JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL.17, NO.1, FEBRUARY, 2017 ISSN(Print) 1598-1657
https://doi.org/10.5573/JSTS.2017.17.1.162 ISSN(Online) 2233-4866

Earliest Virtual Deadline Zero Laxity Scheduling for Improved Responsiveness of Mobile GPUs

Seongrim Choi, Suhwan Cho, Jonghyun Park, and Byeong-Gyu Nam*

***Abstract*—Earliest virtual deadline zero laxity (EVDZL) algorithm is proposed for mobile GPU schedulers for its improved responsiveness. Responsiveness of user interface (UI) is one of the key factors in evaluating smart devices because of its significant impacts on user experiences. However, conventional GPU schedulers based on completely fair scheduling (CFS) shows a poor responsiveness due to its algorithmic complexity. In this letter, we present the EVDZL scheduler based on the conventional earliest deadline zero laxity (EDZL) algorithm by accommodating the virtual laxity concept into the scheduling. Experimental results show that the EVDZL scheduler improves the response time of the Android UI by 9.6% compared with the traditional CFS scheduler.**

***Index Terms*—GPU scheduler, responsiveness, BFS, EDZL, mobile GPU, smart devices**

I. INTRODUCTION

Responsiveness is one of the most important performance factors in user interface (UI) on modern smart devices as it indicates the latency that users experience from user inputs to the display outputs on smart devices [1]. Therefore, GPU throughput plays a key role to this responsiveness but little was taken into account in scheduling GPU tasks on GPU device drivers.

Traditionally, mobile GPU drivers have relied on the completely fair scheduling (CFS) algorithm [2] that allocates GPU resource to the process with the smallest virtual runtime. However, this policy hurts the responsiveness of devices because of its algorithmic complexity to maintain its scalability to many-core systems.

Recently, the BFS scheduler [3] is proposed for mobile devices with a limited number of cores exploiting the earliest virtual deadline first (EVDF) algorithm [4]. The EVDF reduces the scheduling complexity for an improved responsiveness by introducing the virtual deadline which involves simpler calculation compared with the virtual runtime used in the CFS algorithm. However, the limited scalability of the EVDF leads to virtual deadline misses on multi-core environments which are becoming common in mobile GPUs thereby increasing its response time. The original idea of the EVDF can be found from the earliest deadline first (EDF) algorithm in the real-time domain [5]. The EDF is famous for its optimal scheduling in uniprocessor domain but has a deadline missing problem in multi-core systems, which is a similar phenomenon to the virtual deadline misses associated with the EVDF algorithm. This problem was solved by the least laxity first (LLF) and earliest deadline zero laxity (EDZL) algorithms that exploit the laxity of a task representing the spare time to the deadline [6, 7]. In this letter, we take a similar approach and propose the earliest virtual deadline zero laxity (EVDZL) algorithm for an improved responsiveness in mobile GPUs by incorporating the virtual laxity into the EVDF, thus resolving the multi-core scheduling problem of the EVDF. The EVDZL inherits the lower complexity of the EVDF by exploiting the virtual deadline and prevents the violation of virtual deadline via the virtual laxity that takes the multi-core domain into account, resulting in an improved

Manuscript received Jan. 6, 2017; accepted Feb. 7, 2017
Department of Computer Science and Engineering, Chungnam National University, 99, Daehak-ro, Yuseong-gu, Daejeon, 305-764, Korea
E-mail : bgnam@cnu.ac.kr (Corresponding Author: Byeong-Gyu Nam)

responsiveness of smart devices.

II. EVDZL SCHEDULER

Completely fair scheduling (CFS) [2] is the most widely used in GPU scheduling since it fairly allocates GPU resource to each task by exploiting a virtual runtime according to the Eq. (1). The virtual runtime considers not only the weight of each task but also the execution time of it to achieve the fairness in allocation of GPU time to each task. However, the scheduling latency increases because the Eq. (1) requires a division operation in its calculation by incorporating the task weight which is inversely proportional to task priority.

$$VR(\tau_i,t) = \frac{\omega_0}{W(\tau_i)} \times A(\tau_i,t) \quad (1)$$

where $VR(\tau_i,t)$ is the virtual runtime of a task τ_i at time t,
ω_0 is the weight of a task with priority of zero,
$W(\tau_i)$ is the weight of a task τ_i,
$A(\tau_i,t)$ is the execution time of a task τ_i at time t.

The BFS scheduler [3] based on the earliest virtual deadline first (EVDF) algorithm [4] was proposed to reduce this scheduling complexity of the CFS and thus improving the responsiveness. As shown in Eq. (2), the EVDF scheduling algorithm avoids the division of Eq. (1) by introducing the virtual deadline which is proportional to the task priority. The virtual deadline becomes a guideline to improve the timeliness of a task.

$$VD(\tau_i,t) = t + (\pi(\tau_i) \times RR_i) \quad (2)$$

where $VD(\tau_i,t)$ is the virtual deadline of a task τ_i at time t,
$\pi(\tau_i)$ is the priority of a task τ_i,
RR_i is the round robin interval.

However, the virtual deadline in this algorithm might be missed in multi-core systems (e.g. GPUs) because the EVDF algorithm tries to run a task with the earliest virtual deadline without any considerations on the time slack to the virtual deadline, thereby hurting its responsiveness. In real-time domain, a similar problem appears in the earliest deadline first (EDF) algorithm [5] that misses the deadline in multi-core environments. This problem was resolved in the least laxity first (LLF) [6] by accommodating the time slack to deadline, which is represented as the laxity, in task scheduling. The LLF algorithm schedules the tasks with the least laxity first thereby meeting the deadline, but the laxity-based scheduling incurs a huge overhead from frequent switching of tasks. Earliest deadline zero laxity (EDZL) algorithm [7] was studied to combine the strong points of the EDF and LLF algorithms. It basically schedules tasks based on their deadlines instead of the laxity and switches to the laxity-based scheduling if it finds any tasks with zero-laxity because the laxity of zero indicates the task needs an immediate start or it will miss its deadline. As a result, its task switching overhead gets lower than that of the LLF algorithm by minimizing the chances for the laxity-based scheduling.

Unfortunately, the laxity itself does not make sense in non-realtime domain as it requires the deadline to be specified in its calculation. Therefore, we introduce the *virtual laxity* as in Eq. (3) that indicates the time slack to the virtual deadline. This virtual laxity can be used to avoid the virtual deadline violations associated with the EVDF algorithm. We propose the earliest virtual deadline zero laxity (EVDZL) algorithm based on this virtual laxity as described in Algorithm 1. The proposed EVDZL algorithm basically runs on the virtual deadline but it switches to the virtual laxity if it finds any task with a virtual laxity of zero from the scheduling queue. Fig. 1 shows the behaviors of the EVDF and EVDZL algorithms in a multi-core environment. We assume the virtual deadline (*VD*) of a task τ_1, τ_2, and τ_3 are at time 2, 3, and 1, respectively, and the virtual laxity (*VL*) of the τ_1, τ_2, and τ_3 at time of 0 are 1, 0, and 0, respectively. In the EVDF algorithm of Fig. 1(a), the task τ_1 with earlier deadline is scheduled before the task τ_2 even though the task τ_1 has time slack to the deadline, so the task τ_2 misses its virtual deadline. On the other hand, the EVDZL in Fig. 1(b) allocates tasks with the virtual laxity of zero first to each core, and thus the task τ_2 is scheduled before the task τ_1 thereby meeting its virtual deadline. As described in Fig. 1, the EVDZL exploits the virtual

Algorithm 1. EVDZL

```
1: while (true) do
2:   if there is any task with zero VL in the queue then
3:     if there is any idle core in the GPU then
4:       Execute the zero VL task on the idle core
5:     else if there are any current tasks with positive VL then
6:       Preempt the current task with largest VD
7:     else
8:       Fail to schedule
9:     end if
10:  else
11:    if there is any idle core in the GPU then
12:      Execute the earliest VD task on the idle core
13:    else
14:      Preempt the current task with largest VD
15:    end if
16:  end if
17: end while
```

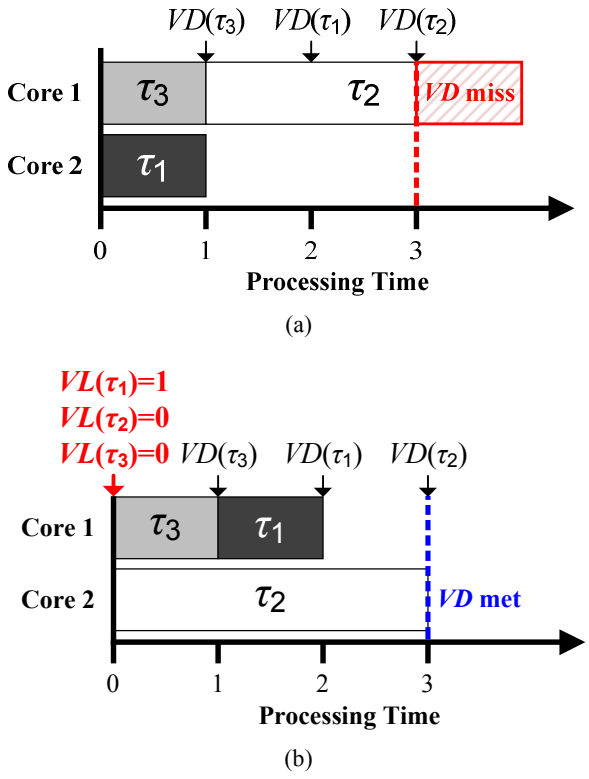

Fig. 1. Comparison of the EVDF and EVDZL algorithms in multi-core environment (a) EVDF violates virtual deadline, (b) EVDZL meets virtual deadline.

deadline for a reduced scheduling complexity and the virtual laxity to prevent the violations of virtual deadlines. As a result, the proposed EVDZL scheduler improves the responsiveness of mobile GPUs by exploiting the strong points of both the virtual deadline and virtual laxity together.

$$VL(\tau_i,t) = VD(\tau_i,t) - t - e(\tau_i,t) \quad (3)$$

where $VL(\tau_i,t)$ is the virtual laxity of a task τ_i at time t, $e(\tau_i,t)$ is the remaining execution time of a task τ_i at time t.

III. Experimental Results

We use the Samsung Exynos5422 AP with the ARM Mali-T628 GPU as a testbed for mobile GPU cores. All GPU-related experiments are conducted on the Odroid-XU3 board running the Android 4.4.2 Kitkat on Linux kernel version 3.10.9.

As we need to know the remaining execution time of a given task to evaluate its virtual laxity as in Eq. (3), we acquire it by profiling GPU tasks in advance. In order to evaluate the responsiveness of user interface (UI), we used the System UI and SurfaceFlinger in Android platform that provide built-in UI and display manager of the platform, respectively. Responsiveness of the Android UI is tested while the tasks for 3D graphics or GPGPU are running on as background workloads to show the EVDZL scheduling performance under multiple GPU tasks.

Table 1 shows that the responsiveness of Android UI under the EVDZL scheduling is improved by 9.6% and 4.9% in average for the 3D graphics and GPGPU background tasks, respectively, compared to those on the CFS scheduler.

IV. Conclusions

In this paper, a novel GPU scheduling algorithm exploiting the virtual laxity as well as the virtual deadline is proposed for mobile GPUs. The virtual laxity is proposed to avoid the violations of virtual deadlines in

Table 1. Latency comparison of the EVDZL and CFS schedulers regarding the background task categories

Algorithm	3D Graphics (ms)	GPGPU (sec)
CFS	8.58	11.28
EVDZL	7.76	10.73

multi-core systems and the virtual deadline is adopted for a reduced complexity of the scheduling algorithm. Thanks to these virtual timelines exploited together, the proposed EVDZL scheduler demonstrates a 9.6% improvement in the responsiveness of Android UI compared with the CFS algorithm.

ACKNOWLEDGMENTS

This work was supported by research fund of Chungnam National University.

REFERENCES

[1] A. Ng et al., "Design for Low-Latency Direct-Touch Input," In *Proc. of ACM UIST'12*, pp. 453-464, Oct., 2012.

[2] T. Li et al., "Efficient and Scalable Multiprocessor Fair Scheduling Using Distributed Weighted Round-Robin," in *Proc. of the ACM Symp. on Principles and Practice of Parallel Programming (PPoPP)*, pp. 65-74, Feb. 2009.

[3] T. Groves et al. (2009). "BFS vs. CFS – Scheduler Comparison," The University of New Maxico, http://cs.unm.edu/~eschulte/classes/cs587/data/bfs-v-cfs_groves-knockel-schulte.pdf (accessed Jan. 5, 2017).

[4] Y. J. Choi and H.-M. Kim, "A New Scheduling Scheme for High-Speed Packet Networks: Eaeliest Virtual Deadline First," *Comput. Commun. Elsevier J.*, Vol. 30, No. 10, pp. 2291-2300, July, 2007.

[5] 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, Jan. 1973.

[6] M. L. Dertouzos and A. K. Mok, "Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks," *IEEE Trans. Software Engineering*, Vol. 15, No. 12, pp. 1497-1506, Dec., 1989.

[7] S. K. Lee, "On-Line Multiprocessor Scheduling Algorithms for Real-Time Tasks," in *Proc. of the IEEE Region 10's Ninth Ann. Int'l Conf.*, pp. 607-611, Aug. 1994.

Seongrim Choi received the B.S. and M.S. degrees in computer science and engineering from the Chungnam National University (CNU), Daejeon, in 2011 and 2013, respectively, where he is currently working toward the Ph.D. degree. His current research interests include object recognition processor and wearable SoC. He received the IEEE Asian Solid-State Circuits Conference (A-SSCC) Distinguished Design Award in 2016.

Suhwan Cho received the B.S. degree in computer science and engineering from the Chungnam National University (CNU), Daejeon, in 2015, where he is currently working toward the M.S. degree. His current research interests include machine learning SoC and mobile GPU. He received the Distinguished Design Award in the IEEE Asian Solid-State Circuits Conference (A-SSCC) 2016.

Jonghyun Park received the B.S. and M.S. degrees in computer science and engineering from the Chungnam National University (CNU), Daejeon, in 2012 and 2015, respectively. His research interests include embedded OS design and GPU scheduler design.

Byeong-Gyu Nam received his B.S. degree (*summa cum laude*) in computer engineering from Kyung-pook National University, Daegu, Korea, in 1999, M.S. and Ph.D. degrees in electrical engineering and computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2001 and 2007, respectively. His Ph.D. work focused on low-power GPU design for wireless mobile devices. In 2001, he joined Electronics

and Telecommunications Research Institute (ETRI), Daejeon, Korea, where he was involved in a network processor design for InfiniBand™ protocol. From 2007 to 2010, he was with Samsung Electronics, Giheung, Korea, where he worked on world first low-power 1-GHz ARM Cortex™ microprocessor design. Dr. Nam is currently with Chungnam National University, Daejeon, Korea, as an associate professor. He is serving as a vice director of the System Design Innovation and Application Research Center (SDIA), KAIST and a member of steering committee of the IC Design Education Center (IDEC), KAIST. His current interests include mobile GPU, embedded microprocessor, low-power SoC design, and embedded software platforms. He co-authored the book Mobile 3D Graphics SoC: From Algorithm to Chip (Wiley, 2010) and presented tutorials on mobile processor design at IEEE ISSCC 2012 and IEEE A-SSCC 2011. He is serving as the chair of Digital Architectures and Systems (DAS) subcommittee in ISSCC and a member of the TPC for IEEE ISSCC, IEEE A-SSCC, IEEE COOL Chips, VLSI-DAT, ASP-DAC, and ISOCC. He served as a Guest Editor of the IEEE Journal of Solid-State Circuits (JSSC) and is an Associate Editor of the IEIE Journal of Semiconductor Technology and Science (JSTS).