Necessary and Sufficient Conditions for Thermal Schedulability of Periodic Real-Time Tasks

Rehan Ahmed Parameswaran Ramanathan Kewal K. Saluja
Department of Electrical and Computer Engineering
University of Wisconsin Madison, Madison,WI 53706, USA
rahmed3@wisc.edu, parmesh@ece.wisc.edu, saluja@ece.wisc.edu

***Abstract*—With growing need to address the thermal issues in modern processing platforms various performance throttling schemes have been proposed in literature (DVFS, clock gating etcetera). In real-time systems such methods are often unacceptable as they can result into potentially catastrophic deadline misses. As a result real-time scheduling research has been focused in developing algorithms which meet the compute deadline while satisfying power and thermal constraints. Basic bounds that can determine if a set of tasks can be scheduled or not were established in the 70's based on computation utilization of processing power and no new results have been forthcoming that deal with thermal effect based bounds. In this paper we address the problem of thermal constraint schedulability of tasks and derive necessary and sufficient conditions for thermal feasibility of periodic tasksets for a uni-core system. We then extend some of these results to multi-core processing environment. We demonstrate the efficacy of our results through extensive simulations.**

I. Introduction

Embedded real-time systems are commonly used to monitor and control the dynamics of the underlying physical processes in many cyber-physical applications. The objectives are to enhance the performance, reliability, and safety of the applications. Examples include modern day automobiles, aerospace applications, smart electric grids, cyber-physical medical applications, and next generation transportation systems. For instance, modern day automobiles are embedded systems with hundreds of processors executing many real-time tasks. The processors in the Powertrain Control Module execute critical real-time tasks that monitor and control key operational parameters such as engine speed and position, fuel injection settings, and the timing of next spark signal. Similarly, other processors in the vehicle may execute non-critical tasks such as those in the car's entertainment system. Both types of tasks usually have stringent real-time deadline constraints by which they must complete their execution. Failure to meet the deadline constraints may have severe consequences on performance, reliability, and/or safety of the vehicle.

Research over the past four decades has resulted in extensive theory and algorithms for meeting the real-time constraints of such embedded applications. The theory and the algorithms deal with different processing models (uni-core, multi-core, or multiprocessor), different application models (preemptive, non-preemptive, tasks with resource constraints, etc.), and different guarantee needs (hard, firm, or soft) of the real-time constraints. However, certain important emerging challenges of modern day processors are not well-addressed by these extensive theory and algorithms.

TABLE I: Recent papers on real-time applications in RTSS, RTAS and ECRTS conferences considering power and thermal constraints

Model		Constraints	
Processor	Application	Ther. Only	Pow. & Ther.
Single	Periodic	[1] [2] [3] [5] [6] [7]	[4]
Single	Periodic Aperiodic	[8]	
Multi	Periodic	[9] [10] [11] [14] [15]	[12] [13]
Multi	Periodic Sporadic	[16]	[17] [18]

One such emerging challenge is the need for thermal management in next generation processing systems. As technology scales, on-chip power consumption and power density are increasing rapidly. International Technology Roadmap for Semiconductors (ITRS) predicts that power densities will more than double over the next decade. As power consumption and density increase, their on-chip spatio-temporal variations will create thermal hot spots that are detrimental to both processor performance and reliability. For example, studies have shown that a 10-15°C increase in temperature can more than double the probability of failure of the underlying semiconductor device. In addition, increases in on-chip temperature, increases device leakage currents, which increases on-chip power consumption, which in turn, increases the temperature. This positive feedback between temperate and leakage currents can cause "thermal runaways" that are extremely detrimental to processor performance and reliability. The challenge, therefore, is to schedule and complete the application tasks in such a way that the peak on-chip temperature stays below a predetermined threshold.

Some the recent works that deal with power and thermal issues are categorized in Table I. The scope of these works and their limitations are discussed in Section II. Broadly speaking most of the exiting literature, even those which address power and thermal issues, assume a very simple constant power model and they rely on speed scaling to manage temperature.

In this work, we propose a necessary and sufficient condition for thermal feasibility of periodic tasksets. Furthermore, we also present a constructive algorithm for scheduling periodic tasks such that the timing constraints of periodic tasks are met and the thermal constraints of the processor are not

violated. We also extend the proposed concepts to multi-core processing environment. To summarize, this work has following vital contributions:

1) Using our recently introduced concepts of *Total Thermal Impact* and *Thermal Utilization* [4], we derive simple necessary and sufficient conditions for thermal feasibility of periodic real-time tasksets on uni-core processors.
2) Our derivation of sufficient conditions for thermal schedulability includes a constructive approach for scheduling of periodic real-time tasks such that the peak processor temperature is minimized in uni-core systems.
3) We extend the concept of *Total Thermal Impact* and *Thermal Utilization* to account for thermal coupling between the processing cores in multi-core environments. We exploit this concept to propose an optimization based task allocation approach and an associated scheduling algorithm (TrUMPS) to minimize the peak temperature among the multiple cores.
4) For a given periodic taskset, we derive a lower bound for the peak temperature among the multiple cores over all possible allocations and scheduling.

We empirically evaluate the temperatures obtained from our multi-core allocation and scheduling approach to the lower bound on peak temperature. The empirical evaluation shows that, for more than 90% of the $> 10,000$ tasksets we considered, the difference between the peak temperature in our scheme and the lower bound is less than 1°C. This empirically demonstrates that the proposed approach for thermal-aware multi-core scheduling has excellent performance.

This paper is organized as follows: Section II covers the related research in the scheduling of real-time tasks under thermal constraints. Section III covers the system and task model used in this work. This is followed by the concept of Thermal Impact/Utilization [4] in Section IV. We also prove that Generalized Processor Sharing (GPS) is thermally optimal for scheduling periodic tasks in uni-core systems in Section IV. We then present the analysis/validation of theoretical results for uni-core systems in Section V. The concepts developed for uni-core systems are extended in Section VI to a multi-core platform. Results for multi-core are presented in Section VII followed by conclusion in Section VIII.

II. Related Research

In this section we discuss the research work dealing with thermal aware scheduling and its limitations. As mentioned before Table I succinctly categorizes recent work from literature on thermal-aware scheduling in real-time systems. In particular, the table categorizes papers published in the years 2004–2013 in three premier conferences (Real-time Systems Symposium, Real-Time Technology and Applications Symposium, and Euromicro Conference on Real-Time Systems) based on assumed models.

Most of the existing works in uni-core context rely on scaling down the processor speed (through a combination of voltage and frequency scaling) to reduce its power consumption and thereby temperature. They can be broadly subdivided into either reactive schemes [3] [6] or proactive schemes [1] [20] [5]. Reactive schemes reduce processor speed only when temperature gets above a certain threshold. Wang et al. perform schedulability analysis for a two speed reactive speed scaling scheme in [3]. They also perform delay analysis in [6]. Their scheme is improved upon by Chen et al. [1] by using proactive speed scheduling. In proactive schemes, instead of reacting to a thermal trigger, the speed is set *proactively*, based on the current processor temperature. Chen et al. also derive approximation bounds on the maximum temperature for Earliest Deadline First (EDF) schedule compared to a thermally optimal schedule in [12]. The 2-speed reactive solution is also improved upon by Chaturvedi et al. [21] who use an m-oscillating scheme to schedule periodic tasks. The processor speed oscillates between m levels to keep the system temperature below threshold. Thermally constrained scheduling of periodic tasks is also studied by Quan et al. [20] [5]. They derive thermal feasibility checks and also formulate constructive speed scheduling algorithms for periodic tasks under thermal constraints.

Among existing schemes that consider multi-core model, Chantem et al. [22] [23] use integer linear programming to minimize the maximum temperature for a given set of real-time tasks. They use an equivalent circuit model to estimate the transient core temperatures. Fisher et al. [16] use the thermal characteristics of processing cores to derive preferred speeds for processing elements in a homogeneous multi-core system to minimize system temperature. Hung et al. [24] focus on real-time task scheduling for processing elements without dynamic voltage scaling capability to target maximal power saving in a heterogeneous multi-core system. Liu et al. [25] propose a two-stage approach, in which they first utilize the retiming and software pipelining technique to transform a periodic dependent task graph into a set of independent tasks, and then iteratively adjust the task schedule and voltage selection by incorporating the dynamic voltage scaling and dynamic power management techniques.

In [10] [11], Schor et al. perform worst case temperature analysis for real-time tasks running on multi-core systems. They use a more generalized task model by using the concepts of real-time calculus [26] to model arrivals and computation demands of real-time tasks. The worst case computation methodology is further used in [14] to perform task and frequency assignment for real-time tasks executing on multi-core systems. The problem is formulated and solved as a binary optimization problem. Kumar et al. [7] perform worst case delay analysis on a stream of jobs following arrival and computation patterns based on real-time calculus. In their work, Hettiarachchi et al. [17] propose a new resiliency parameter for real-time systems with thermal constraints. Their work focuses on the design of real-time systems that operate in unpredictable thermal environments where the ambient temperature can change. This work is extended to multi-core processing environment in [18]. In [13], Yun et al. consider scheduling of periodic tasks on a multi-core system under soft thermal constraints. They design a proactive peak temperature

manager which periodically estimates peak processor temperature. It uses machine learning to predict high temperature on a given core. If a core is predicted to overheat, dynamic power management techniques are applied to cool down the core without violating task timing constraints. Kumar et al. [9] address the problem of minimizing end-to-end delay on a set of real-time tasks with inter-dependencies; on a thermally constrained distributed system. They prove that the delay is minimized if all tasks are executed *as soon as possible* under the system thermal constraints.

Specifically, all of the mentioned works have the following primary limitations which are addressed by this work:

1) Most of them which deal with thermal aware scheduling consider dynamic power consumption to be only a function of processor speed. Therefore, it is assumed that power consumption is independent of the task executing on the system. This is clearly a restrictive power model, considering that different applications exhibit vastly different power characteristics [27]. Therefore, to guarantee *hard* thermal constraints, the schemes which consider no power variation across tasks have to schedule tasks based on the *worst-case* power/thermal behavior; which may be very pessimistic. In contrast, this work assumes that dynamic power consumption is a function of both (a) the task executing on the system and (b) processor speed.
2) Except for [21] [22] [23], all of the schemes rely on speed scaling to manage temperature. Therefore, their implementation is restricted to systems which support speed scaling and cannot be applied to systems which do not have this support. The scheduling scheme proposed in this work does not have this reliance on speed scaling.
3) None of the existing works have ability to determine schedulability of a taskset a priory, without alluding to finding a schedule, i.e. they lack the necessary and sufficient conditions for thermal schedulability of periodic tasksets.

III. System Model and Problem Definition

a) Task Model: The real-time application is a set of n periodic, preemptive, independent tasks. Let Γ denote the set of tasks in the application. Each task τ is characterized by three parameters: (i) worst-case computation time C_τ, (ii) period T_τ, and worst-case computational activity a_τ. Furthermore, all tasks are assumed to be in-phase. Each task instance also has a hard deadline equal to the arrival time of the next instance of the corresponding periodic task. Due to the computational activity of a task, the processor consumes a certain amount of dynamic power whenever a corresponding task instance executes. The dynamic power consumed by a task instance is assumed to be proportional to its worst-case computational activity. In general, the dynamic power consumed by a task instance is also a strong function (typically cubic) of the speed of the processing core at which the execution occurs. There are many solutions in literature, including some by the authors of this paper [4], that optimize the assignment of processing speeds to tasks and/or task instances to reduce its dynamic power consumption without jeopardizing its ability to meet its associated deadlines (see Section II for a brief review of these solutions). In this paper, we do not focus on speed assignment and assume that one of the existing solutions in literature is used to assign a speed s_τ for each task. The solution proposed in this paper works in conjunction with the speed scaling solution. For simplicity of presentation, we assume that the value of C_τ is the worst-case computation time of the task at its assigned speed s_τ and its worst-case dynamic power consumption is $P_\tau = a_\tau s_\tau^3$. As mentioned earlier, the tasks may differ in their dynamic power consumption even if they are executing at the same processor speed because of the difference in their computational activities. This is distinguishing aspect of our solution approach.

b) Processor Model: The processor has a set M of m identical cores for some $m \geq 1$. Some of the results in this paper are specific to uni-core system, i.e, $m = 1$. However, most results in this paper are applicable and effective on multi-core systems, i.e, where $m > 1$. We define $\widehat{\Delta}$ as the thermal constraint of the processor. The temperature of each core m has to be less than $\widehat{\Delta}$ for safe operation. The power consumed by a core has two components: (i) dynamic power due to computational activities of a task, and (ii) leakage power due to leakage currents. Studies have shown that leakage currents are a strong function of temperature. However, within a typical operating temperature of a processor, power due to leakage currents can be adequately modeled as a linear function of temperature [28]. As a result, if task τ is executing on core i at time t, then the total power consumed by core i at time t can be written as

$$P_{\text{tot},i}(t, \Psi(t)_i) = P_\tau + \delta\Psi(t)_i + \rho, \tag{1}$$

where P_τ is the dynamic power due to computational activities of task τ, $\Psi(t)_i$ is the temperature of core i at time t and where ρ and δ are modeling constants.

c) Thermal Model: The power consumption on a core causes its temperature to increase. Due to thermal coupling between them, the power consumption on a core also causes temperature increases on other cores of a multi-core system. Studies have shown that thermal effects of power consumption can be modeled as linear resistance-capacitance (RC) circuits [29] [16] [23]. Accurate thermal models of multi-core processors typically include multiple discrete thermal elements such as processing core, thermal interfaces, heat spreaders, and heatsinks. Using the well-known Fourier law, thermal behavior of a processor can be written as set of differential equations, which in turn can be succinctly rewritten in matrix form as follows.

$$\boldsymbol{C}^\top \boldsymbol{\Psi}(t)' = \boldsymbol{P}(t) - \boldsymbol{A}\boldsymbol{\Psi}(t), \tag{2}$$

where $\boldsymbol{C}$ is a vector characterizing the thermal capacitances of different elements in the model, $\boldsymbol{\Psi}(t)$ is a vector of temperature at time t of all the thermal elements in the model, $\boldsymbol{P}(t)$ is a vector of total power consumption of the different elements in the model at time t, and $\boldsymbol{A}$ is matrix which can

be constructed using the thermal conductances between the different elements in the model.

Let $\mathbf{\Psi}_{\text{idle,ss}}$ denote a vector of steady state temperatures when all cores are idle, i.e., when the dynamic power consumption on every core is zero. Define a new variable $\mathbf{\Theta}(t) = \mathbf{\Psi}(t) - \mathbf{\Psi}_{\text{idle,ss}}$. In other words, $\mathbf{\Theta}(t)$ is a column vector of temperature at time t of all the thermal elements in the model expressed as a relative increase with respect to the idle steady-state temperatures. Following a few algebraic simplifications, one can show that

$$\boldsymbol{C}^{\top}\mathbf{\Theta}(t)' = \boldsymbol{P_{dyn}}(t) - (\boldsymbol{A} - \delta\boldsymbol{I})\mathbf{\Theta}(t). \tag{3}$$

Note that, the above equation only contains dynamic power consumptions. Leakage power is accounted for by expressing temperature as relative increase with respect to idle steady-state temperatures and subtracting δ from diagonals of A matrix. For simplicity of presentation, the rest of this paper assumes that all temperatures are expressed relative to the idle steady-state temperatures. The thermal constraint of the processor relative to steady-state temperature is a vector $\mathbf{\Delta}$ such that:

$$\mathbf{\Delta} = \widehat{\Delta} \cdot \mathbf{1} - \mathbf{\Psi}_{\text{idle,ss}}$$

where $\mathbf{1}$ is a column vector in which all entrees are 1.

d) Problem Definition.: Given a periodic taskset Γ and a processing system with m cores, the problem is to first partition Γ into m sets $\Gamma_1, \ldots, \Gamma_m$ and then schedule the task instances of Γ_i on core i, $i \in M$, such that: (i) the deadline constraints of all tasks are satisfied, and (ii) the temperature of each core i is less than or equal to its corresponding threshold temperature Δ_i at all times.

Note that, in this problem definition, all instances of a task are executed on the same core. This is often referred to as the *No migration* model [30].

IV. Uni-Core: Thermal Schedulability

Definition 1: Suppose that a single instance of a unit power task executes for one time unit on a uni-core processor starting at an initial reference temperature of 0°C. Its power consumption will cause the temperature of the core to increase. Let $\theta(t)$ denote the temperature of the core at time t as a result of this execution. Then, *Unit Thermal Impact* (UTI) of the core is defined as

$$\zeta = \int_0^{\infty} \theta(t)dt.$$

Note that, due to the linearity of the system, if a single instance of a task with power P executes for C time units on the core, then its *Total Thermal Impact* on the core will be $P \cdot C \cdot \zeta$. Similarly, due to linearity of the system, it follows from superposition principle that total thermal impact of executing more than one instance is the sum of the total thermal impacts of each instance.

Also, it is proved in [4], [19], [20] that if we repeatedly execute a periodic schedule Π, the temperature at the start of the hyperperiod increases monotonically until the temperature at the start and end of hyperperiod are approximately equal. This scenario where the temperature at the start of hyperperiod stops increasing is called *thermal steady state*. We call this temperature Θ_{Π}.

Theorem 1 (Theorem 1 from [19]): Let a periodic taskset be Γ and schedule be $\Pi(t)$. Suppose that all tasks are executed as per $\Pi(t)$ and $\Theta(0) = \Theta_{\Pi}$, then

$$\int_0^L \Theta(t) = L \cdot \zeta \sum_{\tau \in \Gamma} \frac{P_\tau C_\tau}{T_\tau}, \tag{4}$$

where L is the hyperperiod.
Note that, the value of the integral in the above theorem is only a function of the task characteristics and UTI of the processor. In particular, it is independent of the schedule Π. We call the value of integral in Equation 4 the Hyperperiod Thermal Impact of periodic taskset Γ (HTI(Γ)).

Theorem 2: [Theorem 3 from [19]] Consider a periodic taskset Γ with schedule Π starting at an initial temperature $\Theta(0) = \Theta_{\Pi}$. Let $\phi(\Pi)$ be the corresponding maximum temperature during the periodic schedule, i.e., $\phi(\Pi) = \max_{0 \le t \le L}(\Theta(t))$. Then,

$$\phi(\Pi) \ge \zeta \sum_{\tau \in \Gamma} \frac{P_\tau C_\tau}{T_\tau}. \tag{5}$$

This theorem specifies a lower bound on the maximum temperature of periodic taskset Γ at thermal steady state.

Based on this lower bound, we now define *Thermal Utilization* of a taskset Γ as follows.

$$\Upsilon(\Gamma) = \frac{\zeta}{\Delta} \sum_{\tau \in \Gamma} \frac{P_\tau C_\tau}{T_\tau}. \tag{6}$$

It therefore follows that $\Upsilon(\Gamma)$ must be less than or equal to 1 in order for a periodic taskset Γ to have a thermally feasible schedule. Otherwise, a lower bound on the maximum core temperature for any schedule of this taskset will be greater than Δ, which in turn violates the thermal constraint of the processor. In other words, if the thermal utilization of a periodic taskset is greater than 1, there is no feasible schedule which meets the thermal constraint of the processor. In this paper, we show that if computational and thermal utilizations of a taskset are ≤ 1, then there exists a thermal and timing feasible schedule for Γ on a uni-core processor.

Theorem 3: The lower bound on maximum temperature for periodic taskset Γ at thermal steady state, identified by Theorem 2, is achieved if and only if the processor temperature is constant throughout hyperperiod and is equal to $\frac{\text{HTI}(\Gamma)}{L}$
Proof: We prove this by contradiction.
Case1 $\Theta(t_1) < \frac{\text{HTI}(\Gamma)}{L}$**:** If the processor temperature at time t_1, $(\Theta(t_1)) < \frac{\text{HTI}(\Gamma)}{L}$, the processor temperature at some other time t_2 will have to be greater than $\frac{\text{HTI}(\Gamma)}{L}$ for the integral of processor temperature during hyperperiod to equal $\text{HTI}(\Gamma)$. In this case, we do not achieve the lower bound on temperature since $\Theta(t_2) > \frac{\text{HTI}(\Gamma)}{L}$.
Case2: $\Theta(t_1) > \frac{\text{HTI}(\Gamma)}{L}$**:** If the processor temperature at time t_1, $\Theta(t_1) > \frac{\text{HTI}(\Gamma)}{L}$, then we do not achieve the lower

bound on temperature because of temperature at t_1. Therefore, processor temperature has to be constant at $\frac{\text{HTI}(\Gamma)}{L}$ for us to achieve lower bound on maximum temperature for periodic taskset Γ at thermal steady state.

Corollary 1: The lower bound on maximum temperature for periodic taskset Γ, is achieved if and only if the processor power consumption is constant throughout hyperperiod and is equal to $\sum_{\tau\in\Gamma}\frac{C_\tau P_\tau}{T_\tau}$

Proof: From Theorem 3, the lower bound on maximum temperature is achieved for periodic taskset Γ if temperature is constant at $\frac{\text{HTI}(\Gamma)}{L} = \zeta\cdot\sum_{\tau\in\Gamma}\frac{C_\tau P_\tau}{T_\tau}$. This constant temperature can only be achieved with constant power consumption; since any variation in power consumption would cause variation in temperature. The energy consumption during one hyperperiod for periodic taskset Γ can be computed as $L\cdot\sum_{\tau\in\Gamma}\frac{C_\tau P_\tau}{T_\tau}$. Therefore, to achieve lower bound on maximum temperature, power consumption must be constant at $\sum_{\tau\in\Gamma}\frac{C_\tau P_\tau}{T_\tau}$.

We now prove that a scheduling scheme inspired by Generalized Processor Sharing (GPS) [31] achieves this lower bound on temperature. Generalized Processing Sharing is a service policy for entities sharing a common resource. In the context of this paper, the different entities sharing a common resource are the individual periodic tasks in periodic taskset Γ. The shared resource in this case is the core itself with a resource capacity of 1.

Lemma 1: If computational utilization of a periodic taskset $\Gamma \leq 1$ and GPS algorithm with weight for task $\tau\in\Gamma$ equal to C_τ/P_τ is used for scheduling, all periodic task instances of task τ will finish exactly at their absolute deadlines.

Proof: For a periodic task τ, GPS guarantees a rate of execution equal to its weight ($\frac{C_\tau}{T_\tau}$) if the sum of all weights in the system is less than system capacity ($\sum_{\tau\in\Gamma}\frac{C_\tau}{T_\tau}\leq 1$). Since the computation utilization of taskset $\Gamma\leq 1$, GPS will guarantee execution rate of exactly $\frac{C_\tau}{T_\tau}\ \forall\tau\in\Gamma$. Now suppose that we have a periodic task instance j of task τ with arrival time t_1 and deadline t_1+T_τ. By definition, GPS will execute j for exactly $\frac{C_\tau}{T_\tau}(t_2-t_1)$ where $t_1<t_2\leq t_1+T_\tau$. For a periodic task instance to complete, it needs to execute for C_τ units of time. At time $t+T_\tau$, GPS will schedule instance j for exactly $\frac{C_\tau}{T_\tau}((T_\tau+t_1)-t_1)=C_\tau$ time units. Therefore, instance j will complete exactly at its deadline. Also, since the next instance of periodic task τ arrives at time t_1+T_τ, periodic task τ is always *Backlogged*.

Theorem 4: If computational and thermal utilization of a periodic taskset Γ are ≤ 1, then Generalized Processor Sharing (GPS) algorithm with weight for task τ equal to C_τ/P_τ finds a schedule for Γ that meets all deadline constraints and keeps the core temperature at or below threshold Δ.

Proof: By Lemma 1, there will be no timing violations if periodic taskset Γ is scheduled using GPS and computation utilization of $\Gamma\leq 1$. Now, consider any time interval $(t,t+h)\in[0,L)$. By Lemma 1, we know all all periodic tasks are always backlogged. Therefore, by definition GPS will ensure that each task $\tau\in\Gamma$ executes for $h\cdot C_\tau/T_\tau$ time units in this interval. The average power of all task instances executed in this interval is $\sum_{\tau\in\Gamma}\frac{C_\tau P_\tau}{T_\tau}$. Now if we make the length of this interval infinitesimally small (i.e. $h\to 0$), we can consider the power consumption constant at $\sum_{\tau\in\Gamma}\frac{C_\tau P_\tau}{T_\tau}$ throughout the hyperperiod. By Corollary 1, a constant power of $\sum_{\tau\in\Gamma}\frac{C_\tau P_\tau}{T_\tau}$ achieves the lower bound on maximum temperature for periodic taskset Γ. This constant temperature is equal $\frac{\text{HTI}(\Gamma)}{L}=\zeta\sum_{\tau\in\Gamma}\frac{C_\tau P_\tau}{T_\tau}$ (Theorem 3). Therefore, if Thermal utilization of periodic taskset $\Gamma\leq 1$, i.e. ($\frac{\zeta}{\Delta}\sum_{\tau\in\Gamma}\frac{P_\tau C_\tau}{T_\tau}\leq 1$), maximum temperature given by GPS schedule $\leq\Delta$; hence the theorem.

Worst Case Fair Weighted Fair Queuing (WF2Q): WF2Q [32] is a scheduling scheme which approximates GPS when the system is not fluid (processor is not infinitely preemptible). We use this scheme in the analysis of results presented in section V. In the context of this work, WF2Q will assign each task τ a weight of $\frac{C_\tau}{T_\tau}$. WF2Q makes scheduling decisions for contiguous time periods called *Execution Intervals*. If WF2Q is used, the execution done for periodic task τ will have an absolute difference of at most 1 *Execution Interval* from its execution in a GPS schedule [32]. Furthermore, we conjecture that due to periodicity of task arrivals, WF2Q scheduling scheme will not cause any deadline misses if computation utilization <1 and scheduling ties are broken based on deadlines (earlier deadline tasks given higher priority).

V. Uni-Core: Validating Results

In this section, we empirically validate the theoretical results presented in section IV. Several periodic tasksets are synthetically generated with computation utilization varying from 0.6-1.0 and thermal utilization varying from 0.5-1.2. The results in this section compare the thermal feasibility of the schedules given by the GPS based scheduling scheme, against EDF (Earliest Deadline First) schedules for these tasksets. The rest of section will explain the thermal characteristics of the simulated system, how periodic tasksets are generated, and validation results. We also analyze the thermal feasibility of WF2Q schedules with various *Execution Interval* lengths.

A. Validation Setup

The simulations conducted in this section use a single RC thermal model [4]. For single RC model, from Equation 3 we know that the processor temperature is governed by the following equation:

$$\Theta(t)=\Theta(0)e^{-\beta t}+\frac{P}{\beta}(1-e^{-}\beta t)$$

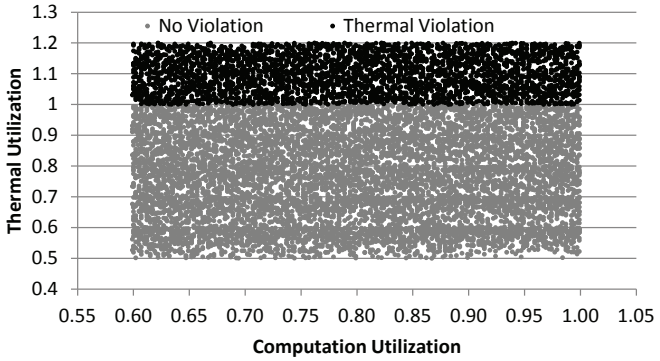

Fig. 1: Computation and thermal utilization of periodic tasksets simulated; and their thermal feasibility with GPS scheduling policy.

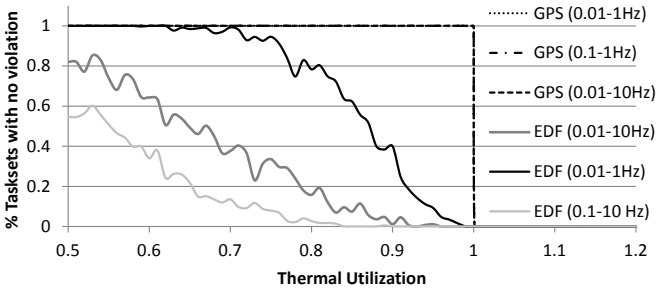

Fig. 2: Fraction of periodic tasksets which are thermally feasible using GPS, and EDF scheduling policies. Different periodic task frequencies simulated.

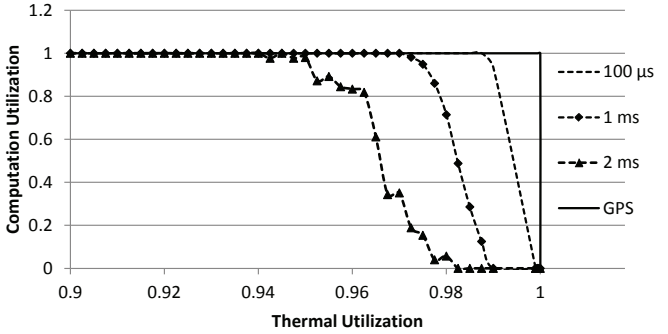

Fig. 3: Fraction of thermally feasible periodic tasksets under WF2Q scheduling policy with various Execution Interval lengths.

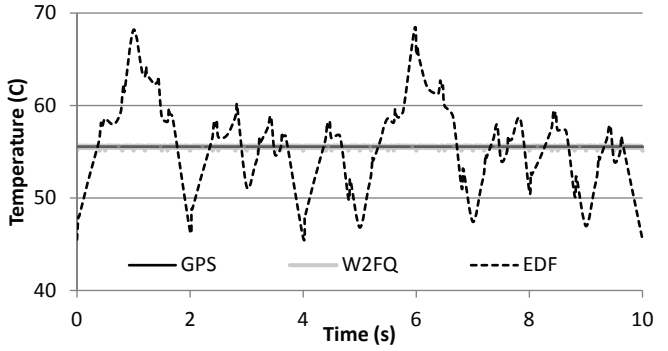

Fig. 4: Processor temperature as a result of schedule given by GPS, WF2Q, and EDF scheduling schemes for a periodic taskset.

where P is the power consumption of the processor between time $[0, t]$ and $\Theta(0)$ is the processor temperature at time 0. Threshold temperature ($\widehat{\Delta}$) is assumed to be 75°C and $\Psi_{\text{idle,ss}}$ is assumed to be 40° C. $\beta = 3.47s^{-}1$ and $\zeta = \frac{1}{\beta} = 0.288$

We use UUniFast algorithm [33] to generate periodic tasksets with computation utilizations varying from 0.6-1.0 in increments of 0.0025. The power consumption of the periodic tasks in a periodic taskset is uniformly distributed between 30W and 250W. To ensure that the generated tasksets are unbiased with respect to low/high thermal utilizations, we divide the thermal utilizations into following ranges: (0.6-0.7], (0.7-0.8], (0.8-0.9], (0.9-1.0], (1.0-1.1], (1.1-1.2]. For each computation utilization and thermal utilization range, we simulate exactly 10 periodic tasksets. This leads to exactly 11270 periodic tasksets simulated (161 computation utilizations $\times$ (7×10) thermal utilization ranges).

Figure 1 shows the computation and thermal utilizations of all periodic tasksets. It shows the periodic tasksets which did not have any thermal violations using GPS, as gray dots. The periodic tasksets which had thermal violations are shown as black dots. As Theorem 4 states, a thermal utilization of ≤ 1 is both necessary and sufficient condition for thermal feasibility of a periodic taskset. Therefore, all periodic tasksets with thermal utilization ≤ 1 have no thermal violations. Figure 1 also shows that all periodic tasksets with thermal utilization > 1 violated thermal constraint.

Figure 2 shows the fraction of periodic tasksets which were accepted for each thermal utilization. The figure shows the Idealized GPS results (All periodic tasksets with thermal utilization ≤ 1 do not cause thermal violations to occur). Figure 2 also shows the acceptance curves for Earliest Deadline First scheduling algorithm. We conducted EDF simulations with three different sets of periodic tasksets. In the first set of results, the computation times and periods of all periodic tasksets are time scaled such that periodic tasks have frequencies varying between [0.01Hz, 1Hz]. For the second set of results, the time scaling factors are kept such that all periodic tasks have frequencies varying between [0.1Hz, 10Hz]. For the third set of results, periodic tasks have frequencies varying between [0.01Hz, 10Hz]. As Figure 2 shows, periodic tasksets with high frequency periodic tasks have a higher thermal acceptance ratio as compared to low frequency periodic tasksets when EDF is used as a scheduling algorithm. This is because, when periodic tasks have low frequency, it is likely that EDF will schedule a high power task for a large time duration; causing the processor temperature to exceed threshold temperature. However, when periodic tasks have high frequency, periodic tasks will be preempted more often; making the execution of high power tasks for large duration of time less likely. Therefore, high frequency periodic tasks have a higher acceptance ratio as compared to low frequency periodic tasks. On the contrary, the acceptance ratio of GPS scheduling strategy does not depend on the frequency of periodic tasks. The acceptance ratio for GPS for all periodic tasksets with thermal utilizations $\leq$1 is 1 irrespective of frequency.

We now compare the acceptance ratios of periodic tasksets with thermal utilizations > 0.9 and with various lengths of WF2Q Execution Intervals. We conducted simulations for

execution intervals of length $100\mu s$, $1ms$, and $2ms$. The results are shown in Figure 3. It is evident from the figure that the acceptance ratio is higher for WF2Q with shorter execution intervals. This is because WF2Q with a short execution interval emulates GPS more closely compares to WF2Q with a long scheduling interval. Figure 4 shows the temporal variation in temperature when GPS, WF2Q and EDF schedules for a periodic taskset are executed on a core. As shown in the figure, EDF schedule has the highest variation in temperature. It also has the highest maximum temperature among all scheduling schemes ($68.32°$C). The curves for GPS and WF2Q schedules are almost overlapping; with temperature remaining almost constant at around $55.5°$ C. The execution interval for WF2Q schedule result shown in Figure 4 is $1ms$.

VI. Multi-Core: Thermal Schedulability

We now extend the presented concepts to a multi-core processing environment. As stated previously all instances of a task are executed on the same core (*No migration* model). Let Λ denote an assignment of of periodic tasks in Γ to processing cores in M. In this section, we first extend the theoretical foundation presented in section IV to multi-core environment. We then present our task assignment and scheduling approach followed by simulation results.

A. Theoretical Results

In a multi-core environment, the power consumption on a given core j results in temperature increase of all cores $i \in M$. Suppose that a single instance of unit power executes for unit time on core j. Also suppose that the initial reference temperature of all cores is 0. Let $\theta(t)_{i,j}$ be the temperature of core i at time t as a result of this execution. Then, we redefine Unit Thermal Impact for multi-core ($\boldsymbol{\zeta}$) as an $m \times m$ matrix where $m = |M|$ and the element $\zeta_{i,j}$ of this matrix is such that:

$$\zeta_{i,j} = \int_0^\infty \theta(t)_{i,j} dt \tag{7}$$

As was the case for uni-core, due to the linearity of the system, if a single instance of a task with power P executes for C time units on the core j, then its *Total Thermal Impact* will be a vector equal to $PC \cdot \boldsymbol{\zeta} \boldsymbol{e}_j$ where $\boldsymbol{e}_j$ is the j^{th} unit vector. Similarly, due to linearity of the system, it follows from superposition principle that total thermal impact of executing more than one instance is the sum of the total thermal impacts of each instance.

If we repeatedly execute a periodic schedule Π, the temperature of all cores at the start of the hyperperiod will converge to a vector $\boldsymbol{\Theta}_{\boldsymbol{\Pi}}$ such that the processor temperature at the end of hyperperiod is also almost equal to $\boldsymbol{\Theta}_{\boldsymbol{\Pi}}$. This scenario where the temperature at the start of hyperperiod approaches temperature at the end of hyperperiod, as before, is called *thermal steady state*.

Theorem 5: Consider a periodic taskset Γ with assignment Λ and schedule $\Pi(t)$. Also suppose that $\boldsymbol{\Theta}(0) = \boldsymbol{\Theta}_{\boldsymbol{\Pi}}$ and assignment Λ is such that all instances of periodic task τ execute on core j in schedule $\Pi(t)$, then

$$\int_0^L \boldsymbol{\Theta}(t) dt = L \cdot \sum_{\tau \in \Gamma} \frac{P_\tau C_\tau}{T_\tau} \boldsymbol{\zeta} \boldsymbol{e}_j \tag{8}$$

This theorem is an extension of Theorem 1 and follows from the definition of $\boldsymbol{\zeta}$ matrix and the linearity of RC thermal model. We call the value on integral in Equation 8 the Hyperperiod Thermal Impact of periodic taskset Γ with assignment Λ ($\mathbf{HTI}(\Gamma, \Lambda)$). Note that ($\mathbf{HTI}(\Gamma, \Lambda)$) is a vector with number of elements equal to the number of cores.

Theorem 6: Consider a periodic taskset Γ with schedule Π starting at an initial temperature $\boldsymbol{\Theta}(0) = \boldsymbol{\Theta}_\Pi$. Let $\boldsymbol{\phi}(\Pi)_i$ be the corresponding maximum temperature of core i during the periodic schedule, i.e., $\phi(\Pi)_i = \max_{0 \le t \le L} (\Theta(t)_i)$. Then,

$$\phi(\Pi)_i \ge \frac{\text{HTI}(\Gamma, \Lambda)_i}{L}. \tag{9}$$

This theorem specifies a lower bound on the maximum temperature of each core $i \in M$ for periodic taskset Γ and assignment Λ. This theorem is a direct extension of Theorem 2 in section IV. Based on this lower bound on temperature, we also define thermal utilization of core $i \in M$ for a given periodic taskset Γ and assignment Λ:

$$\Upsilon(\Gamma, \Lambda)_i = \frac{\text{HTI}(\Gamma, \Lambda)_i}{L \Delta_i} \tag{10}$$

Theorem 7: If thermal utilization of any core $i > 1$, then no scheduling algorithm can meet thermal constraint for i.

This theorem follows from the lower bound on temperature given in Theorem 6 and the definition of $\Upsilon(\Gamma, \Lambda)_i$.

B. Task Thermal Utilization

Since the integral of processor temperature is not constant for a periodic taskset (it also depends on assignment Λ), *Thermal Utilization* of a periodic taskset in multi-core processing environment can not be defined as it was done in section IV for uni-core. For multi-core, the thermal utilization of periodic taskset Γ is defined as:

$$\Upsilon(\Gamma) = \inf_{\Pi} \left\{ \max_{i \in M} \left(\frac{\phi(\Pi)_i}{\Delta_i} \right) \right\} \tag{11}$$

where $\phi(\Pi)_i$ is the maximum temperature of core i when periodic schedule $\Pi(t)$ is executing on the multi-core system and Δ_i is the threshold temperature for core i. i.e. $\Upsilon(\Gamma)$ is the lower bound on $\max_{i \in M} \left(\frac{\phi(\Pi)_i}{\Delta_i} \right)$ for any schedule ($\Pi(t)$) of periodic taskset Γ. Based on this definition, the thermal utilization of ≤ 1 is a necessary condition for thermal feasibility of a periodic taskset.

To evaluate the value of $\Upsilon(\Gamma)$, we solve a linear optimization problem. We first relax the deadline constraints of periodic taskset Γ by transforming it to taskset $\widehat{\Gamma}$. For each task $\tau \in \Gamma$, there is a corresponding task $\hat{\tau} \in \widehat{\Gamma}$ such that: $P_{\hat{\tau}} = P_\tau$, $C_{\hat{\tau}} = \frac{C_\tau L}{T_\tau}$, $T_{\hat{\tau}} = L$.

Note that the amount of computations done for a given periodic task in Γ in one hyperperiod are the same as the

computations done for the corresponding task in $\widehat{\Gamma}$. However, $\widehat{\Gamma}$ does not have any task deadlines within the hyperperiod. We also relax the *No Migration* constraint on the execution of periodic tasks. With these constraints relaxed, we solve the following optimization problem:
Let $x_{\hat{\tau},j}$ be a positive variable which is the amount execution done for periodic task $\hat{\tau} \in \widehat{\Gamma}$ on core $j \in M$ in periodic schedule $\widehat{\Pi(t)}$. Based on the values of $x_{\hat{\tau},j}$, the integral of temperature of core i for one hyperperiod at thermal steady state is given by:

$$\int_0^{\infty} \Theta(t)_i dt = \sum_{\hat{\tau} \in \overline{\Gamma}} \sum_{j \in M} x_{\hat{\tau},j} P_{\hat{\tau}} \zeta_{i,j} \quad \forall i \in M \tag{12}$$

where $\mathbf{\Theta}(0) \approx \mathbf{\Theta}(L)$. Then based on Theorem 6, lower bound on the value of $\frac{\phi(\widehat{\Pi})_i}{\Delta_i}$ is given by

$$\frac{\phi(\widehat{\Pi})_{i,\mathrm{LB}}}{\Delta_i} = \frac{1}{L \cdot \Delta_i} \sum_{\hat{\tau} \in \overline{\Gamma}} \sum_{j \in M} x_{\hat{\tau},j} P_{\hat{\tau}} \zeta_{i,j} \quad \forall i \in M \tag{13}$$

In this formulation, we try to find the optimal values of $x_{\hat{\tau},i}$ variables, such that $\Upsilon_{\max} = \max_{i \in M} \left(\frac{\phi(\widehat{\Pi})_{i,\mathrm{LB}}}{\Delta_i} \right)$ is minimized with the following constraints:

$$\sum_{\hat{\tau} \in \Gamma} x_{\hat{\tau},i} \leq L \qquad \forall i \in M \tag{14}$$

$$\sum_{i \in M} x_{\hat{\tau},i} = C_{\hat{\tau}} \qquad \forall \hat{\tau} \in \widehat{\Gamma} \tag{15}$$

Constraint in Equation 14 ensures that no core is assigned more computation that it is able to complete within the hyperperiod. Constraint in Equation 15 ensures that all tasks in the periodic taskset $\widehat{\Gamma}$ complete within the hyperperiod. Since $\widehat{\Gamma}$ is a relaxation of periodic taskset Γ with all deadline constraints removed, the optimal value of $\Upsilon_{\max}$ is the lower bound of $\inf_{\Pi} \left\{ \max_{i \in M} \left(\frac{\phi(\Pi)_i}{\Delta_i} \right) \right\}$; where $\Pi(t)$ is a schedule of periodic taskset Γ. Therefore: $\Upsilon_{\Gamma} = \Upsilon_{\max}$ in the optimal solution of the LP problem presented in this section. The lower bound on maximum temperature of periodic taskset Γ can be computed as $\Upsilon_{\Gamma} \cdot \Delta_i$ where $i = \arg\max_{i \in M} \left(i : \frac{\phi(\widehat{\Pi})_{i,\mathrm{LB}}}{\Delta_i} \right)$.

C. TrUMPS

In this section, we present our proposed scheduling and assignment approach. We call the proposed solution TrUMPS (**Th**e**r**mal **U**tilization **M**inimization with **P**artitioned **S**cheduling). This is a joint assignment and scheduling approach for periodic tasks executing on a multi-core system.

In our proposed solution, we first find the task to core assignments such that maximum value of core Thermal Utilizations ($\Upsilon(\Gamma, \Lambda)_i$) across all cores is minimized. Once we have the solution for task assignments, we use GPS on individual cores to schedule periodic tasks. GPS ensures that the maximum value of temperature of each core is equal to its thermal utilization times its threshold temperature i.e.($\Upsilon(\Gamma, \Lambda)_i \cdot \Delta_i$). Since we minimize the maximum value of core thermal utilization in the assignment stage of the proposed solution, we conjecture that TrUMPS scheme yields the minimum value of maximum un-adjusted temperature for a given periodic taskset when execution of tasks is constrained by *No Migration* model.

The objective of this optimization problem is to find assignment Λ which minimizes $\Upsilon_{\max} = \max_{m \in M}(\Upsilon(\Gamma, \Lambda)_m)$ while adhering to the following constraint:

$$\sum_{\tau \in \Gamma} \frac{C_\tau}{T_\tau} \boldsymbol{e}_j \leq \mathbf{1} \quad \text{where task } \tau \text{ is assigned to core } j \text{ in } \Lambda \tag{16}$$

Constraint given in Equation 16 ensures that the computation utilization of each individual core is less than 1. Once we find the optimal task-to-core assignments, GPS is used on individual cores to minimize their temperature.

VII. Multi-Core: Experimental Evaluation

A. Multi-core Results

In this section, we conduct simulations for thermal feasibility of periodic tasksets. We conduct experiments on a 3-core system. Several periodic tasksets are simulated with computation utilization varying from 1.5-3.0 and thermal utilization varying from 0.5-1.2. The results in this section compare the maximum temperatures given by the proposed TrUMPS assignment/scheduling strategy against the lower bound on maximum temperature; which is computed as explained in section VI-B.

B. Simulation Setup

We conducted simulations using HOTSPOT [29] to evaluate $\boldsymbol{\zeta}$ for our 3-core floorplan. For our simulations:

$$\boldsymbol{\zeta} = \begin{pmatrix} 0.72225 & 0.156 & 0.156 \\ 0.156 & 0.55375 & 0.16525 \\ 0.156 & 0.16525 & 0.55375 \end{pmatrix}$$

Threshold temperature ($\widehat{\Delta}$) is assumed to be 75° C and $\mathbf{\Psi}_{\mathrm{idle,ss}}$ is assumed to be 40° C for all cores. We use UUniFast Discard algorithm [34] to generate periodic tasksets with computation utilizations varying from 1.5-3.0 in increments of 0.01. The power consumption of the periodic tasks in a periodic taskset is uniformly distributed between 10W and 125W. To ensure that the generated tasksets are unbiased with respect to low/high thermal utilizations, we divide the thermal utilizations into following ranges: (0.6-0.7], (0.7-0.8], (0.8-0.9], (0.9-1.0], (1.0-1.1], (1.1-1.2]. For each computation utilization and thermal utilization, we simulate exactly 10 periodic tasksets. This leads to exactly 10570 periodic tasksets simulated (151 computation utilizations $\times$ (7×10) thermal utilization ranges). Note that some periodic tasksets may not be computationaly feasible even though their computation utilization is < 3 [30]. This is because there may be no possible task to core assignments which result in the computation utilization of each core being

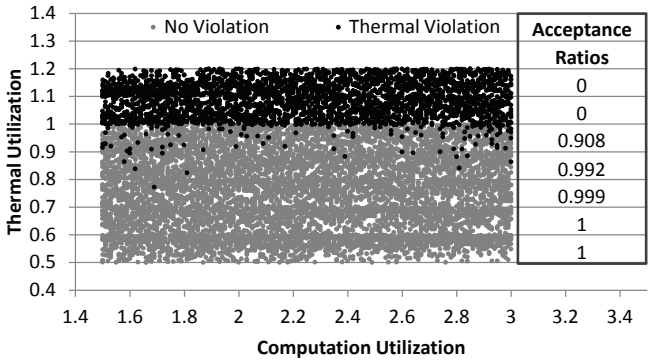

Fig. 5: Computation and thermal utilization of periodic tasksets simulated; and their thermal feasibility using TrUMPS

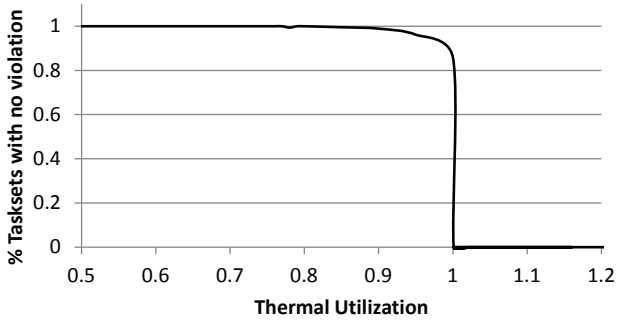

Fig. 6: Fraction of periodic tasksets which are thermally feasible using TrUMPS

≤ 1. In this section, only tasksets which are computationally feasible are included in the results.

C. Simulation Results

Figure 5 shows the computation and thermal utilizations of all periodic tasksets. It shows the periodic tasksets which did not have any thermal violations using GPS, as gray dots. The periodic tasksets which have thermal violations are shown as black dots. Contrary to the results in the uni-core section, having thermal utilization ≤ 1 is not sufficient for the thermal feasibility of periodic taskset. This is because the periodic tasks considered in this section are deadline constrained and their execution is constrained by *No Migration* Model. Both of these restrictions were removed when lower bound on temperature and thermal utilization was computed/defined in section VI-B. Therefore, as Figure 5 shows, there are some periodic tasksets which violate thermal constraints when scheduled using TrUMPS; even though their thermal utilizations are < 1. Furthermore, Figure 5 shows that all periodic tasksets with thermal utilization > 1 violated thermal constraints. This is because having thermal utilization ≤ 1 is a necessary condition for thermal feasibility of periodic taskset. Figure 5 also shows the ratio of thermally feasible periodic tasksets for all 7 thermal utilization ranges.

The likelihood of thermal violation increases as thermal utilization increases. This trend is illustrated in Figure 6 which shows how the fraction of thermally feasible periodic tasksets varies with thermal utilization. As shown in the figure, most periodic tasksets with thermal utilization ≤ 0.9 do not have any thermal violations. However, thermal violations increase

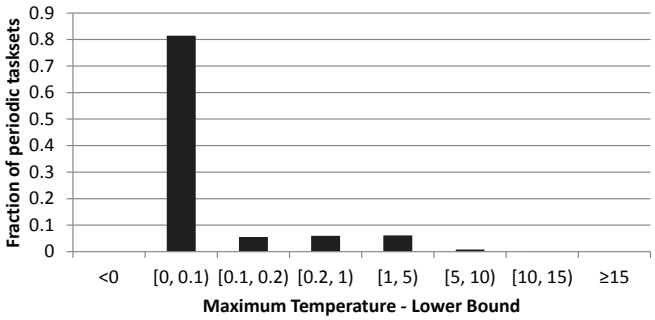

Fig. 7: Difference in maximum temperature given by TrUMPS and the lower bound on temperature

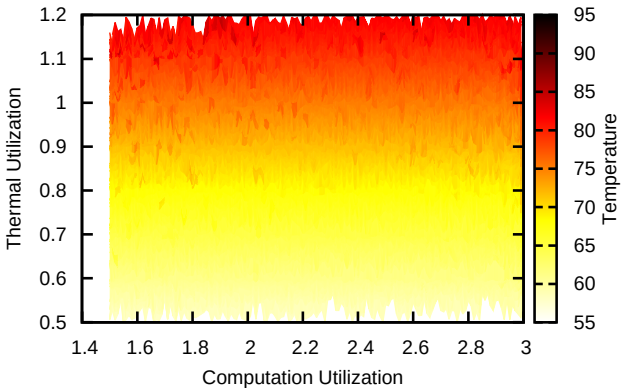

Fig. 8: Maximum temperature given by the TrUMPS for all tasksets

as thermal utilization increases beyond 0.9. Figure 7 shows the difference in maximum temperature given TrUMPS from the lower bound on temperature explained in Section VI-B. As shown in figure, more than 80% of the periodic tasksets have the difference of less than $0.1°$C from the lower bound on temperature. Furthermore, there are very few periodic tasksets with difference greater than $10°$C (7 periodic tasksets out of 10570), and no periodic tasksets with difference > 15. Figure 8 illustrates how the maximum temperature given by TrUMPS varies with the thermal and computation utilizations of periodic tasksets. As shown in the figure, the maximum temperature is not dependent on computation utilization. However, it rises linearly as thermal utilization is increased; which is the expected result.

VIII. Conclusion

In this paper, we make several key contributions on thermal constrained scheduling of periodic tasks. Analogous to the computation utilization bound for schedulability established in 1973 [35], we establish a necessary and sufficient condition for thermal feasibility of periodic tasksets on uni-core systems in Section IV. Furthermore, we prove that a scheduling approach based on Generalized Processor Sharing always yields a thermally feasible schedule if the proposed condition is satisfied (Theorem 4). The detailed analysis in Section V validates our theoretical results.

We also extend the presented concepts for multi-core and propose a necessary condition for thermal feasibility of periodic tasksets in section VI. This involves computation of

lower bound on maximum temperature with any task execution/migration model. Furthermore, we present a partitioned scheduling algorithm (TrUMPS) which minimizes processor temperature when task executions are constrained by *No Migration Model*. Simulations conducted for multi-core show that TrUMPS performs exceptionally well (maximum temperature has a difference of less than 1°C from lower bound for 90% of simulated tasksets). This is especially impressive considering that task executions in TrUMPS are constrained by *No Migration Model*. We believe that we can get closer to the lower bound on temperature with scheduling strategies which use *Task Migration* or *Instance Migration* model. Formulation of such scheduling strategies and and reformulation of thermal feasibility condition for multi-core, such that it is necessary and *sufficient*; is part of ongoing research.

REFERENCES

[1] J.-J. Chen, S. Wang, and L. Thiele, "Proactive speed scheduling for real-time tasks under thermal constraints," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, April 2009, pp. 141–150.

[2] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. Koutsoukos, and H. Wang, "Feedback thermal control for real-time systems," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*. IEEE, Apr. 2010, pp. 111–120.

[3] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," in *Proceedings of the Euromicro Conference on Real-time Systems*, Jul. 2006, pp. 160–170.

[4] R. Ahmed, P. Ramanathan, and K. Saluja, "On thermal utilization of periodic task sets in uni-processor systems," in *RTCSA 2013*. IEEE, 2013.

[5] G. Quan and Y. Zhang, "Leakage Aware Feasibility Analysis for Temperature-Constrained Hard Real-Time Periodic Tasks," in *Proceedings of the Euromicro Conference on Real-time Systems*. IEEE Computer Society, Jul. 2009, pp. 207–216.

[6] S. Wang, Bettati, and Riccardo, "Delay analysis in temperature-constrained hard real-time systems with general task arrivals," in *Proceedings of the Real-time Systems Symposium*, Dec. 2006, pp. 323–334.

[7] P. Kumar and L. Thiele, "Timing analysis on a processor with temperature-controlled speed scaling," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*. IEEE, Apr. 2012, pp. 77–86.

[8] Y. Ahn and R. Bettati, "Transient overclocking for aperiodic task execution in hard real-time systems," in *Proceedings of the Euromicro Conference on Real-time Systems*. Los Alamitos, CA, USA: IEEE Computer Society, Jul. 2008, pp. 102–111.

[9] P. Kumar and L. Thiele, "End-to-end delay minimization in thermally constrained distributed systems," in *Proceedings of the Euromicro Conference on Real-time Systems*. IEEE, July 2011, pp. 81–91.

[10] L. Schor, H. Yang, I. Bacivarov, and L. Thiele, "Worst-case temperature analysis for different resource models," *IET Circuits, Devices & Systems*, vol. 6, no. 5, pp. 297–307, 2012.

[11] L. Schor, I. Bacivarov, H. Yang, and L. Thiele, "Worst-case temperature guarantees for real-time applications on multi-core systems," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*. IEEE, Apr. 2012, pp. 87–96.

[12] J. Chen, C. Hung, and T. Kuo, "On the minimization fo the instantaneous temperature for periodic real-time tasks," in *13th IEEE Real Time and Embedded Technology and Applications Symposium, 2007. RTAS'07*, 2007, pp. 236–248.

[13] B. Yun, K. Shin, and S. Wang, "Predicting thermal behavior for temperature management in time-critical multicore systems," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, Apr. 2013, pp. 185–194.

[14] L. Schor, H. Yang, I. Bacivarov, and L. Thiele, "Thermal-aware task assignment for real-time applications on multi-core systems," in *Formal Methods for Components and Objects*. Springer, 2013, pp. 294–313.

[15] D. Rai, H. Yang, I. Bacivarov, J.-J. Chen, and L. Thiele, "Worst-case temperature analysis for real-time systems," in *Proceedings of Design, Automation and Test in Europe*, march 2011, pp. 1 –6.

[16] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, April 2009, pp. 131–140.

[17] P. hettiarachchi, N. Fisher, M. Ahmed, L. Y. Wang, S. Wang, and W. Shi, "The design and analysis of thermal-resilient hard-real-time systems," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, Apr. 2012.

[18] P. Hettiarachchi, N. Fisher, and L. Wang, "Achieving thermal-resiliency for multicore hard-real-time systems," in *Proceedings of the Euromicro Conference on Real-time Systems*, Jul. 2013, pp. 37–46.

[19] R. Ahmed, P. Ramanathan, K. K. Saluja, and C. Yao, "Scheduling aperiodic tasks in next generation embedded real-time systems," *VLSI Design, International Conference on*, vol. 0, pp. 25–30, 2013.

[20] G. Quan, Y. Zhang, W. Wiles, and P. Pei, "Guaranteed scheduling for repetitive hard real-time tasks under the maximal temperature constraint," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*. ACM, 2008, pp. 267–272.

[21] V. Chaturvedi, H. Huang, and G. Quan, "Leakage aware scheduling on maximum temperature minimization for periodic hard real-time systems," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1802–1809.

[22] T. Chantem, X. S. Hu, and R. P. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 10, pp. 1884 –1897, October 2011.

[23] T. Chantem, R. Dick, and X. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 288–293.

[24] C.-M. Hung, J.-J. Chen, and T.-W. Kuo, "Energy-efficient real-time task scheduling for a dvs system with a non-dvs processing element," in *Proceedings of the Real-time Systems Symposium*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2006, pp. 303–312.

[25] H. Liu, Z. Shao, M. Wang, and P. Chen, "Overhead-aware system-level joint energy and performance optimization for streaming applications on multiprocessor systems-on-chip," in *Proceedings of the Euromicro Conference on Real-time Systems*, July 2008, pp. 92–101.

[26] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *Proceedings of International Symposium on Circuits and Systems*, vol. 4. IEEE, 2000, pp. 101–104.

[27] E. Kursun, C. yong Cher, A. Buyuktosunoglu, and P. Bose, "Investigating the effects of task scheduling on thermal behavior," in *In Third Workshop on Temperature-Aware Computer Systems (TACS06*, 2006.

[28] Y. Liu, R. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *Design, Automation Test in Europe*, 2007, pp. 1–6.

[29] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, no. 5, p. 501, 2006.

[30] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Comput. Surv.*, vol. 43, no. 4, Oct. 2011.

[31] A. Pareh and R. Gallagher, "A generalized processor sharing approach to flow control in integrated services network the single node case," *ACM/IEEE Transactions on Networking*, vol. 1, no. 3, pp. 344–357, Jun. 1993.

[32] J. C. R. Bennett and H. Zhang, "Wf2q: Worst-case fair weighted fair queueing," in *Proceedings of INFOCOM*, vol. 1, 1996, pp. 120–128.

[33] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.

[34] P. Emberson, R. Stafford, and R. I. Davis, "Techniques for the synthesis of multiprocessor tasksets," in *1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2010, pp. 6–11.

[35] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.