Mitigating Interactive Performance Degradation from Mobile Device Thermal Throttling

James R. B. Bantock, *Student Member, IEEE*, Bashir M. Al-Hashimi, *Fellow, IEEE*, Geoff V. Merrett, *Senior Member, IEEE*

***Abstract*—Mobile devices are limited in mass and volume reducing the viability of active device cooling implementations, this requires the use of less effective passive techniques to maintain device skin temperature levels. Application performance demands on a modern mobile device are driven by sustained performance workloads, such as 3D games, Virtual and Augmented Reality. Mobile System-on-Chips have corresponding increases in performance through both architectural changes and frequency of operation increases; which has resulted in the peak power consumption exceeding the sustainable thermal envelope defined by device skin temperature requirements. Existing thermal throttling techniques mitigate this by capping the frequency of operation of the System-on-Chip. Through experimentation with a modern smartphone platform using sequences from real-world applications, we demonstrate in this paper that Frequency Capping can have a significant effect on the performance of interactive applications, increasing the number of frame rate defects by up to 146%. We propose Task Utilization Scaling, a new lever for thermal throttling, which scales performance for critical interactive periods by the same factor as non-critical periods. Experiments demonstrate that the proposed approach can result in a decrease in frame rate defects of up to 18% compared with Frequency Capping or a skin temperature reduction of up to 2°C.**

***Index Terms*—Thermal management, mobile devices, frame rate janks, interactive performance**

I. Introduction

More than a decade on from the advent of the smartphone market, the industrial design of mobile devices is maturing. Irrespective of whether the device in question is a smartphone, tablet or notebook, the dominant form-factor has trended towards a thin and light product. The limitations imposed by this form-factor mean that the active cooling techniques employed by desktop, server and some embedded systems are often not applicable to mobile devices. Passive cooling techniques, e.g. heat spreaders and pipes [1], [2], have so far failed to replicate the dissipation characteristics of their active counterparts. The dissipation characteristics define the thermal envelope which can be sustained without the device skin temperature exceeding a level that will cause discomfort or injury to the user; this temperature level is around 45°C [3]–[5]. A typical sustainable thermal envelope for a smartphone is approximately 3.5W [6], although environmental conditions, including how the device is held by the user, can affect this number [4], [7].

In a competitive mobile device market, manufacturers and developers have sought to drive discretionary purchases by supporting applications beyond the performance constraints of older devices. The predominant classes of these applications include 3D video games, Virtual Reality (VR) and Augmented Reality (AR). A common characteristic of these application classes is that they require sustained high System-on-Chip (SoC) performance over the period of execution [8]–[10]. Average power measured during this period of sustained performance can exceed 7W for a modern smartphone SoC [11], which exceeds the aforementioned sustainable thermal envelope of 3.5W. SoC thermal throttling is therefore unavoidable, and intuitively a degradation in average frame rate performance will result [12].

Traditional thermal throttling techniques involve capping the maximum frequency at which an SoC may operate. The relationship between voltage, frequency and dynamic power dictates that higher frequencies are often less power-efficient than lower frequencies [13], [14]. The frequency at which to cap is determined by closed-loop control, in some cases based only on device temperature sensors [15], including standard industry solutions such as the Qualcomm Thermal Engine [16]. In other cases, application performance and user Quality of Experience (QoE) metrics are also monitored [17], [18]; these implementations are effective in thermal throttling whilst achieving stable performance, most commonly by monitoring the average frame rate rendered for an application and adjusting the frequency cap correspondingly. However, there is no consideration of workload interactivity as this is not captured by the average frame rate metric.

In this paper we investigate device thermal throttling using a modern mobile device with three key contributions:

1) We show that Frequency Capping can increase dropped frames by up to 146% in interactive applications
2) A new Dynamic Power Management lever is proposed, Task Utilization Scaling, a drop-in replacement for Frequency Capping in existing thermal throttling systems
3) Task Utilization Scaling is demonstrated to reduce dropped frames by up to 18% or to reduce mobile device skin temperature by up to 2°C at the same performance compared with Frequency Capping

This work was supported in part by the EPSRC under the PRiME programme grant EP/K034448/1 (http://www.prime-project.org)

The authors are with the School of Electronics and Computer Science, University of Southampton, UK. (e-mail: jrbb1g11@ecs.soton.ac.uk, gvm@ecs.soton.ac.uk)

All data openly available at https://doi.org/10.5258/SOTON/D1351.

© 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

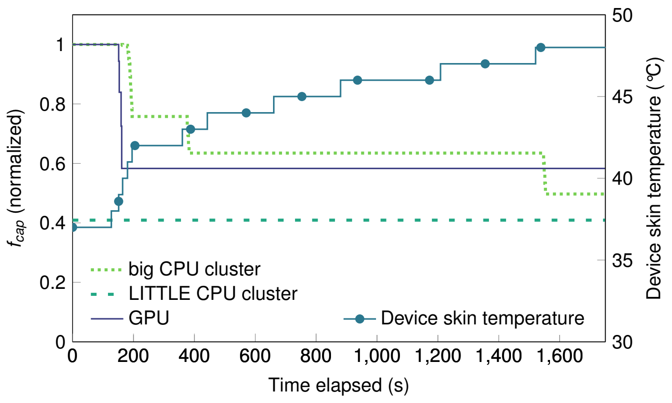


Fig. 1. An execution sequence of 3D video game Player Unknown Battlegrounds demonstrating the Qualcomm Thermal Engine's rules in action; throttling takes effect after 151 seconds before progressively capping CPU and GPU frequencies (f_{cap}).

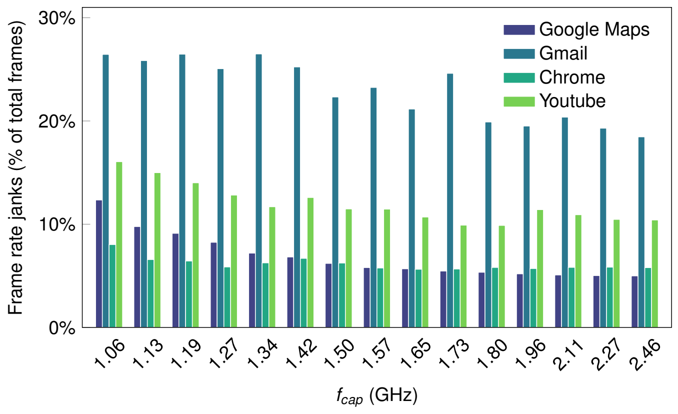
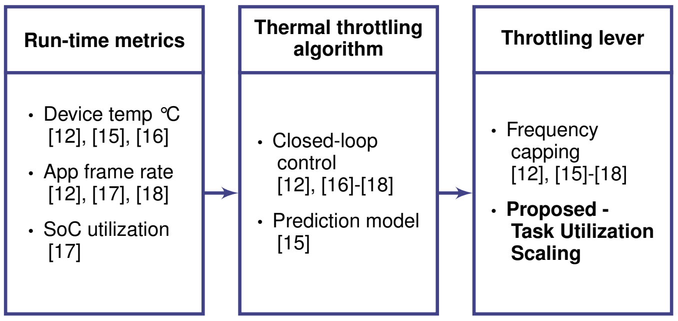


Fig. 2. The effect of Frequency Capping on frame rate janks (defects [19]). Four application sequences are used, the degradation varies between application, Google Maps exhibits the largest relative degradation with a 146% increase in janks after the frequency cap, f_{cap}, is reduced to 1.056Ghz.



Run-time metrics
• Device temp ℃ [12], [15], [16]
• App frame rate [12], [17], [18]
• SoC utilization [17]
Thermal throttling algorithm
• Closed-loop control [12], [16]-[18]
• Prediction model [15]
Throttling lever
• Frequency capping [12], [15]-[18]
• Proposed - Task Utilization Scaling

Fig. 3. Operation of thermal throttling systems, showing how Task Utilization Scaling, the proposed lever, can replace the Frequency Capping lever in existing approaches.

II. Motivation

In this section, we motivate investigation of a new thermal throttling lever by demonstrating the effect of frequency capping on frame rate defects (otherwise known as janks [19]) in interactive applications. Previously, only the average frame rate has been considered [12]. Experimentation was carried out using the Google Pixel 2 smartphone, running the Android 9.0 Pie operating system along with the Energy Aware Scheduling patchset. Thermal throttling is implemented using the Qualcomm Thermal Engine, a programmable cross-platform solution. The input to the controller is a configuration file specifying frequency caps that may be implemented by a PID controller according to temperature sensor readings. The Google Pixel 2 implementation estimates skin temperature using a designated thermistor. At 40°C, throttling is initiated; by 47°C, the big CPU cluster is capped below half its maximum frequency; and at 56°C a device shutdown is enforced. In addition, the GPU frequency is capped dependent on the temperature difference measured between the GPU subsystem and the overall SoC package.

In Figure 1, a 3D video game[1] is executed on the Google Pixel 2 to demonstrate thermal throttling in action. After 151 seconds, throttling is initiated before progressively increasing as the phone temperature rises to 48°C, which exceeds the discomfort level of 45°C [3]–[5]. The big CPU cluster frequency cap, f_{cap}, is reduced to 1.19GHz from an initial 2.46GHz.

Technical constraints limit performance analysis of individual frames for third-party interactive Android NDK applications such as 3D game engines in a production Android build. Instead, after disabling the Qualcomm Thermal Engine, interactive Android SDK applications were executed under set frequency caps analogous to those used by the Qualcomm Thermal Engine to analyze the effect on performance. Experimental sequences were implemented for four Android SDK applications: Google Maps, Gmail, YouTube, and Chrome. Each application was executed using the default input sequences from Workload Automation [20], Figure 2 shows the results; the relative and absolute degradations vary between the applications, with Google Maps exhibiting the highest relative degradation. The variation in frame rate janks for Google Maps from 5.0% of frames to 12.2% represents a 146% increase as the frequency cap, f_{cap}, is reduced.

[1]Player Unknown Battlegrounds, maximum graphics, resolution and FPS

III. Task Utilization Scaling

A characteristic common to all of the interactive workloads used in the previous experiment is that high performance is required for short sporadic periods of time. This may be a response to user input or the conclusion of a delay waiting for external resources such as network access [21]. For the remainder of time, the performance required is lower and unlikely to be at a level that would be affected by a frequency cap. In effect, when an interactive application is executed under a frequency cap, the brief critical events are throttled and the sustained non-critical periods (where throttling would be less noticeable) are not throttled.

An approach to thermal throttling that considers frame rate janks must throttle the system during non-critical periods, as well as critical periods. We propose a new technique called Task Utilization Scaling (TUS), to replace Frequency Capping (FC) in existing thermal throttling systems. Task Utilization Scaling does not replace the existing state-of-the-art thermal management systems entirely [13], [16]–[18], instead it would

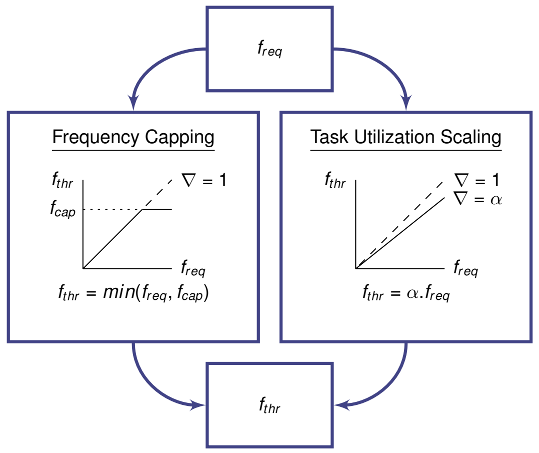


Fig. 4. Effect of Frequency Capping and Task Utilization Scaling levers on the maximum single core frequency available after throttling, f_{thr}, selected by the Energy Aware Scheduler. By throttling all periods of execution equally using a scaling factor, α, Task Utilization Scaling permits a higher maximum f_{thr} than the equivalent frequency cap, f_{cap}, allows.

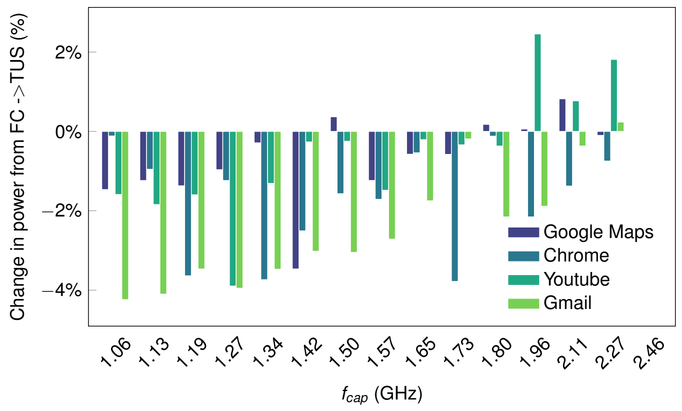


Fig. 5. Change in CPU power consumption from swapping to Task Utilization Scaling from Frequency Capping. α was selected such that for each frequency cap, f_{cap}, the value of α allowed the same maximum frequency, f_{thr}. Frame rate jank performance was unchanged due to the identical f_{thr}.

serve as a drop-in replacement for FC within these systems as shown in Figure 3.

When an application is executed by an operating system such as Android, the utilization of its one or more tasks is monitored. This task utilization information is used to map the tasks to appropriate CPU cores and by the Dynamic Voltage and Frequency (DVFS) governor to select a frequency for the chosen cores. If the task utilizations on a single CPU core are reported in the nominal fashion, the DVFS governor will receive an overall utilization number which, when combined with an extra performance margin will allow a frequency to be chosen that does not throttle any of the tasks.

We propose that the reported task utilization that the DVFS governor receives is intercepted and is multiplied by a scaling factor, α, as a replacement for FC when thermal throttling is necessary. The effect of applying this scaling factor is that the DVFS governor is deceived into believing that the system utilization is lower than the actual value. If the scaled task utilization is below the threshold that triggers a frequency change, then the DVFS governor will select a lower frequency which reduces the thermal load on the device. As α is reduced, the reported task utilization will reduce further leading to further frequency reductions. The TUS scaling factor, α, directly replaces the frequency cap used by FC when used as a drop-in replacement in existing thermal throttling systems. In most existing systems, solely the CPU frequencies are capped; however, in other systems simultaneous capping of CPU and GPU frequencies takes place for which α_{CPU} and α_{GPU} would be varied by the same closed-loop control system instead of the corresponding CPU and GPU frequency caps.

The distinction between TUS and FC is illustrated in Figure 4. The FC response is identical to the frequency required, f_{req}, until a hard performance limit, beyond which only the frequency cap, f_{cap}, is selected. The TUS response is the required frequency but multiplied by a scaling factor, α. In both task utilization scaling and FC, the maximum single core CPU frequency, f_{thr}, will be decreased as the level of throttling is increased; the rate of decrease is, however, slower for TUS. TUS applies a level of throttling which is constant irrespective of the performance required. The result of this is that the system is throttled at all times, not simply when high performance is required during critical periods of execution.

Under normal mobile device usage, device skin temperature throttling reduces the performance of the System-on-Chip before its components reach critical temperatures that would require System-on-Chip thermal throttling measures. However, as a last resort, these measures such as per-CPU frequency capping or shut-off should be retained.

IV. Validation of Proposed Approach

Experimental validation of Task Utilization Scaling (TUS) was carried out for both frame rate janks of interactive Android SDK applications and average frame rate throttling due to thermal constraints of a 3D game, the results were compared against the baseline approach of Frequency Capping (FC).

A. *Interactive Applications*

Findings from Section II establish that the effects caused by thermal throttling vary between applications and are also dependent on the level of throttling implemented. To validate the proposed approach, the same experiment from Figure 2 is repeated for TUS. The values of α were selected such that for each frequency cap, f_{cap}, the corresponding α value permitted the identical maximum frequency, f_{thr}. For example, when f_{cap} was set to 1.80GHz, the value of α compared was 0.73. CPU power consumption was modeled using traced data determining the time spent at each CPU frequency. The results in Figure 5 show that TUS achieves a relative CPU power consumption reduction of up to 4% compared to the FC with the same frame rate jank performance. This reduction is caused by the throttling of non-critical periods of execution implemented by TUS but not FC. If performance is desired, the power saving can be reinvested to increase α, and therefore f_{thr}, when using TUS. The geometric mean improvement in frame rate janks across all four applications varies between

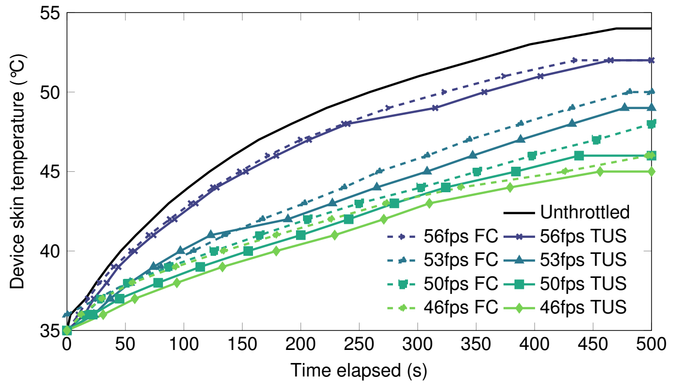


Fig. 6. Device skin temperature measured over time for Frequency Capping (FC) and Task Utilization Scaling (TUS) techniques at {46, 50, 53, 56} frames-per-second performance in 3D video game Player Unknown Battlegrounds. The f_{cap} values used were {1.34, 1.65, 1.96, 2.27} GHz and the α values used were {0.54, 0.67, 0.80, 0.92}. Task Utilization Scaling maintains equivalent performance with a skin temperature of 1-2°C lower than Frequency Capping.

0% when f_{cap} is 2.46GHz to 8.5% at 1.06GHz. When the system is unthrottled, TUS and FC are identical as the TUS, α, is 1.0. As the throttling is increased, the f_{thr} reduction is smaller in TUS and the frame rate janks performance reflects this. Energy Aware Scheduling implemented on the Google Pixel 2 prevents switching of foreground tasks from big CPU cores to LITTLE cores, as such there is negligible variation in task mapping between TUS and FC.

B. Thermal Throttling

Results from the interactive applications experiment show that improved interactive performance may be achieved by replacing FC with TUS by reinvesting power savings achieved at the same performance. However, in some scenarios it is more desirable to not reinvest the power savings in additional performance and instead benefit from reduced device skin temperature. To validate this, an execution sequence taken from the 3D video game Player Unknown Battlegrounds is repeated across four throttled performance points (46, 50, 53 and 56 frames-per-second) for both approaches with the skin temperature measured, the results from this are shown in Figure 6. For all four performance points, TUS operates at the same performance with a lower skin temperature of between 1-2°C. In-line with results from the previous experiment, as throttling increases the skin temperature difference increases.

V. Conclusions

This paper has investigated the negative effects of Frequency Capping, used in existing thermal throttling systems, on interactive applications, including an increase in frame rate janks by up to 146%. Furthermore, the inability of Frequency Capping to throttle time periods with low performance requirements in an application execution sequence has been highlighted. To mitigate the negative effects of Frequency Capping, we propose an alternative lever of Task Utilization Scaling, which throttles all time periods of an application execution sequence by multiplying measured task utilizations by a scaling factor. In existing thermal throttling solutions, this approach can be a drop-in replacement for Frequency Capping. Experimental validation demonstrates that Task Utilization Scaling allows a higher maximum frequency within the same SoC sustainable thermal envelope. Results indicate the proposed approach of Task Utilization Scaling can achieve a reduction in frame rate janks of up to 18%. In addition, the efficacy of Task Utilization Scaling in thermal throttling is demonstrated for a demanding 3D video game with equivalent average frame rate to Frequency Capping with a skin temperature reduction of up to 2°C. These results highlight the potential of Task Utilization Scaling and warrant further investigation across different hardware and applications.

References

[1] V. Chiriac, S. Molloy, J. Anderson and K. Goodson, "A figure of merit for mobile device thermal management," *ITherm*, 2016, pp. 1393–1397.
[2] G. Wagner and W. Maltz, "On the Thermal Management Challenges in Next Generation Handheld Devices," *ASME InterPACK*, 2013.
[3] M. Berhe, "Ergonomic Temperature Limits for Handheld Electronic Devices," *ASME InterPACK*, 2007, pp. 1041–1047.
[4] S. Kang, H. Choi, S. Park, C. Park, J. Lee, U. Lee and S. Lee, "Fire in Your Hands: Understanding Thermal Behavior of Smartphones," *MobiCom*, 2019.
[5] Q. Xie, M. Dousti and M. Pedram, "Therminator: A Thermal Simulator for Smartphones Producing Accurate Chip and Skin Temperature Maps," *ISLPED*, 2014, pp. 117–122.
[6] M. Halpern, Y. Zhu and V. Reddi, "Mobile CPUs Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction," *HPCA*, 2016, pp. 64–76.
[7] F. Paterna and T. Rosing, "Modeling and Mitigation of Extra-SoC Thermal Coupling Effects and Heat Transfer Variations in Mobile Devices," *ICCAD*, 2015, pp. 831–838.
[8] A. Pathania, A. Irimiea, A. Prakash and T. Mitra, "Power-Performance Modelling of Mobile Gaming Workloads on Heterogeneous MPSoCs," *DAC*, 2015.
[9] Y. Leng, C. Chen, Q. Sun, J. Huang and Y. Zhu, "Energy-Efficient Video Processing for Virtual Reality," *ISCA*, 2019, pp. 91–103.
[10] H. Chen, Y. Dai, H. Meng, Y. Chen and T. Li, "Understanding the Characteristics of Mobile Augmented Reality Applications," *ISPASS*, 2018, pp. 128–138.
[11] A. Frumusanu, "The Snapdragon 855 Performance Preview," 2019. [Online]. Available: https://www.anandtech.com/show/13786/snapdragon-855-performance-preview/5
[12] O. Sahin, L. Thiele and A. Coskun, "On the Impacts of Greedy Thermal Management in Mobile Devices," *IEEE Embedded Syst. Letts.*, vol. 7, no. 2, pp. 55–58, 2015.
[13] J. Kong, S. Chung and K. Skadron, "Recent Thermal Management Techniques for Microprocessors," *ACM Computing Surveys*, vol. 44, no. 3, pp. 13:1–13:42, 2012.
[14] ARM, "Energy Aware Scheduling (EAS)," 2019. [Online]. Available: https://developer.arm.com/open-source/energy-aware-scheduling
[15] G. Singla, G. Kaur, A. Unver and U. Ogras, "Predictive Dynamic Thermal and Power Management for Heterogeneous Mobile Platforms," *DATE*, 2015, pp. 960–965.
[16] Qualcomm, "Thermal Debugging Guide," 2016. [Online]. Available: https://developer.qualcomm.com/download/db410c/thermal-debugging-guide-dragonboard-410c.pdf
[17] J. Park, N. Dutt, H. Kim and S. Lim, "HiCAP: Hierarchical FSM-based Dynamic Integrated CPU-GPU Frequency Capping Governor for Energy-Efficient Mobile Gaming," *ISLPED*, 2016, pp. 218–223.
[18] A. Prakash, H. Amrouch, M. Shafique, T. Mitra and J. Henkel, "Improving Mobile Gaming Performance through Cooperative CPU-GPU Thermal Management," *DAC*, 2016.
[19] V.Reddi, H. Yoon and A. Knies, "Two Billion Devices and Counting," *IEEE MICRO*, vol. 38, no. 1, pp. 6-21, 2018.
[20] ARM, "Workload Automation (WA)," [Online]. Available: https://github.com/ARM-software/workload-automation
[21] J. Bantock, V. Tenentes, B. Al-Hashimi and G. Merrett, "Online Tuning of Dynamic Power Management for Efficient Execution of Interactive Workloads," *ISLPED*, 2017.