Thermal-Aware Global Real-Time Scheduling on Multicore Systems

Nathan Fisher*, Jian-Jia Chen†, Shengquan Wang‡, and Lothar Thiele†

* *Department of Computer Science*
Wayne State University, USA
fishern@cs.wayne.edu

† *Computer Engineering*
and Networks Laboratory (TIK)
ETH Zurich, Switzerland
{*jchen,thiele*}*@tik.ee.ethz.ch*

‡ *Department of Computer*
and Information Science
University of Michigan-Dearborn, USA
shqwang@umd.umich.edu

Abstract

As the power density of modern electronic circuits increases dramatically, systems are prone to overheating. Thermal management has become a prominent issue in system design. This paper explores thermal-aware scheduling for sporadic real-time tasks to minimize the peak temperature in a homogeneous multicore system, in which heat might transfer among some cores. By deriving an ideally preferred *speed for each core, we propose global scheduling algorithms which can exploit the flexibility of multicore platforms at low temperature. Compared with load-balancing strategies, the proposed algorithms can significantly reduce the peak temperature by up to* $30\ ^\circ C$ *to* $70\ ^\circ C$ *for simulated platforms.*

Keywords: Thermal-aware scheduling, Dynamic voltage scaling, Global real-time scheduling, Multicore systems.

1. Introduction

As the power density of modern electronic circuits increases dramatically, systems are prone to overheating. High temperature also reduces system reliability and increases timing errors [30]. Thermal management has become a prominent issue in system design. Techniques for thermal management have been explored both at design time through appropriate packaging and active heat dissipation mechanisms, and at run time through various forms of Dynamic Thermal Management (DTM). The packaging cost of cooling systems grows exponentially [5]. Recent estimates have placed the packaging cost at $1 to $3 per watt of heat dissipated [26]. The techniques to reduce the packaging cost of cooling systems (e.g., the amount of cooling hardware in the system) or reduce the temperature in architectural levels have been studied in [9], [16], [26], [30]. As an alternative solution, the DTM [9], [16], [26] has been proposed to control the temperature at run time by adjusting the system power consumption. Many modern computer architectures provide system designers with such flexibility.

In real-time systems, thermal-aware scheduling aims to maintain safe temperature levels or minimize the peak temperature for processors without violating timing constraints for real-time tasks. For uniprocessor systems, thermal-aware scheduling has been explored to optimize the performance by exploiting the DTM [9], [16], [26] to prevent the system from overheating by adopting Dynamic Voltage Scaling (DVS) [18], [32]. Wang et al. [27]–[29] developed reactive speed control with schedulability tests and delay analysis, while Chen et al. [12] developed proactive speed control to improve the schedulability. Bansal et al. [4] developed an algorithm to maximize the workload that can complete in a specified time window without violating the thermal constraints. Zhang and Chatha [33] provided approximation algorithms to minimize the completion time, while each task is restricted to execute at one speed. Chen et al. [11] showed that the schedule with the minimum energy consumption is an e-approximation algorithm in terms of peak temperature minimization for periodic real-time tasks. Bansal et al. [5] show that Yao's algorithm [32] for real-time jobs is a 20-approximation algorithm for peak temperature minimization.

Thermal-aware multiprocessor scheduling has also been explored recently, e.g., [17], [19], [23], [24]. For multiprocessor real-time scheduling, there are typically two choices of scheduling paradigm: *global* or *partitioned*. In the global scheduling paradigm, a real-time job is permitted to migrate between the processors on the processing platform. In partitioned scheduling, a job is statically assigned to a single processor in the platform and migration is not permitted. A significant portion of prior research in thermal-aware multiprocessor systems has focused on the partitioned scheduling paradigm. For multiprocessor systems without heat transfer among the processors, Chen et al. [11] proved that the largest-task-first strategy (also called worst-fit decreasing [2]) has a constant approximation factor for the minimization of peak temperature. If heat transfer between two cores is taken into account, thermal-aware scheduling of real-time tasks has only limited results. Chantem et al. [10] provided a mixed integer linear programming (MILP) formulation for peak temperature reduction by assuming that the power consumption of a task on a processor is fixed and the heat transfer can be estimated by accumulating the power consumption of the other cores. However, the above thermal-aware scheduling algorithms focus on partitioned scheduling of periodic real-time tasks or a set of job instances without periodicity. Applying partitioned scheduling for real-time tasks in a multicore environment is often too conservative. The focus of this paper is obtaining results for thermal-aware scheduling under the global paradigm.

This paper explores thermal-aware scheduling for sporadic real-time tasks to minimize the peak temperature in a homogeneous multicore system. As heat can transfer among cores and heat sinks, the cooling and heating phenomena is modeled by applying the Fourier's cooling model in the literature [10], [17], [23], [24], in which the thermal parameters can be calculated by the RC thermal model. Although heat transfer is a dynamic process, it is not difficult to see that the temperature on a core is non-decreasing if the execution speed on a core is fixed. Moreover, it will end up with a steady state, in which the temperatures on all cores become steady. We show how to approximately minimize the peak temperature at the steady state. This paper proposes a two-stage approach. In the first stage, we derive the *preferred* speeds for execution to minimize the peak temperature under the necessary schedulability conditions of global scheduling. Then, in the second stage, we derive a proper speedup factor to satisfy the sufficient schedulability conditions of global scheduling. The proposed approach is quite general, and can be adopted for global scheduling algorithms that have both a necessary condition and a sufficient condition for the global schedulability of sporadic tasks, such as the global earliest-deadline-first (EDF) scheduling policy and the global deadline-monotonic (DM) scheduling policy. Furthermore, in our approach, we permit each core to have a potentially different speed than the other cores. To evaluate the effectiveness of the proposed algorithms, we use three multicore benchmarks with 4×1, 2×2, 3×2, and layouts for simulations. Compared with load-balancing strategies, the proposed algorithms can significantly reduce the peak temperature by up to 30 °C to 70 °C for simulated platforms.

The rest of this paper is organized as follows: Section 2 shows the system model and problem definition. Section 3 presents how to derive the preferred speeds of cores for minimizing the peak temperature under the necessary schedulability conditions of global scheduling. Section 4 derives the feasible speed scheduling based on the preferred speeds. Section 5 presents performance evaluation over simulated multicore platforms.

2. System Model and Problem Statement

Thermal model We consider a multicore system, in which each core is a discrete thermal element. In the system, there is a set of heat sinks on top of the cores. Those heat sinks generate no power, and are used only for heat dissipation. Figure 1 is an example layout for 4 cores with 2 heat sinks. Heating or cooling is a complicated dynamic process depending on the physical system. We could approximately model this process by applying Fourier's Law [4], [5], [10], [17], [23], [25], [27]–[29], [33], in which the thermal coefficients can be obtained by using the RC thermal model, such as the approaches in [10], [17], [23], [24]. The thermal model adopted in this paper is similar to the recent approaches in [10], [17], [24].

We define $\mathcal{M} = \{1, 2, 3, \dots, M\}$ as the set of the M cores in the multicore system. Suppose that the thermal conductance between Cores j and ℓ in $\mathcal{M}$ is $G_{j,\ell}$, where $G_{j,\ell} = G_{\ell,j}$. Note that if Cores j and ℓ have no intersection for heat transfer, then $G_{j,\ell} = 0$. We assume $G_{j,j}$ be 0 for any j in $\mathcal{M}$. We assume that the capacitance of Core j in $\mathcal{M}$ is C_j.

We define $\mathcal{H} = \{1, 2, 3, \dots, \hbar\}$ as the set of the $\hbar$ sinks in the multicore system. Suppose that the thermal conductance of a heat sink dissipating heat to the environment is $G^\dagger$. We define $\mathcal{H}_j$ as the set of heat sinks connected to Core j. Suppose that the vertical thermal conductance between Core j and Sink h in $\mathcal{H}_j$ is $H_{j,h}$, which depends on the distance and the linking material. For Sinks h and g in $\mathcal{H}_j$, the horizontal thermal conductance between the sinks is $G_{h,g}$, where $G_{h,g} = G_{g,h}$. If there is no heat dissipation from Core j to Sink h, then $H_{j,h} = 0$. We assume the capacitance of Sink h in $\mathcal{H}$ is C_h.

We define $\Theta_j(t)$ and $\Theta_h(t)$ as the temperature at time instant t on Core j and Sink h, respectively. We assume that the ambient temperature Θ_a is fixed. We also define $\Psi_j(t)$ as the power consumption on Core j at time t. Informally, the rate of change in the temperature on a core is proportional to the power consumption times the quantity of the heating coefficient minus the cooling coefficients times the quantity of the temperature gradients among the core, its neighboring cores, and its heat sinks. The heating/cooling process by Fourier's Law can be formulated as

$$\begin{aligned} C_j \frac{\mathrm{d}\Theta_j(t)}{\mathrm{d}t} =& \Psi_j(t) - \sum\nolimits_{h \in \mathcal{H}} H_{j,h}(\Theta_j(t) - \Theta_h(t)) \\ & - \sum\nolimits_{\ell \in \mathcal{M}} G_{j,\ell}(\Theta_j(t) - \Theta_\ell(t)), \qquad \text{(1a)} \\ C_h \frac{\mathrm{d}\Theta_h(t)}{\mathrm{d}t} =& - G^\dagger(\Theta_h(t) - \Theta_a) \\ & - \sum\nolimits_{j \in \mathcal{M}} H_{j,h} \left(\Theta_h(t) - \Theta_j(t)\right) \\ & - \sum\nolimits_{g \in \mathcal{H}} G_{g,h}(\Theta_h(t) - \Theta_g(t)), \qquad \text{(1b)} \end{aligned}$$

where $\frac{\mathrm{d}\Theta_j(t)}{\mathrm{d}t}$ and $\frac{\mathrm{d}\Theta_h(t)}{\mathrm{d}t}$ are the derivatives of the temperatures on Core j and the heat sink, respectively. All these parameters can be derived by applying the RC thermal model for a given platform, e.g., [10], [17], [24].

Power consumption model We explore thermal-aware scheduling on cores, each with an independent DVS capabilities (referred to as DVS cores). As shown in the literature

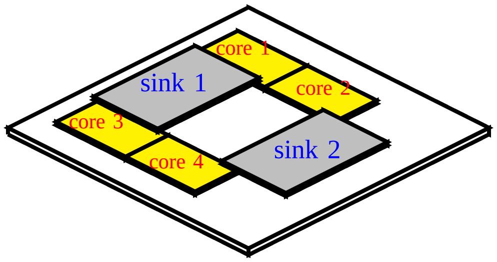


Figure 1: An example for 4 cores.

[1], [10], [18], the power consumption Ψ_j on Core j is contributed by:

- *The dynamic power consumption* $\Psi_{dyn,j}$ mainly resulting from the charging and discharging of gates on the circuits, which can be modeled by $\Psi_{dyn,j} = \alpha s_j^{\gamma}$, where s_j is the execution speed of Core j and both γ (≤ 3) and α are constant.
- *The static power consumption* $\Psi_{sta,j}$ mainly resulting from the leakage current. The static power consumption function is a constant Ω when the leakage power consumption is irrelevant to the temperature [11], [31]. When the leakage power consumption is related to the temperature, it is a super linear function of the temperature [20]. As shown in [10], [21], the static power consumption could be approximately modeled by a linear function of the temperature with roughly 5% error. Hence, the static power consumption in this paper is as follows: $\Psi_{sta,j} = \delta\Theta_j + \Omega$, where Θ_j is the absolute temperature on Core j and both δ and Ω are non-negative constants.

As a result, the following formula is used as the overall power consumption on Core j of speed s_j with temperature Θ_j:

$$\Psi = \Psi_{dyn,j} + \Psi_{sta,j} = \alpha s_j^{\gamma} + \Omega + \delta\Theta_j. \tag{2}$$

Task model In this paper, we consider jobs generated by a *sporadic task system* [22], $\mathbf{T} \overset{\text{def}}{=} \{\tau_1, \tau_2, \ldots, \tau_N\}$. Each sporadic task, τ_i, is characterized by (e_i, d_i, p_i) where e_i is the required execution cycles, d_i is the relative deadline, p_i is the minimum inter-arrival separation parameter (historically, called the period). The interpretation of sporadic task τ_i is that the first job a task τ_i may arrive at any time; however, subsequent job arrivals are separated by at least p_i time units. After every job arrival for task τ_i the processor must execute e_i cycles of the job within d_i time units. If, at any given time t, a job has execution remaining, the job is said to be *active* at time t. For this paper, we assume that $\mathbf{T}$ is a *constrained-deadline* task system; that is, $d_i \leq p_i$ for all $\tau_i \in \mathbf{T}$. Furthermore, we will also assume that tasks are indexed in non-decreasing order of their relative deadline: $d_i \leq d_{i+1}$ for all $1 \leq i < N$.

We define the following metrics on task system workload. The *utilization* of task τ_i is denoted by $u_i \overset{\text{def}}{=} e_i/p_i$. The total system utilization is $u_{\text{sum}}(\mathbf{T}) \overset{\text{def}}{=} \sum_{\tau_i \in \mathbf{T}} u_i$. The *density* of τ_i is denoted by $\delta_i \overset{\text{def}}{=} e_i/\left(\min\left((d_i, p_i)\right)\right)$. The max density (among the first k tasks of $\mathbf{T}$) are respectively defined as:

$$\delta_{\max}(\mathbf{T}, k) \overset{\text{def}}{=} \max_{i=1}^{k}\{\delta_i\}. \tag{3}$$

The *demand-bound function* $\mathsf{dbf}(\tau_i, t)$ quantifies the maximum cumulative execution cycles of τ_i that must execute over any interval of length t. More specifically, $\mathsf{dbf}(\tau_i, t)$ is the maximum cumulative execution of jobs of τ_i that have both arrival times and absolute deadlines in any interval of length t. In [8], it has been shown that for a sporadic task τ_i, the demand-bound function may be computed as follows:

$$\mathsf{dbf}(\tau_i, t) \overset{\text{def}}{=} \max\left(0, \left(\left\lfloor \frac{t - d_i}{p_i} \right\rfloor + 1\right) e_i\right). \tag{4}$$

Using the demand-bound function, we may compute the maximum "load" that first k tasks of $\mathbf{T}$ places upon the processing platform:

$$\mathsf{load}(\mathbf{T}, k) = \max_{t \geq 0}\left\{\frac{\sum_{i=1}^{k} \mathsf{dbf}(\tau_i, t)}{t}\right\}. \tag{5}$$

In general, $\mathsf{load}(\tau, k)$ may be determined exactly in pseudo-polynomial time or approximated to within an arbitrary additive error in polynomial time [14].

Scheduling algorithms Each DVS core on our platform $\mathcal{M}$ is permitted to execute at a potentially different speed than the other cores. The *uniform multiprocessor model* (e.g., see [15]) is a machine-scheduling abstraction which appropriately characterizes DVS multicore processors executing at different speeds. In the uniform multiprocessor model, Core j executes at a rate s_j. Any job (regardless of the generating task) executing upon Core j will complete $s_j \times t$ cycles over any time interval of length t.

For our current work, we consider two priority-driven global scheduling algorithms: EDF and DM. Upon uniform multiprocessor platforms, priority-driven scheduling works by assigning each job a priority and executing, at any time instant, the (at most) M highest-priority active jobs. Furthermore, among the set of at most M highest-priority active jobs, higher-priority jobs are favored over lower-priority jobs, by executing the highest-priority jobs upon the fastest processors. Note that, if there are $a(< M)$ active jobs at time t, then only the a fastest processors execute jobs at time t; the $M - a$ slowest processors are idled at time t. The (global) EDF scheduling algorithm assigns priority to jobs in inverse proportion to their absolute deadline: the earlier a job's deadline the greater its priority. The (global) DM scheduling algorithm assigns priority to each job proportional to the inverse of its relative deadline: the smaller a job's relative deadline the greater its priority. We will summarize some current results concerning global scheduling of sporadic tasks upon uniform multiprocessors in Section 3.2.

Problem definition Given a system $\mathbf{T}$ of sporadic real-time tasks, the *thermal-aware global scheduling* problem is to find an assignment of execution speeds on the multicore system such that all the tasks may complete by their respective deadlines by applying the global scheduling policy (either EDF or DM) and the peak temperature is minimized. This paper obtains an execution-speed assignment approximation algorithm that runs in polynomial time. Without loss of generality, we assume that the initial temperature is equal to the ambient temperature.

3. Deriving Preferred Speeds

This section presents how to derive the preferred speed of each core so that the peak temperature is minimized while the necessary schedulability conditions are satisfied. First, in Section 3.1, we will present how to reformulate the thermal parameters so that we can easily calculate the peak temperature of a speed assignment. Then, in Section 3.2, we will summarize the schedulability conditions of global scheduling in uniform multiprocessor systems, following the derivation of preferred speeds based on the necessary schedulability conditions for global scheduling of sporadic real-time tasks in Section 3.3.

3.1. Thermal Parameters Reformulation

Suppose that Core j is assigned with a constant speed s_j for its execution (and also for idling) all the time. If each core runs at its constant speed, it is clear that the temperature is non-decreasing on each core. Moreover, it will end up with a steady state, in which the temperatures on all cores become steady. Therefore, the peak temperature of Core j is no more than the temperature Θ_j^*, which is the solution to Equation $\frac{\mathrm{d}\Theta_j}{\mathrm{d}t} = 0$. Similarly, we can obtain the peak temperature Θ_h^* of Sink h. By reformulating (1), we know that at the steady state, for all j,

$$\begin{aligned} 0 &= \Psi_j - \sum_{h\in\mathcal{H}} H_{j,h}(\Theta_j^* - \Theta_h^*) - \sum_{\ell\in\mathcal{M}} G_{j,\ell}(\Theta_j^* - \Theta_\ell^*) \\ &= \alpha s_j^\gamma + \Omega + (\delta - \textstyle\sum_{h\in\mathcal{H}} H_{j,h} - \sum_{\ell\in\mathcal{M}} G_{j,\ell})\Theta_j^* \\ &\quad + \textstyle\sum_{h\in\mathcal{H}} H_{j,h}\Theta_h^* + \sum_{\ell\in\mathcal{M}} G_{j,\ell}\Theta_\ell^* \end{aligned}$$

and, for the heat sink h,

$$\begin{aligned} 0 &= -G^\dagger\left(\Theta_h^* - \Theta_a\right) - \textstyle\sum_{j\in\mathcal{M}} H_{j,h}\left(\Theta_h^* - \Theta_j^*\right) \\ &\quad - \textstyle\sum_{g\in\mathcal{H}} G_{g,h}(\Theta_h^* - \Theta_g^*). \end{aligned}$$

We can simplify the above equations by the following notations: for any $1 \le j \ne \ell \le M$ and $1 \le h \ne g \le \hbar$,

$$\begin{aligned} A_{j,j} &= \delta - \textstyle\sum_{h\in\mathcal{H}} H_{j,h} - \sum_{\ell\in\mathcal{M}} G_{j,\ell}, \\ A_{j,\ell} &= G_{j,\ell}, \\ A_{j,M+h} &= A_{M+h,j} = H_{j,h}, \\ A_{M+h,M+h} &= -G^\dagger - \textstyle\sum_{j\in\mathcal{M}} H_{j,h} - \sum_{g\in\mathcal{H}} G_{g,h}, \\ A_{M+h,M+g} &= G_{g,h}. \end{aligned}$$

Then, we know that

$$\begin{pmatrix} A_{1,1} & \cdots & A_{1,\eta} \\ A_{2,1} & \cdots & A_{2,\eta} \\ \vdots & \vdots & \vdots \\ A_{M,1} & \cdots & A_{M,\eta} \\ A_{M+1,1} & \cdots & A_{M+1,\eta} \\ \vdots & \vdots & \vdots \\ A_{\eta,1} & \cdots & A_{\eta,\eta} \end{pmatrix} \begin{pmatrix} \Theta_1^* \\ \Theta_2^* \\ \vdots \\ \Theta_M^* \\ \Theta_{M+1}^* \\ \vdots \\ \Theta_\eta^* \end{pmatrix} = - \begin{pmatrix} \alpha s_1^\gamma + \Omega \\ \alpha s_2^\gamma + \Omega \\ \vdots \\ \alpha s_M^\gamma + \Omega \\ G^\dagger\Theta_a \\ \vdots \\ G^\dagger\Theta_a \end{pmatrix},$$

where η is $M + \hbar$. For notational brevity, let $[\mathcal{A}]$ be the $(M + \hbar)$-dimensional matrix of $A_{j,\ell}$, in which all the elements in matrix $[\mathcal{A}]$ are constants. Let $\vec{\Theta}$ be the vector of the peak temperatures of the cores and the sinks in the above equation. Let $\vec{B}$ be the transposition of the $(M+\hbar)$-dimensional vector $(\overbrace{\Omega, \Omega, \ldots, \Omega}^{M}, \overbrace{G^\dagger\Theta_a, \ldots, G^\dagger\Theta_a}^{\hbar})$. Let $\vec{P}$ be the transportation of the $(M + \hbar)$-dimensional vector of dynamic power consumption on these cores, where the power consumption of the $(M + h)$-th element in $\vec{P}$ is 0 for $1 \le h \le \hbar$.

With these notations, the above equation can be simplified as $[\mathcal{A}]\vec{\Theta} = -\vec{P} - \vec{B}$. Therefore, we have

$$\vec{\Theta} = -[\mathcal{A}]^{-1}(\vec{P} + \vec{B}), \tag{6}$$

where $[\mathcal{A}]^{-1}$ is the inverse of matrix $[\mathcal{A}]$. Since matrix $[\mathcal{A}]$ is only related to the hardware implementation of the multicore platform, we can calculate its inverse $[\mathcal{A}]^{-1}$ off-line. For notational brevity, let $[\mathcal{V}]$ be the inverse matrix of $[\mathcal{A}]$. For vector $\vec{B}$, B_n is the value at the n-th row. For matrix $[\mathcal{V}]$, $V_{j,\ell}$ is its element at the j-th row and the ℓ-th column. Hence, after assigning the execution speed of these M cores, the peak temperature can be easily obtained with the above formula.

We now provide an example to show why speed scaling matters for minimizing the peak temperature. Consider a system with 4 cores as shown in Figure 1 with matrix $[\mathcal{A}]$ defined as follows:

$$\begin{pmatrix} -0.261 & 0.009 & 0.004 & 0.000 & 0.200 & 0.050 \\ 0.009 & -0.121 & 0.000 & 0.004 & 0.050 & 0.060 \\ 0.004 & 0.000 & -0.261 & 0.009 & 0.200 & 0.050 \\ 0.000 & 0.004 & 0.009 & -0.121 & 0.050 & 0.060 \\ 0.200 & 0.050 & 0.200 & 0.050 & -1.725 & 0.300 \\ 0.050 & 0.060 & 0.050 & 0.060 & 0.300 & -1.445 \end{pmatrix}$$

Moreover, suppose that vector $\vec{B}$ is $[0.1, 0.1, 0.1, 0.1, 280.4, 280.4]^T$ and α is 1. The peak temperatures reached on these four cores by executing at speed 1.8 for all cores are $73.1, 102.6, 73.1, 102.6$ °C. Assigning the speed of the four cores as 2.1, 1.5, 2.1, and 1.5 leads to a solution with peak temperatures $87.5, 83.5, 87.5, 83.5$ °C on these four cores. The above speed assignments provide the same computation capability, but are with different peak temperatures. As a result, speed assignment must be done carefully so that the peak temperature can be reduced.

3.2. Preliminary Results for Global Scheduling

In this subsection, we summarize some recent results obtained by Baruah and Goossens [6], [7] for global scheduling upon uniform multiprocessor platforms, in which we will develop our approach based on their results. Let $\pi(i)$ denote the i'th fastest processor (ties broken arbitrarily) of multicore platform $\mathcal{M}$; that is, $s_{\pi(1)}, s_{\pi(2)}, \ldots s_{\pi(M)}$ are the speeds of the processors of $\mathcal{M}$, in non-increasing order. Some important metrics [15] on uniform multiprocessor platforms are:

$$S_\ell(\mathcal{M}) \overset{\text{def}}{=} \sum_{j=1}^{\ell} s_{\pi(j)}, \;\; \lambda(\mathcal{M}) \overset{\text{def}}{=} \max_{\ell=1}^{M} \left\{ \frac{\sum_{j=\ell+1}^{M} s_{\pi(j)}}{s_{\pi(\ell)}} \right\}. \quad (7)$$

We will use the convention that $S_{\pi(0)}(\mathcal{M})$ equals zero.

Sufficient conditions for global scheduling of sporadic task systems upon uniform multiprocessors are known:

Lemma 1 ([6], [7]): A constrained-deadline sporadic task system $\mathbf{T}$ is globally $\mathcal{S}$-schedulable ($\mathcal{S}$ is either EDF or DM) upon a processing platform $\mathcal{M}$, if

$$\mathsf{load}(\mathbf{T}, i) \leq \frac{1}{\phi_{\mathcal{S}}} \left(\mu(\mathcal{M}, \mathbf{T}, i) - \nu(\mathcal{M}, \mathbf{T}, i) \delta_{\max}(\mathbf{T}, i) \right), \quad (8)$$

for $i = N$ if $\mathcal{S} =$ EDF and for all i ($1 \leq i \leq N$) if $\mathcal{S} =$ DM, where

$$\mu(\mathcal{M}, \mathbf{T}, i) \overset{\text{def}}{=} S_M(\mathcal{M}) - \lambda(\mathcal{M}) \delta_{\max}(\mathbf{T}, i), \quad (9)$$
$$\nu(\mathcal{M}, \mathbf{T}, i) \overset{\text{def}}{=} \max\{\ell : S_\ell(\mathcal{M}) < \mu(\mathcal{M}, \mathbf{T}, i)\}, \quad (10)$$

and

$$\phi_{\mathcal{S}} \overset{\text{def}}{=} \begin{cases} 1, & \text{if } \mathcal{S} = \text{EDF} \\ 2, & \text{if } \mathcal{S} = \text{DM} \end{cases} \quad (11)$$

□

Additionally, necessary conditions for global scheduling of sporadic task systems can be obtained using $\mathsf{load}(\mathbf{T}, i)$ and $\delta_{\max}(\mathbf{T}, i)$:

Lemma 2 ([6]): If a task system $\mathbf{T}$ is globally schedulable (either EDF or DM) upon a processing platform $\mathcal{M}$, then for all i ($1 \leq i \leq N$),

$$\mathsf{load}(\mathbf{T}, i) \leq S_M(\mathcal{M}), \quad (12)$$

and

$$\delta_{\max}(\mathbf{T}, i) \leq s_{\pi(1)}. \quad (13)$$

□

3.3. Optimization for Preferred Speeds

For the rest of this section, we present how to derive the lower bound of the peak temperature among all cores and preferred speeds by solving non-linear programming optimally to minimize the peak temperature while the necessary schedulability conditions are satisfied. However, as the maximum density constraint in (13) is a non-linear constraint, we will first derive the peak temperature of the platform for a specified Core r such that $\delta_{\max}(\mathbf{T}, N) \leq s_r \leq s_{\pi(1)}$. Then, among these M solutions by setting $r = 1, 2, \ldots, M$, the corresponding speeds with the minimum peak temperature are returned as the preferred speeds.

Based on the necessary conditions of schedulability in Lemma 2 and the peak temperature formula in Section 3.1, the lower bound Θ_r^*, for a specified r, of the peak temperature can be obtained by solving the following non-linear programming (denoted $\mathsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$):

$$\begin{aligned} &\text{minimize } \Theta_r^* \overset{\text{def}}{=} \max_{1 \leq j \leq M+\hbar} \left\{ \sum_{\ell=1}^{M+\hbar} -V_{j,\ell} (\alpha s_\ell^\gamma + B_\ell) \right\} \\ &\text{subject to } \mathsf{load}(\mathbf{T}, N) \leq \sum_{\ell=1}^{M} s_\ell, \\ &\qquad \delta_{\max}(\mathbf{T}, N) \leq s_r, \\ &\qquad s_\ell \geq 0, 1 \leq \ell \leq M + \hbar. \end{aligned} \quad (14)$$

Obviously, an optimal solution to (14) will set s_{M+j} to zero where $j = 1, \ldots, \hbar$. Thus, we do not specify the constraints of the sinks in the above system.

To our best knowledge, there is no explicit form for an optimal solution of $\mathsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$. Here, we adopt the approach proposed by Dutta and Vidyasagar [13] by solving the above constrained non-linear programming with a transformation to unconstrained non-linear programming. Due to space limitation, we will only summarize the procedure as shown in the appendix, while the proof of optimality can be found in [13]. Moreover, for a given set $\mathbf{T}$ of tasks, the $\mathsf{load}(\mathbf{T}, N)$ is irrelevant to the speed settings. For the rest of this section, we assume that $\mathsf{load}(\mathbf{T}, N)$ is known a priori by applying the exact or approximated methods in [14].

Then the minimum among $\{\mathsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r) : r = 1, \ldots, M\}$ is the lower bound Θ_{min}^* of the peak temperature. Denote $r_{\min} \overset{\text{def}}{=} \arg\min\{\Theta_r^* : r = 1, \ldots, M\}$ and $\Theta_{\text{min}}^* \overset{\text{def}}{=} \Theta_{r_{\min}}^*$. Let $\mathcal{M}_{\min}$ be the system corresponding to Θ_{min}^* with the derived speeds $s_1, s_2, \ldots, s_M$. The following theorem shows that Θ_{min}^* is the lower bound of the peak temperature for feasible speed scheduling [1]:

[1] All proofs of the lemmas and the theorems and the corollaries are put in Appendix (unless otherwise stated).

Theorem 1: $\Theta^*_{\min}$ is a lower-bound on the peak temperature for task system $\mathbf{T}$ schedulable (by any algorithm) upon platform $\mathcal{M}$ with thermal characteristics expressed by matrix $[\mathcal{A}]$ and vectors $\vec{B}$ and $\vec{P}$. □

4. Feasible Speed Scheduling

Given $\mathcal{M}_{\min}$ determined by the preferred-speed calculation of Section 3.3, we now describe the next phase of deriving feasible speed scheduling. In this phase, we will obtain a constant multiplicative factor by which processing platform $\mathcal{M}_{\min}$'s speed would need to increase to guarantee that $\mathbf{T}$ is globally schedulable.

Let $\beta \cdot \mathcal{M}$ denote the platform where each of $\mathcal{M}$'s M processors has their speed increase by a constant factor $\beta \geq 1$; i.e. the speed of each processor ℓ in $\beta \cdot \mathcal{M}$ is $\beta \cdot s_\ell$. The following lemma states some properties of $\beta \cdot \mathcal{M}$ (the proof is straightforward and omitted for space):

Lemma 3: $S_\ell(\beta \cdot \mathcal{M}) = \beta \cdot S_\ell(\mathcal{M})$ and $\lambda(\beta \cdot \mathcal{M}) = \lambda(\mathcal{M})$, for all $\ell = 1, \ldots, M$. □

With the above notation, our objective for the feasible speed scheduling is to obtain a constant $\beta \geq 1$ such that $\mathbf{T}$ is globally schedulable (by EDF or DM) upon $\beta \cdot \mathcal{M}_{\min}$. We propose two methods to compute such a β. The first method derives a pessimistic bound on the speed-up required for both EDF and DM. The second method gives an iterative algorithm which improves upon this pessimistic bound.

4.1. Deriving a Pessimistic Feasible Speed Scheduling

A pessimistic bound on β for global EDF and DM may be achieved by simply deriving a β that satisfies Lemma 1. The following theorem (which follows a similar argument to Lemma 5 in [6]) obtains such a bound.

Theorem 2: For sporadic task system $\mathbf{T}$ and $\mathcal{M}_{\min}$, $\mathbf{T}$ is globally $\mathcal{S}$-schedulable ($\mathcal{S}$ is either EDF or DM) upon $\beta_\mathcal{S} \cdot \mathcal{M}_{\min}$ where $\beta_\mathcal{S}$ is defined as

$$\begin{aligned}&\Big[S_M(\mathcal{M})(s_{\pi(1)} + \phi_\mathcal{S} s_{\pi(M)}) - \lambda(\mathcal{M}) s_{\pi(1)} s_{\pi(M)} \\ &+\Big(\big(S_M(\mathcal{M})(s_{\pi(1)} + \phi_\mathcal{S} s_{\pi(M)}) - \lambda(\mathcal{M}) s_{\pi(1)} s_{\pi(M)}\big)^2 \\ &-4 S_M(\mathcal{M}) \lambda(\mathcal{M}) s^2_{\pi(1)} s_{\pi(M)}\Big)^{\frac{1}{2}}\Big] \Big(2 S_M(\mathcal{M}) s^2_{\pi(M)}\Big)^{-1}\end{aligned} \tag{15}$$

where $\phi_\mathcal{S}$ is defined in (11). □

Using the above theorem, we can obtain an approximation ratio (in terms of the ideal-processor speeds) for the peak temperature of the system, using the speed factor in (15):

Theorem 3: The peak temperature of $\beta_\mathcal{S} \cdot \mathcal{M}_{\min}$ (where $\mathcal{S}$ is either EDF or DM) is at most a factor of $\beta_\mathcal{S}^\gamma$ greater than the peak temperature of the optimal M-processor platform on which $\mathbf{T}$ is globally schedulable. □

4.2. Deriving a Better Feasible Speed Scheduling

The above analysis did not specify the task workload. For specific task workload, we can further improve the feasible speed scheduling. Let $\mathcal{M}_{\min}$ again be the "preferred-speed" processor determined from the previous section. We will now describe an algorithm for more precisely determining a processor $\beta \cdot \mathcal{M}_{\min}$ such that β is minimized. The next two lemmas give upper and lower bounds on the value β must satisfy in order for $\mathbf{T}$ to be global schedulable upon $\beta \cdot \mathcal{M}_{\min}$.

Lemma 4: Given $\mathbf{T}$, $\mathcal{M}$, and $\beta \geq 1$, if $\nu(\beta \cdot \mathcal{M}, \mathbf{T}, i)$ equals ℓ where $\ell \in \{0, 1, \ldots, M-1\}$, then

$$\Gamma(\mathcal{M}, \mathbf{T}, \ell, i) < \beta \leq \Gamma(\mathcal{M}, \mathbf{T}, \ell + 1, i) \tag{16}$$

where

$$\Gamma(\mathcal{M}, \mathbf{T}, \ell, i) \stackrel{\text{def}}{=} \begin{cases} \frac{\lambda(\mathcal{M}) \cdot \delta_{\max}(\mathbf{T}, i)}{S_M(\mathcal{M}) - S_\ell(\mathcal{M})}, & 0 \leq \ell < M - 1 \\ \infty, & \text{otherwise.} \end{cases} \tag{17}$$

□

Given the input task workload, by Lemma 3 we may simply solve (8) in Lemma 1 as shown in the following lemma (the proof is straightforward and omitted for space):

Lemma 5: For global scheduler $\mathcal{S}$ (either EDF or DM), if $\mathbf{T}$ satisfies (8), then there exists $\ell \in \{0, 1, \ldots, M-1\}$, equal to $\nu(\beta \cdot \mathcal{M}, \mathbf{T}, i)$, such that

$$\beta \geq \hat{\Gamma}(\mathcal{S}, \mathcal{M}, \mathbf{T}, \ell, i), \tag{18}$$

for $i = N$ if $\mathcal{S} =$ EDF and for all i ($1 \leq i \leq N$) if $\mathcal{S} =$ DM, where

$$\begin{aligned}\hat{\Gamma}(\mathcal{S}, \mathcal{M}, \mathbf{T}, \ell, i) \quad &\stackrel{\text{def}}{=} \quad \frac{1}{S_M(\mathcal{M})}(\phi_\mathcal{S} \cdot \mathsf{load}(\mathbf{T}, i) \\ &\qquad + (\lambda(\mathcal{M}) + \ell)\delta_{\max}(\mathbf{T}, i)),\end{aligned} \tag{19}$$

and $\phi_\mathcal{S}$ is defined in (11). □

Next we aim to find the minimum β that satisfy Lemmas 4 and 5 upon a processor $\beta \cdot \mathcal{M}_{\min}$. Since $\hat{\Gamma}()$ is an increasing function with respects to ℓ, then we only need to find the minimum ℓ satisfying both lemmas, which is defined as

$$\begin{aligned}\ell_{\min,i} \quad \stackrel{\text{def}}{=} \quad & \min\{\ell \in \{0, 1, \ldots, M-1\} : \\ & \Gamma(\mathcal{M}_{\min}, \mathbf{T}, \ell) < \hat{\Gamma}(\mathcal{S}, \mathcal{M}_{\min}, \mathbf{T}, \ell, i) \\ & \leq \Gamma(\mathcal{M}_{\min}, \mathbf{T}, \ell + 1)\}.\end{aligned} \tag{20}$$

Then the minimum β can be obtained as the following theorem (the proof is straightforward based on the above analysis and omitted for space):

Theorem 4: For sporadic task system $\mathbf{T}$ and $\mathcal{M}_{\min}$, $\mathbf{T}$ is globally EDF-schedulable upon $\beta_{\text{EDF}} \cdot \mathcal{M}_{\min}$ where β_{EDF} is defined as

$$\beta_{\text{EDF}} \stackrel{\text{def}}{=} \hat{\Gamma}(\text{EDF}, \mathcal{M}_{\min}, \mathbf{T}, \ell_{\min,N}, N); \tag{21}$$

$\mathbf{T}$ is globally DM-schedulable upon $\beta_{\text{DM}} \cdot \mathcal{M}_{\min}$ where β_{DM}

is defined as

$$\beta_{\text{DM}} \overset{\text{def}}{=} \max_{i\in\{1,2,\ldots,N\}} \{\hat{\Gamma}(\text{DM}, \mathcal{M}_{\min}, \mathbf{T}, \ell_{\min,i}, i)\}, \qquad (22)$$

where $\hat{\Gamma}()$ is defined in (19) and $\ell_{\min,i}$ is defined in (20). □

5. Performance Evaluation

This section provides performance evaluations of the proposed algorithm for speed assignments under global real-time scheduling. Due to space limitation, we will only present the simulation results by adopting the global EDF scheduling policy for task scheduling. The results for DM scheduling are similar. In the simulations, we evaluate two different algorithms defined as follows:

- Algorithm **Balanced**: first derives speed assignment by applying the necessary schedulability condition so that the speeds are as balanced as possible, and then applies Theorem 4 for speed determination.
- Algorithm **PTO**: first applies sequential quadratic programming for deriving optimal solutions of (14), and then applies Theorem 4 for determining the resulting speeds.

We evaluate the performance in terms of peak temperature of the resulting task partition on three different hardware platforms, in which their layouts are 2×2, 4×1, and 3×2 with 4, 4, and 6 cores, respectively. The corresponding thermal parameters are determined based on the layout. The ambient temperature in the simulations is assumed as 30 °C. The power consumption function $\Psi(s_\ell, \Theta_\ell)$ is $s_\ell^3 + 0.1 + 0.002\Theta_\ell$ Watt, where Θ_ℓ is the absolute temperature of Core ℓ and s_ℓ is with unit of GHz.

We use synthetic sporadic real-time tasks for evaluating the performance, in which the deadline of a task is earlier than its period. The task set generator is based on the approach developed by Baker [3]: An initial $M + 1$ tasks are pseudo-randomly generated and added to the collection of tasks. Subsequent task systems add tasks to this initial set until the task system is no longer feasible upon M unit speed processors, at which time a new set of $M + 1$ tasks is randomly generated. Each task τ_i in synthetic task system had its period parameter p_i uniformly chosen from $[1, 1000]$. A utilization parameter u_i was generated for each task drawn from the inverse exponential distribution from 0.0 to 1.0. A task's deadline is chosen uniformly from the interval $[e_i, p_i]$. In our experiments, we generate 1000 task sets with different numbers of tasks and different values of $\text{load}(\mathbf{T}, N)$. Moreover, for deriving the preferred speeds, we ignore the known parameter $\text{load}(\mathbf{T}, N)$ and apply the approximated calculation proposed in [14] to efficiently derive the upper bound of $\text{load}(\mathbf{T}, N)$ with less than 1% error.

For each task set, we evaluate the peak temperatures of the resulting speed assignments of Algorithm BALANCED and Algorithm PTO. For a specified platform, we conduct the peak temperature for two different settings, based on the numbers of tasks of the input task set and the values of workload $\text{load}(\mathbf{T}, N)$. For each configuration in a setting, the average value of the peak temperatures is reported.

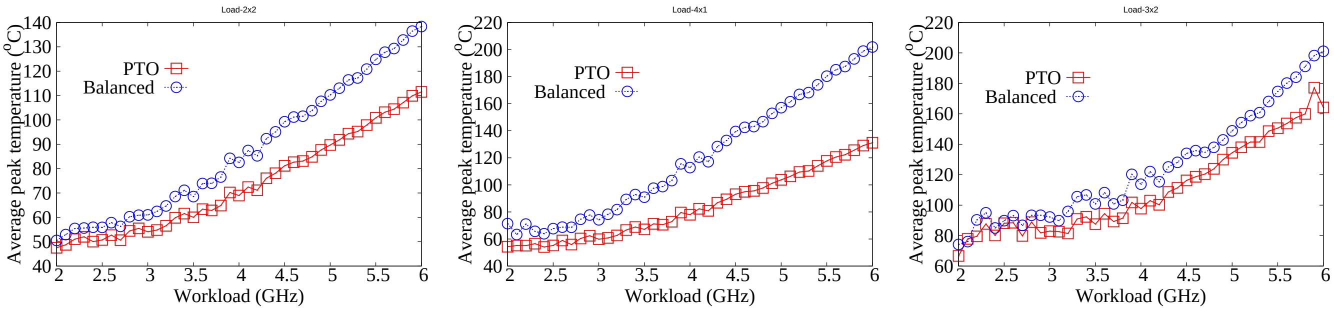
Figure 2 presents the average peak temperatures of the evaluated algorithms for the platforms under different workload settings. As shown in Figure 2, the average peak temperature generally increases when the workload demand increases. When the workload is low, e.g., less than 3 GHz, the difference between the evaluated algorithms is not too much because the power consumptions on the cores are not very high. However, when the workload is higher, a good speed assignment can significantly reduce the peak temperature, as shown in Figure 2 when the workload is more than 4.5 GHz. By applying the algorithm proposed in this paper, we can reduce the average peak temperature by at most 30°C for benchmark 2×2 in Figure 2(a), at most 70°C for benchmark 4×1 in Figure 2(b), and at most 35°C for benchmark 3×2 in Figure 2(c).

Figure 3 illustrates the results by varying the number of tasks. For a specified workload, the less the number of tasks of the input task set, the larger the density/workload of the given tasks is. Therefore, according to the generator of task sets, when the task number is fewer, the average peak temperature is higher in both evaluated algorithms for all the platform benchmarks. Compared to Algorithm **Balanced**, Applying Algorithm **PTO** proposed in this paper can reduce the average peak temperature by at most 50°C for a fixed number of tasks.

6. Conclusion

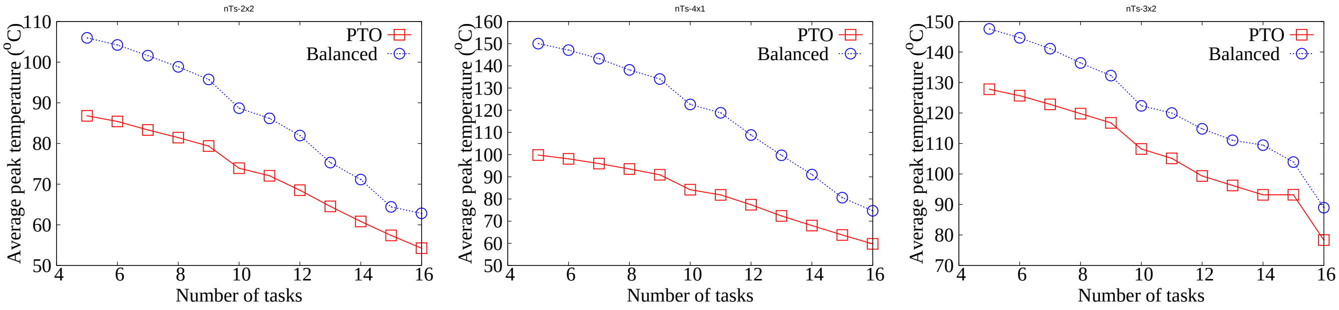
Thermal constraints are becoming increasingly severe for many systems as chip density increases and the size of the system decreases. Heat dissipation in multicore platforms further complicates satisfying thermal constraints due to the transfer of heat between cores on the same chip. In order to respect these constraints, system designers may scale-back the power-consumption to reduce the peak temperature of the system. However, in real-time, thermal-aware systems the system designer must simultaneously ensure that temporal constraints are still satisfied. The focus of our current research is to address the challenge of minimizing the peak-temperature for a multicore platform scheduled by a multiprocessor real-time scheduling algorithm.

In this paper, we focused upon global scheduling of sporadic task systems according to either the EDF or DM scheduling algorithms. Under this setting, we proposed an approach which first derives the preferred speeds of the cores by using necessary conditions for multiprocessor schedulability. The resulting platform executing at the preferred



(a) benchmark 2×2 (b) benchmark 4×1 (c) benchmark 3×2

Figure 2: Simulation results of different demand functions $\mathsf{load}(\mathbf{T}, N)$ for systems with different layouts.



(a) benchmark 2×2 (b) benchmark 4×1 (c) benchmark 3×2

Figure 3: Simulation results of different number of tasks for systems with different layouts.

speeds may be viewed as a uniform multiprocessor platform. We applied known schedulability tests to correctly scale the speed of the preferred speeds to ensure the schedulability of the task system. We showed that our approach is effective in reducing peak temperature by comparing its performance (via simulations over synthetically generated task systems) against an approach which attempts to balance the speed of the processors. The reduction in peak temperature may be as much as 70°C for some multicore platform configurations. Our current approach statically determines the speed of each processor prior to system execution. Future research will investigate whether further temperature reduction is possible in multicore platforms when each core may vary its speed over time.

Acknowledgment

This work is sponsored in part by a Wayne State University Faculty Research Award, Rackham Faculty Research Grant at the University of Michigan, NSF CAREER Grant No. CNS-0746906, Taiwan National Science Council NSC-096-2917-I-564-121, and the European Community's Seventh Framework Programme FP7/2007-2013 project Predator (Grant 216008).

References

[1] H. Aydin, V. Devadas, and D. Zhu. System-level energy management for periodic real-time tasks. In *the 27th IEEE Real-Time Systems Symposium*, 2006.

[2] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *17th International Parallel and Distributed Processing Symposium*, 2003.

[3] T. P. Baker. Comparison of empirical success rates of global vs. partitioned fixed-priority and EDF scheduling for hard real time. Technical Report TR-050601, Department of Computer Science, Florida State University, 2005.

[4] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. In *Symposium on Foundations of Computer Science*, 2004.

[5] N. Bansal and K. Pruhs. Speed scaling to manage temperature. In *Symposium on Theoretical Aspects of Computer Science*, 2005.

[6] S. Baruah and J. Goossens. Deadline monotonic scheduling on uniform multiprocessors. In *International Conference on Principles of Distristributed Systems (OPODIS)*, 2008.

[7] S. Baruah and J. Goossens. The edf scheduling of sporadic task systems on uniform multiprocessors. In *IEEE Real-Time Systems Symposium*, 2008.

[8] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190, Orlando, Florida, 1990. IEEE Computer Society Press.

[9] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *International Symposium on High-Performance Computer Architecture*, 2001.

[10] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs. In *Design, Automation and Test in Europe*, 2008.

[11] J.-J. Chen, C.-M. Hung, and T.-W. Kuo. On the minimization of the instantaneous temperature for periodic real-time tasks. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2007.

[12] J.-J. Chen, S. Wang, and L. Thiele. Proactive speed scheduling for frame-based real-time tasks under thermal constraints. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2009.

[13] S. R. K. Dutta and M. Vidyasagar. New algorithms for constrained minimax optimization. *Journal Mathematical Programming*, (1):140–155, 1977.

[14] N. Fisher, T. Baker, and S. Baruah. Algorithms for determining the demand-based load of a sporadic task system. In *Proceedings of the International Conference on Real-time Computing Systems and Applications*, Sydney, Australia, August 2006. IEEE Computer Society Press.

[15] S. H. Funk. *EDF Scheduling on Heterogeneous Multiprocessors*. PhD thesis, Department of Computer Science, The University of North Carolina at Chapel Hill, 2004.

[16] M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas. A Framework for Dynamic Energy Efficiency and Temperature Management. In *International Symposium on Microarchitecture*, 2000.

[17] W.-L. Hung, Y. Xie, N. Vijaykrishnan, M. T. Kandemir, and M. J. Irwin. Thermal-aware task allocation and scheduling for embedded systems. In *ACM/IEEE Conference of Design, Automation, and Test in Europe*, 2005.

[18] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *the Design Automation Conference*, 2004.

[19] M. Kadin and S. Reda. Frequency planning for multi-core processors under thermal constraints. In *ISLPED*, pages 213–216, 2008.

[20] W. Liao, L. He, and K. M. Lepak. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 24(7):1042–1053, 2005.

[21] Y. Liu, R. P. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *DATE*, pages 1526–1531, 2007.

[22] A. K. Mok. *Fundamental Design Problems of Distributed Systems for The Hard-Real-Time Environment*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1983. Available as Technical Report No. MIT/LCS/TR-297.

[23] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. D. Micheli. Temperature control of high-performance multi-core platforms using convex optimization. In *DATE*, pages 110–115, 2008.

[24] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D. Micheli. Temperature-aware processor frequency assignment for mpsocs using convex optimization. In *IEEE/ACM international conference on Hardware/software codesign and system synthesis*, 2007.

[25] J. E. Sergent and A. Krum. *Thermal Management Handbook*. McGraw-Hill, 1998.

[26] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *International Symposium on Computer Architecture*, 2003.

[27] S. Wang and R. Bettati. Delay analysis in temperature-constrained hard real-time systems with general task arrivals. In *IEEE Real-Time Systems Symposium*, 2006.

[28] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. In *Euromicro Conference on Real-Time Systems*, 2006.

[29] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. *Real-Time Systems Journal*, 39(1-3):658–671, 2008.

[30] Y.-W. Wu, C.-L. Yang, P.-H. Yuh, and Y.-W. Chang. Joint exploration of architectural and physical design spaces with thermal consideration. In *International Symposium on Low Power Electronics and Design*, 2005.

[31] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Moss. Energy efficient policies for embedded clusters. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, 2005.

[32] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Symposium on Foundations of Computer Science*, 1995.

[33] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *International Conference on Computer-Aided Design*, 2007.

Solving $\textsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$

By ignoring the constraint $\delta_{\max}(\mathbf{T}, N) \leq s_r$ and assuming Core q has the highest temperature among all cores, the following relaxation will result in a lower bound of the original optimization:

$$\begin{array}{rl} \text{minimize} & \sum_{\ell=1}^{M+\hbar} -V_{q,\ell}(\alpha s_\ell^\gamma + B_\ell) \\ \text{subject to} & \textsf{load}(\mathbf{T}, N) \leq \sum_{\ell=1}^{M} s_\ell, \\ & s_\ell \geq 0, 1 \leq \ell \leq M + \hbar. \end{array} \tag{23}$$

Then, the above equation can be solved by applying the Lagrange Multiplier Method in $O(M)$, i.e.,

$$-\alpha V_{q,1} s_1^{\gamma-1} = -\alpha V_{q,\ell} s_\ell^{\gamma-1}.$$

Hence,

$$s_{q,1} = \frac{U}{\sum_{\ell=1}^{M}(\frac{V_{q,1}}{V_{q,\ell}})^{\frac{1}{\gamma-1}}}, \quad s_{q,\ell} = s_{q,1}(\frac{V_{q,1}}{V_{q,\ell}})^{\frac{1}{\gamma-1}},$$

where $s_{q,\ell}$ is the speed of Core ℓ under the assumption that Core q is with the highest temperature among all cores, which might not be true. Therefore,

$$\Theta_{r,0}^{*} = \max_{q=1,2,\ldots,M} \left\{ \sum_{\ell=1}^{M+\hbar} -V_{q,\ell}(\alpha s_{q,\ell}^{\gamma} + B_\ell) \right\}$$

is a lower bound of $\textsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$.

Next, starting from $\Theta^*_{r,0}$, we approach the optimal solution of $\textsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$ step by step. That is, for the k-th step, we will derive a new lower bound $\Theta^*_{r,k}$ based on $\Theta^*_{r,k-1}$. Specifically, at the k-th step, we first minimize the following unconstrained non-linear programming by applying the sequential quadratic programming method:

$$\begin{aligned}&\textstyle\sum_{j=1}^{M+\hbar}\left[\max\left\{0, \sum_{\ell=1}^{M+\hbar} -V_{j,\ell}(\alpha s_\ell^\gamma + B_\ell) - \Theta^*_{r,k-1}\right\}\right]^2\\ &+\epsilon_1\left[\max\left\{0, \delta_{\max}(\mathbf{T}, N) - s_r\right\}\right]^2\\ &+\epsilon_2\left[\textsf{load}(\mathbf{T}, N) - \textstyle\sum_{\ell=1}^{M} s_\ell\right]^2,\end{aligned} \tag{24}$$

where ϵ_1 and ϵ_2 are defined positive constants related to the rate of convergence from $\Theta^*_{r,k-1}$ to $\Theta^*_{r,k}$. In general, the constants ϵ_1 and ϵ_2 should be set as large numbers for deriving precise results. Suppose that the optimal solution of (24) is $\Upsilon_{r,k}$. Then, we can set $\Theta^*_{r,k}$ as $\Theta^*_{r,k-1} + (\frac{\Upsilon_{r,k}}{M})^{\frac{1}{2}}$. The above procedure repeats until $(\frac{\Upsilon_{r,k}}{M})^{\frac{1}{2}}$ is a small number. As shown in [13], the resulting speed assignment with the converged $\Theta^*_{r,k}$ is the optimal solution of $\textsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$, when ϵ_1 and ϵ_2 are large numbers.

Proof of Theorem 1

Let $\mathcal{M}$ be the platform defined by processor speeds $s_1, s_2, \ldots, s_M$. By Lemma 2, if $\mathbf{T}$ is schedulable (either EDF or DM) upon $\mathcal{M}$ then $\textsf{load}(\mathbf{T}, i) \leq \textsf{load}(\mathbf{T}, N) \leq \sum_{\ell=1}^{M} s_\ell = S_M(\mathcal{M})$ and $\delta_{\max}(\mathbf{T}, i) \leq \delta_{\max}(\mathbf{T}, N) \leq \max_{\ell=1}^{M}\{s_{\pi(\ell)}\}$ for all $i = 1, \ldots, N$. Thus, by the first and second constraints of $\textsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$, the set

$$\begin{aligned}&\{\mathcal{M} | s_1, s_2, \ldots, s_M\\ &\quad \text{are feasible values of } \textsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)\}\end{aligned}$$

must contain the set of all processors $\mathcal{M}$ with $s_r \geq \delta_{\max}(\mathbf{T}, N)$ where $\mathbf{T}$ is globally schedulable upon $\mathcal{M}$. Thus, the union of all feasible values of $s_1, s_2, \ldots, s_M$ for $\textsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$ over $r = 1, \ldots, M$ must contain the set of all M-processor platforms upon which $\mathbf{T}$ is globally schedulable. It follows that $\Theta^*_{\min}$ is a lower bound on the peak temperature.

Proof of Theorem 2

The satisfaction of Lemma 1 is sufficient for $\mathbf{T}$ to be $\mathcal{A}$-schedulable upon platform $\beta \cdot \mathcal{M}_{\min}$. That is, we will show the following condition holds:

$$\begin{aligned}
\phi_{\mathcal{S}}\textsf{load}(\mathbf{T}, N) &\leq \mu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, N)\\
&\quad -\nu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, N)\delta_{\max}(\mathbf{T}, N)\\
&\Leftarrow \left(\text{since } \left\lceil \tfrac{\mu(\beta\cdot\mathcal{M}_{\min}, \mathbf{T}, N)}{\beta\cdot s_{\pi(M)}}\right\rceil - 1\right.\\
&\qquad \left.\geq \nu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, N)\right)\\
\phi_{\mathcal{S}}\textsf{load}(\mathbf{T}, N) &\leq \mu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, N)\\
&\quad -\left(\left\lceil \tfrac{\mu(\beta\cdot\mathcal{M}_{\min}, \mathbf{T}, N)}{\beta\cdot s_{\pi(M)}}\right\rceil - 1\right)\delta_{\max}(\mathbf{T}, N)\\
&\Leftarrow (\text{since for all } \alpha, \lceil\alpha\rceil - 1 \leq \alpha)\\
\phi_{\mathcal{S}}\textsf{load}(\mathbf{T}, N) &\leq \mu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, N)\\
&\quad -\left(\tfrac{\mu(\beta\cdot\mathcal{M}_{\min}, \mathbf{T}, N)}{\beta\cdot s_{\pi(M)}}\right)\delta_{\max}(\mathbf{T}, N)\\
&\equiv\\
\phi_{\mathcal{S}}\textsf{load}(\mathbf{T}, N) &\leq \mu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, N)\left(1 - \tfrac{\delta_{\max}(\mathbf{T}, N)}{\beta\cdot s_{\pi(M)}}\right)\\
&\equiv (\text{by the definition of } \mu)\\
\phi_{\mathcal{S}}\textsf{load}(\mathbf{T}, N) &\leq [S_M(\beta \cdot \mathcal{M}_{\min}) - \lambda(\beta \cdot \mathcal{M}_{\min})\delta_{\max}(\mathbf{T}, N)]\\
&\qquad \times\left(1 - \tfrac{\delta_{\max}(\mathbf{T}, N)}{\beta\cdot s_{\pi(M)}}\right)\\
&\equiv (\text{by Lemmas 3})\\
\phi_{\mathcal{S}}\textsf{load}(\mathbf{T}, N) &\leq (\beta \cdot S_M(\mathcal{M}_{\min}) - \lambda(\mathcal{M}_{\min})\delta_{\max}(\mathbf{T}, N))\\
&\qquad \times\left(1 - \tfrac{\delta_{\max}(\mathbf{T}, N)}{\beta\cdot s_{\pi(M)}}\right)\\
&\Leftarrow \Big(\text{constraints } \ (\textsf{load}(\mathbf{T}, N) \leq S_M(\mathcal{M}_{\min}))\\
&\qquad \wedge \left(\delta_{\max}(\mathbf{T}, N) \leq s_{\pi(1)} \text{ of } \textsf{SYSTEM}\right)\Big)\\
\phi_{\mathcal{S}} S_M(\mathcal{M}_{\min}) &\leq \left(\beta \cdot S_M(\mathcal{M}_{\min}) - \lambda(\mathcal{M}_{\min})s_{\pi(1)}\right)\\
&\qquad \times\left(1 - \tfrac{s_{\pi(1)}}{\beta\cdot s_{\pi(M)}}\right)\\
&\equiv
\end{aligned}$$

$$\begin{aligned}&s_{\pi(M)}S_M(\mathcal{M}_{\min})\beta^2 - [(s_{\pi(1)} + \phi_{\mathcal{S}} s_{\pi(M)})S_M(\mathcal{M}_{\min})\\ &\quad +\lambda(\mathcal{M}_{\min})s_{\pi(1)}s_{\pi(M)}]\beta + \lambda(\mathcal{M}_{\min})s_{\pi(1)}^2 \geq 0.\end{aligned}$$

Using standard techniques for solving quadratic equations, we obtain $\beta_{\mathcal{S}}$ equal to the solution of the final inequality above.

Proof of Theorem 3

According to Theorem 1, a lower-bound on the peak temperature of such an M-core system that can schedule $\mathbf{T}$. Observe that in (14), $-V_{j,\ell}$ is a positive constant. Thus, by increasing any s_j by $\beta_{\mathcal{S}}$ will increase the peak temperature by at most a factor of $\beta_{\mathcal{S}}^\gamma$.

Proof of Lemma 4

Given $\mathbf{T}$, $\mathcal{M}$, and $\beta \geq 1$, let ℓ equal $\nu(\beta \cdot \mathcal{M}, \mathbf{T}, i)$. We will consider two cases:

If $0 \leq \ell < M - 1$, then the definition of ν implies,

$$\begin{aligned}
&S_\ell(\beta \cdot \mathcal{M}) < \mu(\beta \cdot \mathcal{M}, \mathbf{T}, i) \leq S_{\ell+1}(\beta \cdot \mathcal{M})\\
\Rightarrow\ &\beta \cdot S_\ell(\mathcal{M}) < \beta \cdot S_M(\mathcal{M}) - \lambda(\mathcal{M})\delta_{\max}(\mathbf{T}, i) \leq \beta \cdot S_{\ell+1}(\mathcal{M})\\
\Rightarrow\ &\tfrac{\lambda(\mathcal{M})\delta_{\max}(\mathbf{T}, i)}{S_M(\mathcal{M}) - S_\ell(\mathcal{M})} < \beta \leq \tfrac{\lambda(\mathcal{M})\delta_{\max}(\mathbf{T}, i)}{S_M(\mathcal{M}) - S_{\ell+1}(\mathcal{M})}
\end{aligned}$$

The final implication implies the lemma by substituting Γ into the right-hand side of both inequalities above.

If $\ell = M - 1$, then the definition of ν implies $S_{M-1}(\beta \cdot \mathcal{M}) < \mu(\beta \cdot \mathcal{M}, \mathbf{T}, i)$. By the same implications above, we have $\beta > \frac{\lambda(\mathcal{M})\delta_{\max}(\mathbf{T}, i)}{S_M(\mathcal{M}) - S_{M-1}(\mathcal{M})}$. Thus, $\Gamma(\mathcal{M}, \mathbf{T}, M - 1, i) < \beta \leq \infty$, and the lemma follows.