App-Oriented Thermal Management of Mobile Devices

Jihoon Park, Seokjun Lee, Hojung Cha
Deptartment of Computer Science
Yonsei University, Korea
{jihoonpark, seokjunlee, hjcha}@yonsei.ac.kr

ABSTRACT

The thermal issue for mobile devices becomes critical as the devices' performance increases to handle complicated applications. Conventional thermal management limits the performance of the entire device, degrading the quality of both foreground and background applications. This is not desirable because the quality of the foreground application, i.e., the frames per second (FPS), is directly affected, whereas users are generally not aware of the performance of background applications. In this paper, we propose an app-oriented thermal management scheme that specifically restricts *background applications* to preserve the FPS of foreground applications. For efficient thermal management, we developed a model that predicts the heat contribution of individual applications based on hardware utilization. The proposed system gradually limits system resources for each background application according to its heat contribution. The scheme was implemented on a Galaxy S8+ smartphone, and its usefulness was validated with a thorough evaluation.

CCS Concepts

•Computer systems organization → Embedded Software
•Computer systems organization → Reliability

Keywords

Thermal management, mobile devices, user experience

1. INTRODUCTION

The hardware performance of mobile devices has been continuously improving to handle complex and complicated applications. However, as applications increasingly require more computing power, newer devices generate a large amount of heat, which raises the critical issue of thermal problem. Users often complain about thermal problems [1], and accidents caused by a device's high temperature have been reported [2].

A conventional method for handling the thermal issue of mobile devices is to control the performance of the entire device by adjusting the CPU frequency. Unfortunately, that scheme degrades the performance of all applications running on the mobile device, that is, both foreground applications and background applications. This is particularly problematic for foreground applications because their quality, typically represented as frames per second (FPS), is degraded, and consequently, the user experience is harmed. In contrast, the degraded performance of background applications does not affect the user experience as users are generally not aware of the performance of background applications. This gives us an opportunity to solve the thermal issue of mobile devices by handling *background applications*. In fact, our preliminary experiments revealed that the power consumed by background processes is generally non-trivial, compared to that consumed by foreground applications, and background applications contribute substantially to a device's temperature. This means that the device's temperature can be lowered by restricting background applications while preserving the FPS of foreground applications.

Based on our observations, we propose an app-oriented thermal management scheme that throttles background applications individually according to their heat contribution. Instead of restricting the overall performance of the device to lower the temperature, the scheme aggressively limits the performance of heat-generating background apps. To this end, the proposed system estimates the heat contribution of each background application based on its hardware utilization and gradually adjusts the system resources for individual applications according to the device's temperature. To guarantee normal operation of the Android system, we do not throttle critical background processes that play an important role in the Android system, such as Zygote or the system server. This way, the FPS of the foreground applications does not degrade, maintaining the user experience.

Several challenges should be addressed to realize the proposed system. First, we need to find applications that contribute significantly to the device's temperature. To this end, the heat characteristics of each application should be predicted accurately. We developed a heat prediction model that estimates the heat contribution of an application based on its hardware utilization. Second, the hardware utilization should be collected in run-time in order to realize the heat prediction model. We developed an App Analyzer application that monitors the behavior of applications based on Android BatteryStats [3]. Third, the performance of each application should be controlled individually. We used the soft partitioning method commonly employed to manage the operations of heterogeneous platforms running on a server. By using the method, we restricted the resource usage of exothermic background applications dynamically depending on temperature.

The contributions of this work are as follows. First, we experimentally showed that background applications affect the device's temperature. Second, we developed a heat prediction model that estimates the heat contribution of each application. Third, we proposed an app-oriented thermal management scheme for managing the device's temperature without degrading the FPS. To the best of our knowledge, the proposed scheme is the first approach to guarantee the FPS of mobile applications while controlling the underlying thermal issue of mobile devices.

2. RELATED WORK

Several attempts have been made to manage the device

ISLPED '18, July 23–25, 2018, Seattle, WA, USA

ACM ISBN 978-1-4503-5704-3/18/07…$15.00
https://doi.org/10.1145/3218603.3218622

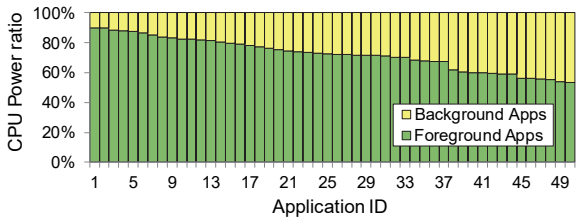


Figure 1. CPU power ratio of foreground and background applications.

Figure 2. Experimental setup.

temperature of mobile devices. Efficient heat management requires accurate heat predictions for the target devices. Previous studies simply predicted the temperature of the Application Processor (AP) with a combination of the AP core, Graphic Processing Unit (GPU), and other components of the AP [4]. More complicated methods tried to predict the internal temperature of mobile devices by calculating the heat transfer between the AP and the external components outside the AP [5-7]. Xie et al. [5] examined the thermal relationship between the battery and the AP, and Paterna et al. [6] investigated the thermal coupling between the AP and the WiFi chip. Ishii and Nakashima [7] modeled the thermal relationship between the inside of a smartphone and the surface using a sophisticated resistance-capacitance (RC) model. However, none of these works estimated the amount of heat generated by user applications.

Together with efforts to predict device temperature, previous work tried to control the heat generated in mobile devices. Thermal management schemes [4] were proposed based on the dynamic voltage and frequency scaling (DVFS) of the CPU/GPU, which are the most exothermic components inside the AP. Paterna et al. [6] controlled the performance of the AP and the WiFi chip concurrently. These works used a variation of DVFS that induced user experience reduction. Compared to previous work, the proposed system limits only the background applications that do not interact with the user and thus, prevents user-perceived QoS degradation of running applications.

3. PRELIMINARY EXPERIMENTS

We conducted an experiment to see how many applications and processes run in the actual user environment. We installed the top 50 apps in Google Play and ran them one by one for ten minutes each on a Galaxy S8+ smartphone. Then, we opened the ADB shell and checked the processes currently running. The result confirmed that 7 foreground processes, 58 background processes, and 119 system background processes were running, on average. In addition, based on previous work [8], we approximately estimated the CPU computing power as follows:

$$Power = \sum_f tick_f \times f^3 , \quad (1)$$

where f and $tick_f$ are the CPU frequency and tick consumed at frequency f, respectively. As shown in Figure 1, we found that the

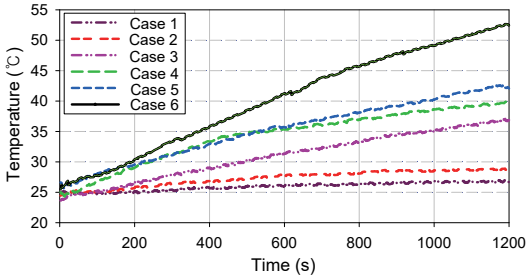
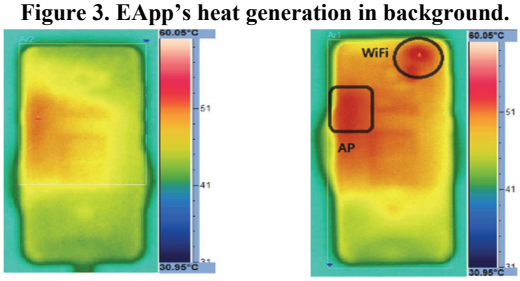


Figure 3. EApp's heat generation in background.



(a) Case 2: Asphalt **(b) Case 4: Heros-evolved**

Figure 4. Thermal images while an application is running in the background.

CPU power consumed by all background applications was about 55% of the power used by foreground applications. The result indicates the possibility of managing a device's temperature by throttling background processes specifically.

To observe the heat generated by background applications, we conducted experiments with a number of applications. We used four different Galaxy S8+ [10] smartphones released in the United States, China, the United Kingdom (UK), and South Korea. The S8+ in the United States and China is equipped with the Qualcomm Snapdragon 835 chip [10] which has the big.LITTLE architecture-based Octa-core (4 × 2.35 GHz Kryo and 4 × 1.9 GHz Kryo) CPU and the Adreno 540 GPU. The models released in South Korea and Europe use the Samsung Exynos 8995 chip [11] which has the big.LITTLE architecture-based Octa-core (4 × 2.31 GHz M 2 and 4 × 1.69 GHz A 53) CPU and Mali G71 GPU. All models run Android 7.1 (Nougat). To measure the device's temperature, we used the Cox CX1000 thermal imaging camera [12]. Figure 2 shows the experiment setup for measuring the temperature of the target devices.

Following public information about the heat generation issue of mobile devices, we installed two problematic apps, Heroes-evolved, and Weibo, which generate heat in the background, and a non-problematic application, the Asphalt game. We then conducted experiments on the following four cases. Case 1: To measure the temperature baseline, the home launcher was initiated on the idle screen, and the temperature was measured. Cases 2 through 4: To analyze the impact of background applications on the device's temperature, we placed each application (Asphalt, Weibo, and Heroes-evolved) in the background while running home launcher application at foreground, and we measured the device's temperature. All experiments were conducted for 20 minutes. During the experiment, the default thermal management policy of the Galaxy S8+ was disabled.

Figure 3 compares the average temperature changes in Cases 1–4. Temperature in Case 2 is similar to that in Case 1 (the baseline), indicating that Weibo and Heroes-evolved generate a large amount of heat in the background whereas Asphalt does not. For a more detailed analysis, we checked the thermal image of a

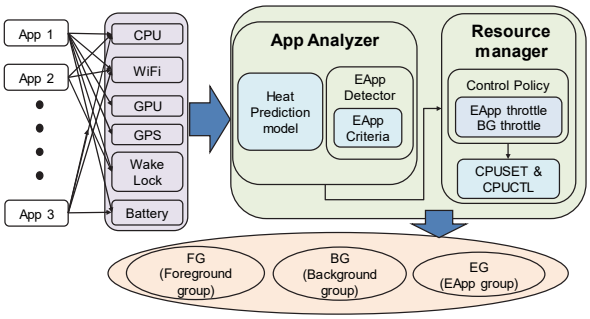


Figure 5. App-oriented thermal management.

smartphone taken with the camera (Figure 4). Figure 4(a) and 4(b) show Case 2 and Case 4, respectively. In Figure 4(a), the overall temperature appears low because Asphalt does not generate much heat in the background. However, in Figure 4(b), the temperature for the AP and WiFi is high because the heat-generating Heroes-evolved app is running in the background. In this experiment, we confirmed the previously reported background thermal problems.

In addition to the four cases, we further conducted experiments to analyze the synergetic effect of exothermic background applications when they run with foreground applications. In Case 5, the temperature was measured while Asphalt was running at foreground. In Case 6, we measured the temperature while Asphalt ran in the foreground and Heroes-evolved was left running in the background. As shown in Figure 3, the temperature increased considerably due to the problematic background application. Based on this observation, we are motivated that the temperature can be managed without FPS loss by restricting the behavior of exothermic background applications, such as Weibo and Heroes-evolved in Case 3, Case 4, and Case 6.

4.APP-ORIENTED THERMAL MANAGEMENT

4.1 System Overview

The key principle of the proposed system is lowering the device's temperature by restricting the performance of the background applications gradually according to the heat contribution. To this end, we developed a thermal prediction model for estimating the heat generated by each application based on its hardware utilization. We then implemented App Analyzer, an Android application, which collects the hardware utilization of each application to predict the heat generated in run-time using the model. When the smartphone's temperature measured by the internal thermal sensor exceeds the threshold, the exothermic background application, called *EApp* hereafter, is determined by the EApp detector based on the App Analyzer's predicted heat value and predefined EApp criteria. Depending on the presence of EApp, the Resource Manager applies the control policy appropriately. Finally, the proposed system adjusts the system resources for each application according to the control policy. Figure 5 illustrates the overview of the proposed system.

4.2 Thermal Prediction Model

Recently, researchers [13] have shown experimentally a strong correlation between the hardware utilization of a mobile device and its skin temperature. The work also validated that the correlation can be expressed as a linear relation. Based on the findings, the contribution of each hardware component to a device's temperature during time window *k* is modeled as Eq. (2):

$$T_C^t = \alpha_C * \sum_{i=t-k}^{t} Util_C^i + \delta_C' \qquad (2)$$

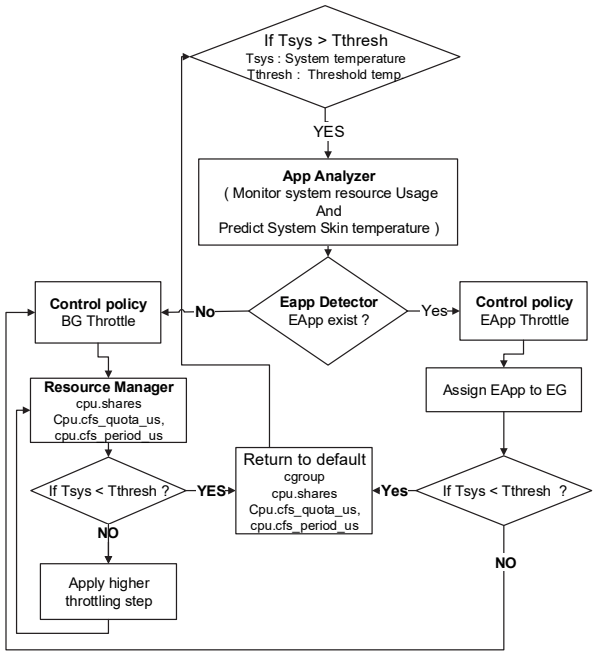


Figure 6. Control policy.

Here, T_C is one of T_{big}, T_{LITTLE}, T_{gpu}, T_{wifi}, T_{cam}, and T_{codec}, and represents the temperature increase during one sample time (i.e., from *t*-1 to *t*) of the big/LITTLE core, GPU, WiFi, camera, and codec, respectively. $Util_C$ is one of F_{big}, F_{LITTLE}, and F_{gpu} and represents the values for the clock frequency of the CPU and GPU, while P_{wifi} is the transmitted packet, R_{size} is the camera recording resolution, and Re_{size} is the resolution when playing a movie. δ_C' and α_E are the coefficients determined by each hardware component. We finally obtain T_{App}^t as follows:

$$T_{App}^t = \beta_{big} * T_{big}^t + \beta_{LITTLE} * T_{LITTLE}^t + \beta_{codec} * T_{codec}^t + \beta_{cam} * T_{cam}^t + \beta_{gpu} * T_{gpu}^t + \beta_{wifi} * T_{wifi}^t . \qquad (3)$$

β_{big}, β_{LITTLE}, β_{codec}, β_{cam}, β_{gpu}, and β_{wifi} are the constants specific to each mobile device. For each hardware component, we determine the thermal coefficient α_C, δ_C', $and\ \beta_C$ by conducting a series of experiments to observe the temperature change according to the component's utilization. During the experiments, all other components except the target component were kept as idle as possible. By summing up the heat generated by all hardware components, the heat generated by an application is finally estimated. Note that we do not consider heat dissipation in this work because our goal is to estimate the heat contribution of each application, not to accurately predict the device temperature.

4.3 App Analyzer

In order to calculate the heat each application generates based on the proposed model, the resource utilization of each application should be acquired accurately. To this end, we developed App Analyzer to collect the hardware utilization of individual applications. To minimize overhead and enhance the generality of the tool, we obtained utilization information from BatteryStats [3], the built-in application behavior monitor in Android. App Analyzer retrieves the utilization of the system components, such as the CPU, GPU, codec, WiFi, and camera. Heat generation is calculated every 100 ms using the heat prediction model. In order to reduce noise, EApp is detected using the average value of the

Table 1. CPU resource assignment for the BG compared to the FG according to the throttling step

	CPU usage ratio	Maximum CPU usage time	Resource reset duration
Default	20%	87.5%	100%
Step 1	10%	50%	200%
Step 2	5%	12.5%	400%

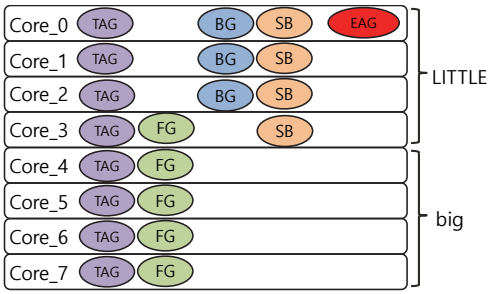


Figure 7. An example of CPUSET configuration in Galaxy S8+.

last 10 temperature data. By predicting the amount of heat each application generates through this process, we compared the heat generated by the background application with the foreground application. When a background application generates more than 20% of the heat compared with the foreground app, it is classified as EApp. Depending on the presence of EApp, we apply one of the policies we describe in Section 4.4.

4.4 Resource Manager

4.4.1 Application Group

Individual management of all the processes is too complicated to be applied in a real environment. Therefore, we divided the processes into five groups based on Android's default process group. The Top-App group (TG) is the most important group which includes the Android core processes, such as Zygote and the system server. The Foreground group (FG) contains the processes that belong to the foreground application. We categorized the background processes into three groups: the system background group (SG), the background group (BG), and the exothermic group (EG). The SG includes the services necessary to operate the system. When the background processes are determined to be exothermic processes, they are assigned to the EG while all other background processes belong to the BG. To guarantee the normal execution of the Android system and the FPS of the foreground application, we do not throttle the TG, FG, and SG. The proposed system focuses on managing the system resources for the BG and the EG.

4.4.2 Control Policy

We designed a control policy for managing the background applications efficiently. First, we classified the thermal throttling situation into two cases. When EApp exists in the system, the situation is considered as EApp Case, otherwise Non-EApp Case. For the EApp Case, limiting EApp is sufficient to lower the temperature because EApp contributes significantly to the device temperature. In this case, we apply the *EApp throttle* policy which moves EApp to the EG where processes are restricted to use only a single little core.

However, EApp does not exist in the Non-EApp Case; therefore, we should throttle all background applications as aggressively as possible to decrease the temperature. To this end, we developed

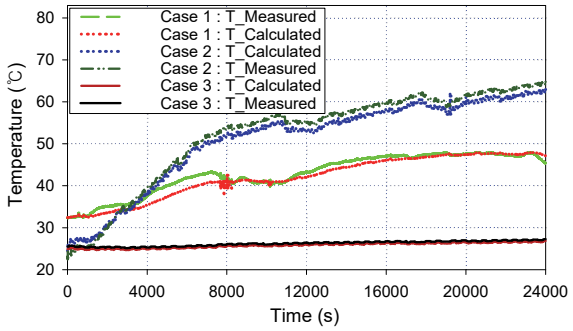


Figure 8. Thermal prediction result.

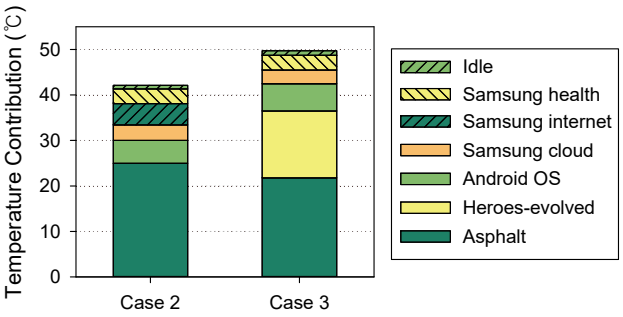


Figure 9. Thermal prediction result.

the *BG throttle* policy. This policy gradually reduces the number of CPU cores available to the BG step by step according to the heat generated. In addition to the number of cores, the available CPU usage time is also restricted for the BG. Table 1 shows the steps for performance control in the BG throttle policy. As the throttling level increases, the CPU usage ratio and absolute available time are decreases for BG. We also lower the bandwidth for the BG while increasing the bandwidth for the FG. Consequently, the amount of system resources for the BG decreases until the temperature goes down. Note that in the EApp Case, when the temperature is maintained or not lowered after a certain period after applying EApp throttling, the BG throttle policy is applied additionally. When the temperature continues to rise due to the high heat generated by the foreground application, the device's default thermal management scheme which limits the performance of the entire device is applied. However, even after the clock is lowered, the proposed scheme would decrease the temperature more quickly because the background applications are restricted. Figure 6 outlines the overall algorithm of the proposed control policy.

5. IMPLEMENTATION

We used Cgroups [14], CPUSET, and CPUCTRL [15] to implement application-level performance management in the Android system. Cgroups is a soft partitioning tool used to provide different services platforms running stably on a server. We used this tool to divide the running applications into groups and assign system resources to each application group differently. With CPUSET and CPUCTRL, we assigned the number of cores and available usage time to each group. Figure 7 shows an example of the application group configuration with the assigned cores in Galaxy S8+.

6. EVALUATION

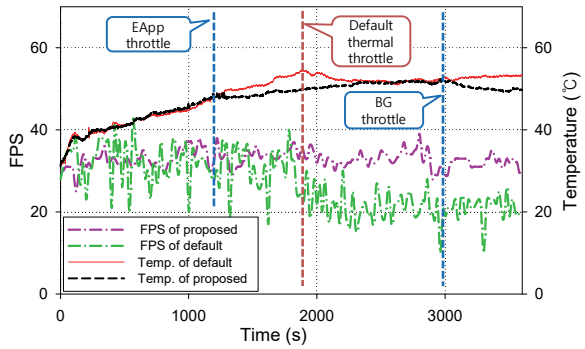


Figure 10. The FPS and temperature trace of the proposed system and the default system.

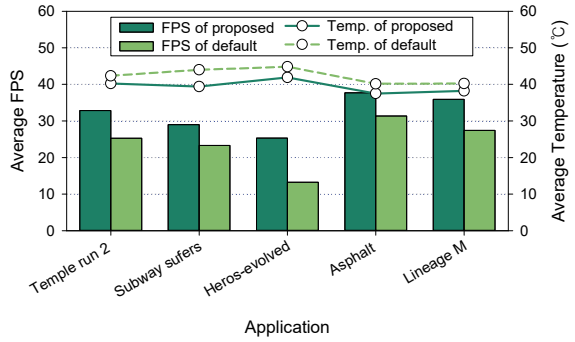


Figure 11. Comparison results for the proposed system and the default system.

6.1 Validating App Analyzer

We first validated the accuracy of App Analyzer. We compared T_{App} calculated by the heat prediction model and $T_{Measured}$ measured using a thermal imaging camera. As stated in Section 4.2, the prediction model does not consider the conduction of the heat each app generates. However, to confirm the accuracy of the prediction model by comparing it with the measured temperature, we must consider the conduction of heat of each application. Based on previous work [13], we estimated the heat dissipation as $\mu * T_{System}^{t-1}$ where the system temperature one unit time ago is multiplied by the heat release rate μ. We used this only to evaluate App Analyzer. Finally, T_{System} is obtained by subtracting the conduction value $\mu * T_{System}^{t-1}$ from the sum of the application heat generation value T_{App}^{t} as follows:

$$T_{System}^{t} = \sum T_{App}^{t} - \mu * T_{System}^{t-1}. \quad (4)$$

Figure 8 shows how accurately App Analyzer estimates the heat of the application with three different cases. In the first case, the home launcher was displayed without an exothermic application, whereas in the second case, Asphalt was running at foreground generating heat. In the third case, Asphalt was running at foreground while Heroes-evolved was executed simultaneously in the background, generating a large amount of heat. The average error of App Analyzer is 1.77%. Next, we verified that App Analyzer detects EApp accurately. Figure 9 shows the heat generation value obtained by App Analyzer while we conducted Case 2 and Case 3. As shown in Figure 9(b), App Analyzer accurately detected the Heroes-evolved application as EApp.

For further analysis, we selected 30 applications in each of the United States, China, South Korea, and UK Google Play stores without duplication. After each application ran in the background for 30 minutes, five EApps were detected: Heroes-evolved (game), Weibo (news), KWAI (movie), and Baidu map (Map) from China's Baidu market and Uber driver (navigation) from the United States' Google Play. We measured the temperature of the smartphone with the thermal imaging camera while running these EApps. The results indicate that EApps generate considerable heat in the background. The detailed analysis of the log of EApp revealed that the EApps were accompanied by system errors. Weibo often increased to more than 40% CPU usage in the background, and at this time, system.err was generated. In Baidu Map, SQLite_ERROR was generated. Due to the nature of the navigation app, the application kept running in the background, which consuming high CPU power and WiFi power.

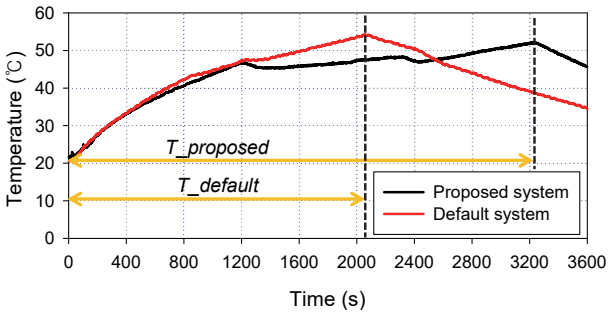


Figure 12. The difference between the proposed system and the default system when the thermal violation point is reached.

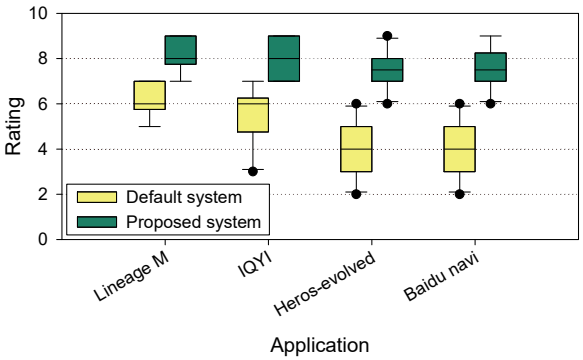


Figure 13. Result of user studies.

6.2 Effects of Thermal Management

We evaluated the proposed system in terms of thermal management and FPS. The temperature and the FPS were measured by thermal camera and the "dumpsys gfxinfo" command, respectively. We compared the proposed system with the default thermal management scheme implemented in Galaxy S8+. When the temperature exceeds the thermal violation point, the default scheme lowers the CPU and GPU frequency to a pre-defined level according to the device temperature. We regard Samsung's default thermal management scheme as a state-of-the-art industry technique. We installed 20 applications, including the six EApps found in Section 6.1, on the both customized Galaxy S8+ device on which the proposed system was enabled and a stock Galaxy S8+ device. Note that Samsung's default thermal management system is activated in both devices. We experimentally set the thermal management threshold as 45 degrees. Therefore, the proposed system started to manage

resource usage after reaching 45 degrees. We first conducted an experiment in which a number of EApps were running in the background, generating a large amount of heat. We put all installed applications in the background and ran WeChat at the foreground for 30 minutes. Figure 10 shows that at around 20 minutes the proposed system first attempted to control the temperature by restricting the EApps. Although the EApp throttle policy made the temperature increase slowly, as the temperature remained above the threshold, the BG throttle policy was applied to decrease the temperature further. As a result, the temperature did not exceed the threshold of thermal violation and guarantees FPS. However, for the default system, the temperature increased beyond the thermal violation point, which ought to limit the device's overall performance. Consequently, the FPS of the default system was much lower than that of the proposed system.

Next, we compared the temperature and the FPS of the proposed system with the default system, while running an application in the foreground and five EApps in the background except for Heros-evolved. As the thermal issue is generally raised by game applications, we selected five widely used game applications. We measured the temperature and the FPS while running game applications on both systems for one hour. For reliable results, we repeated the same usage scenario by Monkey [16]. Figure 11 shows that the proposed system outperformed the default system in terms of temperature and FPS. In particular, for Heros-evolved, one of EApps, the default system severely degraded the FPS because its thermal management scheme limited device performance due to the high temperature. In contrast, the proposed system maintained the FPS at around 25, which is about two times higher than the default system. We analyzed the cause of this difference. We experimented with a case in which a foreground application with a heavy workload pushed the system temperature to the critical threshold. For this case, we installed an application called Hand warmer. This application runs in the foreground and uses all the CPU cores to generate heat out of the device and make the mobile device act as a hand stove. We compared the device's temperature while the application ran for 60 minutes on stock and customized devices. As shown in Figure 12, both types of devices were throttled by the default thermal management scheme at 55 degrees. However, the proposed scheme guarantees the maximum performance of the foreground application up to 60% longer compared with the default system. We believe that this delay greatly reduces the probability of reaching the thermal violation point. As the intensity of the workload varies considerably in real applications, a heavy workload would be maintained only for a short period. In other words, when reaching the critical temperature is prevented for several seconds, the intensity of the workload decreases again, lowering the possibility of a thermal emergency. This is why the proposed system outperformed the default system.

6.3 User Study

We conducted a user study to evaluate the user experience issue related to the proposed thermal management scheme. The experiments were conducted with 10 professional testers working at a software quality assurance company. The participants used four EApps, Heroes-evolved, Weibo, KWAI, and Baidu map, for 20 minutes on two smartphones: one with the proposed thermal management scheme and the other with the default scheme only. Figure 13 shows the results of the responses on the questionnaire that queried the participants' experiences in terms of temperature and FPS. We asked the participants to score their response from 1 to 10 based on their experience. The proposed system was reported to outperform the default system, particularly when high heat was generated, indicating that the app-oriented thermal management system successfully preserves the FPS even when high heat is generated.

7. CONCLUSION

In this paper, we proposed an app-oriented thermal management scheme for mobile devices. The method guarantees the FPS by limiting only background applications that induce high heat. Through a number of experiments with smartphones, the proposed scheme for controlling device temperature without FPS degradation was validated. Because the proposed scheme does not degrade the FPS, we consider that thermal management can start at a very low temperature. We believe that this will lower the possibility of reaching the critical system temperature. In the future work, we will develop enhanced thermal management to control other hardware components, such as the GPU, WiFi, codec, and camera.

8. ACKNOWLEDGEMENT

This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-TB1503-02 and Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. NRF-2017M3C4A7083677).

9. REFERENCES

[1] Brookes, T. 2016. iPhone or iPad getting hot? Here's why & how to fix it. Retrieved from http://www.makeuseof.com/tag/iphone-ipad-getting-hot-heres-fix/.

[2] Japan Today. 2014. Warning issued over "smartphone burns" and other dangers. Retrieved from https://japantoday.com/category/national/warning-issued-over-smartphone-burns-and-other-dangers

[3] Google LLC. 2017. Profile battery usage with BatteryStats and battery historian. Retrieved from https://developer.android.com/studio/profile/battery-historian.html

[4] Singla, G., Kaur, G., Unver, A. K., and Ogras, U. Y. 2015. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In Design, Automation & Test in Europe Conference & Exhibition (DATE '15). IEEE, p. 960-965.

[5] Xie, Q., Kim, J., Wang, Y., Shin, D., Chang, N., and Pedram, M. 2013. Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor. In IEEE/ACM International Conference on Computer-Aided Design (ICCAD '13). p. 242-247.

[6] Paterna, F., Zanotelli, J., and Simunic Rosing, T. 2014. Ambient variation-tolerant and inter components aware thermal management for mobile system on chips. In Proceedings of the conference on Design, Automation & Test in Europe (DATE '14). European Design and Automation Association. p210.

[7] Ishii, M. and Nakashima, Y. 2017. Development of algorithms for real-time estimation of smartphone surface temperature using embedded processor. In 16th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm '17). p. 1088-1093.

[8] Muralidhar, R., Seshadri, H., Bhimarao, V., Rudramuni, V., Mansoor, I., Thomas, S., Veera, B., Singh, Y., and Ramachandra, S. Experiences with power management enabling on the Intel Medfield phone. In Proceedings of the Linux Symposium 2012.

[9] Samsung Electronics Co., Ltd. 2016. Galaxy S8. Retrieved from http://www.samsung.com/us/explore/galaxy-s8/.

[10] Qualcomm Technologies, Inc. 2015. Snapdragon 835. Retrieved from https://www.qualcomm.com/products/snapdragon/processors/835

[11] Samsung Electronics Co., Ltd. 2016. Exynos 9 Series (8895). Retrieved from http://www.samsung.com/semiconductor/minisite/Exynos/Solution/MobileProcessor/Exynos_9_Series_8895.html

[12] Cox. 2015. cx1000 thermal imaging camera. Retrieved from http://coxcamera.com/products/cx1000cx1000-ip-2/

[13] Park, J., Lee, S., and Cha, H. 2018. Accurate Prediction of Smartphones' Skin Temperature by Considering Exothermic Components. In Design, Automation & Test in Europe Conference & Exhibition (DATE '18). IEEE.

[14] Lameter, C. and Jackson, P. 2007. Cgroups. Retrieved from https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

[15] Jackson, P. 2007. CPUSETS. Retrieved from https://www.kernel.org/doc/Documentation/cgroup-v1/cpusets.txt

[16] Android Monkey. Retrieved from https://developer.android.com/studio/test/monkeyrunner/index.html