Proactive Speed Scheduling for Real-Time Tasks under Thermal Constraints

Jian-Jia Chen
Computer Engineering and Networks Laboratory (TIK)
ETH Zurich, Switzerland
jchen@tik.ee.ethz.ch

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

Lothar Thiele
Computer Engineering and Networks Laboratory (TIK)
ETH Zurich, Switzerland
thiele@tik.ee.ethz.ch

Abstract

Thermal management becomes a prominent issue in system design for both server systems and embedded systems. A system could fail if the peak temperature exceeds its thermal constraint. This research studies thermal-constrained scheduling for frame-based real-time tasks on a dynamic voltage/speed scaling system. Our objective is to design speed schedulers for real-time tasks by utilizing dynamic voltage/speed scaling to meet both timing and thermal constraints. Two approaches are proposed: One is based on the minimization of the response time under the thermal constraint, and the other is based on the minimization of the temperature under the timing constraint. We present detailed schedulability analysis for both proposed approaches. Our data show that our proposed proactive approaches outperform existing reactive ones.

Keywords: speed scheduling, dynamic voltage/speed scaling, thermal management.

1. Introduction

With the dramatic increase of power consumption for modern processors in both server systems and embedded systems, thermal management becomes a prominent issue in system design. A system could fail if the peak temperature exceeds its thermal constraint. 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 power management. At design time, the packaging cost of cooling systems grows exponentially as the power consumption of computing devices increases. Current estimates are that cooling solutions are rising at \$1 to \$3 per watt of heat dissipated [12]. Alternatively, at run time with dynamic power management the reduction of power consumption can enhance the system reliability, reduce the packaging cost, and prolong the battery life. For modern processors, Both switching activities and the leakage current are the two major sources of power consumption by a processing unit. To reduce the power consumption due to switching activities, one can apply the dynamic voltage/speed scaling (DVS) approach, in which different supply voltages lead to different execution speeds. For instance, Dynamic Thermal Management (DTM) [3], [7], [12] adopts DVS to prevent the system from overheating. Technologies, such as Intel SpeedStep® and AMD PowerNOW!™, now provide DVS techniques. Well-known example processors for embedded systems are Intel StrongARM SA1100 processor and the Intel XScale.

The objective of the thermal-constrained scheduling is to optimize the system performance under the thermal constraint. In [1], an algorithm was developed to maximize the workload that can complete in a specified time window without violating the thermal constraints. In [14], [15], a reactive speed control scheme was proposed and schedulability tests was presented for frame-based real-time tasks, in which all the tasks have the same period. In [13], delay analysis is performed under reactive speed control for general task arrivals. In [19], approximation algorithms were developed to minimize the completion time, while each task is restricted to execute at one speed. For multiprocessor systems, thermal-aware scheduling has also been explored [4], [6], [8], [9]. Most approaches assume fixed power consumption on a processor, and focus on the issue for the heat transfer between cores/processors, which is not in the scope of this paper.

In this paper, we will explore thermal-constrained speed scheduling of a set of frame-based real-time tasks with the same period. Specifically, we develop *proactive* speed scheduling by utilizing DVS for real-time tasks under the thermal and timing constraints. Different from *reactive* speed scheduling [13]–[15] by using two positive speeds only, more general speed scheduling is allowed in our proactive design. Reactive speed scheduling has the disadvantage that they react only after the system reaches the hardware thermal constraint. Proactive ones can define the processing speed in response to both (asynchronous) thermal triggers, and (synchronous) job-scheduler-issued requests for speed change. In proactive speed scheduling, a control strategy could be used to determine how the speed is chosen over the whole time domain. In this paper, we develop two approaches to proactive speed scheduling:

- *Timing-optimization approach*: We fix the maximum thermal constraint and design a speed scheduling to minimize the completion time of task instances.

- *Thermal-optimization approach*: We fix both timing constraint and maximum thermal constraint and design a speed scheduling to minimize the temperature at the beginning of the period.

We provide the detailed description of proactive speed scheduling and theoretical schedulability analysis for each approach. Our data show that our proposed proactive speed scheduling with both approaches outperforms existing reactive ones.

The rest of this paper is organized as follows: Section 2 shows the system model and the problem definition. Section 3 provides several important lemmas. Section 4 presents the detailed schedulability analysis. Section 5 addresses some practical design issues and the performance evaluation is presented in Section 6. We conclude this paper in Section 7.

2. System Model and Problem Definition

We explore thermal-aware scheduling on DVS processors. There are two major sources of power consumption $\Psi()$ by a processing unit:

- *Speed-dependent power consumption $\Psi_{dep}()$:* It mainly results from the charging and discharging of gates on the circuits. It is generally modelled as a convex function of the processor speed such as the dynamic power consumption in CMOS processors [10]: $\Psi_{dep}(s) = C_{ef} V_{dd}^2 s$, where $s = \kappa_v \frac{(V_{dd}-V_t)^2}{V_{dd}}$. [1] We can further scale s and simplify it as $\Psi_{dep}(s) = s^\gamma$, where $\gamma \leq 3$.
- *Speed-independent power consumption $\Psi_{ind}()$:* It mainly results from leakage current. It could be modelled as a nonnegative constant when leakage power consumption is irrelevant to the temperature [5], [17]. When the leakage power consumption is related to the temperature, it could be approximately modelled by a linear function of the temperature [4]: $\Psi_{ind}(\Theta) = \delta\Theta + \rho$, where Θ is the temperature and δ and ρ are constants.

In this paper, we use the following formula as the overall power consumption

$$\Psi(s, \Theta) = \Psi_{dep}(s) + \Psi_{ind}(\Theta) = s^\gamma + \delta\Theta + \rho, \quad (1)$$

where $\gamma \leq 3$. Both the speed $s = s(t)$ and temperature $\Theta = \Theta(t)$ are functions of time t.

Although cooling in a processor unit is a complicated physical process, it could be approximately modelled by applying Fourier's Law. This is used in most existing thermal-aware system designs, such as those in [1], [2], [4], [11], [13]–[15], [18], [19]. This paper adopts such approximation on the cooling process:

$$\begin{aligned} \Theta'(t) &= \hat{\alpha}\Psi(s(t), \Theta(t)) - \hat{\beta}(\Theta(t) - \Theta_a) \\ &= \hat{\alpha}s^\gamma(t) - (\hat{\beta} - \hat{\alpha}\delta)\Theta(t) + (\hat{\alpha}\rho + \hat{\beta}\Theta_a), \quad (2) \end{aligned}$$

where the ambient temperature Θ_a is assume fixed, $\hat{\alpha}$ is the coefficient for heating, and $\hat{\beta}$ is the coefficient for cooling. For notational simplicity, if we denote $\beta = \hat{\beta} - \hat{\alpha}\delta$ and define the *adjusted temperature* [2] as $\theta(t) \overset{\text{def}}{=} \frac{\Theta(t)}{\hat{\alpha}} - \frac{\hat{\alpha}\rho + \hat{\beta}\Theta_a}{\hat{\alpha}(\hat{\beta} - \hat{\alpha}\delta)}$, then (2) can be simplified as

$$\theta'(t) = s^\gamma(t) - \beta\theta(t). \quad (3)$$

If we know the initial temperature $\theta(t_0)$ at t_0, then we have

$$\theta(t) = \int_{t_0}^{t} s^\gamma(\tau) e^{-\beta(t-\tau)} \mathrm{d}\tau + \theta(t_0) e^{-\beta(t-t_0)}. \quad (4)$$

We assume that θ^* is the hardware thermal constraint in the processor. The *equilibrium* speed s_E is defined as the maximum constant speed that maintains the temperature under the thermal constraint. Based on (3), we can obtain s_E as

$$s_E = (\beta\theta^*)^{\frac{1}{\gamma}}. \quad (5)$$

This paper considers periodic real-time tasks. A periodic task is an infinite sequence of task instances, referred to as jobs. We focus on a set $\mathbf{T}$ of N frame-based periodic real-time, in which all tasks have the same period P [16]. The amount of the required computation cycles of task $T_i \in \mathbf{T}$ is C_i. The relative deadline of task T_i is D_i, in which $D_i \leq P$. We define Δ_i as the response time for task T_i. We assume an Earlier Deadline First (EDF) scheduling is used for job arrivals.

The thermal-constrained frame-based task scheduling problem defined in this paper aims to find a feasible speed scheduling for task set $\mathbf{T}$ so that all the tasks meet two constraints: (i) Timing constraint: $\Delta_i \leq D_i$; (ii) Thermal constraint: $\theta(t) \leq \theta^*$, $\forall t \geq 0$.

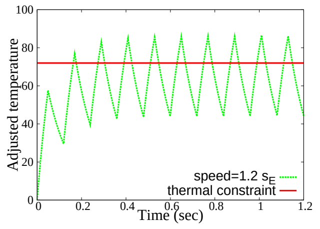
3. Important Lemmas

Traditional real-time task scheduling usually applies schedulability tests in a hyper-period to guarantee the schedulability, where the hyper-period for the tasks in this paper is the period P. However, this does not work for thermal-constrained task scheduling since the initial temperature might be different at the beginning of two hyper-periods. For example, consider the scheduling of a task with period $P = 0.12$ sec, computational requirement $C_i = 0.06 s_E$, and relative deadline $D_i = 55$ msec and the system setting $\hat{\alpha} = 1.25$, $\hat{\beta} = 9.52$, $\gamma = 3$, $\theta^* = 72$.[3]

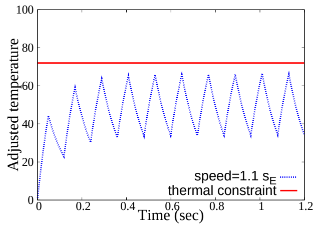
1. C_{ef}, V_t, V_{dd}, and κ_v denote the effective switch capacitance, the threshold voltage, the supply voltage, and a hardware-design-specific constant, respectively. $V_{dd} \geq V_t \geq 0$; $\kappa_v, C_{ef} > 0$.

2. For the rest of the paper, if the context is clear, we will use "temperature" to refer to "adjusted temperature", including thermal constraints.

3. We use the same thermal parameters here as in Section 6.



(a) speed = $1.2s_E$



(b) speed = $1.1s_E$

Figure 1. Comparison of instantaneous temperature during different periods under different speed scheduling.

Figure 1(a) depicts the instantaneous temperature of the speed scheduling by executing the task at speed $1.2s_E$ in time interval $[0, 1.2]$. The speed scheduling by executing the first task instance at speed $1.2s_E$ reaches 57.1 as its highest temperature at time 0.05 sec. Clearly, the speed scheduling can meet both timing and thermal constraints for the first period $[0, 0.12]$. However, applying the same speed scheduling for the second period $[0.12, 0.24]$ results in the highest temperature 76.94, which violates the thermal constraint.

As a result, to guarantee the schedulability of frame-based real-time tasks, one has to perform schedulability tests for each individual period. Fortunately, the following lemmas show that we can guarantee schedulability over all periods if we find a feasible speed scheduling over a single period with a specific initial temperature constraint:

Lemma 1: Let $s_\dagger$ be a feasible speed scheduling in period $[t_0, t_0+P]$ with a *converging initial temperature* $\theta(t_0) = \theta_P$ that satisfies

$$\theta_P = \frac{1}{1-e^{-\beta P}} \int_{t_0}^{t_0+P} s_\dagger^\gamma(\tau) e^{-\beta(t_0+P-\tau)} \mathrm{d}\tau. \tag{6}$$

Then, $s_\dagger$ is a feasible speed scheduling over all time periods. $\square$

The proofs of all lemmas in this paper are given in Appendices.

In the previous example in Figure 1, the speed scheduling by executing the task at speed $1.1s_E$ has a converging initial temperature 33.9 as shown in Figure 1(b).

Moreover, we find out that a feasible speed scheduling with a converging initial temperature is also a necessary condition of the schedulability test as shown in the following lemma:

Lemma 2: If there does not exist any feasible speed scheduling $s_\dagger$ with a converging initial temperature no more than θ^*, there does not exist any feasible speed scheduling for the input task set under the thermal and timing constraints. $\square$

In this paper we intend to derive a feasible speed scheduling by *repetitively* using a feasible speed scheduling $s_\dagger$ if exists as shown in Lemmas 1 and 2. We do not intend to provide different feasible speed scheduling over different periods.

Our ultimate goal is to determine whether there exists such feasible speed scheduling for a given task set. In other words, we want to know what is the maximum amount of workload we can achieve under thermal and timing constraints. The latter goal is equivalent to the former one. In [1] a solution was provided to derive speed scheduling to maximize the workload under the thermal constraints. The workload completed in a time interval $[t_0, t_1]$ is $\int_{t_0}^{t_1} s(t)\mathrm{d}t$. By (3), we have $s(t) = (\theta'(t) + \beta\theta(t))^{\frac{1}{\gamma}}$. Then the optimization of workload can be written as follows:

$$\text{maximize} \quad \int_{t_0}^{t_1} (\theta'(t) + \beta\theta(t))^{\frac{1}{\gamma}} \,\mathrm{d}t \tag{7a}$$
$$\text{subject to} \quad \theta(t_0) = \theta_0, \; \theta(t_1) \le \theta_1, \tag{7b}$$
$$\theta(t) \le \theta^*, \forall t \in [t_0, t_1] \tag{7c}$$

The following lemma provides a preliminary result to maximize the workload by ignoring the hardware thermal constraint:

Lemma 3: In period $[t_0, t_0+P]$, without considering the hardware thermal constraint the temperature curve $\theta(t)$ that achieves the maximum workload can be represented as

$$\theta(t) = ae^{-\beta\Delta_t} + be^{-\frac{\gamma}{\gamma-1}\beta\Delta_t}, \tag{8}$$

where $\Delta_t = t - t_0$, a and b are determined by boundary conditions $\theta(t_0) = \theta_0$ and $\theta(t_1) = \theta_1$ as follows:

$$a = \frac{\theta_1 e^{\beta\Delta} - \theta_0 e^{-\frac{1}{\gamma-1}\beta\Delta}}{1 - e^{-\frac{1}{\gamma-1}\beta\Delta}}, \quad b = \frac{\theta_0 - \theta_1 e^{\beta\Delta}}{1 - e^{-\frac{1}{\gamma-1}\beta\Delta}}. \tag{9}$$

The first derivative of $\theta(t)$ is

$$\theta'(t) = -a\beta e^{-\beta\Delta_t} - \frac{\gamma}{\gamma-1} b\beta e^{-\frac{\gamma}{\gamma-1}\beta\Delta_t}, \tag{10}$$

and the speed scheduling can be written as

$$s(t) = (-b\frac{\beta}{\gamma-1})^{\frac{1}{\gamma}} e^{-\frac{1}{\gamma-1}\beta\Delta_t}. \tag{11}$$

$\square$

If the derived solution $\theta(t)$ in (8) violates the hardware thermal constraint, we need to revise the above solution. A naive solution is to find the time interval $[u, v]$ where the hardware thermal constraint is violated and revise the speed scheduling in $[u, v]$, where u and v satisfy that $t_0 \le u \le v \le t_1$, $\theta(u) = \theta(v) = \theta^*$, and $\theta(t) \ge \theta^*$, $\forall t \in [u, v]$. To avoid the thermal violation, we can adopt the equilibrium speed s_E during $[u, v]$, and for the rest of the time interval, we will still apply $s(t)$ obtained in (11). However, this naive approach might not be the optimal solution. In the following, we aim to develop better strategies to tackle this issue.

4. Schedulability Analysis

At first, we focus on real-time tasks in which all the tasks have the same period P with the same arrival time, where tasks with different arrival times will be discussed in Section 4.3. We aim to find a feasible speed scheduling for task set $\mathbf{T}$ so that all the tasks meet both timing and thermal constraints. By Lemmas 1 and 2, we will only focus on a single period $[t_0, t_0 + P]$. We assume that the task execution starts at time t_0 and ends at t_1 with a response time $\Delta = t_1 - t_0$. We define D as the common deadline of the tasks in $\mathbf{T}$, and C as the required cumulative workload demand of all tasks, i.e., $C = \sum_{T_i \in \mathbf{T}} C_i$. To meet the deadline, we have to ensure that $\Delta \leq D$. For the simplicity of presentation, we will first implicitly focus on ideal processors without speed limitations. Extensions to non-ideal processors will be presented in Section 4.3.

We propose two approaches to the thermal-constrained frame-based task scheduling:

- *Timing-optimization approach*: It aims to complete the execution of the task instances as soon as possible without violating the thermal constraints. Then we can have more time to cool down the processor. In other words, it aims to minimize the delay in completing the tasks under the thermal constraints.
- *Thermal-optimization approach*: We assume the task instance is completed by the end of the deadline. Clearly, the lower the converging initial temperature is, the more the workload can be done in a specified time interval. In other words, it aims to satisfy the thermal constraints for all time instants under the timing constraint of tasks.

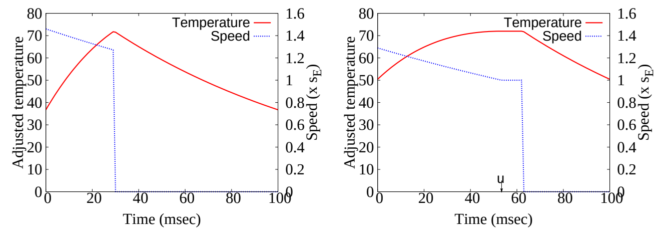
In the following, we will show the detailed description of these two approaches.

4.1. Timing-Optimization Approach

With timing-optimization approach we aim to find a feasible speed scheduling to minimize the response time Δ under the thermal constraints.

We observe that the minimization of the response time $\Delta = t_1 - t_0$ under the thermal constraints will render the temperature $\theta(t)$ reaching the hardware thermal constraint θ^* at the completion time t_1, i.e., $\theta(t_1) = \theta^*$. Otherwise, by contradiction, we assume that $\theta(t_1) < \theta^*$, then the speed profile can be increased without violating the hardware thermal constraint. Therefore, the response time can be reduced, which contradicts the optimal value of the response time. Based on this observation and Lemma 1, in the following we assume the temperature $\theta(t_0) = \theta_P$ and $\theta(t_1) = \theta^*$.

At first, we ignore the hardware thermal constraint in the interval (t_0, t_1), and then we have the following results:



(a) Ignoring θ^* ($P = 100$ msec, $C = 0.04 s_E$) (b) Considering θ^* ($P = 100$ msec, $C = 0.07 s_E$)

Figure 2. Speed scheduling and temperature curve with timing-optimization approach as $\alpha = 1.25$, $\beta = 9.52$, $\theta^* = 72$, and $\gamma = 3$.

Lemma 4 (Ignoring the hardware thermal constraint): The minimum of the response time can be written as

$$\Delta = \frac{\gamma - 1}{\beta} \ln\left(\frac{\beta}{\gamma - 1} (C^{-\gamma} \theta^* (1 - e^{-\beta P}))^{\frac{-1}{\gamma - 1}} + 1\right). \quad (12)$$

During $[t_0, t_1]$, the corresponding speed is

$$s(t) = \left(C \frac{\beta}{\gamma - 1} + (C^{-1} \theta^* (1 - e^{-\beta P}))^{\frac{1}{\gamma - 1}}\right) e^{-\frac{1}{\gamma - 1} \beta \Delta_t}. \quad (13)$$

During $[t_1, t_0 + P]$, the speed $s(t) = 0$. □

Based on Lemma 4, the speed is decreasing during $[t_0, t_1]$ and is 0 during $[t_1, t_0 + P]$ as shown in Figure 2(a).

Next we consider the hardware thermal constraint in the interval (t_0, t_1). With the timing-optimization approach, if $\theta'(t_1) \geq 0$, then $\theta(t) \leq \theta^*$, $\forall t \in [t_0, t_1]$, and hence the derived solution is feasible; otherwise, the hardware thermal constraint is violated at some time instances during $[t_0, t_1]$. Then we have to revise the solution. In this case, we can find an earliest time instant u ($u \in [t_0, t_1]$) and a speed scheduling $s(t)$ so that the resulting temperature curve $\theta(t)$ satisfying that $\theta'(u) = 0$, $\forall t \in [t_0, u]$ and $\theta(u) = \theta^*$ as shown in the following lemma:

Lemma 5 (Considering the hardware thermal constraint): The minimum of the response time Δ can be solved by the fixed point theorem based on the following equations on two dimensional vector (Δ, Δ_u), where $\Delta_u = u - t_0$:

$$\Delta = \Delta_u + C(\beta\theta^*)^{-\frac{1}{\gamma}} - \frac{\gamma - 1}{\beta}\left(e^{\frac{1}{\gamma - 1}\beta\Delta_u} - 1\right), \quad (14)$$

$$\Delta = \Delta_u + \frac{1}{\beta} \ln\left(\gamma - (\gamma - 1) e^{\frac{1}{\gamma - 1}\beta\Delta_u}\right) + P. \quad (15)$$

During $[t_0, u]$, we can use the speed $s(t)$ in (13) by replacing C with $C - (\beta\theta^*)^{\frac{1}{\gamma}}(\Delta - \Delta_u)$. During $[u, t_1]$, the speed $s(t) = s_E$. During $[t_1, t_0 + P]$, the speed $s(t) = 0$. □

Based on Lemma 5, the speed is decreasing during $[t_0, u]$, remains constant s_E during $[u, t_1]$, and is 0 in $[t_1, t_0 + P]$ as shown in Figure 2(b).

Recall that the necessary condition that we can ignore the hardware thermal constraint in the optimization is that $\theta'(t_1) \geq 0$, where $\theta'(t_1)$ can be obtained with the similar analysis in deriving (31) in Appendix. Such constraint will render the following inequality:

$$\frac{\theta_P}{\theta^*}e^{-\beta\Delta} \leq \gamma - (\gamma-1)e^{\frac{1}{\gamma-1}\beta\Delta}. \qquad (16)$$

The above inequality can be further simplified by using θ_P and Δ defined in (28) in Appendix and (12) respectively as follows:

$$\theta^* \leq \beta^{\gamma-1}(\frac{C}{1-e^{-\beta P}})^{\gamma}. \qquad (17)$$

In summary, the following theorem shows feasibility tests of the resulting speed scheduling:

Theorem 1: The following two cases will result in a feasible speed scheduling over all periods:

- If (17) holds, and Δ derived in Lemma 4 satisfies $\Delta \leq D$, and then we use the speed scheduling defined in Lemma 4;
- If (17) does not hold, and Δ derived in Lemma 5 satisfies $\Delta \leq D$, and then we use the speed scheduling defined in Lemma 5.

□

4.2. Thermal-Optimization Approach

We consider the task execution that starts at time t_0 and ends at t_1 with the completion time equal to the deadline, i.e., $\Delta = D$. We aim to minimize the initial temperature θ_P.

At first, we ignore the hardware thermal constraint in the interval (t_0, t_1), and we have the following conclusion:

Lemma 6 (Ignoring the hardware thermal constraint): During $[t_0, t_1]$, the speed $s(t)$ can be represented as (11) with b defined in (9) and the boundary conditions θ_P and θ_1 defined as follows:

$$\theta_P = C^{\gamma}(\frac{\beta}{\gamma-1})^{\gamma-1}(e^{\beta P}-1)^{-1}(1-e^{-\beta\Delta\frac{1}{\gamma-1}})^{1-\gamma}, \quad (18)$$

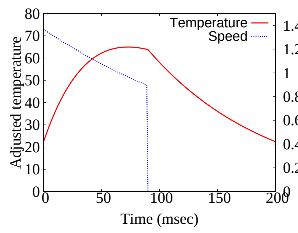
$$\theta_1 = \theta_P e^{\beta(P-\Delta)}. \quad (19)$$

During $[t_1, t_0+P]$, the speed $s(t) = 0$. The corresponding $\theta(t)$ can be written as (8) with the same b, θ_P and θ_1. □

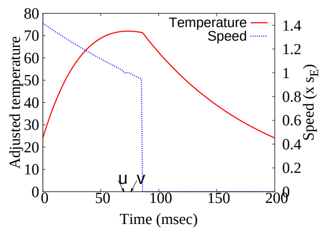
Based on Lemma 6, the speed is decreasing during $[t_0, t_1]$ and is 0 during $[t_1, t_0+P]$ as shown in Figure 3(a).

Next we consider the hardware thermal constraint in the interval (t_0, t_1). If $\theta(t) \leq \theta^*$, $\forall t \in [t_0, t_1]$, where $\theta(t)$ is defined in (8) with the obtained θ_P and θ_1 defined in (18) and (19) respectively, then the derived speed scheduling is optimal; otherwise, the temperature is violated during the execution. Recall that we need to find u and v during time interval $[t_0, t_1]$, $t_0 \leq u \leq v \leq t_1$, as we discussed at the end of Section 3 such that $\theta(u) = \theta(v) = \theta^*$ and $\theta'(u) = \theta'(v) = 0$. Define $\bar{\Delta}_v = t_1 - v$ and $\bar{\Delta}_t = t_1 - t$, and we have the following conclusion:

Lemma 7 (Considering the hardware thermal constraint): Δ_u and $\bar{\Delta}_v$ can be solved by the fixed point theorem based



(a) Ignoring θ^* ($P = 200$ msec, $C = 0.1s_E$, $D = 90$ msec)



(b) Considering θ^* ($P = 200$ msec, $C = 0.1s_E$, $D = 86$ msec)

Figure 3. Speed scheduling and temperature curve with thermal-optimization approach as $\alpha = 1.25$, $\beta = 9.52$, $\theta^* = 72$, and $\gamma = 3$.

the following equations on two dimensional vector $(\Delta_u, \bar{\Delta}_v)$:

$$(\gamma-1)(e^{\frac{1}{\gamma-1}\beta\Delta_u} - e^{-\beta(P-\Delta+\Delta_u+\frac{\gamma}{\gamma-1}\bar{\Delta}_v)}) = \gamma(1-e^{-\beta(P-\Delta+\Delta_u+\bar{\Delta}_v)}), \qquad (20)$$

$$C(\beta\theta^*)^{-\frac{1}{\gamma}} - (\Delta - \Delta_u - \bar{\Delta}_v) = (e^{\frac{1}{\gamma-1}\beta\Delta_u} - e^{-\frac{1}{\gamma-1}\beta\bar{\Delta}_v})\frac{\gamma-1}{\beta}. \qquad (21)$$

During $[t_0, u]$ we can use the speed $s(t)$ defined in (11). During $[u, v]$ the speed remains constant s_E. During $[v, t_1]$

$$s(t) = (\beta\theta^*)^{\frac{1}{\gamma}} e^{\frac{1}{\gamma-1}\beta(\bar{\Delta}_t - \bar{\Delta}_v)}. \qquad (22)$$

During $[t_1, t_0+P]$, the speed $s(t) = 0$. □

Based on Lemma 7, the period $[t_0, t_0+P]$ can be divided into four parts as shown in Figure 3(b): $[t_0, u]$, during which the speed is decreasing; $[u, v]$, during which the speed remains constant s_E; $[v, t_1]$, during which the speed is decreasing; $[t_1, t_0+P]$, during which the speed is 0.

In summary, the following theorem shows feasibility tests of the resulting speed scheduling:

Theorem 2: Given the response time equal to the deadline, i.e., $\Delta = D$, the following two cases will result in a feasible speed scheduling over all time periods:

- If $\theta(t)$ derived in Lemma 6 satisfies $\theta(t) \leq \theta^*$, $\forall t \in [t_0, t_0+D]$, and then we use the speed scheduling defined in Lemma 6;
- If the above condition does not hold, but the variables Δ_u and $\bar{\Delta}_v$ derived in Lemma 7 satisfy $\Delta_u + \bar{\Delta}_v \leq \Delta$, and then we use the speed scheduling defined in Lemma 7.

□

4.3. Extensions

Tasks with different deadlines So far, we assumed that all tasks have the same relative deadlines. Otherwise, according to EDF scheduling, the task instance with the earliest deadline has the highest priority for execution. Therefore, we order task instances in $\mathbf{T}$ non-decreasingly according to their relative deadlines in period $[t_0, t_0 + P]$. We assume $D_i \leq D_j$ as $i < j$. The workload demand right before D_i is $\sum_{j=1}^{i} C_j$. We define Δ_i as the response time of task T_i. To meet timing constraints, we need to ensure that $\Delta_i \leq D_i$ for $i = 1, 2, \ldots, N$.

The derived speed scheduling in Section 4.1 might not satisfy the timing and thermal constraints for all tasks. Therefore, we revise the speed scheduling derived in Section 4.1 as follows:

```
1:  $i \leftarrow N$;
2:  **while** true **do**
3:    Apply Theorem 1 for the cumulative tasks during $[t_0, t_0 + D_i]$ by replacing the workload $C$ with $\sum_{j=1}^{i} C_j$ and setting the speed at $s_E$ from $t_0 + D_i$ to the completion time;
4:    **if** there exists a feasible speed scheduling $s_\dagger$ **then**
5:      calculate $\Delta_j$, $\forall j \leq i$, based on $s_\dagger$;
6:      calculate $\Delta_j = \sum_{k=1}^{i} \Delta_k + \sum_{k=i+1}^{j} C_j / s_E$, $\forall i + 1 \leq j \leq N$;
7:      **if** $\Delta_j \leq D_j$, $\forall j \leq N$ **then**
8:        return the resulting speed scheduling $s_\dagger$;
9:      **else if** $\exists j >= i$ with $\Delta_j > D_j$ **then**
10:       return "not schedulable";
11:     **else**
12:       find the smallest $i^*$ with $\Delta_j > D_j$;
13:       $i \leftarrow i^*$;
14:     **end if**
15:   **else**
16:     return "not schedulable";
17:   **end if**
18: **end while**
```

We start with the task T_N. We can apply Theorem 1 for all cumulative tasks. If there exists a feasible speed scheduling $s_\dagger$, then we calculate each individual response time Δ_j. If $\Delta_j \leq D_j$, $\forall j \leq N$, then we return the feasible scheduling $s_\dagger$. Otherwise, we find earlier-deadline task T_{i^*} with timing violation. We then apply Theorem 1 for the cumulative tasks during $[t_0, t_0 + D_{i^*}]$ by replacing the workload C with $\sum_{j=1}^{i^*} C_j$ and setting the speed at s_E from $t_0 + D_{i*}$ to the completion time. The same procedure is repeatedly applied. Clearly, the while loop in the procedure has at most N iterations. The output returns either a feasible speed scheduling or "not schedulable". We can also apply the above algorithm with the result into Section 4.2 with some revision. We skip the detail here.

Tasks with different arrival-times As shown in [14], [15], the *critical instant* for thermal-constrained scheduling of periodic real-time tasks is at the moment that all instances arrive at the same time when all tasks are with the same relative deadline. As a result, we can perform schedulability test based on the assumption that all the tasks arrive at the same time. However, for executions, we have to be careful, since it might lead to a solution that intends to execute some workload before it arrives. Suppose $s_\dagger$ is the speed scheduling that converges at temperature θ_P. Then, if applying $s_\dagger$ to $\mathbf{T}$ leads to a feasible speed scheduling for task set $\mathbf{T}$, we do not have to do anything for revision. However, if applying $s_\dagger$ to task set $\mathbf{T}$ leads to a speed scheduling that executes some computation before the workload arrives, we have to set speed to 0 when there is no task instance in the system. When a task instance arrives at time t with instantaneous temperature $\theta(t) < \theta^*$, we refer to the speed scheduling $s_\dagger$ to find time instant w with $\theta(t_0 + w) = \theta(t)$ and $s_\dagger(t_0 + w) > s_E$ and then follow the speed scheduling $s_\dagger$ to serve the incoming tasks. When tasks have the same relative deadline but different arrival times, with similar arguments in Lemma 1, it is not difficult to see that the above speed scheduling does not make any task miss its deadline under the thermal constraint if $s_\dagger$ is a feasible speed scheduling.

5. Practical Design Issues

As current DVS platforms have limitations on the maximum and minimum available speeds, we will show how to revise the approaches in this paper to deal with systems with bounded speeds. Moreover, since modern commercial processors have discrete speeds only, we will explore how to deal with systems with discrete speeds.

Bounded speeds Suppose that $s_{\max}$ is the maximum available speed of the processor. Without loss of generality, we assume $s_{\max} > s_E$. Otherwise, the solution is trivial. We only consider the case that the derived solution violates the speed constraint $s_{\max}$ at some time instance. In such a case, we have to revise the derivation of speed scheduling in Sections 4. Clearly, the speed should start to run at $s_{\max}$ for a while and then slow down at a time instant $t_0 + \Delta_w$. The corresponding temperature curve is increasing until the temperature reaches θ^*. Suppose that the temperature at the moment that the speed scheduling starts to slow down from $s_{\max}$ is $\theta^\flat$. Then, starting from $t_0 + \Delta_w$, the temperature curve follows the speed scheduling (11). Suppose that it takes Δ_u to heat from $t_0 + \Delta_w$ to $u = t_0 + \Delta_w + \Delta_u$. By (11), we know that $s_{\max} = (\frac{\theta^* e^{\beta \Delta_u} - \theta^\flat}{1 - e^{-\beta \Delta_u \frac{1}{\gamma - 1}}} \frac{\beta}{\gamma - 1})^{\frac{1}{\gamma}}$. Since the processor runs at speed $s_{\max}$ from time t_0 to $t_0 + \Delta_w$, we have $\theta^\flat = \frac{1}{\beta} s_{\max}^{\gamma} (1 - e^{-\beta \Delta_w}) + \theta_P e^{-\beta \Delta_w}$. Moreover, the workload is constrained by $C - s_E(\Delta - \Delta_u) - s_{\max} \Delta_w = \int_w^u s(t) dt$. Putting all these equations together, we can simply revise the approaches in Sections 4 by adding the above equalities. For the case that there is a minimum speed for execution, the procedure is similar.

Processors with discrete speeds To maximize the workload under the timing and the thermal constraints, one has to change the execution speeds frequently. For commercial processors, frequent speed switching might incur too much overhead and only discrete speeds might be available in the processor. Then we can first derive a solution by assuming continuously available speeds as the approaches in Sections 4. Then, we use some available speeds to approximate the speed scheduling $s_\dagger$ by restricting and perform schedulability tests by applying similar procedures. For instance, we can find a high speed s_H to approximate the non-constant heating speed scheduling derived in Section 4. Suppose that $s_\dagger$ is a speed scheduling that converges at temperature θ_P. We can set s_H by using the following equation: $\theta^* = \frac{1}{\beta} s_H^\gamma (1 - e^{-\beta\Delta_u}) + \theta_P e^{-\beta\Delta_u}$, where $t_0 + \Delta_u$ has the highest temperature in speed scheduling $s_\dagger$. Whether the task set is schedulable under the thermal and timing constraints can then be derived. Similarly, for cooling speed scheduling derived Section 4, we can use a speed $s_L < s_E$ to approximate it.

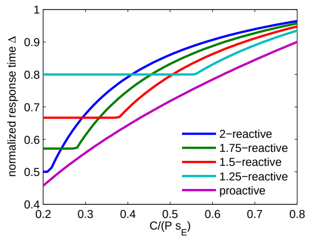
6. Performance Evaluation

This section provides performance evaluation of proactive speed scheduling with both the timing-optimization approach and the thermal-optimization approach. We compare it with reactive speed scheduling in [13], [15]. The reactive speed scheduling uses two speeds for execution, in which one is the equilibrium speed and the other is a speed $s_H > s_E$. Once the temperature is lower than the thermal constraint, the approach executes jobs at speed s_H; otherwise the processor executes at speed s_E. We choose the settings of s_H/s_E as 1.25, 1.5, 1.75, and 2, where the results by using $s_H/s_E = 1.25$ is denoted by 1.25-reactive and so on.

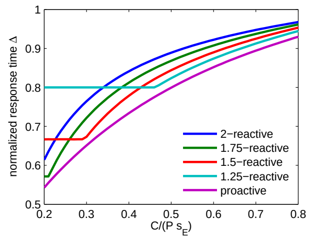
Throughout this section, we use absolute temperature for presenting our results. The ambient temperature (the temperature constraint, respectively) is set to be 30°C (120°C, respectively). The power consumption function $\Psi(s, \Theta)$ is assumed to be $s^3 + 0.001\Theta + 0.1$ Watt. By adopting the same cooling factor in [18], the cooling factor $\hat{\beta}$ of the processor is set as $1/0.105$ sec^{-1}, while $\hat{\alpha}$ is 1.25 K/Joule.[4] For a specified x, we generate a set of 10 frame-based tasks with a specified period P in the range of 0.01 sec and 0.1 sec, while the total amount C of execution cycles at the equilibrium speed s_E is $x \cdot P \cdot s_E$.

As the timing-optimization approach tries to minimize the response time while meeting the thermal constraints, we evaluate the response time of our approaches with comparison to the reactive speed scheduling. For the thermal-optimization approach, we evaluate the converged temperature with comparison to the reactive mechanism. We adopt the *normalized response time* and the *normalized converged temperature* as the performance metric, while the former is the response time of the approach divided by that of the reactive mechanism, and the latter is the converged temperature θ_P of the approach divided by that of the proactive mechanism. If the reactive mechanism fails to schedule a task set, the normalized converged temperature is set as 0.

4. K is the unit of absolute temperature.

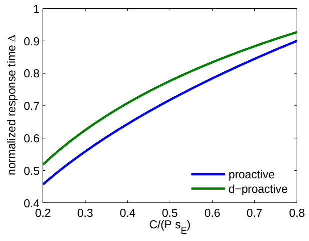


(a) $P = 0.01$ sec

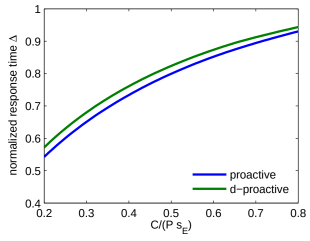


(b) $P = 0.1$ sec

Figure 4. Evaluation results for timing optimization under different settings of periods.



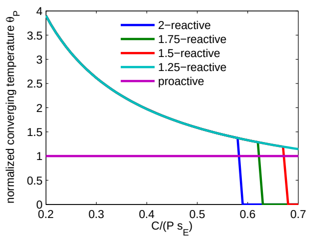
(a) $P = 0.01$ sec



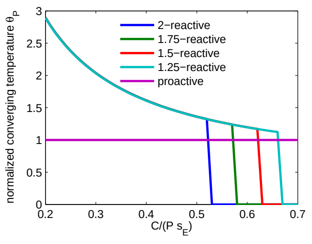
(b) $P = 0.1$ sec

Figure 5. Evaluation results for using discrete speeds.

Figure 4 presents the evaluation results of the average normalized response time by varying utilization $\frac{C}{P s_E}$ at the equilibrium speed from 20% to 70% when $P = 0.01$ sec in Figure 4(a) and $P = 0.1$ sec in Figure 4(b). The other results for different periods are omitted since they are quite similar to the presented results in Figure 4. As the proactive approach can minimize the response time, it always outperforms the reactive speed mechanism. The improvement of the response time for the reactive speed mechanism depends on the selection of the higher speed s_H. The normalized response time of the reactive speed mechanism might converge, e.g., when $s_H = 1.25 s_E$ and the utilization at the equilibrium is less than 0.5, since running at s_H is sufficient to minimize the response time. The improvement of the proactive speed mechanism in terms of response time minimization could be in the range of 5% to 10%, for all the evaluated settings of s_H. Therefore, for critical task sets with stringent deadlines, the timing-optimization approach will have more chance to meet the timing constraint. However, when the utilization becomes large, as there is not too much room for cooling, the improvement of the timing-optimization approach becomes



(a) $P = 0.01$ sec



(b) $P = 0.1$ sec

Figure 6. Evaluation results for thermal optimization under different settings of periods.

marginal. Moreover, Figure 5 illustrates the results for using discrete speeds for proactive speed scheduling, denoted by d-proactive in the legend in Figure 5. As shown in Figure 5, applying discrete speeds is more practical without sacrificing too much in the optimality of response time minimization.

Figure 6 shows the evaluation results of the average normalized converged temperature, where the relative deadline is $\frac{C}{s_E} \times 0.9$. Similar to the result in Figure 4, the normalized converged temperature is smaller when the utilization is lower. The lower the converged temperature is, the more workload the system can support if there are potential tasks to arrive later. Moreover, the reactive speed mechanism could make tasks miss their deadlines, while the proactive approach derives feasible speed scheduling when $\frac{C}{Ps_E} \leq 0.7$.

7. Conclusion

In this paper, we have presented proactive speed scheduling for real-time tasks to meet both timing and thermal constraints. We proposed two different approaches in order to achieve schedulability tests of real-time tasks, i.e., timing-optimization approach and thermal-optimization approach. As we can see, the latter approach aims to minimize converging initial temperature so that the system can tolerate the thermal constraint to complete more workload, while the former approach aims at the minimization of the response time during a period regardless of the temperature at the beginning of the period. Both approaches render our proactive speed control mechanisms outperform the reactive one designed in [14], [15]. For future research, we would like to explore whether the proactive speed scheduling can be applied to general task sets.

Acknowledgment

This work is sponsored in part by 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] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. In *Symposium on Foundations of Computer Science*, 2004.

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

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

[4] 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.

[5] 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.

[6] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele. Thermal-aware global real-time scheduling on multicore systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2009.

[7] 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.

[8] 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.

[9] 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.

[10] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits*. Prentice Hall, 2nd edition, 2002.

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

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

[13] 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.

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

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

[16] R. Xu, D. Mossé, and R. G. Melhem. Minimizing expected energy in real-time embedded systems. In *EMSOFT*, 2005.

[17] 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.

[18] L. Yuan, S. Leventhal, and G. Qu. Temperature-aware leakage minimization technique for real-time systems. In *IEEE/ACM international conference on Computer-aided design*, 2006.

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

Proof of Lemma 1

Given a time interval $[t_0+kP, t_0+(k+1)P]$, $k = 0, 1, \ldots$, we assume that $s_\dagger$ is a feasible speed schedule over this interval, where t_0 is the arrival time of $\mathbf{T}$ and $s_\dagger$ is independent of interval index k, i.e., $s_\dagger(t+kP) = s_\dagger(t)$. Both thermal and delay constraints can be met with this speed schedule over any period. In the following, we want to see how temperature curve changes over each interval. Since $s_\dagger(t+kP) = s_\dagger(t)$, the instantaneous temperature at any time $t \in [t_0+kP, t_0+(k+1)P]$ can be written as

$$\begin{aligned}\theta(t) &= \int_{t_0}^{t} s_\dagger^\gamma(\tau) e^{-\beta(t-\tau)} \mathrm{d}\tau \\ &\quad + e^{-\beta(t-(t_0+kP))}\theta(t_0+kP). \end{aligned} \tag{23}$$

Then the instantaneous temperature at the end of the period, i.e., $t = t_0 + kP$, can be obtained as

$$\begin{aligned}\theta(t_0+kP) &= \frac{1-e^{-\beta kP}}{1-e^{-\beta P}} \int_{t_0}^{t_0+P} s_\dagger^\gamma(\tau) e^{-\beta(t_0+P-\tau)} \mathrm{d}\tau \\ &\quad + e^{-\beta kP}\theta(t_0). \end{aligned} \tag{24}$$

As $k \to \infty$, the above equality will converge to a fixed value θ_P defined in (6).

Based on (24) and (6), if $\theta(t_0) \le \theta_P$, [5] we have $\theta(t_0+kP) = \theta_P + e^{-\beta kP}(\theta(t_0)-\theta_P) \le \theta_P$. If $\theta(t_0) = \theta_P$, then $\theta(t_0+kP) = \theta_P$. Then the lemma is proved.

Proof of Lemma 2

Here we sketch the proof instead of showing the detailed proof. The non-existence of such feasible speed schedule leads to a simple conclusion that any feasible speed schedule in a period $[t_0+kP, t_0+(k+1)P]$ is with $\theta(t_0+kP) < \theta(t_0+(k+1)P)$ for any k. As a result, any speed schedule will eventually violate the thermal or the timing constraint.

Proof of Lemma 3

The optimal solution to (7) can be derived by applying the Euler-Lagrange equation in calculus of variations [1]. Since the temperature and the speed settings are needed for schedulability tests, for completeness, we present how to solve (7) in the rest of this section.

5. We can assume that the temperature at time t_0 stay at the lowest temperature (equal to the ambient temperature) since t_0 is the arrival time for the first job. Therefore we can assume $\theta(t_0) \le \theta_P$.

We begin by ignoring the constraint in (7c). This relaxed problem can be solved by applying calculus of variations. If we define $F \overset{\text{def}}{=} (\theta'(t) + \beta\theta(t))^{\frac{1}{\gamma}}$, then we know that

$$\begin{aligned} F_\theta = \frac{\partial F}{\partial \theta} &= \frac{\beta}{\gamma}\left(\theta'(t)+\beta\theta(t)\right)^{\frac{1}{\gamma}-1}, \\ F_{\theta'} = \frac{\partial F}{\partial \theta'} &= \frac{1}{\gamma}\left(\theta'(t)+\beta\theta(t)\right)^{\frac{1}{\gamma}-1}. \end{aligned}$$

By the Euler-Lagrange equation $F_\theta = \frac{d}{dt}F_{\theta'}$ [1], i.e.,

$$\beta\left(\theta'(t)+\beta\theta(t)\right) = \left(\frac{1}{\gamma}-1\right)\left(\theta''(t)+\beta\theta'(t)\right). \tag{25}$$

The above differential equation is solvable. $\theta(t)$ in (25) can be solved as (8). Based on (8), other formulas in the lemma can be easily derived. Then the lemma is proved.

Proof of Lemma 4

During $[t_0, t_1]$, the amount of workload C has to be completed. Recall that the workload is determined by the speed schedule $s(t)$, which can be obtained in (11). Therefore, the workload $C = \int_{t_0}^{t_1} s(t)dt$ in $[t_0, t_1]$ is constrained as:

$$C = (-b)^{\frac{1}{\gamma}}\left(\frac{\gamma-1}{\beta}\right)^{\frac{\gamma-1}{\gamma}}\left(1-e^{-\frac{1}{\gamma-1}\beta\Delta}\right). \tag{26}$$

Applying b defined by (9) into (26), together with $\theta(t_1) = \theta^*$ and $\theta_0 = \theta_P$, we have

$$C^\gamma\left(\frac{\beta}{\gamma-1}\right)^{\gamma-1} = (\theta^* - \theta_P e^{-\beta\Delta})\left(e^{\frac{1}{\gamma-1}\beta\Delta}-1\right)^{\gamma-1}. \tag{27}$$

During $[t_1, t_0+P]$, the processor is idle, and the temperature at t_0+P changes from θ^* to θ_P. As a result, we have

$$\theta^* = \theta_P e^{\beta(P-\Delta)}. \tag{28}$$

Applying (28) into (27) leads to (12).

Now we are ready to derive $s(t)$. Based on (11) and (26), we have

$$s(t) = \frac{\beta}{\gamma-1}\frac{C}{1-e^{-\frac{1}{\gamma-1}\beta\Delta}}e^{-\frac{1}{\gamma-1}\beta\Delta_t}. \tag{29}$$

Applying (12) into (29), the corresponding speed schedule can represented as (13). Therefore, during $[t_0, t_1]$ we can use $s(t)$ in (13); during $[t_1, t_0+P]$ the speed $s(t) = 0$.

Proof of Lemma 5

We consider the time interval $[t_0, u]$ first. Recall that the optimization will render $\theta'(u) = 0$, where $\theta'(u)$ can be written as (similar to (10))

$$\theta'(u) = -a\beta e^{-\beta\Delta_u} - \frac{\gamma}{\gamma-1} b\beta e^{-\frac{\gamma}{\gamma-1}\beta\Delta_u} \quad (30)$$

$$= \frac{\beta}{\gamma-1}\frac{\theta^*}{e^{\frac{1}{\gamma-1}\beta\Delta_u}-1} (\gamma-(\gamma-1)e^{\frac{1}{\gamma-1}\beta\Delta_u} - \frac{\theta_P}{\theta^*}e^{-\beta\Delta_u}). \quad (31)$$

Therefore $\theta'(u) = 0$ results in the following formula:

$$\frac{\theta_P}{\theta^*}e^{-\beta\Delta_u} = \gamma-(\gamma-1)e^{\frac{1}{\gamma-1}\beta\Delta_u}. \quad (32)$$

During $[u, t_1]$ the processor keeps running at the equilibrium speed $s_E = (\beta\theta^*)^{\frac{1}{\gamma}}$. Therefore, the workload in $[t_0, u]$ can be constrained by $C - s_E(\Delta-\Delta_u) = \int_{t_0}^{u} s(t)dt$. Similar to the analysis in deriving (27), we replace Δ with Δ_u and C with $C - s_E(\Delta-\Delta_u)$ in (27), then we have

$$(C-(\beta\theta^*)^{\frac{1}{\gamma}}(\Delta-\Delta_u))^{\gamma}(\frac{\beta}{\gamma-1})^{\gamma-1} = (\theta^*-\theta_P e^{-\beta\Delta_u})(e^{\frac{1}{\gamma-1}\beta\Delta_u}-1)^{\gamma-1}. \quad (33)$$

Based on (32) and (33), then we can get (14).

During $[t_1, t_0+P]$, the processor is idle, and (28) still holds here. Applying (28) into (32), then we can get (15).

Then Δ and Δ_u can be solved by the fixed point theorem based on Equations (14) and (15).

With the above analysis, now we can easily derive the speed schedule. During $[t_0, u]$, we can use $s(t)$ in (13) by replacing C with $C-(\beta\theta^*)^{\frac{1}{\gamma}}(\Delta-\Delta_u)$; during $[u, t_1]$ the speed $s(t) = s_E$; during $[t_1, t_0+P]$ the speed $s(t) = 0$.

Proof of Lemma 6

During $[t_0, t_1]$, the amount of workload C has to be completed. Similar to the result in (27), we have

$$C^{\gamma}(\frac{\beta}{\gamma-1})^{\gamma-1} = (\theta_1-\theta_P e^{-\beta\Delta})(e^{\frac{1}{\gamma-1}\beta\Delta}-1)^{\gamma-1}. \quad (34)$$

During $[t_1, t_0+P]$ the processor is idle, then we get (19). Based on (34) and (19), we are able to get θ_P as shown in (18). Therefore, during $[t_0, t_1]$, we can use $s(t)$ in (11) with the obtained θ_P and θ_1 used in b; during $[t_1, t_0+P]$ the speed $s(t) = 0$.

Proof of Lemma 7

We consider the interval $[v, t_1]$ first. The temperature curve $\theta(t)$ in time interval $[v, t_1]$ can be written as

$$\theta(t) = \bar{a}e^{\beta\bar{\Delta}_t} + \bar{b}e^{\frac{\gamma}{\gamma-1}\beta\bar{\Delta}_t}, \quad (35)$$

where

$$\bar{a} = \frac{\theta_1 e^{\frac{\gamma}{\gamma-1}\beta\bar{\Delta}_v}-\theta^*}{e^{\frac{\gamma}{\gamma-1}\beta\bar{\Delta}_v}-e^{\beta\bar{\Delta}_v}}, \quad \bar{b} = \frac{\theta^*-\theta_1 e^{\beta\bar{\Delta}_v}}{e^{\frac{\gamma}{\gamma-1}\beta\bar{\Delta}_v}-e^{\beta\bar{\Delta}_v}}. \quad (36)$$

Hence,

$$\theta'(t) = -\bar{a}\beta e^{\beta\bar{\Delta}_t} - \frac{\gamma}{\gamma-1}\bar{b}\beta e^{\frac{\gamma}{\gamma-1}\beta\bar{\Delta}_t}. \quad (37)$$

During $[v, t_1]$ we will slow down the processor for cooling. The speed schedule during $[v, t_1]$ can be written as

$$s(t) = (-\bar{b}\frac{\beta}{\gamma-1})^{\frac{1}{\gamma}}e^{\frac{1}{\gamma-1}\beta\bar{\Delta}_t}. \quad (38)$$

In order to achieve the optimization, we can assume that $\theta'(v) = 0$. Similar to the analysis in deriving (32), we have

$$\frac{\theta_1}{\theta^*}e^{\beta\bar{\Delta}_v} = \gamma-(\gamma-1)e^{-\frac{1}{\gamma-1}\beta\bar{\Delta}_v}. \quad (39)$$

During $[t_1, t_0+P]$, the processor is idle, and (19) still holds. Applying (19) into (39), we have

$$\frac{\theta_P}{\theta^*}e^{\beta\bar{\Delta}_v} = e^{-\beta(P-\Delta)}(\gamma-(\gamma-1)e^{-\frac{1}{\gamma-1}\beta\bar{\Delta}_v}). \quad (40)$$

During $[t_0, u]$, the analysis of speed and temperature curves will be exactly the same as the one we have done in Subsection 4.1. Hence (32) still holds here. Based on (32) and (40), we can get (20).

During $[u, v]$, the processor keeps running at the equilibrium speed s_E. Then the workload in $[t_0, u]$ and $[v, t_1]$ can be constrained by $C - s_E(v-u) = \int_{t_0}^{u} s(t)dt + \int_{v}^{t_1} s(t)dt$ as follows:

$$C - s_E(v-u) = ((-b)^{\frac{1}{\gamma}}(\frac{\gamma-1}{\beta})^{\frac{\gamma-1}{\gamma}}(1-e^{-\frac{1}{\gamma-1}\beta\Delta_u}) + (-\bar{b})^{\frac{1}{\gamma}}(\frac{\gamma-1}{\beta})^{\frac{\gamma-1}{\gamma}}(e^{\frac{1}{\gamma-1}\beta\bar{\Delta}_v}-1)). \quad (41)$$

Applying b and $\bar{b}$ defined in (9) and (36) respectively into (41) together with $v-u = \Delta-\Delta_u-\Delta_v$, we have

$$(C-(\beta\theta^*)^{\frac{1}{\gamma}}(\Delta-\Delta_u-\bar{\Delta}_v))(\frac{\beta}{\gamma-1})^{\frac{\gamma-1}{\gamma}} = (\theta^*-\theta_P e^{-\beta\Delta_u})^{\frac{1}{\gamma}}(e^{\frac{1}{\gamma-1}\beta\Delta_u}-1)^{\frac{\gamma-1}{\gamma}} + (\theta_1-\theta^* e^{-\beta\bar{\Delta}_v})^{\frac{1}{\gamma}}(e^{\frac{1}{\gamma-1}\beta\bar{\Delta}_v}-1)^{\frac{\gamma-1}{\gamma}}. \quad (42)$$

In the right side of the above equation, θ_P can be represented in terms of Δ_u and θ^* based on (32) and θ_1 can be represented in terms of $\bar{\Delta}_v$ and θ^* based on (40). Therefore, with further simplification, (42) can be written as (21).

No explicit formula of Δ_u and $\bar{\Delta}_v$ exists. However, by the fixed point theorem on two dimensional vector $(\Delta_u, \bar{\Delta}_v)$, we can obtain their numerical values in terms of Δ based on (20) and (21) if $\Delta_u + \bar{\Delta}_v \leq \Delta$ holds.

With the above analysis, now we can easily derive the speed schedule. During $[t_0, u]$ we can use $s(t)$ in (11); during $[u, v]$ the speed remains constant s_E; during $[v, t_1]$ we can use $s(t)$ defined in (22), which is obtained by applying $\bar{b}$ in (36) and θ_1 in (39) into (38); during $[t_1, t_0+P]$ the speed is $s(t) = 0$.