IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.8, Issue.4, Apr 2020 E-ISSN: 2347-2693

Calculation of Free Bandwidth for Rate-reservation EDF Scheduling in Flash Storage

Seong-Chae Lim

Department of Computer Science, Dongduk Women's University, Seoul, South Korea

DOI: https://doi.org/10.26438/ijcse/v8i4.14 | Available online at: www.ijcseonline.org

Received: 1/Apr/2020, Accepted: 13/Apr/2020, Published: 30/Apr/2020

***Abstract*—**In the long history of database communities, lots of research efforts had been done for reducing seek-times and rotational delays caused by mechanical components used in HDD (Hard Disk Drive). As an example of those efforts, some I/O scheduling algorithms were devised for the purpose of efficient services of online video streams being pumped up from HDD storage. To this end, a rate-reservation EDF is recently adopted to be incorporated into the recent platform built on flash storage. In this research, a fixed length of time is chosen as a period unit and the disk bandwidth assumption of each video stream is decided based on that time. The previous rate-reservation EDF algorithm is very suitable for serving a mixture of real-time requests and common requests without deadline. In this paper, we propose a new way that can dynamically compute the varying amounts of free bandwidth arising from more-than-reservation reading, while scheduling video streams according to the rate-reservation EDF algorithm. For this, we devised two data structures that can keep information about workloads and free bandwidth over a certain length of period units. Using scheduling information managed in those data structures, our proposed scheme can efficiently utilize slack times that occur unexpectedly from time to time. Because of the efficient reclamation of slack times, our scheme can improve the actual I/O performance of flash storage that is prepared for Web-based streaming services.

Keywords—***Flash Memory, Video Streaming, I/O Scheduling, Real-time EDF Algorithm***

I. INTRODUCTION

In the past time, a lot of research on I/O scheduling was done for the purpose of reducing I/O overheads in HDD (hard disk drive) storage [1-11]. That is, diverse algorithms for optimizing seek times and rotational delays were proposed for efficient usage of HDD storage. Among them, the well-known algorithms such as SCAN and C-SCAN are widely implemented in reality because of their algorithmic simplicity and relatively low overhead of seek times [8, 9, 11]. Since these two scheduling algorithms are not suitable for serving I/O requests with deadlines, they cannot be used for servicing on-line video streams without some modifications to them. In this context, some other algorithms have been proposed to incorporate real-time features into SCAN-style algorithms [8, 10]. Although those algorithms work by considering deadlines of I/O requests in scheduling, they have an inherent limitation due to their feature of batch-style scheduling [3, 7-9].

Recently, [12] propose a I/O scheduling scheme that employs an EDF-style algorithm for serving video streams in flash storage. In this research, scheduling priorities of I/O requests with deadlines are given by using the earliest-deadline-first (EDF) policy. To admit video streams within available storage bandwidth, storage bandwidth is allocated to streams based on the Rate-reservation EDF (RR-EDF) algorithm [5]. In nature, this EDF-style algorithm is suitable for flash storage, rather than HDD storage, because HDD storage suffers from large variations of service times for a given set of I/O requests, depending on their scheduling orders [5]. This is because HDD storage has mechanical parts having seek times and rotational delays [3, 9, 10]. Differently from this, flash memory has no mechanical components, and thus it has very uniform service times of data requests, regardless of their physical locations [12, 13]. Therefore, the EDF-style algorithm can be easily employed in flash storage, since the service orders of data requests barely affect the total read time [14-17].

In this pa98per, we propose a way to compute the amounts of varying slack times, while we are scheduling I/O requests according to the RR-EDF (rate-reservation EDF) algorithm. The proposed technique considers a realistic online streaming service, where a mixture of video data with timing constraints and non-video data without timing constraints are retrieved together. Note that this assumption is feasible for Web-based streaming services. The idea of adaptation of an EDF-style scheduling algorithm was previously proposed for its use in flash storage [12, 13].

In this research, a fixed length of time is selected as a period unit and the disk bandwidth assumption of each video stream is decided based on the period unit. Then, a served stream issues periodically its data requests in proportion to its bandwidth consumption. The period of that stream is equivalent with a certain multiple length of the period unit. Since flash memory can support very uniform read times against varying scheduling orders of a given set of I/O requests, the EDF scheduling algorithm is possible in the case of flash storage. This is different from HDD storage that has varying read times of a given set of I/O requests according to their scheduling orders. The RR-EDF algorithm for flash storage uses a rate of disk bandwidth for serving video's requests and allocates the rest of disk bandwidth to non-video requests. The reservation rate can be easily computed by adding the amounts of disk bandwidth consumed by all served streams [5, 6]. By assigning the rest free disk bandwidth to non-video requests, the RR-EDF algorithm can diminish the average response time of those requests.

Assume that a RR-EDF scheduler has allocated a given rate of disk bandwidth in a period unit for video streams. If there is no non-video request at this time, the remaining free disk bandwidth will be wasted [12, 14]. Of course, the remaining bandwidth can be used for serving request from any video streams. However, the earlier service of the video requests cannot enhance the system performance without any technique that can be calculated the effect of the earlier services in the view of long-term scheduling scenarios.

In this light, we propose a technique that can assess the effect of earlier services of video data in a long viewpoint. From this, the proposed technique can reserve a greater rate of disk bandwidth that is given to non-video data. For this, we use two kinds of data structures that bookkeeps workload bandwidth and additional free bandwidth. The additional free bandwidth is obtained from the extra reading of video data, which is called by more-than-reservation reading. The more-than-reservation reading is for serving data requests with deadlines. The two information are based on an extension algorithm of the RR-EDF algorithm. To utilize those two data structures, we use a fixed maximum length on scheduling periods of video streams. From this, we can improve the actual I/O performance of flash storage established for Web-based streaming services.

The organization of this paper is as follows. In Section II, we give some preliminary knowledge about HDD scheduling algorithms and their major issues. In Section III, we propose a way used for evaluating slack times obtainable from earlier services of video's data requests, thereby improving performance of flash storage. Lastly, we conclude this paper in Section IV.

II. BACKGROUND

In this section, we address the basic idea about the well-known rate-reservation (RR) EDF algorithm. Originally, the RR-EDF algorithm was devised for scheduling real-time tasks that consume a fixed amount of CPU times in a periodic fashion [5, 13, 15]. By adopting the RR-EDF algorithm, [10] proposed a scheme that can efficiently handle a mixture of video and non-video requests in flash storage. In that research, a fixed length of a scheduling time is chosen as a period unit. Then, the number of data pages retrievable within the period unit is computed by considering I/O specifications of a target flash storage. If that number of retrievable pages is N and the size of the period unit is T, then the disk bandwidth is given with $(D \times N)/T$, where D is the size of a data page.

When admitting a video stream Si requiring bandwidth consumption of b_i, the RR-EDF scheduler chooses two integers p_i and n_i such that $(D \times n_i)/(p_i \ \times T) \geq b_i$. To minimize wastes caused by bandwidth fragmentations, p_i and n_i need to be selected so as to get the smallest bandwidth gap, i.e., $(\mathrm{D} \times n_i)/(p_i \ \times T) - b_i$. For the set of admitted streams S_i $(1 \leq i \leq k)$, the reservation rate U is computed as $\sum_{i=1}^{k}(n_i/(p_i \times N))$, and it is managed to be always below 1. For each period unit, the RR-EDF scheduler reads $U \ \times N$ number of data pages requested by admitted video streams according to the EDF policy. From the schedulability of rate-reservation EDF algorithm, it is proven that an RR-EDF scheduler guarantees deadlines of data requests all the time. Since the remaining bandwidth of $(1 - U) \times N$ is free, the free bandwidth can be assigned for the service of non-video data requests during each period unit. Here, N is the number of pages that can be retrieved in a period unit

To see the RR-EDF algorithm in detail, we present a schedule that is likely to be made for servicing streams S_1 , S_2, and S_3 . We assume that S_1 , S_2, and S_3 equivalently consumes 3 pages for 1, 2, and 3 period units, respectively. In other words, S_i consumes bandwidth of $(3 \times \ D)/(i \ \times T)$ in size. According to the definition of the RR-EDF algorithm, the reservation rate of bandwidth is computed as $\frac{3\times D}{1\times T} + \frac{3\times D}{2\times T} + \frac{3\times D}{3\times T}$. If we have flash storage with N = 8, then the reservation rate is computed as 33/48, approximately, 0.69.

Figure 1 depicts a scheduling scenario, where data requests from streams S_1 , S_2, and S_3 have been served according to the RR-EDF algorithm. In the figure, the rectangle labeled with Si represents the page read for Si, and the top half of that figure shows how the video streams issue their data request for playback. In the figure, the scheduling periods of S_1 , S_2, and S_3 are one, two, and three period units, respectively. Since $\lceil U \ \times N \rceil$ is six in the case of Figure 1, up to six pages are retrieved in a period unit. The deadlines of requests coincide with the time points of next issues of requests from video streams.

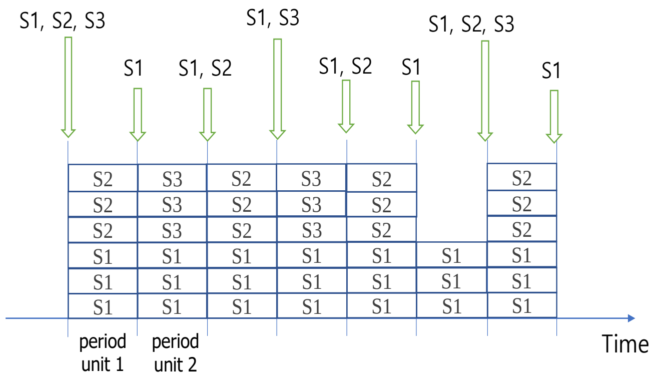

Figure 1. A Scenario for three video streams using an RR-EDF algorithm.

III. Proposed Scheme

A. Basic Ideas

In Figure1, we gave an example of possible I/O schedules that are made by an RR-EDF scheduler. Since the maximum bandwidth of assumed flash storage is that for reading eight pages in a period unit, free bandwidth is the size of reading two pages of non-video data requests in a period unit. If free bandwidth is used for servicing more data requests than an expected number of requests, then earlier services of video's data requests yield larger free bandwidth for period units in the future.

For instance, if we read two pages for stream S_3 additionally in period unit 1, then we obtain additional free bandwidth in period units of 2 and 3. Note that the deadlines of data requests from S3 are the same as the end of period unit 3. Through the earlier service of scheduled data requests, in other words, the RR-EDF scheduler can get additional free bandwidth used for reading data requests of non-video requests. From this, our RR-EDF scheduler can generate very effective real-time schedules for data requests from video streams.

To implement such an RR-EDF scheduler supporting a capability of more-than-reservation reading of videos' data requests, we need to devise a way for calculating the amounts of free bandwidth that varies over future's period units. For that calculation, we employ the concept of exchangeability of scheduling. When we have two data requests r_1 and r_2 with the same deadline, it is the case that these two requests are exchangeable in scheduling order. Therefore, we can freely select a scheduling order between $r_1 \rightarrow r_2$ and $r_2 \rightarrow r_1$.

Based on the before-mentioned scheduling exchangeability, we can trace amounts of free bandwidth available in period units. For this, we select an integer parameter, i.e., *max_period*, and compute the bandwidth required by served video streams until that maximum period. If we pick the value of parameter *max_period* as 10, then we compute bandwidth requirements for 10 period units and enforce every video stream to have its period length that is not greater than the length of 10 period units. For instance, the bandwidth required is computed as follows for the scheduling parameters, N = 8, U = 0.8, and *max_period* = 10.

W_i (r_1 = 8, r_2 = 8, r_3 = 8, r_4 = 8, r_5 = 8, r_6 = 8, r_7 = 8, r_8 = 8, r_9 = 8, r_{10} = 8).

In the notation above, we get the bandwidth requirement by computing $U \times N$ at the time of i-th period unit. If a new video stream is admitted in the i-th period unit and increases U by 0.1, then our scheduler becomes to get the data structure follows as:

W_{i+1} (r_1 = 9, r_2 = 9, r_3 = 9, r_4 = 9, r_5 = 9, r_6 = 9, r_7 = 9, r_8 = 9, r_9 = 9, r_{10} = 9).

This data structure is used for saving varying additional free bandwidth. This extra bandwidth can be used for reading more data in a faster time.

B. Proposed Algorithm

In this research, we consider the modern SSD whose H/W specification is shown in Table 1. The storage unit of that SSD storage is implemented as MLC (Multi-Level Cell) transistor, and the storage capacity is 500 GB.

Table 1. H/W Specs. of Samsung 970 Evo SSD.

Parameters	Data in detail
Storage Capacity	500GB
Cell Type	V-NAND 3-bit MLC
Interfaces Used	PCIe Gen 3.0 x4, NVMe 1.3
Size of a Page	4KB
Random Read Speed	Up to 15K IOPS.
Random Write Speed	Up to 50K IOPS

The proposed scheme is used for computing free bandwidth in a dynamic manner. For this, it is required to compute the extra free bandwidth that is obtainable from more-than-reservation reading of video data. We denote the numbers of data requests that have been processed from more-than-reservation reading by the term of FB_i (e_1 , e_2, e_3, ... , $e_{\max\_p}$). Here, the notation of e_1 denotes the extra free bandwidth that was gained by serving a data request with a deadline of the end point of i-th period unit.

In other words, if e_1 of $FB_i()$ is equal to k, then it is the case that the RR-EDF scheduler has read k number of pages with the same deadline as the end point of (n+i)-th period unit. By dynamically managing the two data structures of W_i and FB_i (), the RR-EDF scheduler can execute very flexible scheduling tasks, thereby improving the performance of flash storage used for servicing video streams in an online mode.

In Figure 2, we present the algorithm used for dynamically calculating free bandwidth, which is gained because of more-than-reservation reads. The algorithm is used for making a schedule that is executed during j-th period unit.

In the algorithm, we dynamically manage data structures of $W_i()$ and $FB_i()$, which are used for bookkeeping bandwidth workloads and free bandwidth caused by earlier reading of video's data requests. The requests to be served are put to request set T_{req} and the set save a mixture of video data and non-video data. If some requests are served using more-than-reservation bandwidth, then the values of $FB_i()$ are properly adjusted.

```
Algorithm Calc_Free_Bandwidth
1.  Let $S_v$ be the set of admitted video streams, and let $U$ be the reservation rate of $S_v$;
2.  Let $Q_n$ be the set of non-video requests;
3.  $T_{req} \leftarrow \phi$; // initialization of a request set
4.  if there is a new stream that has been admitted in period unit j then
5.      Update $W_j(r_1, r_2, r_3, \ldots, max\_p)$; // update of workloads
6.  endif.
7.  Let the value of $r_1$ of $W_j()$ be k;
8.  Put k requests with the deadline equal to the end of period unit j into $T_{req}$;
9.  if $|T_{req}| < N$ then
10.     Get up to ( $|T_{req}| - N$) number of non-video requests from $Q_n$, and put them into $T_{req}$;
11. if $|T_{req}| < N$ then
12.     Select up to ($|T_{req}| - N + FB_j.r1$) number of video data requests according to the EDF policy, and put them into S;
13.     Put the requests in set S into $T_{req}$;
14.     Adjust $FB_j()$ by considering according to the requests in $T_{req}$;
15. endif
16. Read the requests in $T_{req}$ according to their deadline urgency;
```

Figure 2. Algorithm for calculating free bandwith gainable from the RR-EDF scheduling sheme.

IV. CONCLUSIONS

In this paper, we have proposed a way used for calculating free I/O times, while processing I/O requests according to the EDF algorithm. In nature, the EDF algorithm is not suitable for HDD storage because of seek times and rotational delay. However, the RR-EDF algorithm may be possible for processing I/O requests in SSD storage. To use the SSD storage for servicing video streams, the Rate-Reservation EDF algorithm can be employed. In the original Rate-Reservation EDF algorithm, it has no mechanism that can calculate varying slack times by considering earlier-than-reservation service of data requests from videos. Since varying free time cannot be computed in ease, there was no way for utilizing free bandwidth for storage performance. To solve this, we proposed a new way for evaluating free bandwidth that can be obtained by earlier-than-reservation reading. For this, we use two data structures that can keep records of workloads and free bandwidth over period units. From this, the proposed scheme can improve the performance of flash storage serving on-line video streams.

REFERENCES

[1] Hofri Micha, "Disk Scheduling: FCFS vs SSTF Revisited," *Communications of the ACM*, Vol. **23**, No **11**, pp. **645-653**, **1980**.

[2] Houssine Chetto and Maryline Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm," *IEEE Transactions on Software Engineering*, Vol. **15**, No. **10**, pp. **1261-1269**, **1989**.

[3] M. Seltzer, P. Chen, and J. Ousterhout, "Disk Scheduling Revisited," in *Proc. of the USENIX Winter 1990*, pp. **313-324**, **1990**.

[4] Pengliu Tan, Hai Jin, Minghu Zhang, "A Hybrid Scheduling Scheme for Hard, Soft and Non-Real-time Tasks," *In Proceedings of ISORC*, pp. **20-26**, **2006**.

[5] Kang G. Shin and Yi-Chieh Chang, "A Reservation-Based Algorithm for Scheduling Both Periodic and Aperiodic Real-Time Tasks," *IEEE Transactions on Computers*, Vol. **44**, No. **12**, pp. **1405-1419**, **1995**.

[6] Xin Li, Zhiping Jia, Li Ma, Ruihua Zhang, and Haiyang Wang, "Earliest Deadline Scheduling for Continuous Queries over Data Streams," *In Proceedings of ICESS*, pp. **57-64**, **2009**.

[7] Ray-I Chang, Wei-Kuan Shih, and Ruei-Chuan Chang, "Real-Time Disk Scheduling for Multimedia Applications with Deadline-Modification-Scan Scheme," *Real-Time Systems*, Vol.**19** , No. **2**, pp**. 149-168**, **2000**.

[8] R. K. Abbott and H. Garcia-Molina, "Scheduling I/O Requests with Deadlines: A Performance Evaluation," *In Proceedings of the Real-Time Systems Symposium*, pp. **113-125**, **1990**.

[9] E. Balafoutis, M. Paterkakis, and P. Triantallou, "Clustered Scheduling lgorithms for Mixed-Media Disk Workloads in a Multimedia Server," *Cluster Computing Journal*, Vol. **6**, No. **1**, pp. **75-86**, **2003**.

[10] Sungchae Lim, "The Dynamic Sweep Scheme using Slack Time in the Zoned Disk," *In Proceedings of 11st DASFAA*, pp. **404-419**, **2006.**

[11] Mon-Song Chen, Dilip D. Kandlur, and P. Yu, "Optimization of Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Systems," In *Proceedings of the ACM Multimedia*, **1993**.

[12] Seong-Chae Lim, "FlashEDF: An EDF-style Scheduling Scheme for Serving Real-time I/O Requests in Flash Storage," Vol. **10**, No. **3**, pp. **26-34**, **2018**.

[13] Tzu-Jung Huang, Chien-Chung Ho, Po-Chun Huang, Yuan-Hao Chang, Che-Wei Chang, and Tei-Wei Kuo, "Current-aware Scheduling for Flash Storage Devices," *In Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, **2014**.

[14] V. Indhumathi, TRP Maximization Technique based Efficient Scheduling in Grid Environment, Internatilal Journal of Scientific Research in Computer Sciences and Engineering, Vol.**6**, No. **2** , pp. **20-26**, **2018.**

[15] Vidhi Tiwari and Pratibha Adkar, Implementation of IoT in Home Automation using Android Application, Internatilal Journal of Scientific Research in Computer Sciences and Engineering, Vol.**7**, No. **2** , pp. **11-16**, **2019.**

[16] Cheng Ji, Li-Pin Chang, Chao Wu, Liang Shi, and Chun Jason Xue, "An I/O Scheduling Strategy for Embedded Flash Storage Devices With Mapping Cache," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. **37**, No. **4**, **2018**.

[17] Jaechun No and Sung-soon Park, "Exploiting the Effect of NAND Flash-Memory SSD on File System Design," *in Proc. of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. **2012**.

Authors Profile

Prof. S. C. Lim received the B.S. Degree in Computer Engineering from Seoul National University, the M.S. and Ph.D. degrees from KAIST. He is currently working for the Department of Dongduk Women's University from 2005. His research interests include high-performance indexing schemes, flash storage and handling of big data.