A Thermal Margin Preservation Scheme for Interactive Multimedia Consumer Electronics

Nicolás Badano, Youngjoo Woo, Jeaho Hwang, and Euiseong Seo

**Abstract — ***The heterogeneous multicore architecture is considered a cogent solution to match the performance demand for processing the next-generation media formats such as ultra-high definition, 3D or holography. However, the performance cores in a heterogeneous multicore processor dissipate a huge amount of heat. To cope with the thermal risk, most modern embedded processors provide the dynamic thermal management (DTM) feature that forcefully reduces the clock speed of the processors. Although this simple approach can maintain the system temperature below the thermal trip point, the performance of prioritized multimedia or interactive applications can be significantly harmed by the reduced performance even when the thermal crisis is caused mostly by the non-prioritized applications. This paper proposes a novel DTM scheme called Thermal Margin Preservation (TMP). TMP differentiates the thermal trip point for the prioritized applications from that for the non-prioritized ones, and thus forms the thermal margin, which is the temperature gap between the two trip points. Under the proposed scheme, the prioritized applications can run without any disturbance in the thermal margin by sacrificing the performance only of the non-prioritized applications. The evaluation shows that the proposed scheme significantly reduces the quality-of-service degradation for video playback under high temperature conditions*[1].

Index Terms — Dynamic thermal management, Temperature, Thermal model, Heterogeneous multicore

I. Introduction

The next-generation multimedia formats, such as 3D, ultra-high-definition, or holography videos, demand extremely high performance to process. Additionally, the number of third-party applications, often demanding significant performance, is rapidly increasing. At the same time, the energy efficiency and power consumption issues are growing in importance.

The heterogeneous multicore architecture, which integrates high-performance cores and energy-efficient cores on the same die, is considered a promising solution to break through the aforementioned antithetical requirements for the embedded processors of the consumer electronics.

The high-performance cores in a heterogeneous multicore processor usually adopt out-of-order architecture, which consumes a significantly larger amount of power than the normal embedded processors for the sake of high performance. Consequently, the long-term operation of the high-performance cores will dissipate a huge amount of heat, and the accumulated heat will bring the thermal crisis to the system.

However, in contrast to PCs or server systems, which can easily accommodate large heat sinks or multiple fans, the heat dissipation capacity of the thermal solutions used in consumer electronic devices is marginal due to the strict limitations of both design-form factors and manufacturing costs. As a result, thermal management is an important issue for the consumer electronics that adopt the heterogeneous multicore processors.

To overcome the thermal issue, most modern embedded processors are equipped with dynamic thermal management (DTM) schemes that lower the clock speed via dynamic voltage and frequency scaling (DVFS) when the system temperature rises above the predefined thermal trip point. This simple and straightforward approach easily controls the system temperature to prevent it from exceeding the thermal trip point. However, the conventional DTM scheme frequently fails to preserve the quality of services (QoS) of interactive or multimedia applications.

The workloads with timeliness requirements, such as multimedia players or games, are very dynamic when it comes to CPU utilization [1]. They present sporadic CPU-intensive execution phases to maintain the target QoS; frame rate for the aforementioned examples. The conventional DTM scheme blindly slows down the execution of all processes, including the multimedia or interactive workloads [2], even when most of the accumulated heat is generated by long-term background tasks, such as cloud backup, virus scanning, and so on.

To deal with the thermal risk of consumer electronics, this paper proposes *Thermal Margin Preservation (TMP),* a novel DTM scheme. TMP introduces the notion of a thermal margin that is defined as the difference between the software-defined thermal trip-point, which is enforced on non-prioritized

[1] This work was supported in part by the Institute for Information & communications Technology Promotion (IITP) under Grant No.10041244, Smart TV 2.0 Software Platform, and No. B0101-15-0644, Research on High Performance and Scalable Manycore Operating System, funded by the Korea government (MSIP).

N. Badano was with Sungkyunkwan University, Suwon, South Korea. He is now with JP Morgan Chase & Co., Glasgow, United Kingdom. (e-mail: nicolas.badano@gmail.com).

Y. Woo is with Sungkyunkwan University, Suwon, South Korea (e-mail:mongmio@csl.skku.edu).

J. Hwang was with Korea Advanced Institute of Science and Technology, Daejeon, South Korea. He is now with RTST Co. Ltd., Daejeon, South Korea. (e-mail:jhhwang@rtst.co.kr).

E. Seo is with Sungkyunkwan University, Suwon, South Korea (e-mail: euiseong@skku.edu).

Contributed Paper
Manuscript received 12/30/15
Current version published 03/30/16
Electronic version published 03/30/16

applications, and the highest temperature allowed by the hardware vendor. TMP allows the prioritized applications, such as foreground interactive ones, to execute without performance degradation, while the execution of non-prioritized applications is controlled to reduce the system temperature when the current temperature lies in between these two thermal ceilings. Consequently, under the proposed scheme, the QoS of the prioritized applications will be barely affected by the thermal management most of the time.

The proposed scheme was implemented on an embedded system, which is popularly used for smart phones and smart TVs. The prototype implementation was evaluated to measure the improvement in the QoS under high heat conditions.

The remainder of this paper is organized as follows. Section II illustrates the motivation behind this research. Based on this motivation, Section III proposes the TMP scheme. Section IV describes the prototype implementation of the proposed scheme and presents the evaluation results. After the related work is introduced in Section V, Section VI concludes this study and discusses further research.

II. Motivation

The existing DTM defines the appropriate reaction to the predefined temperature levels and carries it out according to the current temperature. This mechanism can be implemented in the firmware of the system-on-chip (SoC). However, in many cases, the DTM is implemented in the thermal driver module of the operating system (OS) kernel.

The embedded system, which was used in this research, shows a typical DTM implementation. The thermal driver of the kernel for the embedded board defines three temperature levels and their corresponding cooling policies as described in TABLE I.

TABLE I
CONVENTIONAL THERMAL MANAGEMENT POLICIES

Temperature (℃)	Policy
110	Clock speed scaling
115	Notification
120	Cut off CPU execution

When the current temperature exceeds 110 ºC, the driver adjusts the clock speed according to the following rules inside a loop that samples the temperature every few milliseconds; i) If the current temperature is above the tripping point and the temperature is rising, then the driver reduces the CPU clock speed by one step. ii) If the current temperature is below the tripping point and the temperature is falling, then the driver increases the clock speed.

At 115 ºC, the thermal management unit (TMU) notifies the OS kernel about the critical condition, and at 120 ºC, the TMU brutally shuts down the processor. Therefore, the OS should regard the clock speed scaling point as the practical upper bound of the system temperature.

This mechanism is considered as a straightforward DVFS in reaction to the current temperature state. It is unaware of the nature of the executing threads, whether with timeliness requirement or not, and it does not care about how the user experience becomes affected by the decreased performance.

To identify the thermal management scheme's performance impact on the interactive applications, diverse system parameters, such as system temperature, clock frequency and processor utilization, were monitored during the execution of two applications, which are extensively described in Section IV and represent a prioritized foreground task and background tasks, respectively.

The target system for this research was chosen to represent a typical heterogeneous multicore architecture, which is widely used by the consumer electronic devices as mentioned previously. In the target system, the four performance cores were grouped as the performance core cluster, while the other four energy-efficient cores were grouped as the energy-efficient core cluster. Either the performance or the energy-efficient cluster could be activated at any point in time, and the power management module would determine the cluster to operate.

TABLE II
AVAILABLE CLOCK FREQUENCIES OF TARGET SYSTEM

Frequency (GHz)
1.6 GHz, 1.5 GHz, 1.4 GHz, 1.3 GHz, 1.2 GHz, 1.1 GHz, 1.0 GHz, 0.9 GHz, 0.8 GHz,

As shown in TABLE II, the processor provides multiple clock levels to the power management module in the kernel. The maximum operating clock frequency of the energy efficient core cluster is 1.2 GHz, and that of the performance core cluster is 1.6 GHz. When the power management module adjusts the clock frequency to a value higher than 1.2 GHz, the performance core cluster will be activated, whereas the energy-efficient core cluster will run. For example, when the power management module raises the target operating clock frequency to 1.3 GHz from 1.2 GHz, the cores in the performance core cluster will exit from the sleep state and run at 1.3 GHz while the cores in the energy-efficient core cluster, which operated at 1.2 GHz, will instead enter the sleep state.

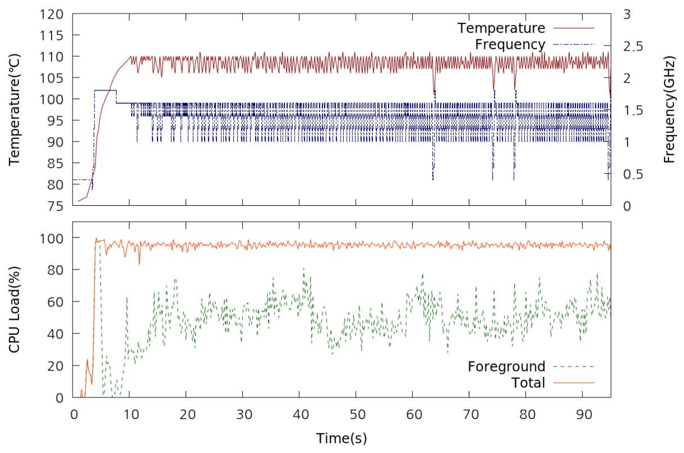


Fig. 1. Thermal management impact on the application execution

As Fig. 1 shows, it was common for the clock frequency to drop by five or more steps due to the thermal crisis. This aggressive downscaling will translate into the saturation of the total processor utilization. Even when the processor utilization saturated and the prioritized tasks were not given sufficient processor resource, the background tasks consumed significant amounts of the processor resource, as shown in Fig. 1.

Interactive or multimedia applications, such as games, web browsers, and multimedia file players, are very unpredictable in terms of the performance demand at any given time partly because they largely depend on human interaction and input data. Therefore, interactive workloads have sporadic execution phases that require high processing power, which is called the *high-performance burst*.

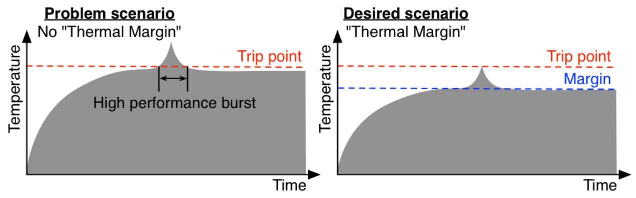


Fig. 2. Notion of thermal margin

If the interactive tasks were the primary source of the thermal crisis, it would be a reasonable and unavoidable solution to retard the execution of them. However, in many cases, the high temperature results from the heat accumulation caused by executing the long-term background tasks. In such a situation, the high-performance burst of a prioritized task may become the small but concluding contributor to the user-perceivable QoS degradation, as shown in the problem scenario of Fig. 2.

Obviously, it would be desirable that the QoS of the prioritized tasks be protected from performance downscaling due to the heat dissipated by the non-prioritized tasks. This would be achievable if the DTM scheme could predict the increase in temperature contributed by the sporadic activation of prioritized applications and could maintain the system temperature below a certain threshold, which would be determined based on the prediction as shown in the desired scenario of Fig. 2.

III. Design of Thermal Margin Scheme

The design objective of the TMP scheme is to preserve the thermal margin so that the temperature does not hit the thermal trip point even when the high-performance bursts of prioritized tasks occur.

In contrast to the conventional DTM mechanisms, which reactively respond to the current temperature, TMP employs a proactive approach. Therefore, TMP can be used together with the conventional DTM. In this case, the conventional DTM, as a safeguard mechanism, will respond to catastrophic situations, where the temperature rises above the hardware-defined thermal trip point, by abruptly adjusting the clock frequency.

Fig. 3 shows the basic workflow of the TMP scheme. A thermal model is required to predict how high the temperature will rise from the current temperature, caused by the heat dissipated by the high-performance bursts of the prioritized tasks. The constructed model will be used to determine whether the system is in a safe state in terms of thermal management. When the system is determined to be in a dangerous condition, TMP restricts the execution of non-prioritized or non-interactive threads, which do not affect the user experience. Therefore, TMP includes the technique that identifies the interactive threads out of all existing threads in the system. Finally, TMP provides the cooling policies and selects one of them according to the situation so that it can control the system temperature while preserving the performance of prioritized applications.

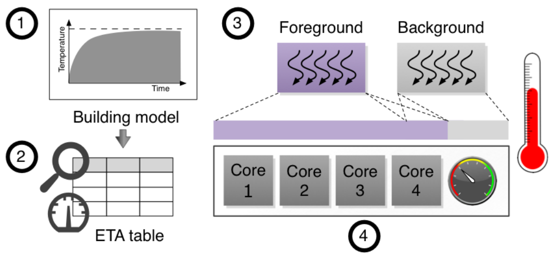


Fig. 3. TMP system overview

The following subsections present detailed descriptions of the introduced components of TMP.

A. *Thermal model*

One of the main contributions of TMP is its predictive nature. What enables TMP to predict the temperature change is a thermal trend model of the system-on-chip (SoC). A thermal model is fundamentally a temperature change curve in the function of time, depending on various factors, including the processor load, clock speed, number of active cores, processor temperature, and ambient temperature. With this curve, TMP is able to predict what the temperature value will be in the future.

The thermal model is built by the model builder process, and the model should be constructed during the first boot and any system reconfiguration. The model builder collects the required data to construct the model while performing a series of stress tests. During the test runs, the model builder proceeds to execute stress tests under all possible core configurations. The model builder records the temperature readings in the function of time and deduces the temperature curve for a given core configuration depending on the time change.

The on-chip temperature sensors may have an uncertainty of several degrees. However, the model is not for estimating the absolute temperature, but for predicting the relative temperature distance from the thermal ceiling, which is also determined by the on-chip temperature sensor. In addition, the model is built from the large set of collected temperature data. Consequently, the inaccuracy of the temperature sensors trivially influences the effectiveness of the model.

The ambient temperature affects how quickly a system dissipates produced heat. Therefore, it determines the sustaining temperature and speed of the temperature change of the system. While most of the modern embedded processors are equipped with processor temperature sensors, they are rarely furnished with ambient temperature sensors. Therefore, the temperature model indirectly infers the ambient temperature from the test run results. Thus, if the ambient temperature changes dramatically, the model should be built again. However, the observation showed that minor changes, such as 10 ºC, of the ambient temperature negligibly impacted the accuracy of the model.

The model built with this approach showed the R^2 value, which is a measure of how well the model fits the observed data, of 0.998 indicating the high accuracy of the curves. Similar approaches or observations were introduced by a number of existing research results [2],[3].

TMP predicts the estimated time of arrival (ETA) at the thermal trip point in the current core configuration based on the built model. To reduce the overhead for the kernel when predicting the ETA, TMP creates the ETA tables based on the thermal model and stores them inside the kernel. Because the temperature granularity is 1-degree Celsius, the tables only took a few KB for the experimental system.

Each of the ETA tables represents a thermal trend curve for a specific core configuration; for example, four high-performance cores running at 1.5 GHz. An entry in a table is indexed by the current temperature, and holds the calculated ETA at the thermal ceiling.

B. *Interactive thread identification*

An interactive thread is defined as a thread that affects the user experience. A typical stimulus-reaction cycle of an application consists of multiple interactive threads working collaboratively, and this chain of interactive threads must finish the cycle within the human perception threshold, usually 50 ms [4]. Therefore, it is essential that TMP be aware of which threads are prioritized ones.

TMP detects interactive threads, and puts them into the interactive thread group, as follows.

First, a user input, such as a screen touch or a button press, is detected via the input framework of the platform. The thread that is responsible for handling this input event will be put into the interactive thread group. Then, the inter-process communication (IPC) mechanisms will be monitored for IPCs between any thread belonging to the interactive thread group and any other thread that is not in this group. When a communication is detected, TMP flags the non-interactive counterpart as an interactive one and puts it into the interactive thread group.

The end result is a set of threads whose members at a given moment are directly or indirectly responsible for the user experience. TMP treats the threads in the interactive groups as prioritized threads.

C. *Scheduling Dynamics*

TMP monitors the processor utilization contributed only by the interactive threads, number of active cores, clock frequency, and current processor temperature at every 10 ms interval to determine the current ETA at the thermal trip point, as well as determines the appropriate cooling policy. The 10 ms monitoring interval is also used by the *ondemand* governor [5], which is used as the default power management module of the Linux kernel, and has been verified to incur insignificant overhead.

The TMP mechanism defines three thermal states as shown in Fig. 4, in which the system will be at any given time. The basic descriptions of the states are as follows:

- ✓ *Safe state*: no thermal crisis, proceed at full capacity.
- ✓ *Alarm state*: thermal crisis ahead, take preventive action.
- ✓ *Danger state*: ongoing thermal crisis, assess the situation and take further action if necessary.

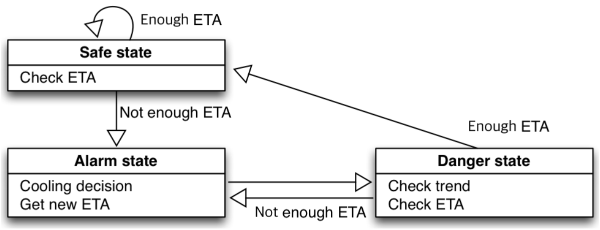


Fig. 4. Thermal state diagram

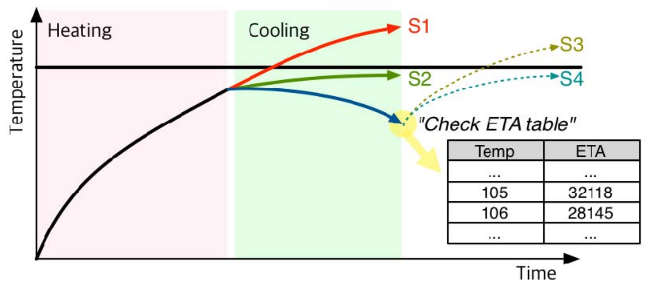
While the system is in the safe state, TMP regularly checks the temperature changes. If the temperature is rising, then TMP consults the thermal model about whether the ETA is larger than the thermal margin, which is the sum of the recorded longest burst cycle of each thread group.

If TMP detects that the ETA at the trip point in the current core configuration is shorter than the thermal margin, then the system enters the alarm state. The alarm state is a transitional state in which the system applies a new cooling policy and moves to the danger state. An appropriate cooling policy to reverse or relieve the temperature change may differ, depending on the circumstances. Section III.C discusses the details of the cooling policies.

Once a cooling policy has been applied TMP will cross over to the danger state. At this state, TMP will assess the impact of the cooling policy applied in the alarm state at every monitoring interval. Since the cooling policy changed the core configuration, TMP selects the appropriate ETA table reflecting the current core configuration.

There are four possible scenarios for the monitored system status in the danger state as shown in Fig. 5.

- ✓ *S1*) The temperature is still ascending, and the ETA is shorter than the determined thermal margin.
- ✓ *S2*) The temperature is still ascending, but the ETA is longer than the determined thermal margin.
- ✓ *S3*) The temperature is descending, but the ETA at the maximum performance level is shorter than the determined thermal margin.
- ✓ *S4*) The temperature is descending, and the ETA is longer than the determined thermal margin even at the maximum performance level.



Temp	ETA
...	...
105	32118
106	28145
...	...

Fig. 5. Four possible scenarios in the danger state

Under S1, the current cooling policy is clearly insufficient to control the temperature, and eventually, the system will hit the thermal ceiling. Consequently, TMP pulls back the system to the alarm state, applies a more aggressive cooling policy, and pushes the system into the danger state again.

TMP maintains the danger state under S2 and S3 because unleashing the performance will likely threaten the system's thermal stability and harm the QoS of the prioritized applications. However, because the temperature is well controlled with the current cooling policy under S2 and S3, TMP keeps monitoring and does not apply any further action to cool down the system more aggressively.

Under S4, the source of the heat may have gone, or at least, the risk has been significantly reduced by the current cooling policy. To improve the performance of the non-interactive threads, TMP reverts the system to the safe state so that the system can function at its maximum performance level.

D. Cooling Policy

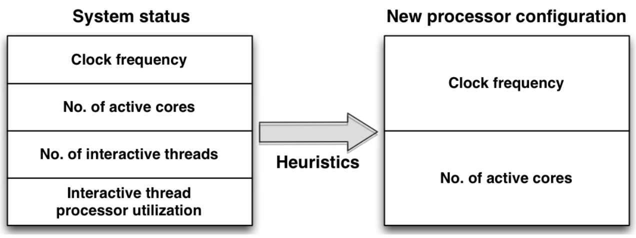

System status
Clock frequency
No. of active cores
No. of interactive threads
Interactive thread processor utilization
Heuristics
New processor configuration
Clock frequency
No. of active cores

Fig. 6. Inputs and outputs of cooling policy decision heuristics

Fig. 6 shows an overview of the cooling decision heuristic's inputs and outputs. It will take the system state values profiled by TMP in a data sampling instance, as explained earlier, apply the decision heuristics to choose an appropriate cooling policy, and create a new core configuration suitable for the current circumstances according to the chosen policy.

The leakage power consumption significantly contributes to the overall heat dissipation; for example, approximately 35% in the target system. However, the leakage power is barely controllable. Therefore, cooling down is mostly enforced by regulating the dynamic power consumption.

There are two alternative policies to cool down the system temperature; shrinking the number of active cores and decreasing the clock speed. Although both approaches will significantly reduce the heat dissipation, reducing the clock speed will retard the execution of all threads, while shrinking the number of active cores will deprive low priority threads of the chances to be scheduled.

Before a cooling policy is applied, the scheduling priorities of non-prioritized threads are adjusted to the lowest level and those of prioritized ones to the highest level so that the non-prioritized threads yield the processor resource to the prioritized threads if possible.

The first heuristic of TMP is to shrink the number of active cores because the execution time of prioritized threads will less likely change in comparison to the other option. It is technically challenging to determine the sweet spot of the number of active cores that will safely maintain the system temperature while minimizing the impact on the execution time. To simplify this problem, TMP turns off cores one by one until the number of interactive threads becomes greater than or equal to twice the amount of active cores. The rationale behind this is that having less than two threads per core will not take advantage of the super-scalar, out-of-order and simultaneous multi-threading (SMT) capabilities of modern processor architecture, while more than two threads per core will possibly generate significant resource contention within the core. However, the adequate number of threads per-core may differ depending on the workload characteristics and the micro-architecture of the processor.

When the number of active cores becomes equal to twice the number of interactive threads, the second heuristic, which lowers the clock speed by one step, is applied.

However, if the processor utilization contributed by the interactive threads rises above a predefined threshold (for example, 70 %) TMP will neither decrease the clock speed nor reduce the number of active cores. TMP will also maintain the current core configuration because the saturated processor utilization will result in the delayed response of interactive applications.

It is noteworthy that these heuristics are straightforward and have a lot of room for improvement, not only in the algorithms but also in using the emerging hardware capabilities that novel architecture provides. For example, the embedded system used in this research can activate either the energy-efficient core cluster or high-performance core cluster, not both at the same time due to the difficulties in the cache coherence protection between both core clusters. However, the cutting edge heterogeneous multicore architecture overcame this limitation and can simultaneously operate both core clusters [6]. With this technology, TMP may migrate non-prioritized threads to the energy-efficient cores while keeping the prioritized ones in the performance cores to cool down the system.

E. QoS Watchdog

So far, the tactics to create the thermal margin and to preserve the performance of the prioritized applications have been described. Of course, in a real situation, a thermal-efficient core configuration will not be sustainable indefinitely. At some point in time, the workload will require an increase in processing power. In other words, as previously defined, sporadic high-performance bursts are expected to happen at any moment without any warning. Therefore, TMP also needs a mechanism to detect these high-performance bursts and change the core configurations to satisfy the newly increased processing demand. This is where the QoS watchdog comes into play.

The QoS watchdog is the last but not the least important component of TMP. As illustrated in Fig. 7, the watchdog component regularly samples the processor utilization of interactive threads. If a predefined threshold is reached, then TMP will proceed to increase the processor performance by one step in an inverse way of the heuristic algorithm to determine the cooling policy. The processor utilization threshold of the QoS watchdog may differ from the one used for the cooling policy determination algorithm, and it is recommended to be higher than that. In the evaluation, the threshold of the QoS watchdog was set to 80 %.

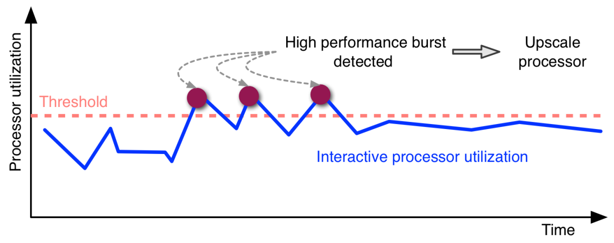


Fig. 7. High-performance bursts detection by QoS watchdog

At this point, TMP has two components fighting for antagonistic interest. On one hand, the cooling decision component pulls to reduce the processor performance; on the other hand, the QoS watchdog pushes to increase the processor performance. These two forces will work together to create the desired equilibrium when it is possible. The state diagram shown in Fig. 8 explains how both antagonistic components work together simultaneously.

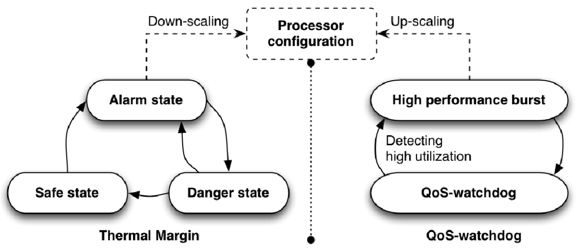


Fig. 8. Dynamics of performance level determination

F. Limitations

Determining the optimal size of the thermal margin would be desirable for the ideal operation of TMP because a large one would degrade the performance of the background workload, and a small one could lead to a thermal crisis. Finding the optimal size of the thermal margin would indicate the mechanism's upfront knowledge of the lengths of the high-performance bursts for a given interactive workload, which would not be possible. Because of this point, the current implementation of TMP tries to build the largest margin possible by summing the longest burst cycle of each interactive thread group. Naturally, this approach adversely impacts the performance of the background workloads.

Nevertheless, this issue can be resolved by an in-depth analysis of the workload behavior. For example, TMP can apply a statistical approach to estimate the acceptable level of thermal margin for a given application. Moreover, to deduce the optimal thermal margin, TMP may accurately simulate what will happen to the overall processor burst cycle when multiple interactive thread groups are competing for the processor resource. The research on the determination of the thermal margin is beyond the scope of this paper, which proposes and realizes the concept of the differentiated thermal management based on the workload characteristics, and is left as a future research theme.

IV. Evaluation

In order to evaluate the effectiveness of TMP in maintaining user experiences for interactive applications, a comparative experiment was conducted. TMP with the out-of-the-box DTM was compared with the power management mechanism included in the Linux distribution built for the target embedded system.

For the workload, a scenario that provided sufficient levels of stress to the processor was emulated in order to reach high temperature. In addition, a possible common use-case for modern consumer electronic devices, such as smart phones, smart TVs, or tablets, was represented in the scenario.

The workload consisted of two applications running simultaneously, a background application and a foreground one, as described in TABLE III. The background task emulated a long-term computing-intensive application, such as background file encryption, pattern matching for malware detection, scanning for memory deduplication, 3D data processing, multimedia file encoding, and so on. The types of such background tasks are expected to increase as the functionalities and features of consumer electronics expand. Three computation workload instances were created and executed to stress all four cores.

TABLE III
Description of Experimental Workloads

Workload	Type	Description
Video player	Interactive	Video playback of 1080p/29.97 fps files
Computation	Background	SPEC2006 benchmarks

As a representative example of a foreground multimedia workload, a video player was chosen because it was easy to quantitatively measure the QoS degradation, which could be indicated by frame skipping. On the other hand, quantitatively measuring the QoS of other interactive applications, such as games or web browsers, would be technically difficult although the experiments with diverse video games clearly showed sensory QoS improvement by applying TMP.

Six threads were forked and executed for the video player, and the processing demand for the video playback significantly fluctuated depending on the characteristics of the frames to be rendered. For example, rendering fast-moving scenes required higher performance than rendering still scenes. Therefore, the high-performance bursts sporadically appeared, and the duration of them would be difficult to predict, as mentioned.

The mixture of the workloads was executed on the target system introduced in Section II. To make the comparison, the workloads were first run on the unmodified Linux kernel, which included the *ondemand* power management module and the default DTM module, which was explained in Section II. Then, the workloads were run on the Linux kernel with TMP. The *ondemand* power management module was disabled to prevent interference with TMP. The workloads were executed for two minutes, and the series of experiments was repeated several times after sufficiently cooling down the board. The time series graphs shown in this paper are select from the repetitions to show the representative characteristics.

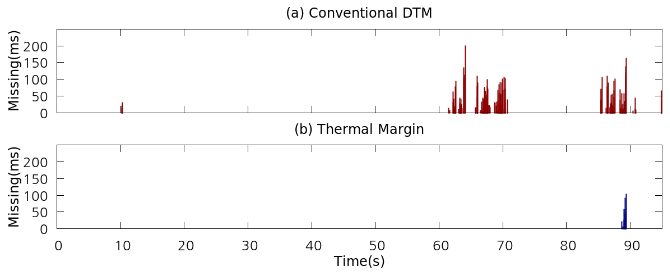


Fig. 9. Time series of frame skip duration under different DTM schemes

To quantitatively measure the user experience degradation, debug data were collected from the video player. The video rendering software would skip frames of footage when the processing power was unavailable. This would translate into a slow frame rate and bad quality video, both directly related to the user experience. The debug data would show the amount of milliseconds of a video frame skipped by the video renderer at any given moment.

Fig. 9 shows the time series of the duration of skipped frames during the first 90 s of the experiment. Each vertical line represents the length of time without rendering the frames on time. A frame skip instance longer than 50 ms is clearly noticeable, and the one longer than 100 ms is significantly annoying to watch.

The results showed that the default DTM generated a significant amount of skipped frames, and the duration of frame skip instances was noticeably long. The sum of the frame skipping duration was 7,491 ms on average during the two minutes of video playback. However, both occurrence frequency and severity of the skipped frames were significantly suppressed by the TMP to the sum of 1,439 ms.

Unfortunately, even TMP could not totally remove frame skipping, and some of the instances reached 100 ms. This was because the thermal margin was not determined to be sufficiently large for the experiment; therefore, the performance scale-up decision by the QoS watchdog was offset by the cooling policy.

The severe performance degradation by the default DTM was explained by its aggressive performance adjustment. The video player started skipping frames below 1.4 GHz. However, the clock speed easily fell below 1.4 GHz with the default DTM, as shown in Fig. 1. On the other hand, TMP avoided frame skipping in most cases because it created a more suitable thermal environment that allowed the video player to use all the processing power needed, which raised the temperature without consequences.

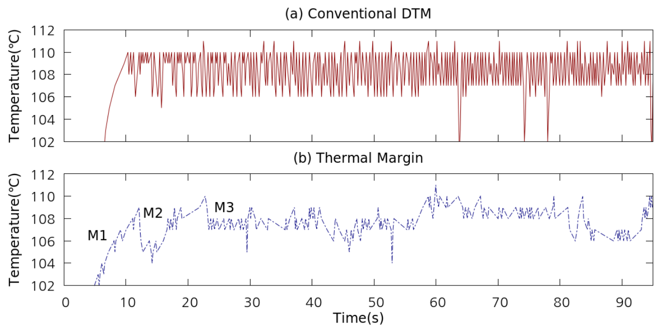


Fig. 10. Temperature changes under different DTM schemes

Fig. 10 shows the time series of the system temperature changes for both experimental configurations. The ambient temperature was set to 25 °C. The first three thermal margins created by TMP are labeled M1, M2, and M3 in Fig. 10 (b). At M1, TMP decided to decrease the number of cores to three, as shown in Fig. 11, which slowed down the temperature increase. However, the temperature steadily rose to the alarm state, and at M2, TMP lowered the clock speed by one step instead of deactivating a core since there were six interactive threads. At M3, TMP reduced the clock speed again by one step, and the temperature fluctuated at around 107 °C after M3.

The system rarely reached the thermal ceiling with TMP, while the default DTM frequently hit the thermal ceiling. TMP also clearly reduced the amplitude of the temperature change compared to the default DTM because of TMP's proactive reaction to the temperature change.

Although lowering the average temperature is not the goal of TMP, it reduced the average operating temperature to some extent. The default DTM scheme managed to keep the average temperature for the given workloads at 108 ºC, with a total sum of 7,491 ms of skipped video rendering. The average temperature under TMP was 106.5 ºC, indicating a 1.8 % reduction in heat accumulation.

Most importantly, TMP accumulated only 1,439 ms of skipped video rendering, representing an 80.8 % improvement in the QoS. The heat reduction average might be insignificant, but the QoS improvement, which is the primary concern of TMP, was substantial.

Fig. 11 shows the time series of the temperature change, interactive thread processor utilization, clock frequency, and number of active cores. The clock frequency stayed within the 1.3 to 1.6 GHz range. Because the temperature was steadily maintained near the thermal ceiling due to the existence of computing-intensive threads, the TMP continually tried to lower the clock speed. However, because the processor utilization of the interactive threads was also high, the QoS watchdog maintained the clock frequency above 1.3 GHz, which was a desirable level for the given workload.

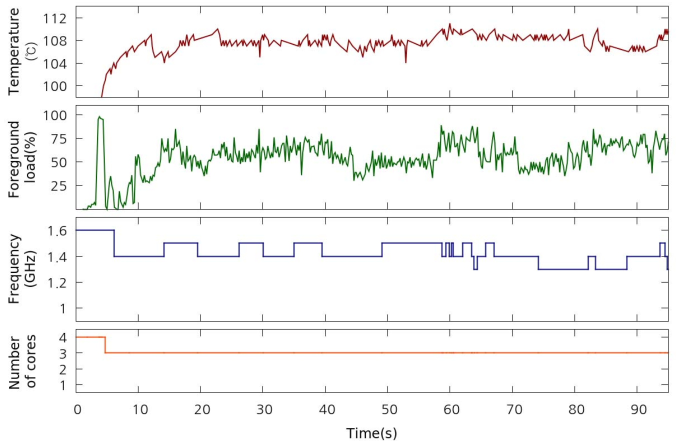


Fig. 11. System status changes under TMP during the experiment

V. Related Work

The thermal design power (TDP) constraint will be one of the most important performance limiting issues when designing new generation processors because even at the constant frequency, a double per-generation core growth will soon exceed TDP [7]. This means that innovative ways to deal with the TDP constraint have to be investigated [3].

Brooks et al. [8] defined and investigated the major components of the DTM schemes. Specifically, they explored the trade-offs between several mechanisms for responding to periods of thermal trauma, and investigated the effects of hardware and software implementations. Skadron et al. [2] also built a microarchitecture thermal model, and used it for comparing several different techniques of the DTM, such as DVFS, migrating computation to spare hardware units, and combinations of fetch gating with DVFS. The cooling policy decision heuristics of TMP can be refined by adopting the diverse thermal management mechanisms considering their characteristics.

Liu et al. [9] proposed a design-time model that analyses workloads and creates task-to-voltage mappings for real-time constraints. The mappings have a number of objectives: energy optimization, thermal optimization, and thermal constrained energy optimization. Scheduling is performed by assigning the best possible mapping to a processor without violating its current thermal and power constraint. Akbari et al. [10] also proposed a dynamic task priority scaling for thermal management that maps and schedules tasks on a multicore processor reflecting their priorities to lower the system temperature. The aim of these approaches are to reduce the total amount of heat dissipation for a given workload, and is orthogonal to that of TMP. They thus can be used together with TMP to earn synergic benefit in terms of the QoS protection in case of thermal crisis.

Preserving the QoS of multimedia applications has been an important issue for the power management schemes because the blind performance management may critically harm the user experience. Kamat [11] proposed a framework that multimedia applications and energy management module collaborate with each other to extend the battery lifetime of multimedia applications. This framework predicts the battery charging patterns and uses that information for power management. However, its goal is not to preserve the QoS of them, on the contrary, it maximizes the battery lifetime by sacrificing the QoS of them based on the prediction.

Hwang et al. [12] designed a hardware-supported power management unit (PMU). The hardware-assisted PMU can efficiently collect the information of I/O usage patterns of applications. Based on the collected information, the PMU carries out predictive power management. While the proposed approach proactively reacts to the workload characteristics, such information is only used for improvement of energy efficiency, not for thermal management.

Srinivasan et al. [13] proposed a predictive DTM scheme, which was designed for supporting multimedia applications. The proposed predictive model is based on an algorithm that profiles applications for an appropriate amount of frames to check the maximum temperatures in all the structures of the chip. Then applies micro-architectural adaptation to reduce temperature for the given workload. This methodology assumes an amount of thermal sensors per chip, which are unavailable in embedded processors yet.

Donald et al. [14] explored novel techniques of the DTM for multicore processors, and evaluated the techniques with simulation. However, the target workloads were mainly the computing-intensive and throughput-oriented ones.

Yi et al. [3] proposed an improved thermal modeling technique that can be used to predict the chip temperature more accurately and efficiently at design time. In addition, they developed a heuristic algorithm to address the static application mapping and scheduling problem based on the proposed thermal model to control the operating temperature.

Chu et al. [15] developed an adaptive online thermal-aware task scheduler for multicore processors. In the same manner as TMP, it monitors certain system parameters to find a thermally-efficient core to map a task. Different from TMP, they assumed that each task would explicitly claim its service level requirement.

Fisher et al. [16] explored how to adapt execution speed of a multicore SoC for real-time tasks in order to cope with thermal constraints. As a thermal model, they used Fourier's cooling model, which is used in most of the related literature in one way or another.

VI. Conclusions

Due to the simultaneous strong demand for both high performance and energy efficiency, the heterogeneous multicore architecture is expected to be popularly used for consumer electronics in the near future. However, because of the manufacturing costs and small design-form factors, the cooling solution will unlikely match the large heat dissipation from the high-performance cores in the heterogeneous processors.

Although the current embedded systems provide the DTM feature, the performance of the interactive applications will be significantly affected in the thermal crisis because the conventional DTM schemes adjust the performance level of the entire system and thus blindly retard the execution of all running threads to cool down the system.

This paper proposed the Thermal Margin Preservation scheme, a novel DTM scheme for consumer electronics. TMP differentiates the thermal ceiling of each thread according to its characteristics. TMP automatically determines the threads that affect the user experience, and prioritizes them in terms of thermal management. By restricting the execution of the non-prioritized threads, TMP maintains the thermal margin, and the thermal margin helps the interactive threads to execute without performance degradation.

The prototype implementation showed that the proposed thermal management scheme dramatically suppressed the occurrences of frame skipping during the video playback while executing a long-term background application.

Although this paper proposed and implemented the concept of differentiated thermal management depending on the thread priority for consumer electronics, further research is required to accurately estimate the thermal margin for a given condition and to predict the temperature changes with less overhead. With the aid of the emerging hardware technology, diverse cooling policies, such as the migration of non-prioritized threads to the energy-efficient cores while keeping the prioritized threads in the high-performance cores, can be additionally integrated into the proposed scheme.

REFERENCES

[1] C. J. Hughes, P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan, “Variability in the execution of multimedia applications and implications for architecture,” *in Proc. International Symposium on Computer Architecture*, Göteborg, 2001, pp. 254–265.

[2] K. Skadron et al., “Temperature-aware microarchitecture: modeling and implementation,” *ACM Trans. Archit. Code Optimization*, vol. 1, no. 1, pp. 94–125, Mar. 2004.

[3] J. Yi et al., “An improved thermal model for static optimization of application mapping and scheduling in multiprocessor system-on-chip,” *in Proc. IEEE Computer Society Annual Symposium on VLSI*, Tampa, 2014, pp. 547–552.

[4] K. Flautner, and T. Mudge, “Vertigo: automatic performance-setting for Linux,” *in Proc. USENIX Symposium on Operating Systems Design and Implementation*, Boston, USA, 2002, pp. 105–116.

[5] V. Pallipadi, and A. Starikovskiy, “The ondemand governor: past, present and future,” *in Proc. Linux Symposium*, 2006, vol. 2, pp. 223–238.

[6] P. Greenhalgh, “Big.Little processing with ARM Cortex-A15 & Cortex-A7,” ARM Inc., San Jose, CA, USA, Sep. 2011.

[7] W. Huang, K. Rajamani, M. R. Stan, and K. Skadron, “Scaling with design constraints: predicting the future of big chips,” *IEEE Micro*, vol. 31, no. 4, pp. 16–29, Jul. 2011.

[8] D. Brooks, and M. Martonosi, “Dynamic thermal management for high-performance microprocessors,” *in Proc. International Symposium on High-Performance Computer Architecture*, Nuevo Leone, 2001, pp. 171–182.

[9] Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang, “Thermal vs energy optimization for DVFS-enabled processors in embedded systems,” *in Proc. International Symposium on Quality Electronic Design*, San Jose, 2007, pp.204–209.

[10] A. Akbari, S. P. Mozafari, H. Noori, and F. Mehdipour, “Dynamic task priority scaling for thermal management of multi-core processors with heavy workload,” *in Proc. Great Lakes Symposium on VLSI,* Pittsburgh, Pennsylvania, USA, 2015, pp. 373–378.

[11] S. P. Kamat, “Energy management architecture for multimedia applications in battery powered devices,” *IEEE Trans. Consumer Electron.,* vol.55, no.2, pp.763–767, May 2009.

[12] Y. Hwang, S. Ku, and K. Chung, “A predictive dynamic power management technique for embedded mobile devices,” *IEEE Trans. Consumer Electron.,* vol.56, no.2, pp.713–719, May 2010.

[13] J. Srinivasan, and S. V. Adve, “Predictive dynamic thermal management for multimedia applications,” *in Proc. 17th ACM/SIGARCH International Conference on Supercomputing*, San Francisco, 2003, pp. 109–120.

[14] J. Donald, and M. Martonosi, “Techniques for multicore thermal management: classification and new exploration,” *ACM SIGARCH Comput. Archit. News,* vol. 34, no. 2, pp. 78–88, May 2006.

[15] H. H. Chu, Y. C. Kao, and Y. S. Chen, “Adaptive thermal-aware task scheduling for multi-core systems,” J. Syst. Softw, vol. 99, pp. 155–174, Jan. 2015.

[16] N. Fisher, J. J. Chen, S. Wang, and L. Thiele, “Thermal-aware global real-time scheduling and analysis on multicore systems,” *in Proc. IEEE Real-Time and Embedded Technology and Applications Symposium,* San Francisco, 2009, pp. 547–560.

BIOGRAPHIES

Nicolás Badano received his BS degree in systems engineering from ORT University, Uruguay in 2011, and MS degree in computer science from Sungkyunkwan University, Korea in 2015. He is currently working at the iOS development team of JPMorgan Chase & Co., Glasgow, UK. His research interests are embedded systems and thermal-aware scheduling.

Youngjoo Woo earned her B.S. degree from Inha University in 2009, and received M.S. degree in electrical and computer engineering at Ulsan National Institute of Science and Technology (UNIST) in 2012. Currently she is a Ph.D. student at Sungkyunkwan University, Korea. Her research interests are embedded systems, power-aware computing, virtualization and cloud computing.

Jeaho Hwang obtained his BS degree in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 2007, and MS degree in computer science from KAIST in 2009. He is currently a researcher at RTST Co., Ltd. His current research interests include operating systems, system reliability, virtualization, and embedded systems.

Euiseong Seo received his BS, MS, and PhD degree in computer science from KAIST in 2000, 2002, and 2007, respectively. He is currently an assistant professor in College of ICE at Sungkyunkwan University, Korea. Before joining Sungkyunkwan University in 2012, he had been an assistant professor at UNIST, Korea from 2009 to 2012, and a research associate at the Pennsylvania State University from 2007 to 2009. His research interests comprise power-aware computing, real-time systems, embedded systems, and virtualization.