Thermal-Aware Global Real-Time Scheduling on Multicore Systems

Nathan Fisher[a], Jian-Jia Chen[b], Shengquan Wang[c], Lothar Thiele[b]

[a]*Department of Computer Science, Wayne State University, USA*
[b]*Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland*
[c]*Department of Computer and Information Science, University of Michigan-Dearborn, USA*

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. We perform simulations to evaluate the performance of the proposed approach.

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

Email addresses: `fishern@cs.wayne.edu` (Nathan Fisher), `jchen@tik.ee.ethz.ch` (Jian-Jia Chen), `shqwang@umd.umich.edu` (Shengquan Wang), `thiele@tik.ee.ethz.ch` (Lothar Thiele)

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 [1]. 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 [2]. Recent estimates have placed the packaging cost at \$1 to \$3 per watt of heat dissipated [3]. 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 [3, 4, 5, 1]. As an alternative solution, the DTM [3, 4, 5] 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 [3, 4, 5] to prevent the system from overheating by adopting Dynamic Voltage Scaling (DVS) [6, 7]. Wang et al. [8, 9, 10] developed reactive speed control with schedulability tests and delay analysis, while Chen et al. [11] developed proactive speed control to improve the schedulability. Bansal et al. [12] developed an algorithm to maximize the workload that can complete in a specified time window without violating the thermal constraints. Zhang and Chatha [13] provided approximation algorithms to minimize the completion time, while each task is restricted to execute at one speed. Chen et al. [14] 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. [2] show that Yao's algorithm [6] for real-time jobs is a 20-approximation algorithm for peak temperature minimization.

Thermal-aware multiprocessor scheduling has also been explored recently, e.g., [15, 16, 17, 18]. 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. [14] proved that the largest-task-first strategy (also called worst-fit decreasing [19]) 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. [20] 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 [16, 20, 15, 17], 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 platforms with 4×1, 2×2, 4×2, and layouts for simulations.

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. We will conclude the paper in Section 6.

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 [2, 12, 15, 20, 13, 21, 8, 9, 10, 17], in which the thermal coefficients can be obtained by using the RC thermal model, such as the approaches in [16, 20, 15, 17]. The thermal model adopted in this paper is similar to the recent approaches in [16, 20, 15].

We define $\mathcal{M} = \{1, 2, 3, \ldots, M\}$ 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 $\mathcal{H} = \{1, 2, 3, \ldots, \hbar\}$ 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

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

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., [16, 20, 15].

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 [7, 20, 22], 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 [23, 14]. When the leakage power consumption is related to the temperature, it is a super linear function of the temperature [24]. As shown in [25, 20], 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:

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

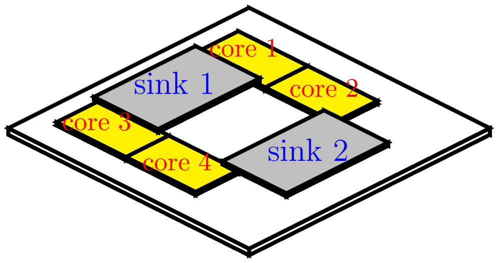


Figure 1: An example for 4 cores.

Task model In this paper, we consider jobs generated by a *sporadic task system* [26], $\mathbf{T} \overset{\text{def}}{=} \{\tau_1, \tau_2, \ldots, \tau_N\}$. 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. The *utilization* of task τ_i is denoted by $u_i \overset{\text{def}}{=} e_i/p_i$. For this paper, we consider two special subclasses of sporadic task systems: *implicit-deadline* and *constrained-deadline*. An implicit-deadline sporadic task system requires that for each $\tau_i \in \mathbf{T}$, the relative deadline equals the period (i.e., $d_i = p_i$). For an implicit-deadline task system, we will assume that the tasks are indexed in non-decreasing order of utilization: $u_i \leq u_{i+1}$ for all $1 \leq i < N$. A constrained-deadline task system requires $d_i \leq p_i$ 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 total utilization of the first k tasks ($1 \leq k \leq N$) is defined as:

$$u_{\text{sum}}(\mathbf{T}, k) \stackrel{\text{def}}{=} \sum_{i=1}^{k} u_i. \tag{3}$$

The maximum utilization over all tasks of $\mathbf{T}$ is denoted by $u_{\max}(\mathbf{T})$. The *density* of τ_i is denoted by $\delta_i \stackrel{\text{def}}{=} e_i / (\min((d_i, p_i))$. The max density (among the first k tasks of $\mathbf{T}$) are respectively defined as:

$$\delta_{\max}(\mathbf{T}, k) \stackrel{\text{def}}{=} \max_{i=1}^{k}\{\delta_i\}. \tag{4}$$

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 [27], it has been shown that for a sporadic task τ_i, the demand-bound function may be computed as follows:

$$\mathsf{dbf}(\tau_i, t) \stackrel{\text{def}}{=} \max\left(0, \left(\left\lfloor \frac{t - d_i}{p_i} \right\rfloor + 1\right) e_i\right). \tag{5}$$

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

$$\mathsf{load}(\mathbf{T}, k) = \max_{t \geq 0} \left\{ \frac{\sum_{i=1}^{k} \mathsf{dbf}(\tau_i, t)}{t} \right\}. \tag{6}$$

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 [28].

Scheduling algorithms Each DVS core on our platform $\mathcal{M}$ is permitted to execute at a potentially different speed than the other cores. The *uni-*

form multiprocessor model (e.g., see [29]) 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,

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

As Θ_a is fixed, for the rest of the this paper, we can simply take Θ_a as 0 and the temperatures on the cores and sinks are shifted accordingly, i.e., Θ_h^* is $\Theta_h^* - \Theta_a$, Θ_g^* is $\Theta_g^* - \Theta_a$, and Θ_j^* is $\Theta_j^* - \Theta_a$. Therefore, for all j,

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

and, for the heat sink h,

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

We can simplify the above equations by the following notations: for any

$1 \leq j \neq \ell \leq M$ and $1 \leq h \neq g \leq \hbar$,

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

Then, we know that

$$\begin{pmatrix} A_{1,1} & \cdots & A_{1,\eta}\\ A_{2,1} & \cdots & A_{2,\eta}\\ \vdots & \vdots & \vdots\\ A_{M,1} & \cdots & A_{M,\eta}\\ A_{M+1,1} & \cdots & A_{M+1,\eta}\\ \vdots & \vdots & \vdots\\ A_{\eta,1} & \cdots & A_{\eta,\eta} \end{pmatrix} \begin{pmatrix} \Theta_1^*\\ \Theta_2^*\\ \vdots\\ \Theta_M^*\\ \Theta_{M+1}^*\\ \vdots\\ \Theta_\eta^* \end{pmatrix} = - \begin{pmatrix} \alpha s_1^\gamma + \Omega + \delta\Theta_a\\ \alpha s_2^\gamma + \Omega + \delta\Theta_a\\ \vdots\\ \alpha s_M^\gamma + \Omega + \delta\Theta_a\\ 0\\ \vdots\\ 0 \end{pmatrix},$$

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 $(\overbrace{\Omega, \Omega, \ldots, \Omega}^{M}, \overbrace{0, \ldots, 0}^{\hbar})$. Let $\vec{P}$ be the transposition 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 $1 \leq h \leq \hbar$.

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

$$\vec{\Theta} = -[\mathcal{A}]^{-1}(\vec{P} + \vec{B}), \tag{7}$$

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 and 2 sinks with matrix $[\mathcal{A}]$ defined as follows:

$$\begin{pmatrix} -1.7000 & 0.2500 & 0 & 0 & 0.1500 & 1.2000 \\ 0.2500 & -1.0000 & 0 & 0 & 0.0500 & 0.6000 \\ 0 & 0 & -1.3500 & 0.5000 & 0.1500 & 0.6000 \\ 0 & 0 & 0.5000 & -1.8500 & 0.0500 & 1.2000 \\ 0.1500 & 0.0500 & 0.1500 & 0.0500 & -5.0300 & 1.0000 \\ 1.2000 & 0.6000 & 0.6000 & 1.2000 & 1.0000 & -10.000 \end{pmatrix}$$

Suppose that vector $\vec{B}$ is $[4.73, 4.73, 4.73, 4.73, 0, 0]^T$ and ambient temperature Θ_a is 30 °C. The power consumption of a core at 1GHz is 40 ($\alpha = 40$), and $\gamma = 3$. The peak temperatures reached on these four cores by executing at speed 1GHz for all cores are $83.60, 102.08, 95.13, 86.61$ °C. Assigning the speed of the four cores as $1.1, 0.9, 0.95, 1.05$ GHz leads to a solution with peak temperatures $90.45, 93.25, 92.23, 89.55$ °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 schedulability and feasibility results obtained by Funk, Goossens, and Baruah [30, 31, 32, 33] for global scheduling of implicit-deadline and constrained-deadline sporadic task systems upon uniform multiprocessor platforms. We will develop our approach based on these schedulability and feasibility conditions. 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 [29] on uniform multiprocessor platforms are:

$$S_\ell(\mathcal{M}) \stackrel{\text{def}}{=} \sum_{j=1}^{\ell} s_{\pi(j)}, \tag{8}$$

$$\lambda(\mathcal{M}) \stackrel{\text{def}}{=} \max_{\ell=1}^{M} \left\{ \frac{\sum_{j=\ell+1}^{M} s_{\pi(j)}}{s_{\pi(\ell)}} \right\}, \tag{9}$$

$$\widehat{\lambda}(\mathcal{M}) \stackrel{\text{def}}{=} \max_{\ell=1}^{M} \left\{ \frac{\sum_{j=\ell}^{M} s_{\pi(j)}}{s_{\pi(\ell)}} \right\}. \tag{10}$$

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

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

Lemma 1 ([30]). *An implicit-deadline sporadic task system* $\mathbf{T}$ *is globally* EDF-*schedulable upon a processing platform* $\mathcal{M}$*, if*

$$S_M(\mathcal{M}) \geq u_{\text{sum}}(\mathbf{T}, N) + \lambda(\mathcal{M}) \cdot \max\left\{ u_{\max}(\mathbf{T}), \frac{u_{\text{sum}}(\mathbf{T}, N)}{M} \right\}. \tag{11}$$

Lemma 2 ([31]). *An implicit-deadline sporadic task system* $\mathbf{T}$ *is globally* DM*-schedulable upon a processing platform* $\mathcal{M}$*, if*

$$S_M(\mathcal{M}) \geq 2u_{\text{sum}}(\mathbf{T}, N) + \widehat{\lambda}(\mathcal{M}) \cdot u_{\max}(\mathbf{T}). \tag{12}$$

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

Lemma 3 ([32, 33]). *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*

$$\mathsf{load}(\mathbf{T}, i) \leq \frac{1}{\phi_{\mathcal{S}}} \left(\mu(\mathcal{M}, \mathbf{T}, i) - \nu(\mathcal{M}, \mathbf{T}, i)\delta_{\max}(\mathbf{T}, i)\right), \tag{13}$$

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

$$\mu(\mathcal{M}, \mathbf{T}, i) \overset{def}{=} S_M(\mathcal{M}) - \lambda(\mathcal{M})\delta_{\max}(\mathbf{T}, i), \tag{14}$$

$$\nu(\mathcal{M}, \mathbf{T}, i) \overset{def}{=} \max\{\ell : S_\ell(\mathcal{M}) < \mu(\mathcal{M}, \mathbf{T}, i)\}, \tag{15}$$

and

$$\phi_{\mathcal{S}} \overset{def}{=} \begin{cases} 1, & \textit{if } \mathcal{S} = \text{EDF} \\ 2, & \textit{if } \mathcal{S} = \text{DM} \end{cases} \tag{16}$$

Additionally, a necessary and sufficient condition for *feasibility* may be obtained for implicit-deadline sporadic task systems. A task system $\mathbf{T}$ is *feasible* if there exists always exist a way to schedule (by any algorithm) the jobs of $\mathbf{T}$ such that they meet their respective deadlines on $\mathcal{M}$.

Lemma 4 ([30]). *An implicit-deadline sporadic task system* $\mathbf{T}$ *is feasible upon a processing platform* $\mathcal{M}$*, if and only if, the following two conditions hold:*

$$\begin{aligned} u_{\text{sum}}(\mathbf{T}, N) &\leq S_M(\mathcal{M}), & (17) \\ u_{\text{sum}}(\mathbf{T}, k) &\leq S_k(\mathcal{M}), \text{ for all } k = 1, \ldots, M. & (18) \end{aligned}$$

The above lemma can be trivially weakened to obtain a necessary condition for feasibility of implicit-deadline sporadic task systems:

Corollary 1. *An implicit-deadline sporadic task system* $\mathbf{T}$ *is feasible upon a processing platform* $\mathcal{M}$*, if the following two conditions hold:*

$$u_{\text{sum}}(\mathbf{T}, N) \leq S_M(\mathcal{M}), \tag{19}$$

and

$$u_{\max}(\mathbf{T}) \leq s_{\pi(1)}. \tag{20}$$

Necessary conditions for constrained-deadline sporadic task systems can be obtained using $\mathsf{load}(\mathbf{T}, i)$ and $\delta_{\max}(\mathbf{T}, i)$:

Lemma 5 ([33]). *If a constrained-deadline task system* $\mathbf{T}$ *is feasible upon a processing platform* $\mathcal{M}$*, then for all* i $(1 \leq i \leq N)$*,*

$$\mathsf{load}(\mathbf{T}, i) \leq S_M(\mathcal{M}), \tag{21}$$

and

$$\delta_{\max}(\mathbf{T}, i) \leq s_{\pi(1)}. \tag{22}$$

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 feasibility conditions are satisfied. Let us first consider a derivation of a tight lower bound on temperature for implicit-deadline sporadic tasks. Let $\Pi(\mathcal{M})$ be the set of all permutations of $\{1, 2, \ldots, M\}$. Thus, any $\pi \in \Pi(\mathcal{M})$ is a function $\pi : \{1, 2, \ldots, M\} \mapsto \{1, 2, \ldots, M\}$. In the necessary and sufficient conditions of Lemma 4, the second condition (Equation 18) states that the k'th fastest processors must have total computational capacity greater than the k'th largest utilization tasks. Therefore, we need to consider all permutations of processors as candidates for the different relative orderings according to speed. Based on the necessary and sufficient condition for feasibility of implicit-deadline tasks (Lemma 4) and the peak temperature formula in Section 3.1, the lower bound Θ^*_π on peak temperature, for a specified permutation of processors, $\pi \in \Pi(\mathcal{M})$, can be obtained by solving the following non-linear programming (denoted $\mathsf{SYSTEM}^*([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, \pi)$):

$$\begin{aligned}
&\text{minimize } \Theta^*_\pi \overset{\text{def}}{=} \max_{1 \le j \le M+\hbar} \left\{ \sum\nolimits_{\ell=1}^{M+\hbar} -V_{j,\ell}(\alpha s_\ell^\gamma + B_\ell) \right\} \\
&\text{subject to } u_{\text{sum}}(\mathbf{T}, N) \le \sum\nolimits_{\ell=1}^{M} s_\ell, \\
&\qquad u_{\text{sum}}(\mathbf{T}, k) \le s_{\pi(k)}, \qquad\qquad 1 \le k \le M \\
&\qquad s_\ell \ge 0, \qquad\qquad 1 \le \ell \le M + \hbar. \qquad (23)
\end{aligned}$$

Obviously, an optimal solution to (23) 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.

The minimum among $\{\mathsf{SYSTEM}^*([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, \pi) : \pi \in \Pi(\mathcal{M})\}$ is a "tight" lower bound $\Theta^*_{\min}$ of the peak temperature. (The bound is tight since we derived it from a necessary and sufficient condition). Denote $\pi_{\min} \overset{\text{def}}{=} \arg\min\{\Theta^*_\pi : \pi \in \Pi(\mathcal{M})\}$ and $\Theta^*_{\min} \overset{\text{def}}{=} \Theta^*_{\pi_{\min}}$. Let $\mathcal{M}_{\min}$ be the system corresponding to $\Theta^*_{\min}$ with the derived speeds $s_{\pi_{\min}(1)}, s_{\pi_{\min}(2)}, \ldots, s_{\pi_{\min}(M)}$. An optimal multiprocessor global scheduling algorithm for implicit-deadline task systems upon uniform multiprocessors (e.g., see [34]) may be used to schedule $\mathbf{T}$ upon $\mathcal{M}_{\min}$ obtaining the minimum obtainable peak temperature. However, for other online scheduling algorithms such as EDF and DM, we must further modify the speeds of $\mathcal{M}_{\min}$ before we can ensure that all deadlines of $\mathbf{T}$ will be met. Section 4 will describe algorithms for determine the values of speeds to ensure EDF and DM schedulability.

A major drawback of the above approach is that it requires calculation of $\mathsf{SYSTEM}^*([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, \pi)$ for all $\pi \in \Pi(\mathcal{M})$. However, there are $M!$ elements of $\Pi(\mathcal{M})$. We may reduce the overall complexity of determining a lower bound on temperature, if we use instead the necessary conditions (Corollary 1 and Lemma 5) on feasibility. Note the second inequality of both the necessary conditions for implicit-deadline and constrained-deadline systems require the fastest processor to have sufficient computational capacity to accommodate the "largest" task in the system; thus, 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. The lower bound Θ^*_r, for a specified r, of the peak temperature can be obtained by solving the following non-linear programming (denoted $\mathsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$):

$$\begin{aligned}
&\text{minimize } \Theta_r^* \overset{\text{def}}{=} \max_{1 \le j \le M+\hbar} \left\{ \sum\nolimits_{\ell=1}^{M+\hbar} -V_{j,\ell}(\alpha s_\ell^\gamma + B_\ell) \right\} \\
&\text{subject to } W(\mathbf{T}, N) \le \sum\nolimits_{\ell=1}^{M} s_\ell, \\
&\qquad L(\mathbf{T}, N) \le s_r, \\
&\qquad s_\ell \ge 0, 1 \le \ell \le M + \hbar. \qquad (24)
\end{aligned}$$

$W(\mathbf{T}, N)$ equals $u_{\text{sum}}(\mathbf{T}, N)$ (resp., $\mathsf{load}(\mathbf{T}, N)$), if $\mathbf{T}$ is an implicit-deadline (resp., constrained-deadline) sporadic task system. Similarly, $L(\mathbf{T}, N)$ equals $u_{\max}(\mathbf{T})$ (resp., $\delta_{\max}(\mathbf{T}, N)$), if $\mathbf{T}$ is an implicit (resp., constrained-deadline) sporadic task system.

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

To our best knowledge, there is no explicit form for an optimal solution of $\mathsf{SYSTEM}^*([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, \pi)$ or $\mathsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$. Here, we adopt the approach proposed by Dutta and Vidyasagar [35] 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 [35]. 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 [28].

The following theorem shows that $\Theta^*_{\min}$ is the lower bound of the peak temperature for feasible speed scheduling [1]:

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 (EDF or DM).

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):

Lemma 6. *$S_\ell(\beta \cdot \mathcal{M}) = \beta \cdot S_\ell(\mathcal{M})$ 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

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

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 on implicit-deadline task systems may be achieved by simply deriving a β that satisfies Lemmas 1 or 2. The following theorem obtains such a bound. The theorems directly follow by solving Equations 11 and 12 (respectively) for the speed-scaling factor β.

Theorem 2. *For constrained-deadline sporadic task system* $\mathbf{T}$ *and* $\mathcal{M}_{\min}$*,* $\mathbf{T}$ *is globally* EDF*-schedulable upon* $\beta^{I}_{\mathrm{EDF}} \cdot \mathcal{M}_{\min}$ *where* β^{I}_{EDF} *is defined as*

$$\frac{u_{\mathrm{sum}}(\mathbf{T}, N) + \lambda(\mathcal{M}_{\min}) \cdot \max\left\{u_1, \frac{u_{\mathrm{sum}}(\mathbf{T},N)}{M}\right\}}{S_M(\mathcal{M}_{\min})}. \tag{25}$$

Theorem 3. *For constrained-deadline sporadic task system* $\mathbf{T}$ *and* $\mathcal{M}_{\min}$*,* $\mathbf{T}$ *is globally* DM*-schedulable upon* $\beta^{I}_{\mathrm{DM}} \cdot \mathcal{M}_{\min}$ *where* β^{I}_{DM} *is defined as*

$$\frac{2u_{\mathrm{sum}}(\mathbf{T}, N) + \widehat{\lambda}(\mathcal{M}_{\min}) \cdot u_{\max}(\mathbf{T})}{S_M(\mathcal{M}_{\min})}. \tag{26}$$

A pessimistic bound on β for global EDF and DM on constrained-deadline task systems may be achieved by simply deriving a β that satisfies Lemma 3. The following theorem (which follows a similar argument to Lemma 5 in [33]) obtains such a bound.

Theorem 4. *For constrained-deadline sporadic task system* $\mathbf{T}$ *and* $\mathcal{M}_{\min}$ *(called* $\mathcal{M}$ *below),* $\mathbf{T}$ *is globally* $\mathcal{S}$*-schedulable (*$\mathcal{S}$ *is either* EDF *or* DM*) upon*

$\beta^C_{\mathcal{S}} \cdot \mathcal{M}_{\min}$ *where* $\beta^C_{\mathcal{S}}$ *is defined as*

$$\Bigg[S_M(\mathcal{M})(s_{\pi(1)} + \phi_{\mathcal{S}} s_{\pi(M)}) + \lambda(\mathcal{M}) s_{\pi(1)} s_{\pi(M)} + \bigg(\Big(S_M(\mathcal{M})(s_{\pi(1)} + \phi_{\mathcal{S}} s_{\pi(M)}) + \lambda(\mathcal{M}) s_{\pi(1)} s_{\pi(M)} \Big)^2 - 4 S_M(\mathcal{M}) \lambda(\mathcal{M}) s^2_{\pi(1)} s_{\pi(M)} \bigg)^{\frac{1}{2}} \Bigg] \Big(2 S_M(\mathcal{M}) s^2_{\pi(M)} \Big)^{-1} \tag{27}$$

where $\phi_{\mathcal{S}}$ *is defined in (16).*

Using the above theorems, we can obtain an approximation ratios (in terms of the ideal-processor speeds) for the peak temperature of the system, using the speedup-factor bounds (Theorems 2, 3, and 4)

Theorem 5. *The peak temperature of* $\beta^X_{\mathcal{S}} \cdot \mathcal{M}_{\min}$ *(where* $\mathcal{S}$ *is either* EDF *or* DM *and* X *is either* C *or* I*) is at most a factor of* $\beta^{\gamma}_{\mathcal{S}}$ *greater than the peak temperature of the optimal* M*-processor platform on which task system* $\mathbf{T}$ *is globally schedulable.*

4.2. Deriving a Better Feasible Speed Scheduling for Constrained-Deadline Systems

The above analysis for constrained-deadline task systems 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 7. *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*

$$\Gamma(\mathcal{M}, \mathbf{T}, \ell, i) < \beta \leq \Gamma(\mathcal{M}, \mathbf{T}, \ell+1, i) \tag{28}$$

where

$$\Gamma(\mathcal{M}, \mathbf{T}, \ell, i) \overset{def}{=} \begin{cases} \frac{\lambda(\mathcal{M}) \cdot \delta_{\max}(\mathbf{T}, i)}{S_M(\mathcal{M}) - S_\ell(\mathcal{M})}, & 0 \leq \ell < M-1 \\ \infty, & \textit{otherwise.} \end{cases} \tag{29}$$

Given the input task workload, by Lemma 6 we may simply solve (13) in Lemma 3 as shown in the following lemma (the proof is straightforward):

Lemma 8. *For global scheduler* $\mathcal{S}$ *(either* EDF *or* DM*), if* $\mathbf{T}$ *satisfies (13), then there exists* $\ell \in \{0, 1, \ldots, M-1\}$*, equal to* $\nu(\beta \cdot \mathcal{M}, \mathbf{T}, i)$*, such that*

$$\beta \geq \widehat{\Gamma}(\mathcal{S}, \mathcal{M}, \mathbf{T}, \ell, i), \tag{30}$$

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

$$\begin{aligned} \widehat{\Gamma}(\mathcal{S}, \mathcal{M}, \mathbf{T}, \ell, i) \;\; & \overset{def}{=} \;\; \frac{1}{S_M(\mathcal{M})}(\phi_{\mathcal{S}} \cdot \mathsf{load}(\mathbf{T}, i) \\ & \qquad + (\lambda(\mathcal{M}) + \ell)\delta_{\max}(\mathbf{T}, i)), \end{aligned} \tag{31}$$

and $\phi_{\mathcal{S}}$ *is defined in (16).*

Next we aim to find the minimum β that satisfy Lemmas 7 and 8 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

$$\begin{aligned}\ell_{\min,i} \overset{\text{def}}{=}\; & \min\{\ell \in \{0, 1, \ldots, M-1\} : \\ & \Gamma(\mathcal{M}_{\min}, \mathbf{T}, \ell, i) < \hat{\Gamma}(\mathcal{S}, \mathcal{M}_{\min}, \mathbf{T}, \ell, i) \\ & \leq \Gamma(\mathcal{M}_{\min}, \mathbf{T}, \ell+1, i)\}. \qquad (32)\end{aligned}$$

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

Theorem 6. *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*

$$\beta_{\text{EDF}} \overset{\text{def}}{=} \hat{\Gamma}(\text{EDF}, \mathcal{M}_{\min}, \mathbf{T}, \ell_{\min,N}, N); \qquad (33)$$

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

$$\beta_{\text{DM}} \overset{\text{def}}{=} \max_{i \in \{1,2,\ldots,N\}} \{\hat{\Gamma}(\text{DM}, \mathcal{M}_{\min}, \mathbf{T}, \ell_{\min,i}, i)\}, \qquad (34)$$

where $\hat{\Gamma}()$ *is defined in (31) and* $\ell_{\min,i}$ *is defined in (32).*

5. Performance Evaluation

This section provides performance evaluations of the proposed algorithm for speed assignments under global real-time scheduling. 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

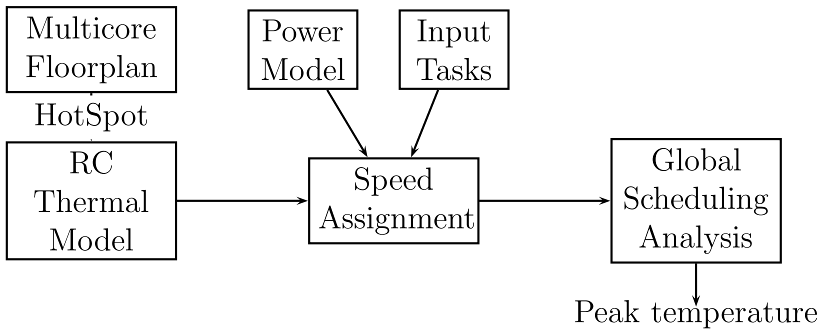


Figure 2: Simulation setup by using Hotspot thermal model.

possible, and then applies Theorem 6 for speed determination. Specifically, for the necessary condition, when $\frac{W(\mathbf{T},N)}{M} < L(\mathbf{T},N)$, one core is assigned with speed $L(\mathbf{T},N)$ and the other cores are with speed $\frac{W(\mathbf{T},N)-L(\mathbf{T},N)}{M-1}$, and we choose the one with the minimum peak temperature.

- Algorithm **PTO**: first applies sequential quadratic programming for deriving optimal solutions of (23), and then applies Theorem 6 for determining the resulting speeds.

5.1. Platform and Simulation Setup

We evaluate the performance in terms of peak temperature of the resulting speed assignments on three different hardware platforms, in which their layouts are 2×2, 4×1, and 4×2 with 4, 4, and 8 cores, respectively. We use HotSpot 4.1 simulator [36] to obtain the RC thermal model for the above platforms. The flowchart of the simulation is in Figure 2. Specifically, we use the thermal configuration for chip specs, heat sink specs, and head spreader specs in the simulator. We consider a core as a block for heat generation and

			4×1		2×2	
	Width	Height	Left (x)	Bottom (y)	Left(x)	Bottom (y)
Core 1	0.006	0.006	0	0	0	0
Core 2	0.006	0.006	0.008	0	0.008	0
Core 3	0.006	0.006	0.016	0	0	0.007
Core 4	0.006	0.006	0.024	0	0.008	0.007

			4×2	
	Width	Height	Left (x)	Bottom (y)
Core 1	0.003	0.006	0	0
Core 2	0.003	0.006	0.004	0.000
Core 3	0.003	0.006	0.000	0.007
Core 4	0.003	0.006	0.004	0.007
Core 5	0.003	0.006	0.0085	0.000
Core 6	0.003	0.006	0.0085	0.000
Core 7	0.003	0.006	0.0125	0.007
Core 8	0.003	0.006	0.0125	0.007

Table 1: Platforms in the simulations (units: meter)

dissipation by using coarse-grained specs. The details of the platforms are illustrated in Table 1.

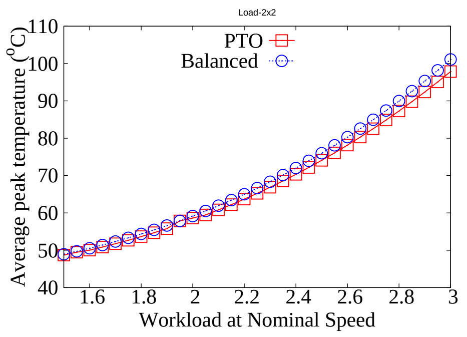
The power consumption function at the nominal speed s_{nom} on absolute temperature Θ_ℓ is assumed $30s_{nom}^3 + 6.9685 + 0.01\Theta_\ell$ Watt. That is, we assume $\Omega + \delta\Theta_a = 10$ Watt.

We use synthetic sporadic real-time tasks for evaluating the performance, in which the deadline of a task is earlier than its period. We consider different workloads for global EDF scheduling and global DM scheduling. For global EDF scheduling, on a given platform, the peak temperature of a speed assignment for Algorithm Balanced or Algorithm PTO depends on two parameters $\mathsf{load}(\mathbf{T}, N)$ and $\delta_{\max}$ only. Therefore, we perform evaluations for different values on $\mathsf{load}(\mathbf{T}, N)$ and $\delta_{\max}$, which covers for both implicit-deadline and constrained-deadline systems. We denote $\frac{\mathsf{load}(\mathbf{T},N)}{s_{nom}}$ as *workload at nominal speed* in the resulting figures. For global DM scheduling, we generate tasks with specified $\sum_{\tau_i \in \mathbf{T}} u_i$. The deadline d_i of a task τ_i is a random variable in $[100, 400]$, and the minimum inter-arrival separation parameter p_i is d_i for implicit-deadline systems or is a random variable in $[d_i, 1.2d_i]$ for constrained-deadline systems.

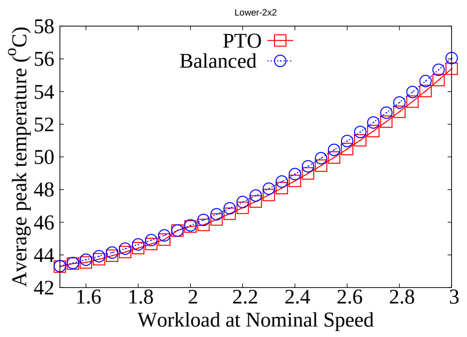
5.2. Simulation Results

Figure 3 presents the peak temperature of the resulting speed assignments of Algorithm Balanced and Algorithm PTO for EDF scheduling when δ_{max} is no more than the average workload on the M cores, i.e., $\delta_{\max} \leq \frac{\mathsf{load}(\mathbf{T},N)}{M}$. Figures 3(a), 3(c), and 3(e) are the results of feasible speed scheduling, while Figures 3(b), 3(d), and 3(f) are for preferred speeds. When the workload is low, 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 3(e). Similarly, when the workload is low, speeding up from the preferred speeds does not increase the resulting peak temperature very much, since the increase of the power consumption is quite limited. When the workload is higher, speedup factor β could significantly increase the power consumption, and leads to larger peak temperature difference between the preferred speeds and the resulting feasible speed scheduling.

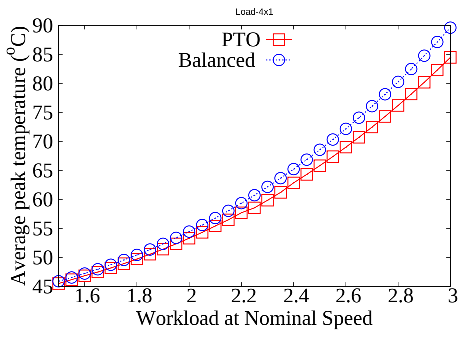
The temperature improvement of Algorithm PTO, compared to Algorithm Balanced, is highly dependent on the simulated platforms, in which



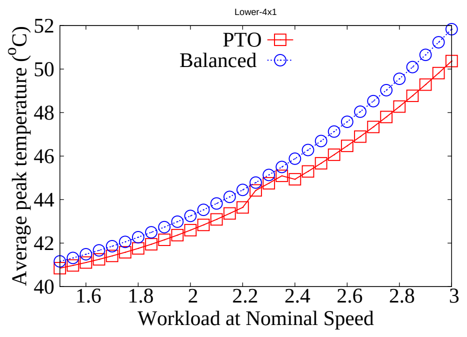
(a) Peak temperature of feasible speed scheduling on platform 2×2



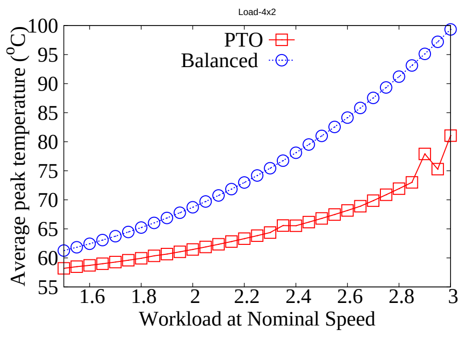
(b) Peak temperature of preferred speedson platform 2×2



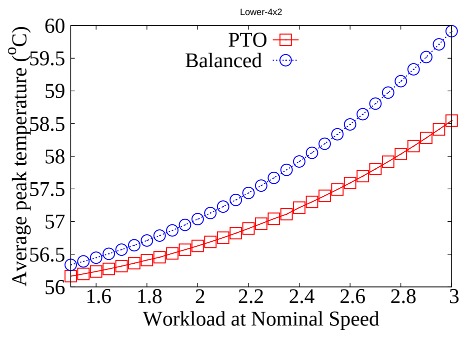
(c) Peak temperature of feasible speed schedulingon platform 4×1



(d) Peak temperature of preferred speeds on platform 4×1



(e) Peak temperature of feasible speed schedulingon platform 4×2



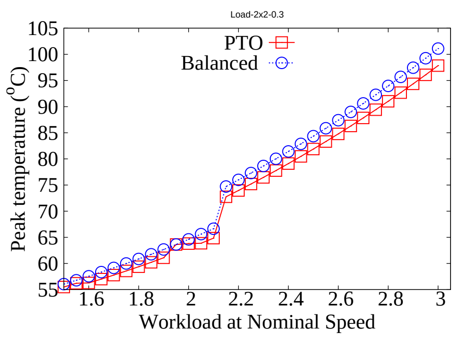
(f) Peak temperature of preferred speedson platform 4×2

Figure 3: Simulation results for EDF scheduling when $\delta_{max} \leq \frac{\mathsf{load}(\mathbf{T},N)}{M}$.

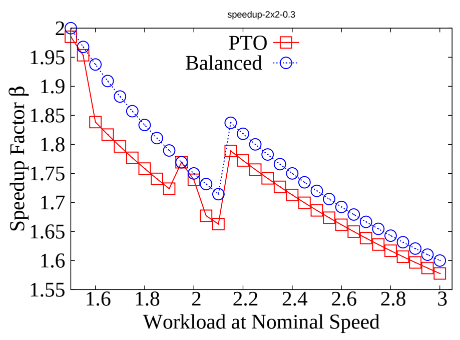
the improvement is at most 3.5°C for platform 2×2 in Figure 3(a), at most 5°C for platform 4×1 in Figure 3(c), and at most 22°C for platform 4×2 in Figure 3(e).

If δ_{max} is larger than the average workload on the M cores, i.e., $\delta_{\max} > \frac{\mathsf{load}(\mathbf{T},N)}{M}$, we always have to find a core to assign with speed $\delta_{\max}$. The performance of the algorithms highly depends on the value of $\delta_{\max}$. Figures 4, 5, and 6 illustrate the evaluation results for different values of $\delta_{\max}$ for EDF scheduling. For such cases, Algorithm Balanced could be better than Algorithm PTO for some cases, especially when $\frac{\delta_{\max}}{\mathsf{load}(\mathbf{T},N)}$ is large. This is because the peak temperature is (almost) dominated by the core with preferred speed $\delta_{\max}$. When we choose the preferred speeds, we try to optimize the speeds for the other cores. However, this affects the derivation of the speedup factor very much. For such a case, compared to the balanced speeds, the improvement on the peak temperature is quite limited by using the optimal preferred speeds. As shown in Figures 4(b), 4(d), 4(f), 5(b), 5(d), 5(f), 6(b), 6(d), and 6(f), the speedup factor matters. When the speedup factor of Algorithm PTO is less than that of Algorithm Balanced, the resulting peak temperature is less than Algorithm PTO. However, when the speedup factor of Algorithm PTO is larger than that of Algorithm Balanced, Algorithm PTO might be worst than Algorithm Balanced. Therefore, when $\delta_{\max} > \frac{\mathsf{load}(\mathbf{T},N)}{M}$, if $\frac{\delta_{\max}}{\mathsf{load}(\mathbf{T},N)}$ is relatively large, Algorithm Balanced could be better. This depends on how to minimize the speedup factor.

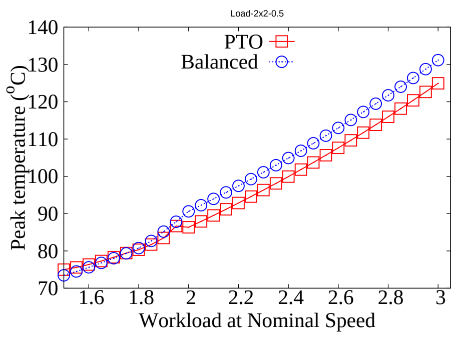
Figure 7 and Figure 8 show the evaluation results of DM scheduling for implicit-deadline and constrained-deadline systems when $\delta_{\max}$ is $0.4 s_{nom}$. The peak temperature is larger than global EDF scheduling, since the factor β is



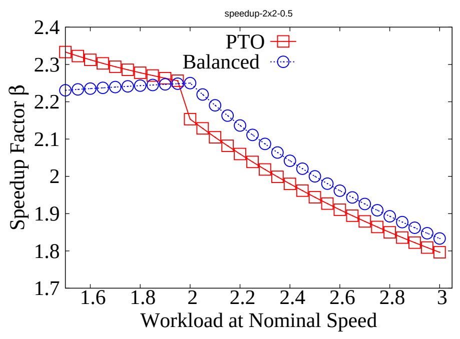
(a) $\delta_{\max} = 0.3s_{nom}$



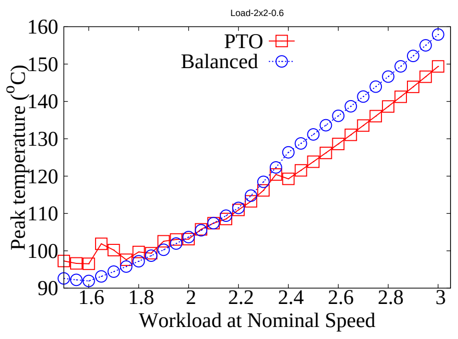
(b) $\delta_{\max} = 0.3s_{nom}$



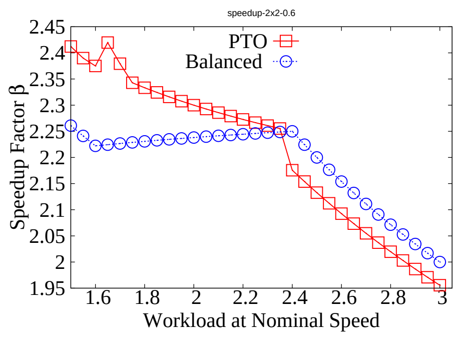
(c) $\delta_{\max} = 0.5s_{nom}$



(d) $\delta_{\max} = 0.5s_{nom}$



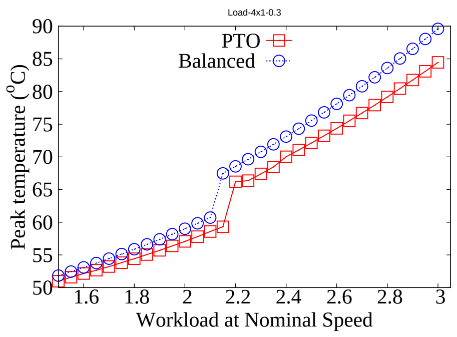
(e) $\delta_{\max} = 0.6s_{nom}$



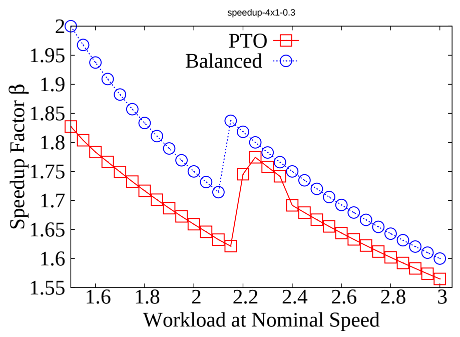
(f) $\delta_{\max} = 0.6s_{nom}$

Figure 4: Simulation results of EDF scheduling for platform with layout 2×2.

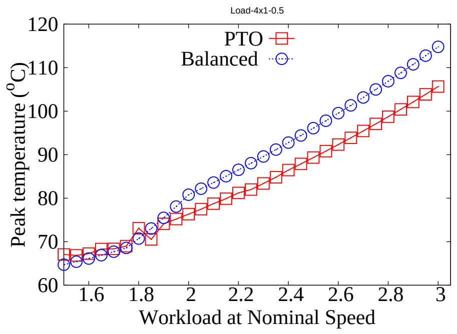
in general larger. For DM scheduling, Algorithm Balanced and Algorithm PTO have similar performance.



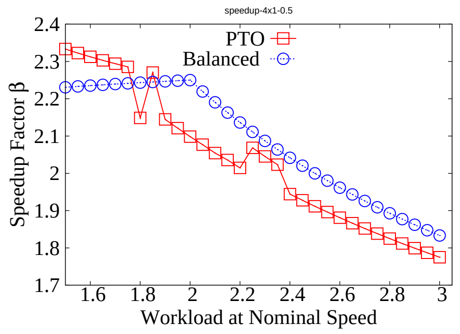
(a) $\delta_{\max} = 0.3 s_{nom}$



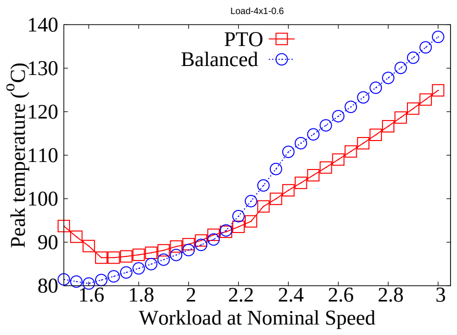
(b) $\delta_{\max} = 0.3 s_{nom}$



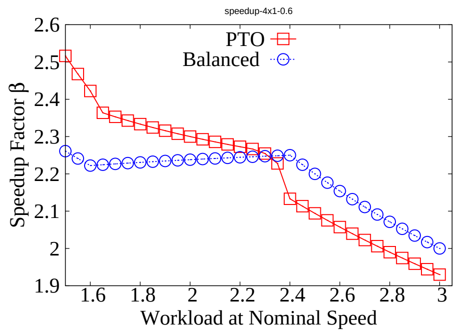
(c) $\delta_{\max} = 0.5 s_{nom}$



(d) $\delta_{\max} = 0.5 s_{nom}$



(e) $\delta_{\max} = 0.6 s_{nom}$

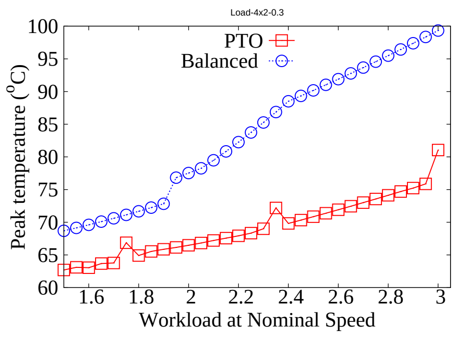


(f) $\delta_{\max} = 0.6 s_{nom}$

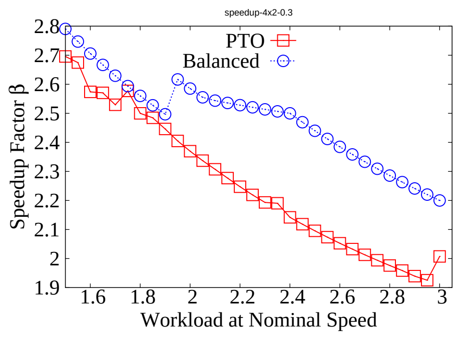
Figure 5: Simulation results of EDF scheduling for platform with layout 4×1.

6. Conclusion

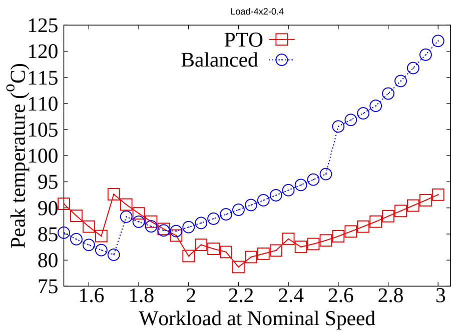
Thermal constraints are becoming increasingly severe for many systems as chip density increases and the size of the system decreases. Heat dissipation



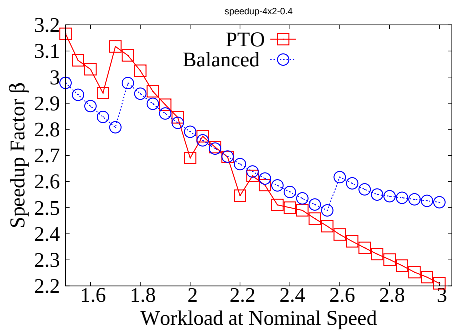
(a) $\delta_{\max} = 0.3s_{nom}$



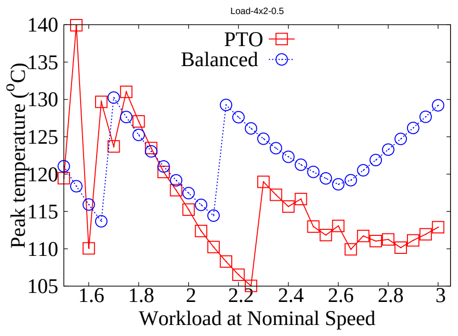
(b) $\delta_{\max} = 0.3s_{nom}$



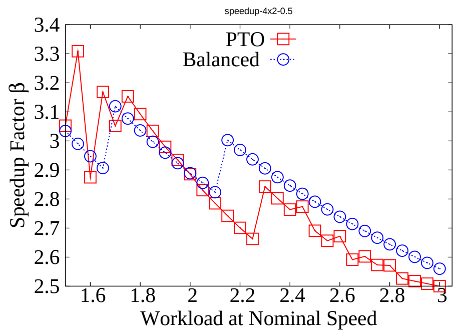
(c) $\delta_{\max} = 0.4s_{nom}$



(d) $\delta_{\max} = 0.4s_{nom}$



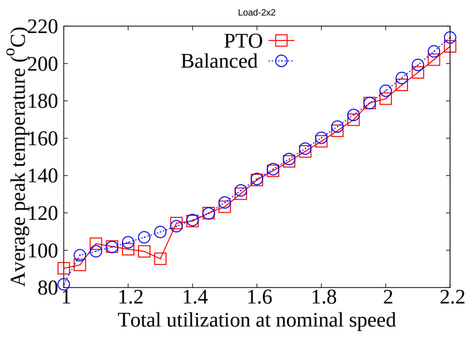
(e) $\delta_{\max} = 0.5s_{nom}$



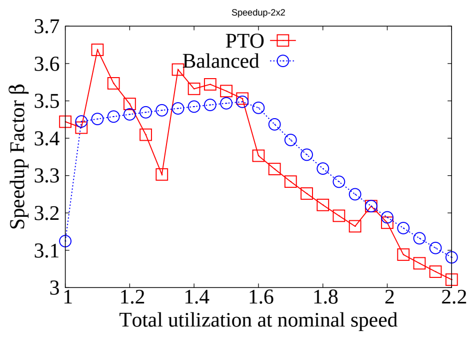
(f) $\delta_{\max} = 0.5s_{nom}$

Figure 6: Simulation results of EDF scheduling for platform with layout 4×2.

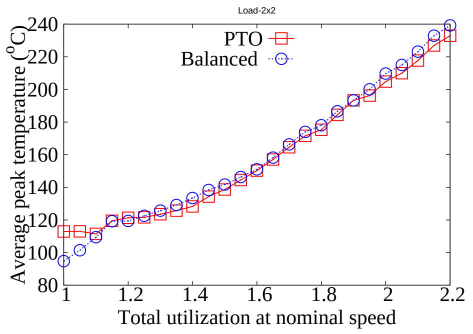
in multicore platforms further complicates satisfying thermal constraints due to the transfer of heat between cores on the same chip. In order to respect these constraints, system designers may scale-back the power-consumption to



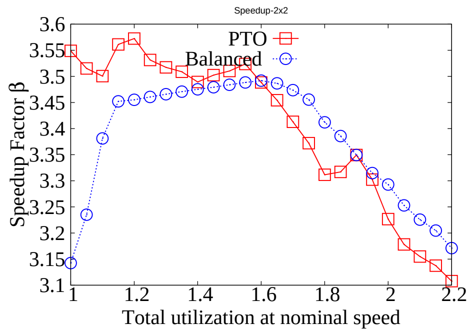
(a) implicit deadline



(b) implicit deadline



(c) constrained deadline

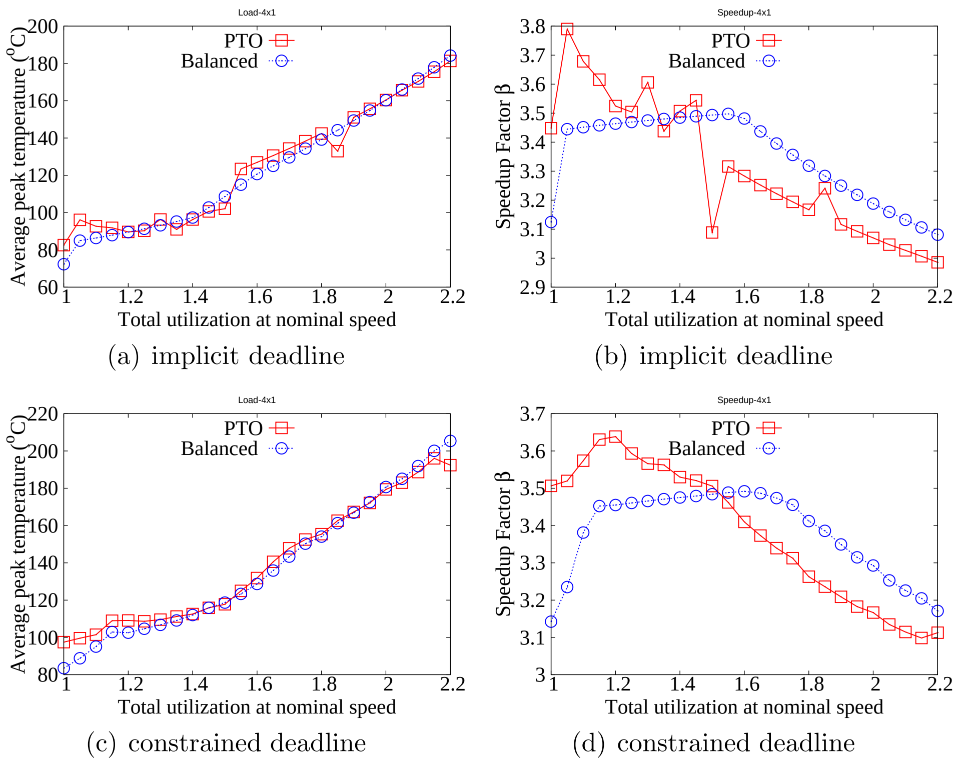


(d) constrained deadline

Figure 7: Simulation results of DM scheduling for the platform with layout 2×2 when $\delta_{\max} = 0.4 s_{nom}$.

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.



(a) implicit deadline (b) implicit deadline

(c) constrained deadline (d) constrained deadline

Figure 8: Simulation results of DM scheduling for the platform with layout 4×1 when $\delta_{\max} = 0.4 s_{nom}$.

The resulting platform executing at the preferred 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 via simulations over synthetically generated task systems. 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).

Appendix A. Solving $\mathsf{SYSTEM}([\boldsymbol{\mathcal{A}}], \vec{\boldsymbol{B}}, \vec{\boldsymbol{P}}, \mathbf{T}, \boldsymbol{r})$

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:

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

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

Next, starting from $\Theta^*_{r,0}$, we approach the optimal solution of $\mathsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$ 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:

$$\sum_{j=1}^{M+\hbar}\left[\max\left\{0, \sum_{\ell=1}^{M+\hbar} -V_{j,\ell}(\alpha s_\ell^\gamma + B_\ell) - \Theta^*_{r,k-1}\right\}\right]^2 \tag{A.2}$$

$$+\epsilon_1\left[\max\left\{0, \delta_{\max}(\mathbf{T}, N) - s_r\right\}\right]^2 \tag{A.3}$$

$$+\epsilon_2\left[\mathsf{load}(\mathbf{T}, N) - \sum_{\ell=1}^{M} s_\ell\right]^2, \tag{A.4}$$

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 (A.2) 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 [35], the resulting speed assignment with the converged $\Theta^*_{r,k}$ is the optimal solution of $\mathsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$, when ϵ_1 and ϵ_2 are large numbers.

Appendix B. Proof of Theorem 1

Let $\mathcal{M}$ be the platform defined by processor speeds $s_1, s_2, \ldots, s_M$. By Lemma 5, if $\mathbf{T}$ is schedulable (either EDF or DM) upon $\mathcal{M}$ then $\mathsf{load}(\mathbf{T}, i) \leq \mathsf{load}(\mathbf{T}, N) \leq \sum_{\ell=1}^{M} s_\ell = S_M(\mathcal{M})$ 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 $\mathsf{SYSTEM}([\mathcal{A}], \vec{B}, \vec{P}, \mathbf{T}, r)$, the set

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

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

Appendix C. Proof of Theorem 4

The satisfaction of Lemma 3 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 for $i = N$ when $\mathcal{S}$ is EDF; for $\mathcal{S}$ equal to DM, the condition must hold for all $i = 1, \ldots, N$.

$$\begin{aligned}\phi_{\mathcal{S}}\mathsf{load}(\mathbf{T}, i) \quad &\leq \mu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, i) \\ &\quad -\nu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, i)\delta_{\max}(\mathbf{T}, i) \\ &\Leftarrow \left(\text{since } \left\lceil \frac{\mu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, i)}{\beta \cdot s_{\pi(M)}} \right\rceil - 1 \right. \\ &\quad \left. \geq \nu(\beta \cdot \mathcal{M}_{\min}, \mathbf{T}, i)\right)\end{aligned}$$

$$\phi_{\mathcal{S}}\mathsf{load}(\mathbf{T},i) \le \mu(\beta\cdot\mathcal{M}_{\min},\mathbf{T},i) - \left(\left\lceil\tfrac{\mu(\beta\cdot\mathcal{M}_{\min},\mathbf{T},i)}{\beta\cdot s_{\pi(M)}}\right\rceil - 1\right)\delta_{\max}(\mathbf{T},i)$$

$$\Leftarrow \text{(since for all } \alpha, \lceil\alpha\rceil - 1 \le \alpha)$$

$$\phi_{\mathcal{S}}\mathsf{load}(\mathbf{T},i) \le \mu(\beta\cdot\mathcal{M}_{\min},\mathbf{T},i) - \left(\tfrac{\mu(\beta\cdot\mathcal{M}_{\min},\mathbf{T},i)}{\beta\cdot s_{\pi(M)}}\right)\delta_{\max}(\mathbf{T},i)$$

$$\equiv$$

$$\phi_{\mathcal{S}}\mathsf{load}(\mathbf{T},i) \le \mu(\beta\cdot\mathcal{M}_{\min},\mathbf{T},i)\left(1 - \tfrac{\delta_{\max}(\mathbf{T},i)}{\beta\cdot s_{\pi(M)}}\right)$$

$$\equiv \text{(by the definition of } \mu)$$

$$\phi_{\mathcal{S}}\mathsf{load}(\mathbf{T},i) \le [S_M(\beta\cdot\mathcal{M}_{\min}) - \lambda(\beta\cdot\mathcal{M}_{\min})\delta_{\max}(\mathbf{T},i)] \times \left(1 - \tfrac{\delta_{\max}(\mathbf{T},i)}{\beta\cdot s_{\pi(M)}}\right)$$

$$\equiv \text{(by Lemmas 6)}$$

$$\phi_{\mathcal{S}}\mathsf{load}(\mathbf{T},i) \le (\beta\cdot S_M(\mathcal{M}_{\min}) - \lambda(\mathcal{M}_{\min})\delta_{\max}(\mathbf{T},i)) \times \left(1 - \tfrac{\delta_{\max}(\mathbf{T},i)}{\beta\cdot s_{\pi(M)}}\right)$$

$$\Leftarrow \Big(\text{constraints } (\mathsf{load}(\mathbf{T},i) \le S_M(\mathcal{M}_{\min})) \wedge \left(\delta_{\max}(\mathbf{T},i) \le s_{\pi(1)} \text{ of } \mathsf{SYSTEM}\right)\Big)$$

$$\phi_{\mathcal{S}}S_M(\mathcal{M}_{\min}) \le \left(\beta\cdot S_M(\mathcal{M}_{\min}) - \lambda(\mathcal{M}_{\min})s_{\pi(1)}\right) \times \left(1 - \tfrac{s_{\pi(1)}}{\beta\cdot s_{\pi(M)}}\right)$$

$$\equiv$$

$$s_{\pi(M)}S_M(\mathcal{M}_{\min})\beta^2 - [(s_{\pi(1)} + \phi_{\mathcal{S}}s_{\pi(M)})S_M(\mathcal{M}_{\min}) + \lambda(\mathcal{M}_{\min})s_{\pi(1)}s_{\pi(M)}]\beta + \lambda(\mathcal{M}_{\min})s_{\pi(1)}^2 \ge 0.$$

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

Appendix D. Proof of Theorem 5

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 (23), $-V_{j,\ell}$ 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 $\beta_{\mathcal{S}}^{\gamma}$.

Appendix E. Proof of Lemma 7

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,

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

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.

References

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

[2] N. Bansal, K. Pruhs, Speed scaling to manage temperature, in: Symposium on Theoretical Aspects of Computer Science, 2005.

[3] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, D. Tarjan, Temperature-aware microarchitecture, in: International Symposium on Computer Architecture, 2003.

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

[5] D. Brooks, M. Martonosi, Dynamic thermal management for high-performance microprocessors, in: International Symposium on High-Performance Computer Architecture, 2001.

[6] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: Symposium on Foundations of Computer Science, 1995.

[7] R. Jejurikar, C. Pereira, R. Gupta, Leakage aware dynamic voltage scaling for real-time embedded systems, in: the Design Automation Conference, 2004.

[8] S. Wang, R. Bettati, Reactive speed control in temperature-constrained

real-time systems, in: Euromicro Conference on Real-Time Systems, 2006.

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

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

[11] J.-J. Chen, S. Wang, 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.

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

[13] S. Zhang, K. S. Chatha, Approximation algorithm for the temperature-aware scheduling problem, in: International Conference on Computer-Aided Design, 2007.

[14] J.-J. Chen, C.-M. Hung, 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.

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

[16] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, 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. doi:http://doi.acm.org/10.1145/1289816.1289845.

[17] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, G. D. Micheli, Temperature control of high-performance multi-core platforms using convex optimization, in: DATE, 2008, pp. 110–115.

[18] M. Kadin, S. Reda, Frequency planning for multi-core processors under thermal constraints, in: ISLPED, 2008, pp. 213–216.

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

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

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

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

[23] R. Xu, D. Zhu, C. Rusu, R. Melhem, D. Moss, Energy efficient policies

for embedded clusters, in: ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems, 2005.

[24] W. Liao, L. He, 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) (2005) 1042–1053.

[25] Y. Liu, R. P. Dick, L. Shang, H. Yang, Accurate temperature-dependent integrated circuit leakage power estimation is easy, in: DATE, 2007, pp. 1526–1531.

[26] A. K. Mok, Fundamental design problems of distributed systems for the hard-real-time environment, Ph.D. thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, available as Technical Report No. MIT/LCS/TR-297 (1983).

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

[28] N. Fisher, T. Baker, 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, IEEE Computer Society Press, Sydney, Australia, 2006.

[29] S. H. Funk, EDF scheduling on heterogeneous multiprocessors, Ph.D.

thesis, Department of Computer Science, The University of North Carolina at Chapel Hill (2004).

[30] S. Funk, J. Goossens, S. Baruah, On-line scheduling on uniform multiprocessors, in: Proceedings of the IEEE Real-Time Systems Symposium, IEEE Computer Society Press, 2001, pp. 183–192.

[31] S. Baruah, J. Goossens, Rate-monotonic scheduling on uniform multiprocessors, IEEE Transactions on Computers 52 (7) (2003) 966–970.

[32] S. Baruah, J. Goossens, The edf scheduling of sporadic task systems on uniform multiprocessors, in: IEEE Real-Time Systems Symposium, 2008.

[33] S. Baruah, J. Goossens, Deadline monotonic scheduling on uniform multiprocessors, in: International Conference on Principles of Distristributed Systems (OPODIS), 2008.

[34] S.-Y. Chen, C.-W. Hsueh, Optimal dynamic-priority real-time scheduling algorithms for uniform multiprocessors, Proceedings of the IEEE Real-Time Systems Symposium (2008) 147–156.

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

[36] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, M. R. Stan, Hotspot: A compact thermal modeling methodology for early-stage vlsi design, IEEE Trans. VLSI Syst. 14 (5) (2006) 501–513.