available since they are responsible for servicing I/O requests to the service area containing the video file e. If $S1$ is avalilable, the request are moved from request-queue to service group $S1$. Then, if e[0] is stored in the disk 2, $S1$ start service for the new request when accessing the disk 2. Since $S1$ contains requests only for the video d, e and f, it will retrieve the data with less seek delays. Also, since $S2$ that used the disk 2 during prior period may contains requests only for the video g, h and i, if the new request is scheduled first, the server retrieves data for $S1$ from the disk 2 in this period with less seek delay. Figure 3 and Figure 4 show the examples of disk arm movement. In these figures, the boxes represent the disks, and the upper(lower) side of the boxes are the outter(inner)

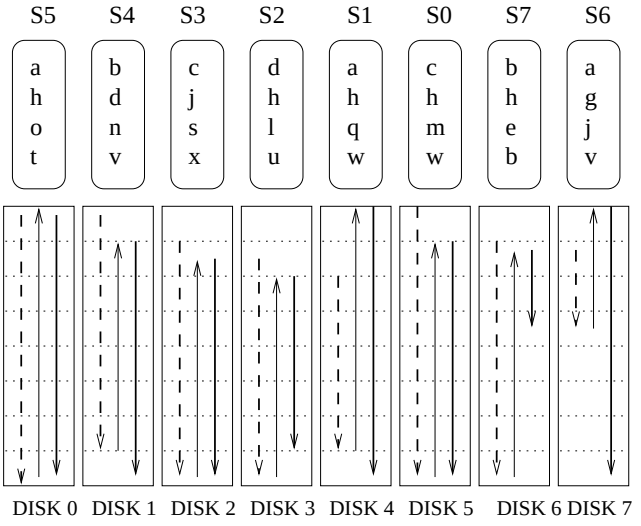

Figure 3. Example of Arm Movement in Disks ($P = 1$)

tracks of disks. In each box, the dotted line is the seek range which were resulted from the previous service group. The left straight line in the box shows the arm movement needed to read the data block which should be read in first accoring to the CSCAN scheduling policy for the current service group. The right straight line shows the arm movement needed to read the data blocks for the current service group. The figures shows that, when service groups are dedicated to serve data access in short range of disk area, the disks have reduced seek delay.

One problem with this configuration is that it does not consider the fact that each service group can be under different workload. It is known that some videos are more popular than others [6, 7]. Hence, the service area that contains the more popular videos will be frequently accessed, and the service groups dedicated to those areas will be under higher load than others. Furthermore, client requests that require popular videos can be stalled in situations where the load on the whole is relatively low. To remedy this problem, the popularity of each video file must be considered when we set the service area.

To balance the workloads among service groups, we adjust service areas as follows:

- Gather the popularity information for video based on the arrived client requests and estimate the popularity of each service area.

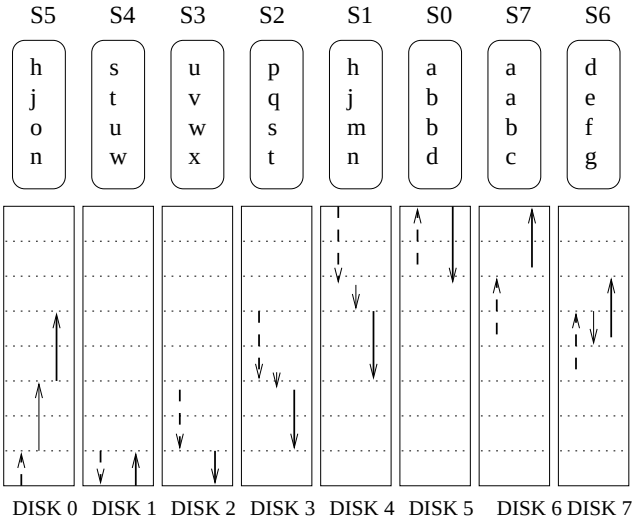

Figure 4. Example of Arm Movement in Disks ($P = 8$)

- Adjust the number of video files that are contained in each service area to balance the workload based on service area popularity. (For example, enlarge the service area if its popularity is higher than that of others)

Since the popularity of each video can be changed dynamically, this adjustment should also be done dynamically.

4. EXPERIMENTAL RESULTS

We performed simulation to evaluate the performance of the proposed scheme. This simulation was done with three different block sizes and we assumed the VoD server with 100 disks. The capacity of each disk is assumed to be 1.5 GB [8]. To simulate the disk operation, we use the disk simulator module in $raidSim$ (RAID simulator) [9]. We assume 75∼170 minutes length of video files.

Figure 5 and Figure 6 shows the experimental results. As shown in Figure 5, the number of streams increases with the increasing number of service areas. In the case of 192 KB(it is close to two track size of common disks), the number of streams increased by roughly maximum 10%. Also, the average user waiting time was reduced along with the increased number of streams. It is shown in the Figure 6. From the two figures, we can see that as the data block

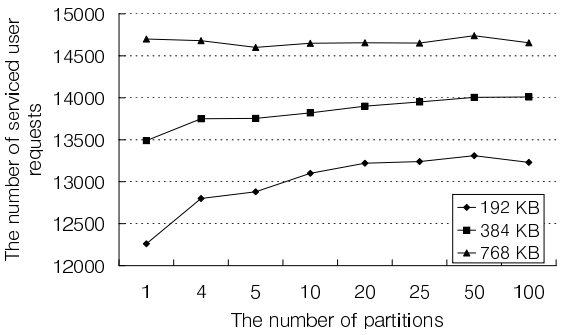

Figure 5. Results for Admission Capacity

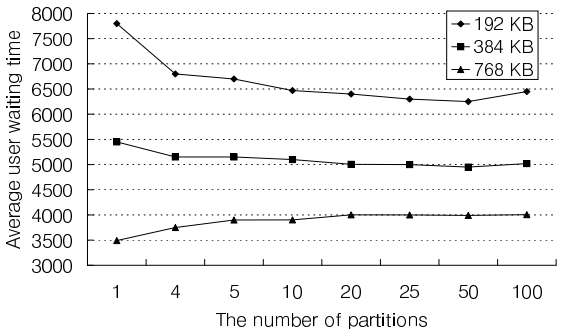

Figure 6. Results for Service Latency

size increases, the number of serviced streams increases, but the benefits from the partitioning decreases. It can be addressed as follows. Since the larger block size requires longer transfer time, the service time per stream can be larger than the reduced seek time. In this case, the benefits from the reduced seek time cannot be used to support additional user requests.

We also perform the simulation to evaluate the effect of user access pattern and its change. Figure 7 shows the effect of adaption in popularity variation. In this experiment, the result were logged after 10,000 seconds of initial period which were needed to regulate the workload over disks. In this initial period, popularities of video files conform to Zipf distribution [6], so that each video file may have different popularity from others. After the initial period, the popularity distribution was changed to uniform distribution

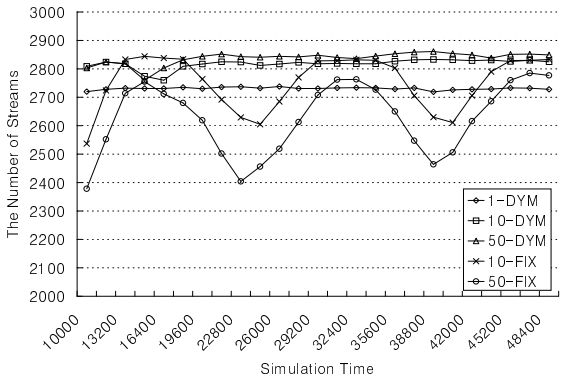

Figure 7. Results for Adaptation in Workload Variation

- entire video files have similar popularity. Then, 8,000 seconds later, the popularity distribution was re-changed to Zipf distribution. Such changes in popularity were repeated at every 8,000 seconds.

In Figure 7, 'n-DYM' means that the server is configured with n partitions and performs adaptation to the changed popularity, while 'n-FIX means no adaptation with n partitions. As shown in the result, our adaptation technique balances the load overall service groups, and, regardless of the popularity variation, the server gives stabilized performance.

5. CONCLUSION

In this paper, we proposed an efficient periodic request grouping technique in disk array. This scheme reduces the disk seek delay that are needed in serving periodic I/O requests. To achieve this, this scheme groups the periodic requests that access adjacent regions into one, and arranges the groups in a pre-determined order. We also presented the adaptation technique for variation of user access patterns. Experimental results showed that our scheme maximized the number of streams that can be served simultaneously without increasing the memory requirement per stream, and our adaptation technique made the system to sustain the change of user access patterns.

References

[1] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks(RAID). In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 109–116, January 1988.

[2] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered Striping in Multimedia Information Systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 79–89, June 1994.

[3] M. S. Chen, D. D. Kandlur, and P. S. Yu. Optimization of the Grouped Sweeping Scheduling(GSS) with Heterogeneous Multimedia Server. In *Proceedings of the ACM Multimedia'93*, pages 235–242, August 1993.

[4] B. Ozden, R. Rastogi, and A. Silberschatz. Disk Striping in Video Server Environments. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 580–589, June 1996.

[5] P. Shenoy and H. M. Vin. Efficient Striping Techniques for Multimedia File Servers. In *Proceedings of the 7th International Workshop on NOSSDAV*, pages 25–36, May 1997.

[6] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In *Proceedings of the 2nd Annual ACM Multimedia Conference and Exposition*, pages 15–23, October 1994.

[7] C. Griwodz, M. Bar, and L. C. Wolf. Long-term Movie Popularity Models in Video-on-Demand Systems or The Life of an on-Demand Movie. In *Proceedings of the ACM Multimedia'97*, pages 349–357, November 1997.

[8] Quantum
Corporation. Quantum Viking Online Specifications. http://www.quantum.com/products/hdd/viking/.

[9] E. K. Lee. *Performance Modeling and Analysis of Disk Arrays*. PhD thesis, Computer Science, University of California at Berkeley, 1993.