IEICE TRANS. COMMUN., VOL.E88–B, NO.5 MAY 2005

PAPER

Practical Issues Related to Disk Scheduling for Video-On-Demand Services*

Ilhoon SHIN[†a)], ***Student Member***, **Kern KOH**[†], ***and*** **Youjip WON**[††], ***Nonmembers***

SUMMARY This paper discusses several practical issues related to the provision of video-on-demand (VOD) services, focusing on retrieval of video data from disk on the server. First, with regard to system design, the pros and cons of cycle-based scheduling algorithms for VOD servers are compared, and an adequate policy according to system configuration is presented. Second, we present a way to tune the cycle-based scheduling algorithm so that it maximizes profit. Third, a method to overcome the cons of cycle-based scheduling algorithms is proposed, and its cost is analyzed.
key words: *VOD, cycle-based disk scheduling, cycle length, prefix caching*

1. Introduction

The use of computer technology in the entertainment industry is becoming more widespread with the penetration of high technology into all aspects of society. The games industry already has a large user base, which is growing rapidly every year. Computing devices are also increasingly being used to watch movies, TV, and other kinds of moving pictures. Proposals and design specifications for this kind of service, designated video-on-demand (VOD), began to appear in the late 1980s and early 1990s. However, computer hardware and software technology was not sufficiently advanced at that time to provide VOD service in a cost-effective manner. However, with the rapid progress in computer hardware (e.g., CPU, memory, disk, network, etc.) and software (e.g., image compression and extraction), watching moving pictures over networks is now becoming cost effective.

Playing moving pictures over networks is a real-time application. As images should be displayed in real time on the clients' screen, time constraints of images should be guaranteed throughout the server, network, and client. Among these parts, we focus on the video server and discuss practical issues related to retrieval of images from the disk in time.

Traditional disk scheduling algorithms, such as FIFO [1], SCAN [1], and C-SCAN [2], are designed to maximize disk throughput and to guarantee fairness, and thus have difficulty in guaranteeing timely retrieval of blocks for VOD service. Modified versions of traditional algorithms have been proposed that take into consideration both time constraints and performance [3]–[6]. These algorithms share the notion of cycles, guaranteeing continuous playback by issuing disk requests in a periodic manner. The data required for current playback are read from the disk in a previous cycle. Similarly, the data to be played in the next cycle are read in the current cycle. In addition, cycle-based disk scheduling can improve disk throughput by ordering requests according to traditional scheduling within a cycle.

Cycle-based scheduling algorithms are classified into two groups according to how they determine cycle length. In the first, cycle length is adjusted according to aggregate playback bandwidth, while in the second, cycle length is set to a predetermined value regardless of aggregate playback bandwidth. Theoretically, cycle length should be adjusted so that the number of blocks read in the current cycle is the same as the number of blocks to be played in the next cycle. If the number of blocks read is less than that to be played, jitter[**] will occur owing to the lack of the required data. In contrast, in the opposite case, memory buffer will be wasted by continuous accumulation of unnecessary data in the buffer. Hence, cycle length fluctuates according to the number of blocks to be read in each cycle.

However, as noted in [7], fluctuations in cycle length causes inequality between the amount of data read in the previous cycle and that played in the current cycle, which results in jitter. To avoid this, many researchers advocate setting cycle length to a large value [8]–[10]. Use of a sufficiently large cycle length has the advantage of simplicity and results in no jitter. However, it may cause long service startup latency[***] and consume large amounts of buffer memory. Due to these problems associated with large cycle length, Y. Won et al. [7] adopted a varied cycle length based on aggregate playback bandwidth. Varied cycle length has the advantages of relatively short service startup latency and of minimal buffer requirements although, as described above, fluctuation of cycle length causes jitter.

From the standpoint of VOD server design, there are several practical issues to address. The first issue is which approach is more suitable for the VOD server in question,

Manuscript received December 25, 2003.
Manuscript revised August 5, 2004.
[†]The authors are with the School of Computer Science and Engineering, Seoul National University, South Korea.
[††]The author is with the Division of Electrical and Computer Engineering, Hanyang University, South Korea.
Work of this author is supported by grant No. R08-2003-000-11104-0 from the Basic Research Program of the Korea Science & Engineering Foundation.
*This research was supported by Brain Korea 21.
a) E-mail: jeje@oslab.snu.ac.kr
DOI: 10.1093/ietcom/e88–b.5.2156

**Jitter means that playback is temporarily paused owing to a violation of timely constraints of data.
***The interval between the time that a client sends a request and the time that playback starts.

which may depend on the VOD server configuration, such as request arrival rate, movie playback rate, client characteristics, network configuration, etc. Second, if a fixed cycle length is adopted, it is necessary to determine the most cost-effective length. A large cycle length allows the system to service a large number of streaming sessions, but it degrades quality of service because of the long startup latency. The third issue is whether there is a method to overcome the weaknesses of each approach and the associated cost.

In this paper, we will discuss these issues with regard to server design. Cycle-based scheduling is explained in more detail in Sect. 2. In Sect. 3, we compare and analyze the merits and demerits of each approach through simulation and present a more suitable approach for each VOD configuration. In Sect. 4, we present a method to find the most cost-effective cycle length using mathematical modeling. In Sect. 5, we discuss how to overcome the weaknesses of each approach and analyze their associated costs. Finally, our conclusions are presented in Sect. 6.

2. Cycle-Based Disk Scheduling

Cycle-based disk scheduling satisfies the time constraints of VOD service while maximizing disk throughput as much as possible. Figure 1 illustrates how cycle-based scheduling achieves this. We assume that there are two streaming sessions ($s1, s2$) in the system. Three horizontal lines can be seen in the figure. The top and bottom lines indicate playback of sessions $s1$ and $s2$, respectively. The middle line expresses block reading from the disk. For continuous playback of sessions, each block should be read from the disk before its playback time. As guaranteeing this deadline on an individual basis as in EDF [11] causes excessive disk head movement, cycle-based scheduling batches disk requests by cycle and guarantees deadlines per cycle. For example, the blocks of session $s1$ and session $s2$ to be played in cycle $i+1$ are all read in cycle i. Their deadlines are all T_i. Likewise, the data to be played in cycle $i+2$ are all read in cycle $i+1$. Their deadlines are all T_{i+1}. Continuous playback can be guaranteed if the blocks to be played in the current cycle have all been read in a previous cycle. In contrast to EDF, cycle-based scheduling can improve disk throughput by arranging the read order of data blocks within a cycle such that disk head movement can be minimized. It is known that C-SCAN minimizes disk head movement while considering fairness [12].

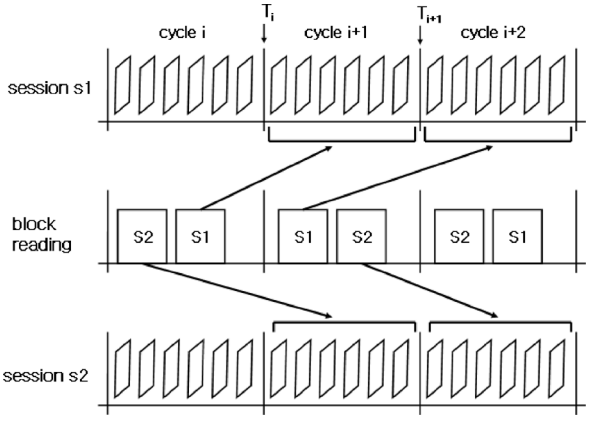


Fig. 1 Cycle-based disk scheduling.

The conditions for continuous playback can be expressed by the following two equations. Eq. (1) indicates that the data read in a previous cycle should be sufficient for playback in the current cycle. Eq. (2) indicates that the previous cycle length should be long enough to read all the required blocks. T, τ_i, n_i, b, θ, s, and $T_{overhead}$ denote cycle length, playback rate of session i, the number of blocks of session i read in the previous cycle, block size, disk transfer rate, the number of concurrent sessions, and the total disk head movement overhead including rotational delay during a cycle, respectively. The cycle length that does not cause jitter is calculated by Eq. (3) from Eq. (1) and Eq. (2). Detailed explanations are given in [7], [13].

$$T \times \tau_i \le n_i \times b \tag{1}$$

$$T \ge \frac{\sum_{i=1}^{s} n_i \times b}{\theta} + T_{overhead} \tag{2}$$

$$T \ge \frac{T_{overhead}}{1 - \frac{1}{\theta} \times \sum_{i=1}^{s} \tau_i} \tag{3}$$

As cycle length is the primary factor that determines service startup latency and buffer requirements, the minimum cycle length that satisfies Eq. (3) has advantages with regard to startup latency and memory cost. However, as noted in [7], use of the minimum cycle length may give rise to problems when a new streaming session is added. From Eq. (3), the addition of a session increases the minimum cycle length that satisfies Eq. (3), and this increased cycle length causes a lack of the required data by violating Eq. (1). Greater increases in cycle length are associated with greater degrees of data deficiency. This can be prevented by setting the cycle length to a sufficiently large value that it satisfies Eq. (3). As cycle length does not fluctuate, no deficiency of data occurs. The disadvantage of a large fixed cycle length is that it may cause long service startup latency and waste buffer memory.

Both approaches have their own strengths and weaknesses and it is difficult to conclude which is better. In the next section, we compare the performance of the two approaches under various conditions and present an adequate approach for each environment.

3. Performance Comparison

The factors that determine the performance of VOD disk scheduling are the occurrence of jitter, service startup latency, buffer consumption, and the number of concurrent streams that can be served, etc. Varied cycle length and fixed cycle length show differences in performance with respect to jitter, startup latency, and buffer consumption, while they can service the same number of concurrent streams. Therefore, we compared the two approaches with respect to jitter,

Table 1 The primary attributes of the simulation disk.

Capacity	120 GB
Sector Size	512 bytes
Aggregate sustained data rate	35 MB/s
Aggregate rotational delay	4.17 ms
Aggregate seek time	8.5 ms
Full seek time	15 ms

startup latency, and buffer consumption.

The model VOD server contains 100 movie files, which are each 100 minutes in length. Movies are all MPEG1 streams that are encoded with a constant bit rate (CBR) of 1.5 Mbps. The requests of clients arrive according to the Poisson distribution and clients pick up the movie according to the Zipf distribution with a parameter of 0.271. Zipf distribution models a skewed access pattern that each object is accessed with different probability. The parameter of 0.271 has been shown to match with the empirical data from video rental stores [14]. The VOD storage device is modeled for an IBM Deskstar 120 GXP hard disk [15]. The primary attributes of the disk are shown in Table 1. Movie files are assumed to be stored consecutively on the disk and disk requests are scheduled with C-SCAN within a cycle. As requests are scheduled by C-SCAN within a cycle and movies are stored consecutively, $T_{overhead}$ can be approximated as shown in Eq. (4). One seek and one rotation should be proceeded to retrieve the data blocks of each session, and C-SCAN returns the disk arm to the first cylinder with a full seek in each cycle. T_{lat}, T_{seek}, T_{full}, and s denote aggregate rotational delay, aggregate seek time, full seek time, and the number of concurrent sessions, respectively. The block size is 4 KB and the server runs for 8 simulated hours.

$$T_{overhead} = s \times (T_{lat} + T_{seek}) + T_{full} \quad (4)$$

3.1 The Occurrence of Jitter

When cycle length is extended according to the aggregate playback bandwidth, a temporary shortage of data occurs and playback is temporarily paused, as described in Sect. 2. The time playback is paused is proportional to a shortage of data and can be calculated using the amount of data that are short. For example, if the data of 128 KB are short within a cycle for a session of a playback rate of 1 Mbps, playback is paused for one second. If the data of 256 KB are short, the playback will be paused for two seconds.

We studied the severity of this problem by measuring the total time playback was paused. The request mean inter-arrival time[†] is varied to 30 and 60 seconds. A mean inter-arrival time of 30 seconds fully utilizes disk bandwidth capacity with playback of a movie 100 minutes in length with a playback rate of 1.5 Mbps, while a mean inter-arrival time of 60 seconds utilizes about half of the disk bandwidth capacity. The results are shown in Fig. 2, in which the X-axis is the upper bound of cycle length, which was varied from 1 to 20 seconds, and the Y-axis is the total time that playback was paused. Var denotes scheduling with varied cycle length. We plotted only the results obtained with varied cycle length because fixed cycle length does not cause jitter. Note that the admission of requests is controlled by the upper bound of cycle length. If the cycle length extended by newly arrived requests exceeds the upper bound of cycle length or becomes a minus[††], the requests are rejected. Only when the extended cycle length is a plus and less than the upper bound, the requests are admitted and served.

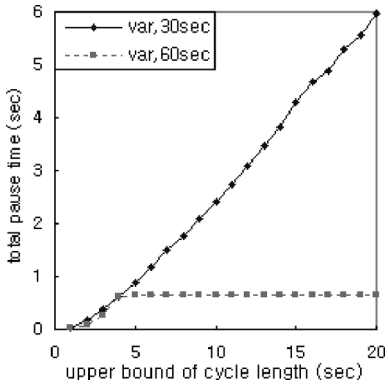


Fig. 2 The total pause time with various cycle length upper bounds.

Figure 2 shows that the occurrence of jitter was not severe when the mean inter-arrival time was 60 seconds. The maximum total pause time was less than 1 second, which is acceptable. In contrast, with a mean inter-arrival time of 30 seconds, the total pause time increased to 6 seconds. This difference in the total pause time was due to the difference in the degree of cycle length fluctuation. When disk bandwidth is under-utilized with a small number of concurrent sessions (i.e., with 60 seconds of mean inter-arrival time), the fluctuation of cycle length is not severe from Eq. (3). In contrast, when disk bandwidth is fully utilized with a large number of concurrent sessions (i.e., with a mean inter-arrival time of 30 seconds), the degree of cycle length fluctuation increases. The difference in the bandwidth utilization according to the request arrival rate causes the difference in the total time playback was paused.

To study the impact of playback bandwidth and movie length, we fixed the upper bound of cycle length to 10 seconds, and varied playback rate and movie length. Playback rate was varied to 128 kbps (MPEG4), 1.5 Mbps (MPEG1), and 19.2 Mbps (STAC)[†††]. Movie length was varied to 10 and 100 minutes. Figures 3 and 4 illustrates the total pause time for each configuration. Figure 3 is the results when movie length was 100 minutes and Fig. 4 is the results when movie length was 10 minutes. The X-axis denotes the different configurations. The upper and bottom strings indicate playback bandwidth (i.e., 128 K = 128 kbps, etc.) and movie length, respectively. The results indicate that play-

[†]The mean time between a request and the next request.

[††]A minus implies that the aggregate playback bandwidth exceeds disk bandwidth capacity, as seen in Eq. (3).

[†††]In the simulation with STAC streams, the VOD server is assumed to contain only 8 movie files owing to the restriction of disk space.

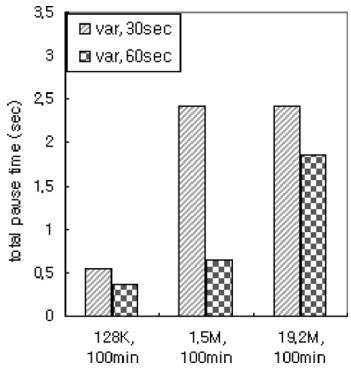


Fig. 3 The total pause time with a movie length of 100 minutes and various playback rates.

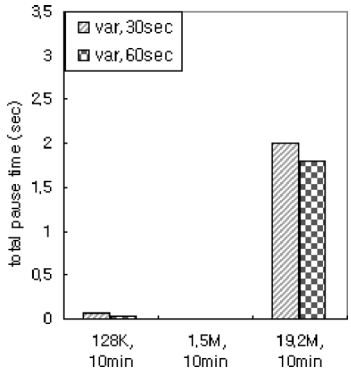


Fig. 4 The total pause time with a movie length of 10 minutes and various playback rates.

back bandwidth has a significant impact on the occurrence of jitter. With a playback rate of 128 kbps, the total pause time was less than 1 second regardless of request arrival rate and movie length. This was because disk bandwidth was hardly utilized with 128 kbps playback, even though requests arrived every 30 seconds. In contrast, with a playback rate of 19.2 Mbps, the total pause time increased to around 2 seconds, regardless of configuration. This was because disk bandwidth can be fully utilized with a small number of concurrent sessions at high playback rate, and therefore the fluctuation of cycle length was significant even with a mean inter-arrival time of 60 seconds and short movie length (10 minutes). With a playback rate of 1.5 Mbps, the total pause time differed according to configuration. The total pause time was less than 1 second with short movies or infrequent request arrival; jitter rarely occurred with 10-minute movies (Fig. 4). However, the total pause time increased to around 2.5 seconds with a mean inter-arrival time of 30 seconds and a movie length of 100 minutes (Fig. 3). As disk bandwidth is fully utilized in this configuration, fluctuation of cycle length becomes significant and the total pause time therefore increases.

When disk bandwidth is under-utilized with short movie length, low playback rate, or infrequent request arrival rate, the occurrence of jitter is not severe. The total pause time is less than 1 second in most cases and the weakness of variable cycle length becomes acceptable. However, when the playback rate is very high or when requests arrive at high frequency, disk bandwidth becomes fully utilized and the fluctuation of cycle length becomes significant. In these cases, the total pause time increases to more than 2 seconds and will increase further according to the upper bound of cycle length as shown in Fig. 2. For these configurations, a fixed cycle length will be an adequate alternative if guaranteeing continuous playback is important.

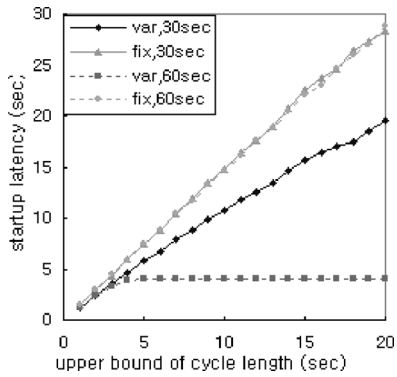


Fig. 5 Average startup latency with various cycle length upper bounds.

3.2 Start-Up Latency

Startup latency is a primary factor that determines QoS of VOD service with the occurrence of jitter. Long startup latency will cause clients to lose interest in VOD services. We compared the performance of varied cycle length and fixed cycle length with regard to startup latency. Startup latency was calculated using the time a request arrives and the time playback starts. As in the case of jitter, we first fixed the playback rate to 1.5 Mbps, and movie length to 100 minutes. The upper bound of cycle length was varied from 1 to 20 seconds, and the mean inter-arrival time was varied to 30 and 60 seconds. In the case of fixed cycle length, cycle length was set to the maximal value that satisfied Eq. (3) without exceeding the upper bound. Figure 5 illustrates the average startup latency of streams served during a given running time. Fix denotes scheduling with fixed cycle length, while var denotes scheduling with varied cycle length.

As shown in Fig. 5, fixed cycle length generates similar startup latency regardless of request arrival rate. This phenomenon is easily explained, as cycle length was set to the maximal value regardless of current aggregate playback bandwidth. Average startup latency increased linearly according to the upper bound of cycle length, with a maximum value of about 28 seconds. The ratio of the increase to the upper bound of cycle length was about 1.4. In contrast, varied cycle length generated significantly different startup latencies according to the request arrival rate. When requests arrived every 30 seconds on average, and thus disk bandwidth was fully utilized, startup latency increased linearly according to the upper bound of cycle length with an increase ratio of about 0.9. The maximum startup latency

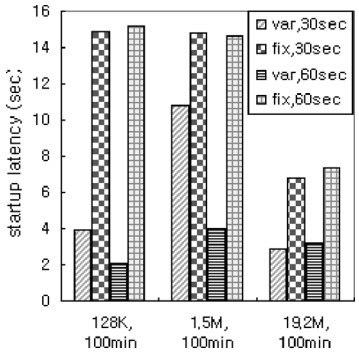


Fig. 6 Average startup latency with a movie length of 100 minutes and various playback rates.

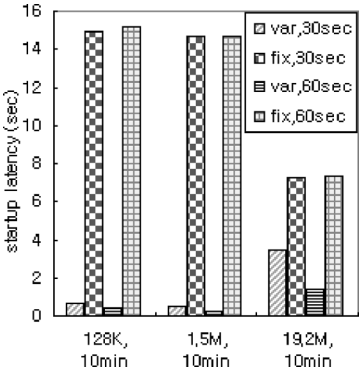


Fig. 7 Average startup latency with a movie length of 10 minutes and various playback rates.

was about 19 seconds, which was shorter by 9 seconds than the fixed cycle length. When requests arrived every 60 seconds, and thus disk bandwidth was under-utilized, startup latency showed marked decreases with varied cycle length. The maximum startup latency was less than 5 seconds. The maximum difference from fixed cycle length was more than 23 seconds.

We also studied the impact of playback bandwidth and movie length. We varied playback rate to 128 kbps, 1.5 Mbps, and 19.2 Mbps, and movie length to 10 and 100 minutes. The upper bound of cycle length was set to 10 seconds. Fixed cycle lengths were 10.0 seconds for 128 kbps, 9.8 seconds for 1.5 Mbps, and 4.8 seconds for 19.2 Mbps, respectively, which satisfy Eq. (3) without exceeding the upper bound of cycle length[†]. Figures 6 and 7 illustrates the average startup latency for various configurations. Figure 6 is the results when movie length was 100 minutes and Fig. 7 is the results when movie length was 10 minutes. The results indicate that variable cycle length generates significantly shorter startup latency when disk bandwidth is under-utilized due to low playback rate, short movie length, or infrequent request arrival. With a playback rate of 128 kbps, variable cycle length showed shorter startup latency by more than 11 seconds, regardless of request arrival rate or movie length. A playback rate of 1.5 Mbps showed similar differences, excluding the case with a mean inter-arrival time of 30 seconds and movie length of 100 minutes, where the difference was about 4 seconds (Fig. 6). With a playback rate of 19.2 Mbps, the startup latency with a varied cycle length was about half that with a fixed cycle length. The difference was around 4 seconds. With 19.2 Mbps playback, the difference in startup latencies was not significant because a small number of streams are sufficient to fully utilize disk bandwidth.

In conclusion, when disk bandwidth is under-utilized with a low playback rate, short movie length, or infrequent request arrival, variable cycle length generates much shorter startup latency due to its adaptation to current playback bandwidth. The difference is more than 10 seconds. Even when disk bandwidth is fully utilized, variable cycle length has a shorter startup latency than fixed cycle length, with a difference of about 4 seconds.

3.3 Buffer Consumption

Continuous playback and startup latency are the factors in which clients are most interested. In contrast, service providers are most interested in system cost, i.e., the cost required to design a system. From the viewpoint of retrieving data blocks from disk, disk bandwidth and memory buffer are the primary factors determining system cost. As both scheduling algorithms consume the same disk bandwidth to service the same number of concurrent sessions, we compared only the amount of memory buffer consumed by both algorithms.

Cycle based scheduling uses two buffers for each session. One buffer contains the data being played in a current cycle, and the other buffer is used to load the data to be played in the next cycle. As the size of each buffer should be $T \times \tau$ for continuous playback, the amount of memory buffer allocated to each session is $2 \times T \times \tau$. T and τ denote cycle length and playback rate, respectively. Figure 8 illustrates the average maximal buffer consumption of sessions served during a given running time, which is computed by the above equation. The playback rate was fixed at 1.5 Mbps and movie length at 100 minutes. The upper bound of cycle length was varied from 1 to 20 seconds, and the mean inter-arrival time was varied between 30 or 60 seconds. The results show that with a mean inter-arrival time of requests of 30 seconds, both scheduling algorithms consumed the same amount of buffer. The gain of varied cycle length was zero. This was because the varied cycle length increases to the same value as the fixed cycle length when disk bandwidth is fully utilized. In contrast, when requests arrived every

[†]Let us show how to calculate a fixed cycle length for 1.5 Mbps. First, the cycle length with one concurrent session is calculated. It is 0.028 second and does not exceed the upper bound. In order to find the maximal value without exceeding the upper bound, cycle length is repeatedly calculated increasing the number of concurrent session by 1, until cycle length exceeds the upper bound. In this example, cycle length exceeds the upper bound when the number of concurrent sessions becomes 151. Therefore, the maximal value without exceeding the upper bound is the cycle length with 150 concurrent sessions, which is about 9.8 seconds.

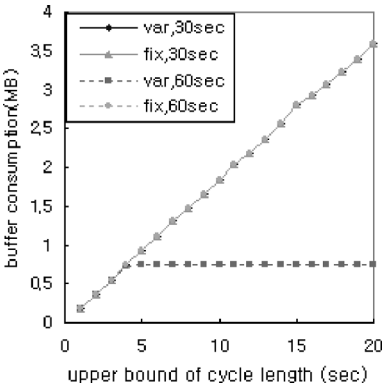


Fig. 8 Average buffer consumption with various cycle length upper bounds.

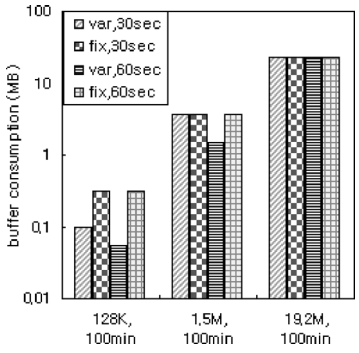


Fig. 9 Average buffer consumption with a movie length of 100 minutes and various playback rates.

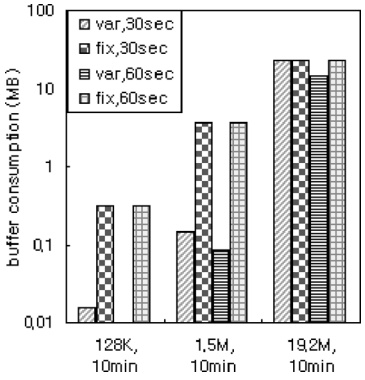


Fig. 10 Average buffer consumption with a movie length of 10 minutes and various playback rates.

60 seconds, varied cycle length consumed much less buffer. The maximal gain per session was more than 2.5 MB, due to the shorter cycle length resulting from under-utilization of disk bandwidth.

Figures 9 and 10 depict buffer consumption for various playback rates and movie lengths, and show similar results. Note that the Y-axis is on a log-scale to depict the wide range of values of buffer consumption. When disk bandwidth is under-utilized with low playback rate, short movie length, or infrequent request arrival, varied cycle length consumes much less buffer memory (up to 10-fold less). However, with a playback rate of 19.2 Mbps, or with a movie length of 100 minutes and a playback rate of 1.5 Mbps, the gain with varied cycle length was less significant. This is because a high playback rate makes it easy to fully utilize all the available disk bandwidth, and long movie length increases the likelihood that a large number of sessions will be served concurrently, thereby fully utilizing the disk bandwidth.

3.4 Discussion

Startup latency, buffer consumption, and the occurrence of jitter are all closely related to cycle length, which is primarily determined by disk bandwidth utilization, as can be seen from Eq. (3). When disk bandwidth utilization is low, varied cycle length has an advantage over fixed cycle length: the total pause time is less than 1 second, startup latency is shorter by about 10 seconds, and it consumes about 10-fold less buffer memory. However, when disk bandwidth is fully utilized with high playback rate, frequent request arrival, or long movie length, it is difficult to say which approach is better. Even though varied cycle length still has the advantages of shorter startup latency and less buffer consumption, the gain is not as great as in the case of under-utilization of disk bandwidth. In addition, jitter occurs more frequently and the total pause time becomes longer. Hence, system designers would be better to choose fixed cycle length for configurations where guaranteeing continuous playback is important. Otherwise, varied cycle length will probably be the better approach.

4. Determining Cycle Length

In varied cycle length, cycle length is determined to the minimum value satisfying Eq. (3). However, in fixed cycle length, the system administrator determines the cycle length. A short cycle length reduces startup latency and thereby improves the quality of each streaming session, while the number of sessions that can be served concurrently is limited, as seen in Eq. (3). In contrast, a long cycle length makes it possible that more streaming sessions can be served concurrently, while startup latency becomes long and the quality of each streaming session is reduced. Most previous studies [8]–[10], [14], [16] that adopted cycle-based disk scheduling with fixed cycle length did not examine which cycle length was the most cost-effective. In this section, we present a method to find the most cost-effective cycle length.

The profit of the VOD service is determined by the number of served streaming sessions and by the price of each session. First, we calculate the number of served streaming sessions for one day. The number of concurrent sessions that can be served with cycle length T, $S_{capacity}$, is calculated by Eq. (5) from Eqs. (3) and (4), and the number of requests that arrive during playback of a movie, $S_{arrival}$, is calculated by Eq. (6). λ and T_{movie} denote request arrival rate and movie length, respectively. As the real number of concurrent sessions with cycle length T, S_{real}, is limited by both $S_{capacity}$ and $S_{arrival}$, S_{real} is the minimal value of

$S_{capacity}$ and $S_{arrival}$, and is expressed by Eq. (7). From S_{real}, the number of streaming sessions served for one day, S_{day}, can be approximated by Eq. (8). T_{run} denotes running time of the server for one day.

$$S_{capacity} = \left\lceil \frac{\theta \times (T - T_{full})}{\theta \times (T_{seek} + T_{lat}) + \tau \times T} \right\rceil \quad (5)$$

$$S_{arrival} = \lceil \lambda \times T_{movie} \rceil \quad (6)$$

$$S_{real} = min(S_{capacity}, S_{arrival}) \quad (7)$$

$$S_{day} = \left\lfloor S_{real} \times \frac{T_{run}}{T_{movie}} \right\rfloor \quad (8)$$

The price of a streaming session can be determined from the QoS experienced by each session. As there is no realistic model for determining price [17]†, we introduce a mathematical function (Eq. (9)) to determine price, based on the observation of fast food marketing studies [18], [19] that customer satisfaction decreases as waiting time increases and waiting time influences customers' return frequency. As long waiting time reduces future arrival rate as well as customer satisfaction, we carefully supposed that price should rapidly decrease as startup latency increases. Eq. (9) was derived from the assumption††. In Eq. (9), M denotes an ideal price when cycle length is 0, and M_T denotes the price when cycle length is T. C is a constant that determines the ratio of decrease of the price. When cycle length, T, is equivalent to C, the price becomes \$0. Meanwhile, from the results of Eq. (8) and Eq. (9), we calculate the total profit for one day as Eq. (10). The optimal cycle length is the length that provides the maximum value of P_{day}. If we assume the above IBM Deskstar hard disk, movie length of 100 minutes, playback rate of 1.5 Mbps, mean inter-arrival time of 30 seconds, a running time of 8 hours, a C value of 20, and ideal price of \$2, the per day profit according to cycle length is illustrated in Fig. 11. The figure shows that the most cost-effective cycle length is about 8 seconds. In the same way, VOD server designers can determine the most cost-effective cycle length from Eq. (10) according to the configuration of the VOD server.

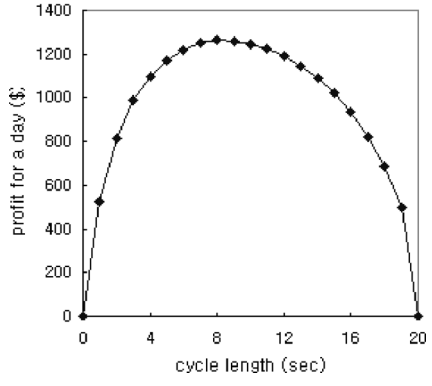


Fig. 11 Total per day profit according to cycle length.

$$M_T = M \times \sqrt{1 - (T^2/C^2)} \quad (9)$$

$$P_{day} = S_{day} \times M_T \quad (10)$$

5. Prefix Caching

As described in Sect. 3, there are tradeoffs between the occurrence of jitter, startup latency, and buffer consumption. Varied cycle length reduces startup latency and buffer consumption at the cost of the occurrence of jitter, while fixed cycle length prevents the occurrence of jitter at the cost of long startup latency and increased buffer consumption. In this section, we introduce prefix caching, which eliminates startup latency and the occurrence of jitter at the cost of additional buffer utilization. Prefix caching preloads an initial portion of movies into memory to absorb startup latency and to prevent the occurrence of jitter. This technique has also been discussed in various VOD papers in applications such as video smoothing [20] and video broadcasting [21]. However, these previous studies did not apply prefix caching to on-demand video retrieval from the server, and therefore did not analyze various practical issues, such as how much data or how many movies should be preloaded. We discuss these questions in this section.

In cycle-based scheduling, the upper bound of the amount of data to completely eliminate startup latency is $2 \times T \times \tau$ bytes. τ denotes playback rate and T is the determined cycle length in the case of fixed cycle length or the upper bound of cycle length in the case of varied cycle length. Figure 12 illustrates the worst case in which $2 \times T \times \tau$ bytes of data are required to completely eliminate startup latency. The request that arrives at the beginning of cycle i starts to read data from the disk at the beginning of cycle $i + 1$ and the movie playback starts at the beginning of cycle $i + 2$. The startup latency of the request is $2 \times T$, and therefore $2 \times T \times \tau$ bytes of data should be preloaded for instant playback. Preloading an amount of data smaller than $2 \times T \times \tau$ may cause a deficiency of data according to the point of time at which the request arrives.

The issue of how many movies to preload into memory is related to both cost and QoS. Preloading as many movies as possible achieves better QoS, while it requires more memory. We present here a method to find the most

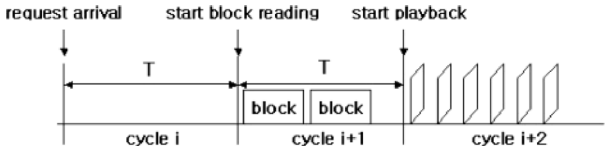


Fig. 12 Worst case of prefix caching.

†The realistic model can be created by marketing techniques such as polls and surveys [17].

††Other mathematical functions can be derived from the assumption, and it is possible to find the most cost effective cycle length in a similar way with regard to other functions.

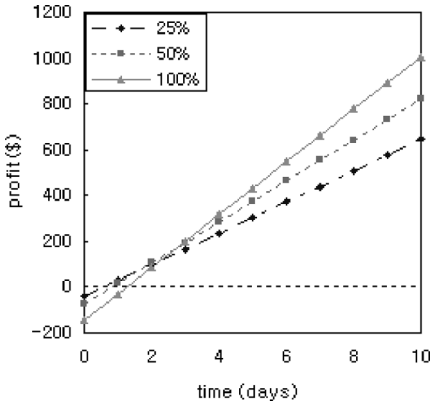


Fig. 13 The total profit of 1.5 Mbps playback from prefix caching.

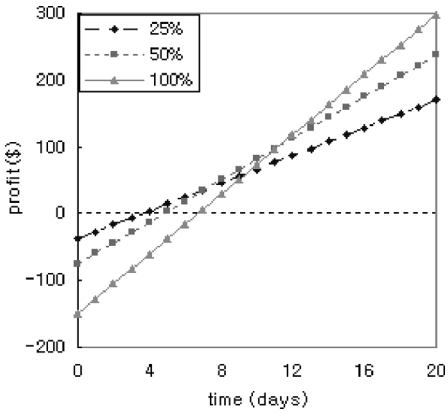


Fig. 14 The total profit of 19.2 Mbps playback from prefix caching.

profitable degree of prefix caching. We assume that popularity of movies follows the Zipf distribution with parameter z, and streaming price follows the function of Eq. (9). From the definition of Zipf distribution, we can calculate the probability $Pr(i)$ that a client picks a movie i as shown in Eq. (11). S_{total} denotes the number of movies offered by the VOD server. From Eq. (8) and Eq. (11), the total number of requests per day for movie i can be calculated by Eq. (12). The profit gained from preloading movie i is calculated by Eq. (13) from Eq. (12) under the assumption that we preload $2\times T\times\tau$ bytes of data for each movie. Note that this profit is a gain from prefix caching for one day. Meanwhile, the initial investment cost to preload movie i is calculated by Eq. (14). M_{mem} denotes the memory price per byte. From Eq. (13) and Eq. (14), the final profit sum gained from preloading n popular movies for k days is calculated by Eq. (15).

$$Pr(i) = \frac{1}{i^{1-z} \times \sum_{i=1}^{S_{total}} (1/i^{1-z})} \tag{11}$$

$$N(i) = S_{day} \times Pr(i) \tag{12}$$

$$P_{prefix}(i) = N(i) \times M \times (1 - \sqrt{1 - T^2/C^2}) \tag{13}$$

$$Cost = 2 \times T \times \tau \times M_{mem} \tag{14}$$

$$P_{total}(n,k) = k \times \sum_{i=1}^{n} (P_{prefix}(i)) - Cost \times n \tag{15}$$

VOD server designers can determine how many movies to preload from Eq. (15) to maximize $P_{total}(n,k)$. For example, if we assume that the configuration of the VOD server is the same as described in the experiments in Sects. 3 and 4, $P_{total}(n,k)$ can be calculated as Figs. 13 and 14. Figure 13 shows the result with a playback rate of 1.5 Mbps and an ideal session price of $2, and Fig. 14 shows the result with a playback rate of 19.2 Mbps and an ideal session price of $4. The X-axis shows the days for which the VOD server runs and the Y-axis shows the total gain from prefix caching. We assume 512 MB of memory (Samsung DDR DRAM), which costs $250. The indicators 25%, 50%, and 100% mean that the server preloads 25%, 50%, and 100% of top-ranked movies by popularity, respectively. Figure 13 shows that with a playback rate of 1.5 Mbps, preloading all the movies produces the most profit. After 3 days, 100% produces the most profit and the difference from the others increases according to the days. A playback rate of 19.2 Mbps also shows the similar result. Until 7 days, 25% produces the most profit. However, from about 12 days, 100% is the most profitable. From the results, we can conclude that preloading all the movies is the best choice for the configuration assumed.

6. Conclusion

This paper discussed several practical issues related to disk scheduling for VOD service.

1. Cycle-based disk scheduling, which has been adopted for most VOD studies, was classified to scheduling with varied cycle length and scheduling with fixed cycle length by determining cycle length. Varied cycle length has the drawback that it generates jitter, while fixed cycle length causes long startup latency and large buffer consumption. Which approach is better for a given application is dependent on the VOD server environment. Simulations showed that varied cycle length is the better approach for environments where disk bandwidth is under-utilized with low playback rate, short movie length, or infrequent request arrival rate. Under these conditions, the occurrence of jitter was negligible, and both startup latency and buffer consumption were markedly reduced. In contrast, when disk bandwidth is fully utilized, it is difficult to conclude which is the better approach. If guaranteeing continuous playback is important, fixed cycle length is the better choice because varied cycle length generates considerable jitter. Otherwise, varied cycle length will be the better approach because it still has the advantage of short startup latency and minimum buffer consumption, although the advantage is not as great as in an environment where disk bandwidth is under-utilized.
2. Cycle length is an important parameter that influences QoS of VOD and thereby determines how much profit can be gained from operating a VOD server. With varied cycle length, aggregate playback bandwidth deter-

mines cycle length automatically. However, with fixed cycle length, the VOD server administrator decides the cycle length. In this paper, we presented a method to find the cycle length that maximizes profit. In the configuration assumed in this study, a cycle length of about 8 seconds produced the maximum profit.

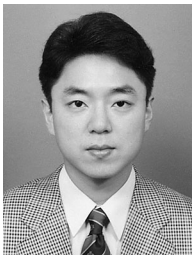
3. Prefix caching masks the weaknesses of both fixed cycle length and varied cycle length by preloading the initial portions of movies into memory. We analyzed how much data must be preloaded to mask startup latency completely, and presented a method to determine how many movies to preload to produce the maximum profit. For the configuration assumed in this study, preloading all the movies was the best choice.

References

[1] A. Silberschatz, P.B. Galvin, and G. Gagne, Operating System Concepts, Sixth ed., John Wiely & Sons, 2002.
[2] P.H. Seaman, R.A. Lind, and T.L. Wilson, "An analysis of auxiliary-storage activity," IBM Syst. J., vol.5, no.3, pp.158–170, 1966.
[3] M.-S. Chen, D.D. Kandlur, and P.S. Yu, "Optimization of the grouped sweeping scheduling (gss) with heterogeneous multimedia streams," Proc. ACM Multimedia, pp.235–242, 1993.
[4] D.R. Kenchammana-Hosekote and J. Srivastava, "Scheduling continuous media in a Video-On-Demand server," Proc. IEEE ICMCS, pp.19–28, 1994.
[5] A.L.N. Reddy and J. Wyllie, "Disk scheduling in a multimedia I/O system," Proc. ACM Multimedia, pp.225–233, 1992.
[6] Y. Rompogiannakis, G. Nerjes, P. Muth, M. Paterakis, P. Triantafillou, and G. Weikum, "Disk scheduling for mixed media workloads in a multimedia server," Proc. ACM Multimedia, pp.297–302, 1998.
[7] Y. Won, K. Cho, and S. Park, "Mitigating impact of starting new session in zoned disk," Proc. ACM Multimedia, pp.549–551, 2001.
[8] W.J. Bolosky, R.P. Fitzgerald, and J.R. Douceur, "Distributed schedule management in the tiger video fileserver," Proc. ACM SOSP, pp.212–223, 1997.
[9] P.J. Shenoy and H.M. Vin, "Cello: Disk scheduling framework for next generation operating system," Proc. ACM SIGMETRICS, pp.44–55, 1998.
[10] R. Wijayaratne and A.L.N. Reddy, "Techniques for improving the throughput of VBR streams," Proc. ACM/SPIE MMCN, pp.216–227, 1999.
[11] C.L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment," J. ACM, vol.20, no.1, pp.46–61, 1973.
[12] B.L. Worthington, G.R. Ganger, and Y.N. Patt, "Scheduling algorithms for modern disk drives," Proc. ACM SIGMETRICS, pp.241–251, 1994.
[13] S. Ghandeharizadeh, S. Kim, and C. Shahabi, "Continuous display of video objects using multi-zoned disks," Technical Report, Univ. of Southern California, 1995.
[14] W. Shi and S. Ghandeharizadeh, "Trading memory for disk bandwidth in video-on-demand servers," Proc. ACM SAC, pp.505–512, 1998.
[15] IBM Deskstar 120GXP, http://ssddom01.hgst.com
[16] A. Dan and D. Sitaram, "Buffer management policy for an on-demand video server," U.S. Patent no.5572645, 1996.
[17] P. Basu and T.D.C. Little, "Pricing considerations in Video-on-Demand systems," Proc. ACM Multimedia, pp.359–361, 2000.
[18] A.K.Y. Law, Y.V. Hui, and Z. Xiande, "Modeling repurchase frequency and customer satisfaction for fast food outlets," Int'al Journal of Quality Reliability Management, vol.21, no.5, pp.545–563, 2004.
[19] M.M. Davis and T.E. Vollmann, "A framework for relating waiting time and customer satisfaction in a service operation," The Journal of Services Marketing, vol.4, no.1, pp.61–69, 1990.
[20] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," Proc. IEEE INFOCOM, pp.1310–1319, 1999.
[21] J.F. Paris, D.D.E. Long, and P.E. Mantey, "Zero-delay broadcasting protocols for video-on-demand," Proc. ACM Multimedia, pp.189–197, 1999.

Ilhoon Shin received the B.S. and M.S. degrees in computer science from Seoul National University in 1998 and 2000. He is currently working toward the Ph.D. degree at Seoul National University. His current research interests include storage management for VOD server, QoS adaptive multimedia streaming, and web caching.

Kern Koh received the Ph.D. degree in computer science from University of Virginia. Now he is a Professor in the School of Computer Science and Engineering of Seoul National University. His current research interests include file system, web cache, multimedia system, and operating system.



Youjip Won received his B.S. and M.S. degree in Computer Science from the Department of Computer Science and Statistics, Seoul National University in 1990 and 1992, respectively and a Ph.D. in Computer Science from the University of Minnesota, Minneapolis, 1997. He worked for Intel Corp. from 1997 to 1999 as a server performance analyst. Since 1999, he has been on board of faculty members of Department of Electronics and Computer Engineering, Hanyang University, Seoul, Korea where he is currently Associate Professor. His current research is in the areas of Operating System, Computer Network, Performance Modeling and Analysis and etc. He is a member of IEEE and ACM. His professional activities include being on various program committees, and refereeing for journals and conferences.