An Adaptive Scheduling Mechanism for Real-Time Streaming Servers

HYUNJUN LEE, KYUNGOH LEE, YOUNGHAN LEE
Dept of Computer Science
Sunmoon University
Tangjungmun Asan Chungnam
Republic of Korea

Abstract: - An innovative dynamic scheduling scheme is proposed to improve the efficiency of video-on-demand servers. We first introduce a paged segment striping model that makes dynamic scheduling possible. Based on this striping scheme, we propose a dynamic scheduling scheme that adapts to frequently changing workloads. In particular, we can change the round length without any additional disk access so that it can be adapted to changing request trends with a negligible cost in performance. This dynamic scheduling scheme always shows better performance than the static scheduling scheme in simulation. Although the dynamical scheme introduces additional scheduling overhead, it is very small when compared with the performance degradation in the static scheme.

1 Introduction

Storage servers for digital movie retrieval are the central components of many multimedia systems. The designers of such servers attempt to maximize performance while minimizing cost. Due to the immensity multimedia objects and their consequent data transfer requirements, these multimedia servers will undoubtedly be founded on large disk arrays. Disk arrays achieve high performance by servicing multiple concurrent I/O requests, and utilize several disks to service even single requests in parallel.

Due to the periodic nature of media playback a multimedia server can service multiple clients simultaneously by proceeding in rounds, the duration of which (known as the round length) is governed by the response time requirements of clients [8,14]. During each round, the server retrieves a fixed number of media units (e.g., video frames or audio samples) for each client. To ensure continuous playback, the amount of data accessed for each stream during a round must be large enough to meet their respective playback requirements. Furthermore, the service time (i.e., the total time spent in retrieving media units during a round) should not exceed the duration of the round [16]. To effectively utilize a disk array, and hence to maximize the number of clients that can be serviced simultaneously, a multimedia server must interleave the storage of each media stream among the disks in the array [3, 9, 15, 17]. The unit of data interleaving, referred to as a striping unit, denotes the maximum amount of logically contiguous data that is stored on a single disk. Successive striping units of a stream are placed on consecutive disks using a round-robin allocation algorithm.

Recently, several researchers have developed techniques to find the optimal striping unit size [4, 6, 12]. The optimal value may vary significantly if heterogeneous streams (such as HDTV streams, MPEG-1 movies and audio streams) are serviced; that is, an optimal value in one situation may be far from optimal in other situations. If most streams have a low data rate (say, 1.5 M bps, as in MPEG-1) a long round length will show better performance, but if most streams have high data rate (say, 10 M bps, as in HDTV) a short round length will show better performance. Low data rate streams require relatively small buffer space but more frequent disk accesses as long as the round length is suitable; however, since high data rate streams require much more buffer space, the latter becomes the bottleneck of performance. In this case, a short round length will show better performance [11].

It would be convenient if these systems exhibit a single optimal round length. Unfortunately, usage trends in video-on-demand vary frequently as time

passes. This implies that the round length should also be allowed to vary. If changing the round length induces additional disk seeks, then the performance may become worse. But, in the PSS (Paged Segment Striping) scheme that we suggest, we can change the round length without any additional disk access. Based on this striping model, we propose a dynamic scheduling algorithm in which the round length can be adaptively changed according to the changing request trends. In particular, we can improve the performance by enlarging (reducing) the round length if there is enough available buffer space (disk bandwidth) but insufficient disk bandwidth (buffer space). We have verified that the proposed dynamic scheduling scheme always shows better performance than a static scheduling scheme through extensive simulation. Furthermore, the additional scheduling overhead of the dynamic scheme is relatively small.

The rest of this paper is organized as follows. Section 2 introduces the striping model, section 3 explains the dynamic scheduling scheme, section 4 shows the results of our experiments and, finally, section 5 presents our conclusions.

2 STRIPING MODELS

Since fine-grained striping schemes waste disk bandwidth due to an excessive number of disk seeks, they are typically outperformed by coarse-grained schemes [3]. Constant time length (CTL) schemes also show better performance than constant data length (CDL) schemes [4], since CDL schemes tend to require much more buffer space [7]. Accordingly, we adopted a coarse-grained CTL as our basic striping scheme. Reducing the number of seeks required to read the necessary data blocks is much more important in optimizing the throughput than reducing the seek time or the rotational latency. By making it possible for a stream to get enough data for one round from only one access to one disk, we can minimize the frequency of disk seeks.

A. RTL(Round Time Length Striping)

If we let DT, the playback time of a striping unit, be the same as the round length, T, then each stream needs to access exactly one disk in a round. A CDL striping scheme, however, may require two or more blocks from round to round, even if a similar configuration is adopted. Thus, we divide each multimedia object into segments that have the same playback duration, and these segments are striped along the disks. When placing the first segments of objects we apply a staggered striping scheme [2]. This reduces the possibility of load imbalance by storing more segments on one disk than on the others. This striping scheme is illustrated in Fig. 1 and we call the scheme RTL (Round Time Length) striping [10].

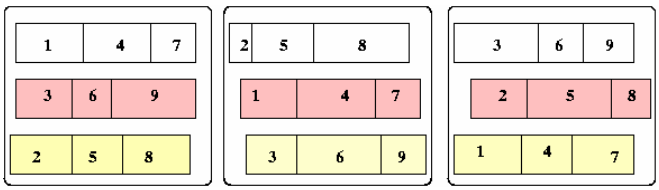

Fig. 1. RTL(Round Time Length) Striping

If k active streams are being serviced from disk i at the current service round, then these streams will be serviced from disk (i+1) mod n at the next round where n is the number of data disks in the disk array under the RTL striping model. If we assume that every disk in the disk array is identical, then it follows that if we have serviced a set of streams without any problems at round j, then there will be no problems at round j+1.

If the worst-case assumption (i.e., that each stream requires peak data rate, all the time) is used to prevent starvation or overflow, then resources are wasted. Instead of using the peak data rate, we use the maximum segment data rate - R(DT), which is much smaller than the peak data rate. Assume that we split a multimedia object i into segments that have equal playback time DT, and let ri,j be the average data rate of the jth segment of object i, then we can define the maximum segment data rate Ri(DT) of object i as follows.

$$Ri(DT) = MAX(ri,j(DT) \text{ for all } j) \quad (1)$$

This equation is the same as the following equation.

$$Ri(DT)=MAX(Size\_of\_Segments)/\#\text{ of playback time of the segment} \quad (2)$$

For example, if maximum segment size is 5M bit, and DT is 2 seconds, then R(DT) becomes 2.5Mbps.

In many cases, the worst-case assumption is used to prevent the hiccup or jitter. Yet, even though we use R(DT) instead of the peak data rate for resource reservation, there will be neither starvation nor overflow since each segment is stored contiguously

and retrieved segment by segment. If DT is the same as the playback time of a frame, then R(DT) is the same as the peak data rate, but if DT is the same as the playback time of a whole video object, then R(DT) is the same as the average data rate of that video object. Thus, R(DT) becomes smaller as DT becomes larger, and the following equation holds globally.

$$R(DT1)<R(DT2) \text{ if } DT1>DT2 \quad (3)$$

B. Paged Segment Striping (PSS)

As users' request trends vary, so does the optimal round length. Thus, in a multimedia server that serves heterogeneous streams, we should change the round length from time to time to maximize the performance. The change in round length should not, however, cause additional disk overhead as disk bandwidth is commonly the bottleneck of performance. For example, assume that we stripe the objects by RTL and the optimal round length is altered from DT to 1.5DT or 2 DT. We need only one disk access per stream with round length DT, but 2 accesses are required if we increase the round length to 1.5 DT or 2 DT. This means that it requires twice as much seek overhead, which is unacceptable.

To solve this problem, we propose the Paged Segment Striping (PSS) scheme. Each object is split into segments as in RTL and each segment is divided into the same number of pages with equal playback time. All the pages in a segment are stored contiguously on one disk. Fig. 2 shows one case of PSS with 6 pages in each segment.

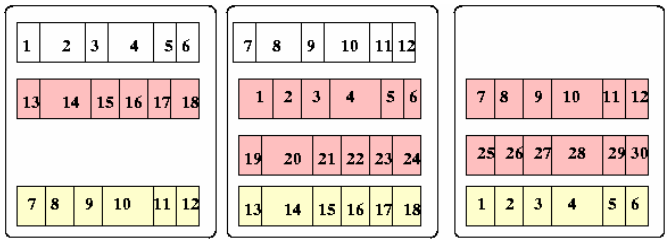

Fig. 2. Paged Segment Striping Scheme (Each segment has 6 pages in it)

The optimal round length becomes maximum or minimum when 100% of the streams are the lowest or highest data rate streams, respectively. Consequently, the optimal round length varies between these minimum and maximum values according to the workload. The size (say, playback time) of each segment is set to the maximum round length.

Let n be the number of pages in a segment. We allocate from the 1st to the nth pages (the first segment of an object) on the 1st disk contiguously and from the (n+1)th to the 2nth on the 2nd disk, and so on. We store each segment of an object on the disks in round robin manner. In choosing n, we should take an integer that has many divisors, but is not too large. In particular, 6, 8, 12, 24, etc. are suitable candidates for n.

3 Dynamic Scheduling Scheme

A new stream cannot be accepted if the server does not have sufficient resources such as disk bandwidth or buffer space. If both these resources are inadequate, then the new stream has no chance of being accepted; however, if only one resource is inadequate, then there is still a chance. Let us suppose that we run out of disk bandwidth but still have some buffer space left. By increasing the round length, we can use the disk bandwidth more effectively since it requires fewer disk seeks per unit time; but this increases the buffer space requirement. If we have enough buffer space to increase the round length, then the new stream may be accepted. On the other hand, let us suppose a new stream cannot be accepted due to an absence of buffer space left while some disk bandwidth still remains. In this case, reducing the round length may make the new stream acceptable.

Example) Let a segment have 6 pages and p be the playback time of a page. Then the possible round lengths which do not induce additional seeks are p, 2p, 3p, and 6p. If the current round length is 2p, then each stream accesses the same disk 3 times. After 3 rounds, each stream starts to read data from the next disk in the array. Additional disk seeks are incurred in some cases, however, even if we change round length using the divisors. Assume that a stream reads page 3 and 4 in Fig. 2 in the current round and the server decides to change the round length from 2 p to 3p. Then, the stream needs to read page 5, 6 and 7 in the next round. This means this stream should access two disks to read sufficient data for one round. This doubles the disk seek frequency; however, if we change the round length after a round so that it will read pages 7, 8 and 9, there will be no additional seeks. Therefore, if changing (increasing or decreasing) the round length causes additional overhead, then we should wait a few rounds so that reading a round is always aligned at the

disk boundary. We call this postponement of the changing round Segment Alignment delay.

Assume that we stripe each object by PSS. In this case, each segment has n pages and the playback time of a page is p. If we have to change the round length in some situation, then we change round length as m • p (m is one of the divisors of n) so that no additional disk seek is induced; that is, Segment Alignment needs to be considered. As we have stated before, there are many choices for n. If n is a large integer that has many divisors, then a fine-grained change of a round is possible, but scheduling and admission control becomes more complex. For example, if n is 24, then the round can be one of p, 2p, 3p, 4p, 6p, 12p, or, occasionally, 24p, while if n is 6, then the round can be one of p, 2p, 3p, or 6p. Initially, the round length is set to 2p or 3p if n is 6 and 4p or 6p if n is 24 so that the round length can be increased or decreased afterwards. Since 6 showed reasonably good performance in our experiments and scheduling is simple compared to other values, we adopt 6 as the number of pages in one segment.

A. Admission Control Algorithm

There are many constraints in admission control, but a buffer constraint (is there enough buffer to store necessary data even if a new stream is accepted?) and a disk constraint (is there enough disk bandwidth to read all the necessary data even if a new stream is accepted?) are the most important for a multimedia storage server. If the admission of a stream does not break either of these constraints, it can be accepted safely.

A.1 Buffer Constraint

We need to store the retrieved data in a buffer before sending to clients. Let m be the number of streams to be serviced, Avail_Buf be the total buffer size, DT be the playback time of a segment, n be the number of pages necessary for one segment, and T be the round length. To accept a new stream, the following constraints should hold:

$$Total\_Amount\_of\_Data\_to\_be\_Stored \le Available\_Buffer\_Size \quad (4)$$

$$2 \cdot \sum_{i=1}^{m} R_i(DT) \cdot T \le Avail\_Buf \quad (5)$$

$$2 \cdot \sum_{i=1}^{m} R_i(DT) \cdot n \cdot DT \le Avail\_Buf \quad (6)$$

If we assume $R_i(DT)$ is all the same as R(say, constant) for all streams, then we can simplify (6) as follows:

$$2 \cdot m \cdot R \cdot n \cdot DT \le Avail\_Buf \quad (7)$$

In (7), we can easily see m is inversely proportional to DT; however, since $R_i(DT)$ is a function of DT as in (3), we can not simplify $R_i(DT)$ as a constant to get correct solution. But we can still guess m is generally in inverse proportion to DT in (5).

A.2 Disk Bandwidth Constraint

Disk service time consists of seek time, rotational latency, transfer time of data, and other latencies[8]. Let d be the number of disks in the array, m be the number of streams to be serviced, k be the number of segments to be retrieved from the most heavily loaded disk. If there are many active streams and many disks in the array, then by the central limit theorem we can approximate $k \cong \left\lceil \frac{m}{d} \right\rceil$. If the service time to read k blocks from a disk is shorter than a round, we can service m streams without any starvation. In particular, if the following equations hold

$$Seek\_Time + Transfer\_Time + Rotational\_Latency \le T \quad (8)$$

$$Seek(k) + \sum_{i=1}^{k} \left(\frac{R_i(DT) \cdot DT}{Transfer\_Rate} + Rotation\_Time \le T \right) \quad (9)$$

disk access rates will be adequate. In (9), we can guess k is generally proportional to DT. Since m is proportional to k, we can say that m is generally proportional to DT.

A.3 Admission Control Algorithm using PSS

The dynamic admission control algorithm using PSS is shown in Fig. 3. Assume that m active streams are in the system, disk i has a service list queue $Q[i]$, and a new stream wants to be served from disk i (i.e., the first segment for that stream is stored on disk i). If a new stream passes both the Buffer_Test and the Disk_Test, then it is added to $Q[i]$, but if it fails in

both tests, it is rejected. However, if it fails only in one test, it still has a chance. Let us suppose a new stream passes the Buffer_Test (Disk_Test) but fails in the Disk_Test (Buffer_Test). In this case, we may accept the stream by increasing (decreasing) the round length. Sometimes, it is impossible to change the round length since the round length is the same as the largest (smallest) value or for the *Segment Alignment*. Then, this stream cannot be accepted. If a new stream can pass both of the tests only after changing the round length, then the round length is changed and the new stream is added to *Q[i]*.

```
1: Boolean
Dynamic_Admission_Control(New_Stream)
2: case 1 - if New_Stream passes both
                          Buffer Test and Disk Test;
3:          then accept the New_Stream;
4: case 2 - if it fails both of tests;
5:          then reject the stream;
6: case 3 - if it fails only in Disk Test;
7:       try to change T to T' which is less than T;
8:       if it is impossible to change the round length,
9:          then reject the stream;
10:      if it is possible to change the round length
              but it does not pass both of tests,
11:         then reject the stream;
12: case 4 - if it fails only in Buffer_Test;
12:      try to change T to T' which is larger than T;
13:      if it is impossible to change the round length,
14:         then reject the stream;
15:      if it is possible to change the round length
              but it does not pass both of tests,
16:      then reject the stream;
```

Fig. 3. Dynamic Admission Control Algorithm

The *Disk_Test* and *Buffer_Test* are required in most admission control algorithms. One more *Disk_Test* or *Buffer_Test* may be issued in the dynamic scheme than in the static scheme, and there can be the overhead in changing the round length. However, since these kinds of overhead are required only when the characteristics of workloads vary significantly, the additional overhead is small. Indeed, in normal cases, the round length remains unchanged.

4 Experimental Results

We have conducted extensive simulations to compare the performance. In our experiment, we used 20 MPEG-1 traces from [18].

In many situations, multimedia servers support several kinds of heterogeneous streams such as HDTV quality video, NTSC quality video, and CD quality audio, etc., which differ greatly in their characteristics. In this case, the requesting probability has an important implication. To simulate an environment of serving heterogeneous streams (i.e., where the difference of data rates among streams are non-negligibly large), we multiply the original traces by 10 to simulate MPEG-2 traces. We have carried out simulations with 20 high data rate objects and 20 low data rate objects. In the simulation, we used the characteristics of an IBM-HDD3200 disk. The access patterns to each video depend on the popularity of the video. We use a Zipf distribution [19] to find the probability of choosing *k*th video from a collection of *V* videos. The probability of choosing the *k*th most popular video from a collection of *V* videos, is found by Z/k where $Z = 1/(1 + 1/2 + 1/3 + ... + 1/V)$. The order in which they appear is also an indication of their popularity, the first video being the most popular and the last video the least popular. Using the Zipf distribution, if the overall arrival rate of requests to the VOD system is λ, the arrival rate of *k*th video is determined by $\lambda_k = \lambda \cdot p_k$ where $p_k = Z/k$.

In Fig. 4, the performance of the dynamic scheduling scheme is compared with that of the static scheme. If users' trends are exactly the same as expected at the system design time and there is no change in the trends, then there is almost no performance difference between the two schemes.

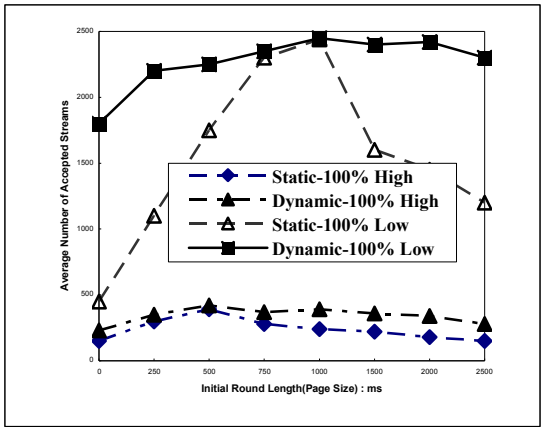

Fig. 4. Comparison of the dynamic scheme and the static scheme

Let us suppose that we have predicted that 100% of requests will be for high bandwidth objects. Then, 500 ms will be chosen as the round length. However if user requests become concentrated on

the low bandwidth objects sometime later, then there will be severe performance degradation, since 1000 ms is optimal for the low bandwidth regime. If we use 500 ms as the round length for 100% low bandwidth workloads, then there is about a 33% ~~of~~ performance degradation using the static scheme. This is in stark contrast to the dynamic scheme where there is almost no performance degradation since the round length can be changed to 1000 ms if the playback time of a page (or initial round length) is 500 ms.

Let us consider the opposite case. In particular, we predict 100% of requests will be for low bandwidth objects, so the optimal round length is chosen to be 1000 ms; however, if users request only high bandwidth objects later, then 1000 ms is no longer optimal. Using the wrong round length, there is a 46% performance degradation using the static scheme while there is almost no performance degradation in the dynamic scheme (since we can decrease the round length down to half (500 ms) of a page playback time (1000 ms)).

If 750 ms is chosen as the round length, the performance degradation from the optimum point is not much in either case. As we can see in Fig. 5,750 ms is the average of 500 ms (which is the optimal round length for high bandwidth streams) and 1000 ms (which is the optimal round length for low bandwidth streams). However, if the system configuration (buffer size, number of disks, and work load) becomes different, the performance degradation could be more severe

5 Conclusion

In this paper, we have studied the problem of how to maximize the throughput of a contiguous-media system, given amounts of buffer space and disk bandwidth both predetermined at design time. We have presented a new striping model (Paged Segment Striping) which makes it possible to change the round length dynamically without additional seek overhead.

In the dynamic scheme, even a significant change in the users' trends induces only a small performance degradation since it adapts to the situation by varying the round length. By building on the dynamic scheme's ability to self-adjusting round length, we have increased the performance up to 46% with the same resources given even if there is a severe change in the users' trends. Furthermore, since it requires little more calculation overhead than the static scheme, we are convinced that unless usage trends are guaranteed to be constant the dynamic scheduling scheme is superior to the static scheme.

Reference

[1] Scott A. Barnett and Gary J. Anido. Performability of disk-array-based video servers. ACM Multimedia Systems Journal, 6(1):60–74, January 1998.

[2] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered striping in multimedia information systems. In Proc. of ACM SIGMOD, pages 79–90, Minnesota, USA, June 1994.

[3] Jin B. Kwon, Heon Y. Yeom, " Generalized Data Retrieval for Pyramid-based Periodic Broadcasting of Videos", Future Generation Computer Systems, Vol.20, No.1, Jan. 2004, pp.157-170

[4] E. Chang and A. Zakhor. Cost analysis for vbr video servers. In IS&T/SPIE Int'l Symposium on Electronic Imaging: Science and Technology, pages 29–31, California, January 1996.

[5] Huagn-Jen Chen and T.D.C. Little. Storage allocation policies for time-dependent multimedia data. IEEE Trans. on Knowledge and Data Engineering, 8(5):855–864, October 1996.

[6] K. H. Yeung, C. C. Wong and K. Y. Wong, "A Cache Replacement Policy for Transcoding Proxy", IEICE Transactions on Communications, Vol. E87-B, No. 1, Jan. 2004.

[7] J. Dangler, E. Biersack, and C. Bernhart. Deterministic admission control strategies in video servers with variable bit rate. In Proc. of International Workshop on Interactive Distributed Multimedia Systems and Services(IDMS'96), volume 1045 of LNCS, pages 245–264, Heidelberg, Germany, March 1996.

[8] J. Gemmell, Harrick M. Vin, Dilip D. Kandlur, and Venkat Rangan. Multimedia storage servers: A tutorial and survey. IEEE Computer, 28(5):40–49, May 1995.

[9] Sharam Ghandeharizadeh, S.H. Kim, and C. Shahabi. On configuring a single disk continuous media server. In Proc. of ACM SIGMETRICS/PERFORMANCE Conf. on Measurement and Medeling of Computer Systems, pages 37–46, May 1995.

[10] KyungOh Lee and Heon Y. Yeom. Deciding round length and striping unit size for multimedia servers. In Proceedings of the 4th International Workshop on Multimedia Information Systems(MIS'98), pages 33–44, Istanbul, Turkey, September 1998.