Computers & Graphics xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Issue on CAD/Graphics 2017

Energy-efficient global illumination algorithms for mobile devices using dynamic voltage and frequency scaling

Q1 SeongKi Kim[a], Takahiro Harada[b], Young J. Kim[c,*]

[a] Sangmyung University, 20, Hongjimun 2-gil, Jongno-gu, Seoul, Republic of Korea
[b] AMD, CA, USA
Q2 [c] Ewha Womans University, 11-1, Daehyun-Dong, Seodaemun-Gu, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 12 June 2017
Revised 8 July 2017
Accepted 9 July 2017
Available online xxx

Keywords:
DVFS
Path tracing
Global illumination
OpenCL
GPGPU
Mobile device

ABSTRACT

With increases of display resolution and graphical quality demands, global illumination techniques are sought after to meet such demands. However, global illumination algorithms have a long processing time on mobile devices, such as smartphones or tablet PCs. Besides the performance issue, the algorithms consume a large amount of battery-powered energy. The performance and energy consumption have a trade-off relationship. The dynamic voltage and frequency scaling (DVFS) algorithms control the balance between the performance and the energy consumption by adjusting the GPU and CPU's frequencies. To improve the performance of the global illumination algorithm and reduce the high energy consumption of the current DVFS algorithm, we suggest new algorithms with new metrics for estimating the amount of workload for GPUs as well as their memory workload. Using our new DVFS algorithms, we increase the energy efficiency, the performance per watt, of a global illumination algorithm by 33.59% in comparison to a conventional general-purpose DVFS algorithm, without modifying the underlying algorithm core.

1. Introduction

The display resolutions of mobile devices have increased significantly over the last decade. For example, the resolution of Nexus One was WVGA (800×480) in 2010, but those of Nexus 5X and Nexus 6P were QHD (1440×2560) and FHD (1080×1920) in 2015. In addition, an increasing number of graphics-intensive software, such as video games, graphical user interfaces, and augmented reality (AR) and virtual reality (VR) applications require realistically rendered scenes on mobile devices. Global illumination could fulfill these requirements for high-quality images, and research efforts have been made also for mobile devices [1–4]. Further, global illumination techniques for rapidly generating high-quality images on mobile devices demand more research efforts [1-4]. However, real-time global illumination on mobile devices is still challenging even with the recent hardware improvements and GPGPU languages such as OpenCL that enable the use of mobile GPU for general-purpose computation besides rasterization-based graphic rendering.

The form factor and thermal problems limit the performance of mobile GPUs, as mobile devices are designed to be small for portability. Thus, it is hard to house a cooling fan inside the mobile processor. The limited battery power is another factor that hinders the high-performance of mobile GPUs because the battery is drained in a short time if it has to continuously supply power to drive the high-performing GPU. Owing to these performance limitations, it is quite challenging to execute complex rendering algorithms, such as global illumination, on mobile devices.

Besides the low-performance problem, energy consumption is another important issue to consider for global illumination on mobile devices, as battery technologies have been relatively slowly improved compared to computational and graphical processing technologies such as those for the CPUs and GPUs. For instance, if we use global illumination on off-the-shelf, mobile GPUs to generate an image from a small-sized scene consisting of only tens of triangles, the mobile device can last for less than 5 h. Due to the necessity for power-efficient rendering, power-saving surface and volume rendering techniques have been proposed recently [5,6].

GPUs use dynamic voltage and frequency scaling (DVFS) technology to minimize battery consumption. Conventional DVFS algorithms predict the GPU workloads based on the old utilizations of the GPU and its working time within a given period, and adjust the GPU's voltage and frequency after comparing the old utilization with a hand-tuned threshold value, a criterion for determining the frequency. DVFS technology successfully ensures energy-saving with minor performance decreases, as observed in many past research works [7,8]. In addition, most current mobile devices employ this technology.

* Corresponding author.
E-mail addresses: skkim9226@gmail.com (S. Kim), Takahiro.Harada@amd.com (T. Harada), kimy@ewha.ac.kr (Y.J. Kim).

http://dx.doi.org/10.1016/j.cag.2017.07.017
0097-8493/

However, the existing DVFS mechanism does not work well for global illumination algorithms. For instance, the GPU's old utilizations may not adequately represent the amount of work that the GPU needs to process currently, since the utilizations may include redundancies; for instance, the GPU waits for completing irregular memory requests including traversing a bounding volume hierarchy (BVH) for testing intersections between rays and objects.

Our paper offers the following contributions to global illumination on mobile devices. First, we propose a novel mathematical model for better estimating a GPU's workload for global illumination, which does not depend on any hardware-specific performance counters. As a result, this model can be applied to any GPUs or even to CPUs. Second, we propose novel DVFS algorithms that control the voltages and the frequencies of GPU as well as its memory with the proposed estimation model, and a new metric for the GPU workload. As a result, using our new DVFS algorithm, the total energy efficiency, that is, the performance per watt, increases by more than 33.59% for global illumination in comparison to a conventional, interval-based DVFS algorithm.

The rest of this paper is organized as follows. Section 2 describes the global illumination for mobile devices and a DVFS as related works. Section 3 describes our analysis for improving the energy-efficiency of the global illumination and their rationales. Section 4 details our DVFS algorithms. Section 5 implements the global illumination and the DVFS algorithms on mobile devices and discusses their energy consumption and the performance. Section 6 draws the conclusion.

2. Related works

In this section, we describe prior works related to our DVFS work for global illumination.

2.1. Global illumination and path tracing

Monte Carlo path tracing can be used to render an image with a photorealistic quality [9]. Path tracing is the most popular choice today owing to its robustness and efficiency, although there are studies on more sophisticated algorithms [10,11]. It solves the rendering equation using Monte Carlo integration, which is done by casting random rays.

One of the most computationally expensive operations in the path tracing algorithm is ray-object intersection. Acceleration structures such as the BVH [12] or Kd-Tree [13] are often used to make this operation faster. With the computational power of GPUs, path tracing can be executed at an interactive speed [14]. However, it is still challenging to run path tracing on mobile devices, and thus, dedicated hardware has been introduced [1-3] for mobile devices, or image quality is sacrificed [4].

In this paper, we will use a concept of the original path tracing and improve its run-time performance while minimizing its energy consumption using DVFS. Additionally, our technique can be applied to other advanced algorithms such as bidirectional path tracing or the Metropolis light transport (MLT) algorithm.

2.2. Dynamic voltage and frequency scaling (DVFS)

On mobile devices powered by battery, the relationship between the consumed energy E of CPU or GPU, and its power consumption P, working time t, voltage V, and frequency F can be captured by the following Eq. (1) [15]:

$$E = Pt = cV^2Ft. \tag{1}$$

Here, c is a constant specific to CPU or GPU. Even though the voltage, V, is not directly proportional to the frequency, F, a higher frequency requires a higher voltage for the device's stable operation. Thus, as the frequency increases, the consumed power, and energy increase quadratically, but the performance increases linearly [16]. Thus, it is important to determine the optimal frequency and voltage to process given GPU workloads while minimizing the consumed energy and maximizing the performance according to Eq. (1).

The interval-based DVFS algorithm periodically measures the GPU's working time. Utilization U_i is defined as the working time over a unit time interval. The algorithm uses an averaged utilization over the past n time periods for predicting a future one, U_{i+1}. U_{i+1} is predicted by Eq. (2) where T_i, w_i and n are a time interval, the running time on GPUs and the window size, respectively.

$$U_{i+1} = \frac{1}{n}\sum_{j=0}^{n-1}\frac{w_{i-j}}{T_{i-j}} \quad when\ i \geq n-1 \tag{2}$$

After the future utilization U_{i+1} is predicted using Eq. (2), it is compared with system-defined thresholds; if U_{i+1} is less than a down-threshold, the frequency decreases; if it is higher than an up-threshold, the frequency increases; otherwise, the current frequency is retained.

Another class of DVFS algorithms known as inter-task algorithms investigates source codes, dynamically profile the real-time hardware information, and determine the frequency [17]. The intra-task algorithms monitor a single task or a process and determine the frequency [18].

Among these different types of DVFS algorithms, most of the recent GPU kernel drivers embed interval-based DVFS algorithms with the window size 1.

3. Analysis and modeling of DVFS for path tracing

This section describes the characteristics of path tracing and their impacts upon interval-based DVFS.

3.1. Analysis of path tracing for DVFS

Path tracing typically requires four processes: ray generation, tree (Kd-tree or BVH) traversal, intersection tests, and shading. Monte Carlo sampling in path tracing makes the hit ratio of a cache low due to incoherent memory accesses. Moreover, path tracing on GPU uses a vast number of work-items (e.g. 230,400) that also increases the number of memory accesses. Moreover, the cache size in mobile GPU is small compared to desktop GPU. As a result, path tracing on mobile GPU results in low performance while consuming high energy.

Besides the memory access issue, using conventional DVFS algorithm, Fig. 1 shows an example where three DVFS time intervals are required to draw a single frame. The path tracing algorithm starts at T_1^1, but the GPU cannot finish the entire tasks and the DVFS time interval is expired, and the frequency is adjusted. At T_2^1, the GPU can finish all and becomes idle. At the end of T_2^1, the time interval is expired again, and frequency is also adjusted again. At T_3^1, the GPU becomes busy again because it should draw the shaded colors back into a display.

DVFS algorithms periodically control voltage and frequency for GPUs as well as for memory bus, based on a history of old utilizations to save battery power. However, naive interval-based DVFS might incorrectly affect performance as well as energy consumption of the path tracing since GPU is designed for rasterization-based graphics.

Please cite this article as: S. Kim et al., Energy-efficient global illumination algorithms for mobile devices using dynamic voltage and frequency scaling, Computers & Graphics (2017), http://dx.doi.org/10.1016/j.cag.2017.07.017