Energy Efficient Continuous Multimedia Processing Using the Tegra K1 Mobile SoC

Kristoffer Robin Stokke, Håkon Kvale Stensland, Carsten Griwodz, Pål Halvorsen
Simula Research Laboratory & University of Oslo, Norway
{krisrst, haakonks, griff, paalh}@ifi.uio.no

ABSTRACT

Energy consumption is an important issue for mobile devices, as the technological development in battery technology has not kept pace with the power requirements of mobile hardware. In this paper, we use a video rotation filter to study the effects of CPU and GPU frequency scaling in terms of performance and energy. Our platform is the Tegra K1 mobile processor with a quad-core CPU and a CUDA-capable GPU. We find that most energy can be saved by minimising CPU frequency while meeting the filter's framerate requirement. Interestingly, the frequency scaling affects GPUs differently, where the best frequency is always moderately higher than the minimum which meets the framerate requirement. Using these heuristics, it is possible to save up to 10 % energy compared to the standard Linux frequency scaling algorithms, which use processor utilisation to adjust processor frequency.

Categories and Subject Descriptors

C.1.3 [**Other Architecture Styles**]: Heterogeneous (hybrid) systems;
C.1.4 [**Parallel Architectures**]: Mobile processors;
J.2.0 [**Physical Sciences and Engineering**]: Electronics

Keywords

Multimedia, Tegra K1, CUDA, energy, performance, CPU-GPU frequency scaling

1. INTRODUCTION

Battery capacity is a severe limitation of modern mobile devices. Dynamic Voltage and Frequency Scaling (DVFS) [1] algorithms minimise power usage of CPUs and GPUs by lowering operating frequency, and is an effective way to save energy when full performance is not needed. There have been many proposals from the research community on how DVFS can be improved for mobile devices. However, these consider GPUs that do not support general purpose computing, such as CUDA [5, 4]. Those that consider DVFS for CUDA-capable GPUs [2] typically target high-performance applications found in data centres. We argue that common mobile applications are lighter processes where the goal is not to finish processing quickly, but to meet application-specific QoS constraints such as a specific framerate. In this paper, we consider a video processing filter implemented for both CPU and GPU, where the goal is to provide an acceptable video quality of 25 FPS. We study the impact that CPU and GPU frequency settings have on application performance and energy using a Tegra K1 mobile SoC, and find that 10 % energy can be saved by using workload specific frequency settings compared to the standard Linux DVFS algorithms.

MoVid'15, March 18-20 2015, Portland, OR, USA
Copyright 2015 ACM Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3353-5/15/03…$15.00
http://dx.doi.org/10.1145/2727040.2727044 ...$15.00.

2. SYSTEM SETUP

Our system is based on a Jetson-TK1 mobile development kit equipped with a custom power measurement sensor. The workload used in our study is an image rotation filter. The filter rotates each frame of a 25-FPS video stream by a continuously increasing angle. Its operations resemble that of a video stabiliser. The filter has been implemented for both the CPU and the GPU using CUDA. To study the effects of frequency scaling we disable the Linux DVFS algorithms and set CPU and GPU operating frequency manually while processing frames as follows:

1. While the CPU or the GPU is busy processing a frame, its respective frequency is set to a higher frequency which we vary throughout our experiments.
2. If the CPU or GPU processing ends before the frame deadline (40 ms for 25 FPS), CPU frequency is lowered to 204 MHz for the CPU and 72 MHz for the GPU. The CPU sleeps for the remaining time.

It is important to note that the performance of the filter never exceeds 25 FPS, but that it can be lower than this if the manually set processing frequencies are too low.

3. EXPERIMENTAL RESULTS

We run our experiments using our setup with three input resolutions (see Table 1). Each test is run ten times for each frequency, stopping if the encoding time is longer than 12 s. This is to equalise the energy consumption of idle

Resolution	Userspace P-SAV			Linux Governor (“ondemand”)		Improvement
	Core	Max Frequency	Energy	Core	Energy	
352x288	CPU	204 MHz (CPU)	9.22 mWh	CPU	9.02 mWh	-2.2 %
640x480	CPU	304 MHz (CPU)	11.33 mWh	CPU	12.86 mWh	11.9 %
1920x1080	GPU	804 MHz (GPU)	26.00 mWh	GPU	29.16 mWh	10.8 %

Table 1: The most energy efficient frequency configurations compared with the best Linux DVFS algorithm.

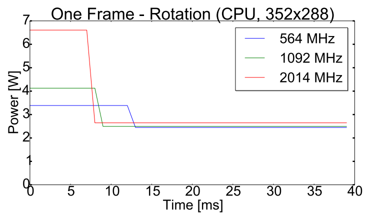

Figure 1: A single CPU frame encoding snapshot for different operating frequencies.

components from each run. The framerate and total energy usage for CPU- and GPU-execution can be seen in Figure 2. The best frequencies are marked with red. As expected, we see that higher frequencies increase the framerate. For the CPU, the best frequency is very close to the point where the 25 FPS requirement is reached. After this, increasing the operating frequency further only increases the total energy usage on the CPU. A likely reason for this observation is that a doubling in frequency effectively doubles the power usage of the processor [1], but does not reduce the frame encoding time by a corresponding amount, which can be seen in Figure 1.

The GPU experiments show a different trend than that of the CPU. For the mid-resolution video, 25 FPS is reached at 72 MHz. The best frequency is four frequency steps above, at 324 MHz, reducing the energy usage by 23 %. In other words, the frequency should not be minimised as for the CPU. The effect is similar, but not as clear, for the high- and low-resolution videos. However, we have run other types of video filters which confirm this observation. Unfortunately, we could not include them here due to space restrictions.

We also run the experiments with the standard Linux DVFS algorithms [3], where the CPU and GPU operating frequencies are automatically adjusted in response to changes in processor utilisation. There is only one GPU DVFS algorithm, but of the four CPU algorithms, the “ondemand” algorithm was consistently better. Compared to these, up to 10 % energy can be saved by using workload-specific frequencies (see Table 1).

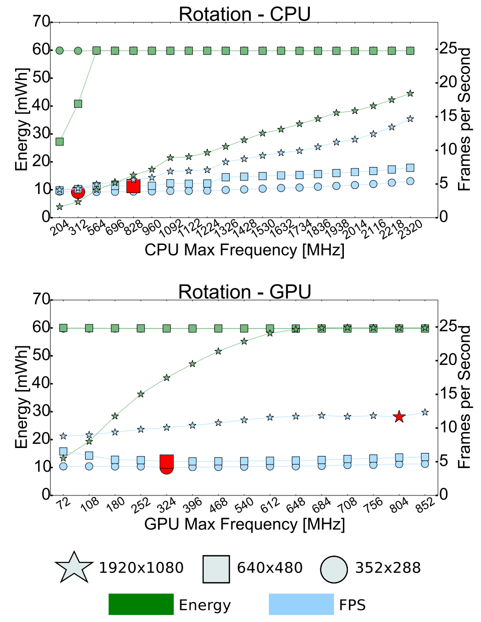

Figure 2: Frequency scaling experiments. The frequencies shown correspond to the one used while the CPU or GPU is actively processing a frame.

4. CONCLUSION

In this paper, we study the impact of CPU and GPU frequency settings on a mobile processor. Our workload is a video rotation filter implemented for both the CPU and the GPU using CUDA. We find that up to 10 % energy can be saved by minimising CPU frequency such that a framerate of 25 FPS is met generally saves energy over the standard Linux DVFS algorithms. It is clear that the standard Linux DVFS governors, which change processor frequency in response to changes in utilisation [3], can be improved if QoS requirements such as framerate were considered. The effect of DVFS is different for the GPU, where the best frequency tends to lie moderately above the minimum which achieves 25 FPS. As this is work-in-progress, we do not yet know the exact reasons behind this observation. For future work, it would be interesting to see if existing GPU DVFS algorithm proposals [4, 5] can be improved for our platform.

5. REFERENCES

[1] A. Castagnetti, C. Belleudy, S. Bilavarn, and M. Auguin. Power consumption modeling for DVFS exploitation. In *Proc. of Euromicro DSD-AMT*, pages 579–586, 2010.

[2] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong. Effects of dynamic voltage and frequency scaling on a k20 gpu. In *Proc. of ICPP*, pages 826–833, 2013.

[3] V. Pallipadi and A. Starikovskiy. The ondemand governor. In *Proc. of the Linux Symposium*, volume 2, pages 215–230, 2006.

[4] A. Pathania, Q. Jiao, A. Prakash, and T. Mitra. Integrated cpu-gpu power management for 3d mobile games. In *Proc. of the 51st Annual Design Automation Conference*, pages 1–6, 2014.

[5] D. You and K. Chung. Quality of service-aware dynamic voltage and frequency scaling for embedded gpus. *IEEE Computer Architecture Letters*, 2013.