International Journal of Advanced Smart Convergence Vol.13 No.3 31-40 (2024)
http://dx.doi.org/10.7236/IJASC.2024.13.3.31

IJASC 24-3-4

Periodic I/O Scheduling for the Storage of MPEG-DASH Video Servers

Seong Chae Lim

Professor, Dept. of Computer Science, Dongduk Women's University, Korea
sclim@dongduk.ac.kr

Abstract

The proliferation of video streaming services has led to a need for flexible networking protocols. As a result, the Dynamic Adaptive Streaming over HTTP (MPEG-DASH) protocol has emerged as a dominant streaming protocol due to its ability to dynamically adjust playback bitrates according to the end-user's network conditions. In this paper, we propose a novel I/O scheduling scheme tailored for the storage of MPEG-DASH-enabled video servers. Using the renowned rate-reservation (RR) algorithm and bulk-SCAN mechanism, our proposed scheme improves storage bandwidth utilization while ensuring seamless playback of streams with varying bitrates. In addition, we provide a mechanism for reclaiming the idle I/O time typically incurred while retrieving video segments from storage. Consequently, our scheme offers practical solutions for reducing the storage costs of MPEG-DASH video servers. With a simple cost model, we evaluate the performance enhancements achieved by our proposed I/O scheduling scheme.

Keywords: *MPEG-DSAH, Video Server, I/O Scheduling, Streaming Service, Periodic Scheduling.*

1. Introduction

In recent decades, there has been the remarkable growth of online video streaming services on the Internet. During this time, the ratio of network bandwidth caused by video streaming services has increased by around 24% annually, and now accounts for more than 65% of total Internet bandwidth [1]. This proliferation of streaming services partially relies on the advent of efficient protocols that can seamlessly transfer video contents to end-users' devices [2-4]. For a quality end-user experience, these protocols have been developed to adjust the data transfer rate depending on the varying network conditions of end users [3, 4]. For instance, when end-user's network conditions deteriorate, it is desirable to downgrade the bitrate of a video stream in service, thereby preventing hiccups. Conversely, if the network condition improves, the initial bitrate will be restored to enhance playback quality [4].

To facilitate such dynamic bitrate adjustments of video streams, the computing community has accepted the Dynamic Adaptive Streaming over HTTP (MPEG-DASH) protocol as a prominent protocol [2, 4]. The MPEG-DASH protocol enables video servers to store various bitrates of video segments for a single video stream. Correspondingly, an end-user has the ability of switching between different bitrates of video streams based on

Manuscript Received: July. 5. 2024 / Revised: July. 12. 2024 / Accepted: July. 17. 2024
Corresponding Author: sclim@dongduk.ac.kr
Tel: +82-02-940-4589, Fax: +82-02-940-4170
Professor, Dept. of Computer Science, Dongduk Women's University, South Korea

its network conditions. The widespread adoption of MPEG-DASH is largely due to its inherent openness that leads to extensive support by web browsers [5]. Currently, major Tech. companies such as Google, Apple, Netflix, and Amazon provide their video streaming services based on the MPEG-DASH protocol [6-8]. In this context, we propose an efficient I/O scheduling scheme that can improve the performance of storage systems of MPEG-DASH video servers.

In the MPEG-DASH specification, the playback time of a video content is divided into equal-sized time intervals, called Periods [3]. For the support of multiple bitrates of playback, different resolutions of video segments are stored for each Period. Since an MPEG-DASH stream repeatedly issues data requests according to its Periods, periodic I/O scheduling may be advantageous for MPEG-DASH streaming services. To enable this periodic scheduling, we employ the rate-reservation (RR) scheduling algorithm, originally devised for CPU scheduling of periodic tasks [9]. To apply the RR algorithm to our scheduling problem, we introduce the concept of a time unit in I/O scheduling. Based on this scheduling time unit, we execute bulk-SCANs to meet deadlines of data requests asking for video segments [10]. To further enhance the effectiveness of the bulk-SCAN, our proposed scheme reclaims idle times, which usually occur at the end of bulk-SCANs. This approach helps the devolvement of cost-effective MPEG-DASH servers. Using a cost model tailored for storage, we demonstrate the performance advantages of the proposed I/O scheme.

The paper is organized as follows. In Section 2, we describe the MPEG-DASH specification and our basic idea based on this protocol. Section 3 presents the proposed I/O scheduling scheme, and Section 4 shows performance benefits of the proposed scheme. Finally, we conclude the paper in Section 5.

2. Basic Idea

To design a cost-effective MPEG-DASH storage system, we first need to understand the specification of that. MPEG-DASH utilizes Media Presentation Description (MPD) files to expose playback options and locations of saved contents [3, 7]. The MDP file, downloaded to end-users from an MPEG-DASH server, divides playback duration into equal-sized time intervals, called Periods. For each Period of a stream, multiple Adaptations are defined to present distinct data types such as video, audio, subtitle texts.

Figure 1 gives a MPD file example where three Adaptations are made for the second Period. Let us assume that the first Adaptation is for video data. Then, more than one Representations can be saved for specifying different bitrates of video streams as in Figure 1. Each Representation has metadata and locations of stored video segments. In the figure, the second Representation is for a video stream with bitrate of 1.5 Mbps, and it contains the locations of the video segments that are played back during the second Period. Since the end-user has downloaded the MDP file at its local memory, it can change the bitrates of a stream by using multiple Representations made for a Period [5, 7].

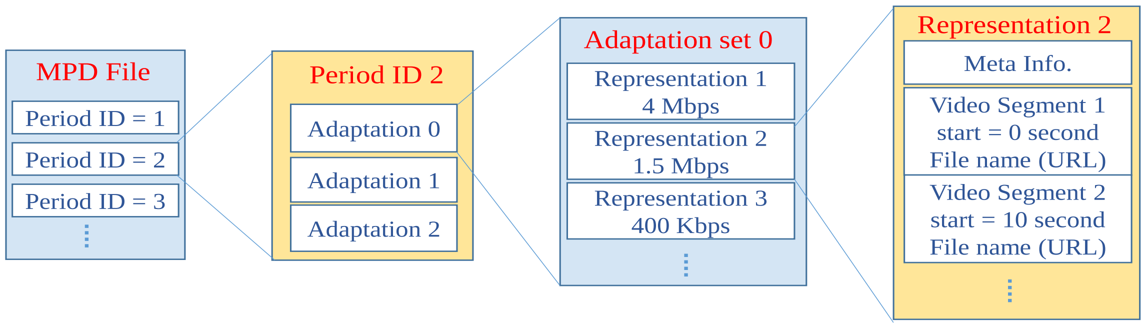

Figure 1. Typical organization of an MPD file.

As depicted in Figure 1 MPEG-DASH streaming is served based on Periods specified in an MDP file. In the respect of storage of a video server, we can make I/O scheduling more efficient by using a periodic scheduling concept. Recall that video segments are requested on the per-Period basis. That is, an MPEG-DASH end-user sends request messages asking for consecutive segments $[v_1,\ v_2,\ v_3, \ldots, v_n]$ for a given Period. In that case, those video segments of $v_i\ (1 \le i \le n)$ have the same deadline. To prevent hiccups, all the video segments for Period k have the deadline of the beginning point of Period k. Such deadline assignment is repeated for each Period during playback.

To leverage the periodic nature of I/O requests in MPEG-DASH streams, we adopt the RR (Rate Reservation) algorithm for our I/O scheduling problem [9, 10]. Since the RR algorithm was originally devised for CPU scheduling of periodic tasks, its adoption for our problem requires a conceptual bridge between the usage of CPU time and the usage of I/O bandwidth.

For this, we introduce the concept of the rate of bandwidth usage, which is conceptually the same as CPU utilization computed for a periodic task. To compute the bandwidth rate, our scheduling scheme utilizes the fact that the endpoints of stream's Periods only arise at multiples of a given time unit. Our time unit concept is analogous to the minimum interval defined in the RR scheduling algorithm. Since there is a minimum time interval between the deadlines of video segments, we can perform bulk-SCANs to reduce seek-time I/O overheads. Further details are presented in the following section.

3. Proposed Algorithm for MPEG-DASH Server Storage

3.1 Rate-Reservation Algorithm

The RR algorithm is a CPU scheduling algorithm used for processing mixed workloads of periodic and sporadic tasks [9]. The algorithm assumes that all the occurrences of periodic tasks have a minimum occurring interval. This interval is termed as the *minimum interval* in the RR algorithm. Moreover, release points of period tasks coincide with any of integral multiples of the minimum interval. Therefore, if we denote the length of the minimum interval by L, then all deadlines of periodic tasks arise time points of $n \times L$ (n = 1, 2, …) [9, 10]. A notation task $\tau(C, k)$ represents a periodic task that requires CPU time of length C with a period length of $k \times L$. The task is first released at time $s \times L$, then its scheduling deadlines are the same as $(s + i \times k) \times L$ (i = 1, 2, 3, …) where s and i are integers.

The RR scheduler computes the CPU utilization of task τ(C, k) as $\frac{C}{k \times L}$, and ensures deadline-guaranteeing scheduling of periodic tasks while their total CPU utilization does not exceed 1. The total CPU utilization U is computed as $\sum_i C_i/(k_i \times L)$, for admitted tasks $\tau_i(C_i, k_i)$. More specifically, the RR scheduler can guarantee deadlines of all periodic tasks by using only the time of U $\times L$ in each minimum interval. During each minimum interval, the RR scheduler serves periodic tasks within U $\times L$ time according to the Earliest Deadline First (EDF) policy. Since the remaining time of (1 – U) $\times L$ is free, it can be utilized for serving sporadic tasks. By repeating this scheduling in every minimum interval, the RR scheduler achieves optimal CPU utilization for mixed workloads of periodic and sporadic tasks [9].

3.2 Adoption of the RR Algorithm for I/O Scheduling

An MPEG-DASH stream issues requests for video segments on a period basis, by choosing a Representation suitable to its network condition [5, 8]. This is illustrated in Figure 2. In the figure, a video stream has three Representations that are made for locating different bitrates of video segments. We also denote the videos stream being coded with different bitrates of video segments by A, B, and C. The representations for those video stream groups are denoted by $R_{A,p}$, $R_{B,p}$, $R_{C,p}$, respectively. Here, p is the ID of the associated Period. In this example, if the stream changes its Representation from $R_{A,1}$ to $R_{B,2}$ in the next Period, then it will issue data requests asking for segments $B_{2,1}$, $B_{2,2}$, …, $B_{2,n}$ for Period 2.

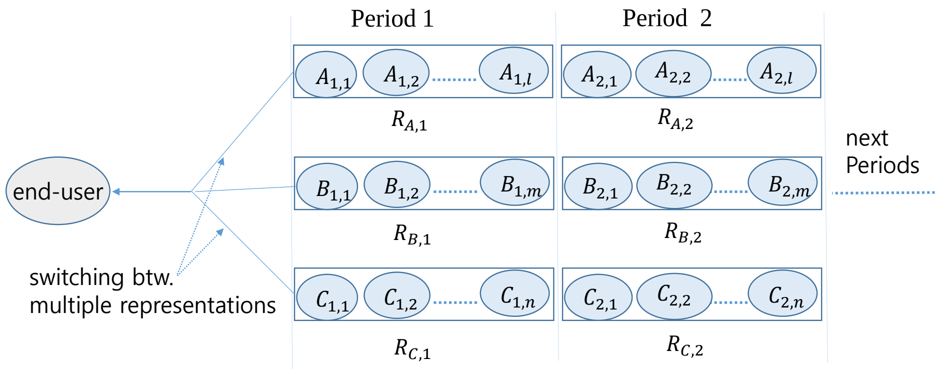
Period 1
Period 2
$A_{1,1}$ $A_{1,2}$ $A_{1,l}$
$A_{2,1}$ $A_{2,2}$ $A_{2,l}$
$R_{A,1}$
$R_{A,2}$
next Periods
end-user
$B_{1,1}$ $B_{1,2}$ $B_{1,m}$
$B_{2,1}$ $B_{2,2}$ $B_{2,m}$
$R_{B,1}$
$R_{B,2}$
switching btw. multiple representations
$C_{1,1}$ $C_{1,2}$ $C_{1,n}$
$C_{2,1}$ $C_{2,2}$ $C_{2,n}$
$R_{C,1}$
$R_{C,2}$

Figure 2. An example of video streaming over two periods.

In our I/O scheduling, the deadlines of video segments are set to the beginning points of their Periods. For example, the deadline of requests for $A_{2,i}$ $(1 \leq i \leq l)$ is set to the beginning point of Period 2. By reading video segments before their Period, hiccup-free playback is guaranteed during that Period [10]. If we make Periods of streams have a minimum interval between them, we can also utilize the concept of the minimum interval time of the RR scheduler. Since Periods of MPEG-DASH streams are specified in seconds [3], we can choose one second as the minimum interval for our I/O scheduling. In this paper, we refer to this as *TUI* (Time Unit for I/O scheduling).

The RR algorithm is based on the concepts of CPU utilization of a periodic task and the total utilization for all served periodic tasks. Similarly, we employ the I/O bandwidth utilization of an MPEG-DASH stream, called the bandwidth usage rate (*BUR*). For a stream requesting *k* video segment per *n* seconds, its BUR is calculated as in Eq. (1).

$$\text{BUR} = \frac{k}{K \times n}, \text{ where } K \text{ is an given I/O parameter.} \tag{1}$$

In Eq. (1), the integral parameter K represents the number of video segments retrieved within a single TUI (i.e. one second). The way to determine K will be described in the following subsection. The total usage of I/O bandwidth is computed by adding up all the BURs of served streams. To retrieve video segments on a per-TUI basis, our I/O scheduler processes a batch of I/O requests within each TUI, selecting those I/O requests based on the RR policy.

3.3 Proposed Scheduling Algorithm

A file system is usually organized and accessed in the unit of a block that resides on a physically continuous region in storage [6, 11]. A video segment is stored in a storage block, and a data request is issued for reading it. For a stream with a greater bitrate, the MPEG-DASH server transits more video segments for each Period.

We consider two types of MPEG-DASH storage: HDD (Hard Disk Drive) and NAND SSD (Solid State Device). In the case of SSD storage, the parameter K is easily obtainable [10, 11]. Since flash memory provides a uniform time for reading a block regardless of its location in storage, the value of K can be computed by using I/O specifications of a target SSD [11]. When we have the time t for reading a block from the SSD specifications, we compute K such that K = $\lceil \frac{TUI}{t} \rceil$.

Unlike SSD storage, HDD storage cannot read a block in a uniform time due to its rotational delay and seek times. Consequently, the number of video segments readable during a TUI varies depending on their locations on HDD. For this reason, deadline-meeting scheduling for HDD storage is not feasible without significantly sacrificing I/O performance [10, 12]. To address this problem, we borrow a SCAN-based disk scheduling scheme proposed in the prior literature [10, 12]. In the literature, the authors introduced the bulk-SCAN scheme, which performs one-directional disk movements to retrieve a group of blocks in batch mode. By retrieving a group of blocks in a single disk scan, the bulk-SCAN scheme significantly reduces seek-time overheads.

In the literature, the time to retrieve k blocks using a bulk-SCAN is computed as shown in Eq. (2). In the equation, T_{rev} denotes the disk revolution time, and T_{cyl} denotes the total number of disk cylinders. The function $T_{seek}(d)$ returns the seek-time required to relocate the disk head across d cylinders. The proof of Eq. (2) can be found in [12].

$$T(k) \leq k \times T_{rev} + (k+1) \times T_{seek}\left(\frac{N_{cyl}}{k+1}\right) \tag{2}$$

Using Eq. (2), we can compute a deadline-guaranteed number of blocks readable within a TUI. By selecting an integer that satisfies Eq. (3) below, we can determine the system parameter *K*.

$$T(K) \leq TUI < T(K+1) \tag{3}$$

With the parameter *K*, we can compute the deadline-guaranteed I/O bandwidth of the storage. We denote that by B_{UTI}, and its value is calculated using Eq. (4).

$$B_{UTI} = \frac{BlockSize \times K}{UTI} \text{ (bytes/sec.)} \tag{4}$$

In Eq. (4), B_{UTI} represents the size of a block in storage. From Eq. (3) and Eq. (4), we have determined two parameters: K and B_{UTI}, respectively. Using these parameters, we schedule I/O requests from MPEG-DASH streams based on the RR algorithm.

We describe the proposed scheduling algorithm using Figure 3, which is executed at the beginning of each TUI. In line 3, the proposed algorithm calculates the total BUI of the video streams in service. Then, it admits new video streams within the free capacity of storage bandwidth by following the steps of lines 4-7. In lines 8-12, the algorithm places data requests for serviced video streams into queue Q_v, while assigning appropriate periodic deadlines to these requests for a bulk-SCAN. The bulk-SCAN then retrieves the data requests in Q_v as described in lines 13-16. Finally, the algorithm computes the size of idle time and performs non-video data retrieval by reclaiming this idle time, as described in lines 17-18.

```
Algorithm 1: Procedure DoBulkScanEachUTI

[Input] : $T_i$ : beginning point of the current UTI;
          $K$ : storage bandwidth parameter;
          $Q_s$, $Q_w$ : queues of stremas in service or waiting for admission;
          $Q_v$ : queue of I/O requests for reading video segments;

1  /* executed at each beginning point of UTIs */
2  Compute the total bandwidth utilization as follows:
3      $U_T = \sum_s U(s)$, for $s \in Q_s$; // U(s) is bandwidth utilization of $s$
4  foreach $s \in Q_w$ do // admission checking
5      if $U_T + U(s) \leq 1$ then
6          Move $s$ from $Q_w$ into $Q_s$;// starting of $s$'s playback
7          $U_T = U_T + U(s)$;
8  foreach $s \in Q_s$ do // request issues for streams' next Periods
9      if $s$'s new Period starts at $T_i$ then
10         Place I/O requests for the per-period video segments of $s$ into $V_{tmp}$;
11         foreach $r \in V_{tmp}$ do
12             Set the deadline of $T_i + p \times UTI$ to $r$, and place r into $Q_v$;
13 Compute a bulk-SCAN capacity $k$ such that $k = \lceil K \times U_T \rceil$;
14 Select up to $k$ requests in $Q_v$ by following the RR algorithm;
15 Perform a bulk-SCAN to retrieve video segments of requests selected above;
16 Transfer the retrieved video segments to the target MPEG-DASH end-users;
17 Calculate idle time $t$ such that $t = T_i + UTI - t_c$; // $t_c$ is current time
18 Retrieve non-video data during time of $t$; // reclamation of idle time
```

Figure 3. Scheduling algorithm executed at a TUI starting point, T_i.

4. Advantages of the Proposed Scheduling Scheme

4.1 Soundness of the Proposed Algorithm

The RR algorithm guarantees deadlines of periodic tasks, while also serving sporadic tasks. The schedulability of the algorithm relies on the assumption that all periodic tasks are released only at the boundaries of a specific minimum interval [9, 10]. Figure 4 illustrates an example of RR scheduling scenarios occurring over two minimum intervals. In the figure, the notation τ_n^i represents the n-th scheduled periodic task during the i-th minimum interval. During these minimum intervals, three and two periodic tasks are scheduled, respectively.

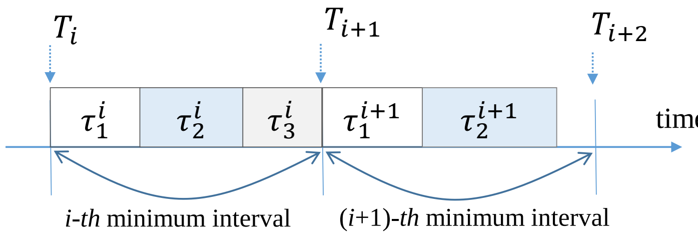

Figure 4. An example of periodic task scheduling using the RR algorithm.

In the case of (*i+1*)-th minimum interval shown in Figure 4, some idle time has arisen at the end of it. This idle time is common in RR scheduling because the CPU utilization of periodic tasks is not 100% full. To reclaim this idle time, the RR scheduler can use it for scheduling sporadic tasks. More specifically, such reclamation time is calculated as the (1 - U_T) portion of a minimum interval. This proactive scheduling of sporadic tasks helps ensure optimal performance when a mixture of periodic and sporadic tasks is scheduled [9].

Now, we demonstrate the scheduling capability of the proposed RR-based I/O scheduling. Let us look into the scheduling of tasks τ_1^i, τ_2^i, τ_3^i in Figure 4. When d(τ) represents the deadline of a task τ, it holds that d(τ_1^i) $\leq d(\tau_2^i) \leq d(\tau_3^i)$. In the case of the RR algorithm, we can freely rearrange the scheduling orders of τ_1^i, τ_2^i, and τ_3^i without missing any deadline. Recall that all deadlines arise only at the ends of minimum intervals, namely T_j (i < j).

The above ability to permute scheduling orders is vital for our I/O scheduling scheme designed for MPEG-DASH servers. When a bulk-SCAN retrieves video segments during a TUI, the retrieval order depends on their relative cylinder locations, regardless of their deadlines. Since the deadlines of video segments coincide with the ends of TUIs, our I/O scheduling scheme does not need to consider the urgency of deadlines of video segments, similar to CPU scheduling of periodic tasks in the original RR scheduler. Consequently, the proposed scheduling algorithm can ensure deadline-compliant I/O schedules by executing bulk-SCANs for HDD storage.

Unlike HDD storage, flash storage offers uniform time for random reads of video segments. As a result, the EDF algorithm can be used for determining the retrieval orders of video segments. If n segments are retrieved in a TUI, then n can simply be used as the parameter K. Consequently, our proposed scheme is applicable to MPEG-DASH services on both HDD storage and SSD storage platforms [13].

4.2 Evaluations of Enhanced I/O Performance

As mentioned previously, our proposed scheduling scheme is applicable to both SSD storage and HDD storage. Since HDD storage is more commonly accepted for streaming services [6], we focus on analyzing the performance advantages in the case our scheme is used for an HDD storage system. To enhance I/O efficiency, our scheduling scheme performs bulk-SCANs with deadlines. Since the time estimation for bulk-SCAN scheduling is based on worst-case I/O scenarios, each TUI usually contains some idle time in itself.

For instance, the (*i*+1)-th TUI contains idle time, as shown in Figure 4. This idle time may occur for two reasons: (*i*) the actual retrieval time is shorter than the pre-estimated worst-case time in scheduling, and (*ii*) the workload of streams does not always consume the maximum bandwidth capacity. Therefore, to improve the utilization of HDD storage, it is crucial to reclaim this idle time efficiently.

To reclaim idle time, we capitalize on the idea used in the original RR scheduler for serving sporadic tasks. Our scheduler allocates a portion of I/O bandwidth to serve data requests issued from MPEG-DASH streams. Then, we estimate the remaining time until the end of the current TUI. When we denote the idle time size by t_{idle}, our scheduler can compute the number of blocks that a bulk-SCAN retrieves within t_{idle}. This computation is done using the function $F(t_{idle})$ described below.

$$n = F(t_{idle})\left\{return\ the\ greatest\ n\ satisfying\ that\ T_{seek}\left(\frac{N_{cyl}}{n+1}\right) + n \times T_{rev} \leq\ t_{idle}\ ;\right\}$$

Using the return value *n*, our scheduler retrieves up to *n* additional blocks. When selecting these additional blocks, our scheduler prefers non-video blocks. This reclamation of idle time is conceptually similar to the way the original RR scheduler serves sporadic tasks in each minimum interval time [9, 10].

To demonstrate the performance advantages of our scheduling scheme, we evaluate the size of I/O bandwidth that is reclaimed by retrieving non-video blocks using idle times. For this, we invent a simple performance evaluation model. The performance model is based on a modern HDD with the hardware specification given in Table 1. In the table, the capacity and speed of a Seagate HDD are described. The HDD has the sector size of 4 KB and the maximum rotational time of about 8.3 milliseconds, respectively.

Table 1. Hardware specification of Seagate ST2000DM008

Prameters	Data
Disk Capacity	2 TBytes
Interface Speed	600 MB/sec (SATA)
Disk Rotational Speed	7200 RPM
Total Cylinder Number (N_{cyl})	16,383
Sector Size	4096 Bytes
Average No. of Sectors per Track	63

In the proposed scheduling scheme, two factors mainly dominate the amounts of idle time within TUIs. First, the length of a TUI has an impact on the size of idle time, and this length varies by adjusting the value of parameter K. The larger the value of K, the longer the TUI. Increasing the length of TUI typically leads to an increased amount of idle time. Second, the block size influences the amount of idle time. An MPEG-DASH video file consists of sequential video segments, each of which is stored in a single disk block. In this situation, as the block size gets smaller, the ratio of rotational delay to the disk revolution time increases.

From these observations related to two factors above, we can estimate performance advantages obtainable in the proposed scheme. Table 2 shows the performance enhancements that are estimated based on our performance evaluation model. In the table, Case A/B/C (x%) represent evaluation environments, where the real number x denotes the percentage of average rotational delay to the disk revolution time. As stated before, this depends on the block size. By varying the values of K, we can achieve I/O enhancement using the proposed scheduling scheme that reclaims idle time.

Table 2. Performance enhancements with respect to scheduling parameters.

Three Simulation Env.	$K = 50$	$K = 100$	$K = 150$
Env. Case A (10%)	6%	7%	7.3%
Env. Case B (20%)	14%	15%	15.3%
Env. Case C (30%)	22%	23%	23.3%

The real number *x* is calculated as (*RT*-*AT*)/*RT*. Here, *RT* represents the revolution time of the HDD listed in Table 1, and *AT* is the actual rotational delay time taken to read a video segment. The magnitude of *AT* depends on the block size. When we use larger block sizes, the rotational delay *AT* decreases. Since a video segment typically occupies less than 4% of a track's size, the ratio of rotational delay is generally below 50%. Therefore, we set the ratio to 10%, 20%, and 30% in Table 2 to reflect this observation,

In Table 2, it is clear that the size of TUI does not significantly affect the performance enhancement. Recall that the size of a TUI is dependent on the value of parameter *K*. Instead, the ratio of rotational delay acts as a crucial factor in our approach. By setting *K* to 100, we adjust the size of TUI to one second. Since the Periods of MPEG-DASH streams are defined in the unit of one second, setting *K* to around 100 is reasonable. Under these conditions, our proposed mechanism improves I/O performance by 7% to 23%, as shown in Table 2. Consequently, we can say that the proposed scheduling scheme ensures fast I/O times for reading non-video data as well as the video segments of MPEG-DASH streams.

5. Conclusion

MPEG-DASH is the most popular protocol used for video streaming services on the Internet. The protocol supports the dynamic adjustments of stream bitrates to ensure seamless playback under challenging network conditions. Correspondingly, an MPEG-DASH video server must dynamically allocate I/O bandwidth to accommodate MPEG-DASH streams with varying bitrates. To this end, we proposed a novel Rate-Reservation (RR) scheduling scheme suitable for the storage needs of MPEG-DASH video servers. To employ the RR algorithm for our I/O scheduling, we used the bulk-SCAN scheme suitable for retrieving video segments in a periodic manner. This periodic scheduling is largely suited for MPEG-DASH streams, which consume data according to the periods specified in their MPD files. Thanks to deadline-guaranteeing bulk-SCANs, our scheme offers improved storage bandwidth and hiccup-free delivery of video segments. Moreover, by reclaiming idle time arising in each bulk-SCAN, we can boost I/O performance for HDD storage. Through analytical modeling, we have demonstrated that the proposed scheme can reduce storage costs for MPEG-DASH streaming services.

Acknowledgement

This work was supported by the Dongduk Women's University grant in 2022.

References

[1] Sandvine, Global Internet Phenomena Report, https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2023/reports/Sandvine%20GIPR%202023.pdf.

[2] CDNetworks, "How MPEG-DASH is Revolutionizing Video Streaming?", https://www.cdnetworks.com/media-delivery-blog/mpeg-dash-revolutionizing-video-streaming/

[3] R. Pantos and E.W. May, "HTTP Live Streaming," IETF Internet Draft, 2017.

[4] Michail Michalos and Stelios Kessanidis, "Dynamic adaptive streaming over HTTP," Journal of Engineering Science and Technology Review, Vol. 5, No. 2, pp. 30-34, June 2012. DOI:10.25103/jestr.052.06.

[5] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP," IEEE Communications Surverys & Tutorials, Vol. 21, No. 1, pp. 562-585, 2019. DOI: 10.1109/COMST.2014.2360940

[6] A. Abdelsalam, M. Luglio, M. Quadrini, C. Roseti, and F. Zampognaro, "Analysis of DASH Performance over Time-varying End-to-end Links," Computers & Electrical Engineering, Vol. 84, June 2020.

DOI: 10.1016/j.compeleceng.2020.106623.

[7] Dorsaf Sebai, "MPEG-DASH Parametrisation for Adaptive Online Streaming of Different MOOC Videos Categories, " Multimedia Tools and Applications, Vol. 80, pp. 33193–33212, August 2021.
DOI:10.1007/s11042-021-11352-7

[8] A. Sideris, E. Markakis, N. Zotos, E. Pallis, and C. Skianis, "MPEG-DASH users' QoE: The Segment Duration Effect," in *Proc*. of the International Workshop on Quality of Multimedia Experience (QoMEX), pp. 262-9, Jul. 2015.
DOI:10.1109/QoMEX.2015.7148117

[9] Kang G. Shin and Yi-Chieh Chang. "A Reservation-Based Algorithm for Scheduling Both Periodic and Aperiodic Real-Time Tasks", IEEE Trans. on Computers, Vol. 44, No. 12, pp. 405–1419, 1995.
DOI: 10.1109/12.477246

[10] Seong-Chae Lim, "Calculation of Free Bandwidth for Rate-reservation EDF Scheduling in Flash Storage," International Journal of Computer Sciences and Engineering, Vol.8, No. 4, Apr 2020.
DOI: 10.26438/ijcse/v8i4.115

[11] Stephan Baumann, Giel de Nijs, Michael Strobel, and Kai-Uwe Sattler, "Flashing Databases: Expectations and Limitations," in *Proc.* of ACM Data Management on New Hardware, pp. 9-18, June 2010.
DOI: 10.1145/1869389.1869391

[12] D. Kandlur M. Chen and P. Yu. "Optimization of Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Systems". in *Proc*. of the ACM Multimedia., pp. 235 –242, Sept. 1993.
DOI: 10.1145/166266.166293

[13] Jaeseung Kim, Seyun Choi, Seunghyun Lee, and Soonchul Kwon, "Real-Time Earlobe Detection System on the Web," International Journal of Advanced Smart Convergence, Vol. 10, No. 4, pp. 110-116, 2021.
DOI: 10.7236/IJASC.2021.10.4.110