On The Design and Application of Thermal Isolation Servers

REHAN AHMED, Computer Engineering and Networks Laboratory, ETH Zürich
PENGCHENG HUANG, Computer Engineering and Networks Laboratory, ETH Zürich
MAX MILLEN, Department of Electrical and Computer Engineering, ETH Zürich
LOTHAR THIELE, Computer Engineering and Networks Laboratory, ETH Zürich

Recently, there has been an increasing trend towards executing real-time applications on multi-core platforms. However, this complicates the design problem, as applications running on different cores can interfere due to shared resources and mediums. In this paper, we focus on thermal interference, where a given task (τ_1) heats the processor, resulting in reduced service (due to Dynamic Thermal Management (DTM)) to another task (τ_2). In real-time domain, where tasks have deadline constraints, thermal interference is a substantial problem as it directly impacts the Worst Case Execution Time (WCET) of the effected application (τ_2). The problem exacerbates as we move to mixed-criticality systems, where the criticality of τ_2 may be greater than the criticality of τ_1, complicating the certification process.

In this paper, we propose a server based strategy (Thermal Isolation Server (TI Server)) which can be used to avoid thermal interference of applications. We also present a heuristic to design TI Servers to meet the timing constraints of all tasks and the thermal constraints of the system. TI Servers are time/space composable, and can be applied to a variety of task models. We also evaluate TI Servers on a hardware test-bed for validation purposes.

CCS Concepts: • **Computer systems organization → Embedded software**; Real-time operating systems; • **Hardware → Temperature simulation and estimation**; **Temperature optimization**; *Safety critical systems*;

Additional Key Words and Phrases: Mixed-Criticality systems, Thermal modeling

ACM Reference format:
Rehan Ahmed, Pengcheng Huang, Max Millen, and Lothar Thiele. 2017. On The Design and Application of Thermal Isolation Servers. *ACM Trans. Embedd. Comput. Syst.* 1, 1, Article 1 (October 2017), 21 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

Traditionally real-time systems have been constructed with simple hardware components geared towards providing predictable execution. However, the performance requirements of real-time applications has been growing in several domains; including automotive, aerospace and medical. This has lead to a paradigm shift towards using multi-core [23] or even custom-of-the-shelf (CotS) platforms [16] for constructing real-time systems. Apart from having a potential to solve the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644080 (SAFURE). This work was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0025. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

1539-9087/2017/10-ART1 $15.00
https://doi.org/0000001.0000001

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

performance gap problem, multi-core systems provide several tertiary benefits. These include reduction of cost, weight and energy consumption. However, shifting to multi-cores introduces several new design challenges. One of them is the increased complexity in determining WCET estimates, since tasks executing concurrently on different cores contend for shared system resources (such as cache, bus, accelerators, e.t.c.) which causes interference. To counter task interference, researchers have proposed several contention mitigation strategies e.g., [19]. However the focus of this paper is bounding thermal interference; which has not received much attention.

The issue of thermal interference arises from the fact that modern processor architectures have strict maximum temperature constraints. For instance, Intel Skylake processors have a maximum temperature constraint (Tj_{MAX}) which is set in a read-only model specific register. Typical values for Tj_{MAX} are between $80° - 90°$ C. When core temperatures reach this value, a hardware thermal control circuit aggressively throttles down the frequency and voltage of the cores; until a safe operating temperature is reached. Such DTM strategies adversely effect performance when activated, and introduce an additional design complexity for real-time systems, where strict deadline constraints have to be met.

DTM also adds an interference medium for tasks sharing the same platform. A given task/s may heat up the platform causing DTM to be triggered; resulting in reduced service and possible deadline miss/es for other task/s. In the domain of mixed-criticality systems [4], the problem becomes more adverse because the effected tasks may have higher criticality compared to the tasks which caused DTM to be triggered. To solve this problem, several thermal constrained scheduling approaches for real-time systems have been proposed [3, 5, 6, 9, 17, 18, 24], which aim to minimize and/or bound peak temperature. However, the existing schemes have limitations which we iterate in section 2.

In this work, we propose a new scheduling construct called TI Server. A TI Server has a thermal budget associated with it; which upper-bounds the temperature increase caused by tasks executed by it. Through detailed theoretical analysis, we prove that a given server will never exceed its thermal budget. This allows us to thermally isolate different sets of tasks and avoid thermal interference by ensuring that DTM is never triggered. The total budget is governed by the thermal constraint of the hardware platform (e.g., Tj_{MAX} for Intel Skylake). Furthermore, we prove that TI Servers are time and space composable; which implies that we can simply *add* the thermal budgets of several servers executing in parallel to compute their net worst-case temperature increase. This composability significantly simplifies the server design problem. We also propose a heuristic to design a set of TI Servers, given the computation requirements of tasks and the thermal constraint of the hardware platform. We evaluate the performance of TI Server via extensive simulations. Furthermore, we also implement the scheme on a hardware test-bed to validate our theoretical analysis/results.

This paper is organized as follows: Section 2 enumerates the related research in the domain of thermal protection in the presence of timing constraints. Section 3 presents our processor, thermal and task model. Section 4 explains how the thermal model is solved and defines/explains thermal composability. Section 5 presents TI Servers and explains how they can be used to avoid thermal interference and provide timing guarantees. Section 6 presents a heuristic for designing TI Servers. Section 7 presents experimental evaluation which shows the efficacy of TI Servers in reducing peak temperature. Section 8 demonstrates how TI Servers can be used to provide thermal isolation/protection for a mixed-critical application executing on a hardware test-bed. This is followed by conclusion and appendix.

2 RELATED WORK

Thermal constrained scheduling in the real-time domain has been an active area of research. Initial schemes focused on Dynamic Voltage and Frequency Scaling (DVFS) to schedule implicit deadline periodic tasks on uni-core platforms [6, 24]. In the uni-core domain, certain performance throttling

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

schemes also exist [1, 12]. However, these schemes have a common restriction of providing solutions for uni-core platforms. The thermal characteristics of multi-cores are different in the following manner: In the uni-core domain, for a given value of constant power dissipation, core temperature always monotonically approaches a steady state value. In the multi-core domain, under the same constant power dissipation assumption, the evolution of temperature to a steady state value is not necessarily monotonic [15]. In fact, transient temperature of a given core may rise to a value greater than both its initial temperature and its final steady state temperature. This is because of the heat transfer across the cores. This difference significantly changes the thermal constrained scheduling problem for multi-cores.

In the multi-core domain, thermal constrained real-time scheduling schemes have been studied. Chantem et al. [5] solve a Mixed Integer Linear Programming (MILP) formulation to find a schedule that minimizes the temperature across the multi-core platform while respecting task timing constraints. Fisher et al. [9] derive speeds of the processing cores based on their thermal characteristics such that peak temperature is minimized and deadline constraints of sporadic tasks are met. Ahmed et al. [3] propose a necessary schedulability condition for implicit deadline periodic tasks executing on a multi-core platform. They also propose a thermally optimal scheme for scheduling implicit deadline periodic tasks, assuming a fluid execution model. Although, all of these are valuable contributions, they have the following limitations:

(1) With the exception of [1, 12], which are fundamentally uni-core schemes, all mentioned thermal protection schemes are designed for specific task models. Extending these schemes to different task models is non-trivial.
(2) To the best of our knowledge, no thermal constrained scheduling scheme has been proposed for mixed-critical systems. As section 8 demonstrates, TI Servers can be used to solve this problem.
(3) With the exception of [3], all schemes address the problem of peak temperature minimization for the entire application; but do not directly address the problem of thermal isolation. Specifically, the following question is not addressed: Given a *sub-set* of tasks, what is the maximum temperature *increase* caused by its execution, independent of other tasks. It is important to answer the thermal isolation question, especially in the context of mixed-critical systems; where isolation is a certification requirement. [3] relies on a fluid scheduling model to compute temperature increase; which is not practical in a real system.

A different line of work focuses on *bounding* peak temperature for a given task model assuming a work-conserving schedule [17, 18]. However, significant temperature reduction is possible if non-work-conserving schedule is used [3]. Therefore, relying only on a temperature bounding scheme for thermal protection, can be pessimistic in-terms of schedulability.

Contributions. : TI Servers provide thermal isolation by construction. TI Servers have an associated *thermal budget* which specifies the upper-bound of temperature increase caused by execution of all tasks associated with that server. To provide timing guarantees, we use compositional real-time analysis [20]. Therefore, TI Servers can be used for scheduling several task models. In this work, we also formalize thermal component based analysis. Using this analysis, we prove that the thermal budgets of TI Servers are time and space composable. This composability significantly simplifies the process of designing TI Servers such that timing constraints of all tasks and thermal constraints of the platform are satisfied. For validation purposes, we implement TI Server on a hardware test-bed and demonstrate how thermal protection/isolation can be provided to a mixed critical application. To the best of our knowledge, this is the first work which provides thermal protection for a mixed-critical system; in either uni-core or multi-core domain.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

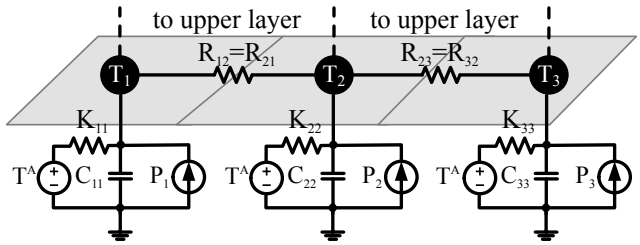


Fig. 1. Single layer in the thermal RC model

3 SYSTEM MODEL

This section formalizes the models used in this work. Bold characters represent vectors and matrices while non-bold characters represent scalars. Subscripts are used to reference individual elements of matrices/vectors; e.g. $H_{k,l}$ denotes the element in k^{th} row and l^{th} column of matrix $\boldsymbol{H}$, and T_i denotes the i^{th} element of vector $\boldsymbol{T}$. $\boldsymbol{I}$ is used to denote identity matrix.

3.1 Processor Model

We consider a multi-core constituting m identical cores. The set of all cores is denoted by M. Each core has two operation modes: *active* and *idle*. The mode of a given core has a direct impact on its dynamic power dissipation. A detailed power model is discussed later in this section. The platform has a thermal constraint $\boldsymbol{T}^{\Delta}$. The temperature of each core $i \in M$ is required to be at or below T_i^{Δ} for safe system operation.

3.1.1 Thermal Model. We develop the thermal model of the multi-core using an equivalent RC network [5, 17, 22]. In this abstraction, temperature is modeled by voltage and power dissipation is modeled by a current source. We model the layout of the chip by four vertical layers, namely heat sink, heat spreader, thermal interface, and silicon die. This is identical to the model adopted by the Hotspot thermal simulator [22]. Each layer is divided into a set of blocks according to the architectural-level units. In our case, we select a processing component abstraction, i.e., we represent each core as an individual node with separate power source and temperature characteristics. Fig. 1 shows the silicon die (processing core) level of our thermal model. $R_{i,j}$ represents the thermal resistance between node i and node j, with $R_{i,i} = \infty$. The processing component abstraction has been shown to be reasonably accurate [7, 25].

In our thermal model, 12 additional nodes are introduced in the heat spreader and heat sink layers to account for the area which is not covered by the subjacent layer. Therefore, a multi-core system with m cores is modeled by $n = 4 \cdot m + 12$ thermal nodes. The n-dimensional temperature vector $\boldsymbol{T}(t)$ at time t is described by a set of first-order differential equations:

$$\boldsymbol{C} \cdot \frac{\mathrm{d}\boldsymbol{T}(t)}{\mathrm{d}t} = (\boldsymbol{G} - \boldsymbol{K}) \cdot \boldsymbol{T}(t) + \mathbf{Power}(t) + \boldsymbol{K} \cdot \boldsymbol{T}^{\mathbf{A}} \tag{1}$$

where $\boldsymbol{C}$ is the thermal capacitance diagonal matrix, $\boldsymbol{G}$ is a square matrix composed with thermal conductances such that:

$$G_{\mathrm{i,j}} = \begin{cases} 1/R_{i,j} & \text{If } i \neq j \\ -\sum\limits_{0 \leq k < n} 1/R_{i,k} & \text{Otherwise} \end{cases} \tag{2}$$

, $\boldsymbol{K}$ is the thermal ground conductance diagonal matrix, $\mathbf{Power}(t)$ is the power dissipation vector at time t, and $\boldsymbol{T}^{\mathbf{A}} = T^{\mathrm{A}} \cdot [1, \ldots, 1]^{\intercal}$ is the ambient temperature vector.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

3.1.2 Power Model. The total power dissipation has leakage and dynamic power components. Leakage power can be approximated to linearly increase with temperature [5, 13]. Dynamic power of a core depends on its mode (*active* or *idle*). Based on these assumptions, the power dissipation of the system is given by the following equation:

$$\mathbf{Power}(t) = \boldsymbol{\phi} \cdot \boldsymbol{T}(t) + \boldsymbol{\psi}(t) \tag{3}$$

where $\boldsymbol{\phi}$ is diagonal matrix of dimension n with constant coefficients, and $\boldsymbol{\psi}(t)$ a vector with n elements such that:

$$\psi_i(t) = \begin{cases} \psi^{\text{active}} & \text{if node } i \text{ is a core, } \textit{active} \text{ at time } t \\ \psi^{\text{idle}} & \text{if node } i \text{ is a core, } \textit{idle} \text{ at time } t \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

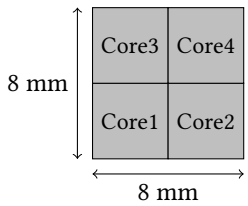
i.e., ψ^{active} and ψ^{idle} are used to denote the temperature independent power dissipation of a core in *active* and *idle* mode respectively. ψ^{d} is used to denote $\psi^{\text{active}} - \psi^{\text{idle}}$. The power dissipation of non-core nodes is zero. Therefore, $\phi_{ii} = 0$ if node i is not a core.

3.2 Application Model

We assume that individual tasks are sporadic. A given task τ_i is characterized WCET E_i, minimum inter-arrival time W_i, relative constrained deadline $D_i \leq W_i$. The set of all tasks is denoted by Π.

3.3 Sample platform architecture

Unless stated otherwise, we use the platform model presented in Fig. 2 for all empirical results, evaluations and examples.



Parameter	Value
ψ^{active}	70 W
ψ^{idle}	20 W
$\boldsymbol{\phi}$	zero matrix
T^{A}	25°C

Fig. 2. System parameters for the sample architecture

The thermal parameters ($\boldsymbol{C}$, $\boldsymbol{G}$ and $\boldsymbol{K}$) are acquired by simulating the sample architecture floorplan with Hotspot thermal simulator's [22] default configuration.

4 THERMAL MODEL SOLUTION AND COMPOSABILITY

Our thermal model is an RC network which is Linear Time Invariant (LTI) system. In LTI systems, superposition can be applied to "compose" the effect of multiple sources. In the context of this work, ability to superpose implies that we can analyze the thermal impact of different executions independently and then compose these effects to get the overall system temperature. We will use the property of thermal composability to design thermal isolation servers in section 5. In this section we first present a closed form solution to (1) for a given value of constant power dissipation and known initial temperature. We then explain how thermal effects of different executions can be composed.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

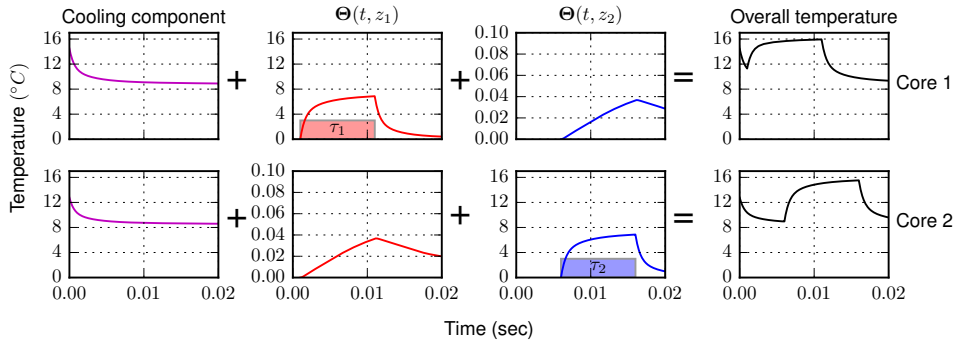


Fig. 3. Composing thermal effects of z_1 and z_2

4.1 Solving the thermal model

To solve the system of differential equations given in (1), we first follow a few algebraic simplification steps to homogenize this system. Firstly, we represent power dissipation as given in (3), and rearrange (1) as temperature dependent and temperature independent terms.

$$\boldsymbol{T}'(t) = \boldsymbol{A} \cdot \boldsymbol{T}(t) + \boldsymbol{B}(t) \tag{5}$$

where $\boldsymbol{A} = \boldsymbol{C}^{-1} \cdot (\boldsymbol{G} + \boldsymbol{\phi} - \boldsymbol{K})$ and $\boldsymbol{B}(t) = \boldsymbol{C}^{-1} \cdot (\boldsymbol{K} \cdot \boldsymbol{T}^{A} + \boldsymbol{\psi}(t))$. This system of differential equations is not homogeneous because of $\boldsymbol{B}(t)$. To homogenize this system, we first assume that power dissipation during interval $[0, t]$ is constant, and is represented by $\boldsymbol{B}(0)$. Furthermore, we perform the following variable transformation:

$$\hat{\boldsymbol{T}}(t) = \boldsymbol{T}(t) - \boldsymbol{T}^{\infty}(\boldsymbol{B}(0)) \tag{6}$$

where $\boldsymbol{T}^{\infty}(\boldsymbol{B}(0))$ is the steady state temperature of the system assuming constant power dissipation as in $\boldsymbol{B}(0)$. We can compute $\boldsymbol{T}^{\infty}(\boldsymbol{B}(0))$ by setting $T'(t) = 0$ in (5):

$$\boldsymbol{T}^{\infty}(\boldsymbol{B}(0)) = -\boldsymbol{A}^{-1}\boldsymbol{B}(0) \tag{7}$$

Substituting $\boldsymbol{T}(t) = \hat{\boldsymbol{T}}(t) - \boldsymbol{A}^{-1}\boldsymbol{B}(0)$ in (5):

$$\hat{\boldsymbol{T}}'(t) = \boldsymbol{A} \cdot \hat{\boldsymbol{T}}(t) \tag{8}$$

This system of differential equations is homogeneous and has the following closed form solution:

$$\hat{\boldsymbol{T}}(t) = \boldsymbol{e}^{\boldsymbol{A} \cdot t} \cdot \hat{\boldsymbol{T}}(0) \tag{9}$$

where $\boldsymbol{e}^{\boldsymbol{A} \cdot t}$ represents matrix exponential [14]. If all eigenvalues of $\boldsymbol{A}$ are distinct, then $\boldsymbol{e}^{\boldsymbol{A} \cdot t} = \boldsymbol{V} \cdot e^{\boldsymbol{\lambda}(t)} \cdot \boldsymbol{V}^{-1}$. Where $\boldsymbol{V}$ is a square matrix with column i representing i^{th} eigenvector of $\boldsymbol{A}$ and $\boldsymbol{\lambda}$ is a diagonal matrix where diagonal is equal to the eigenvalues of $\boldsymbol{A}$. The exponential of $\boldsymbol{\lambda}$ can be computed by exponentiating the diagonal elements and setting the non-diagonal elements to zero. We can use the following equation to compute system temperature assuming that power dissipation is constant in interval $[0, t]$ and $\boldsymbol{T}(0)$ is the known initial temperature vector:

$$\boldsymbol{T}(t) = \boldsymbol{e}^{\boldsymbol{A} \cdot t}(\boldsymbol{T}(0) + \boldsymbol{A}^{-1}\boldsymbol{B}(0)) - \boldsymbol{A}^{-1}\boldsymbol{B}(0) \tag{10}$$

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

4.2 Thermal composability

Let $\Theta(t, z)$ represent the temperature increase due to an execution z which is characterized by a tuple $(\mathrm{st}_z, \mathrm{e}_z, \mathrm{cr}_z)$, specifying its start time, end time, and core. We call $\Theta(t, z)$ the *thermal component* of z. We show in this section that $\Theta(t, z)$ does not depend on initial temperature and/or other executions. Due to this property, thermal components can be composed to compute system temperature.
We now explain with an example how thermal components can be computed and composed in space. For notational brevity, define:

$$B_i^{\mathrm{idle}} = \begin{cases} (K_{i,i} \cdot T^A + \psi^{\mathrm{idle}})/C_{i,i} & \text{if node } i \text{ is a core} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$B_i^{\mathrm{j},\psi^{\mathrm{d}}} = \begin{cases} (\psi^d)/C_{i,i} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Suppose that in a quad-core system, core 1 and core 2 are active indefinitely starting at time 0. Executions on core 1 and core 2 are called z_1 and z_2 respectively. The temperature of the multi-core system can be computed using (10) where $\boldsymbol{B}(0) = \boldsymbol{K} \cdot \boldsymbol{T^A} + [\psi^{\mathrm{active}}\ \psi^{\mathrm{active}}; \psi^{\mathrm{idle}}\ \psi^{\mathrm{idle}}\ 0\ 0\ \cdots]^\intercal$. Splitting this vector $\boldsymbol{B}(0) = \boldsymbol{B^{\mathrm{idle}}} + \boldsymbol{B^{1,\psi^d}} + \boldsymbol{B^{2,\psi^d}}$. Using (10) and (7), temperature can be computed using the following equation:

$$\boldsymbol{T}(t) = \boldsymbol{e^{A \cdot t}}(\boldsymbol{T}(0) - \boldsymbol{T^\infty}(\boldsymbol{B^{\mathrm{idle}}})) + \boldsymbol{T^\infty}(\boldsymbol{B^{\mathrm{idle}}}) + (\boldsymbol{I} - \boldsymbol{e^{A \cdot t}})\boldsymbol{T^\infty}(\boldsymbol{B^{1,\psi^d}}) + (\boldsymbol{I} - \boldsymbol{e^{A \cdot t}})\boldsymbol{T^\infty}(\boldsymbol{B^{2,\psi^d}}) \tag{13}$$

The three additive terms in (13) can be interpreted as:
$\boldsymbol{e^{A \cdot t}}(\boldsymbol{T}(0) - \boldsymbol{T^\infty}(\boldsymbol{B^{\mathrm{idle}}})) + \boldsymbol{T^\infty}(\boldsymbol{B^{\mathrm{idle}}})$: Temperature assuming all cores are idle and initial temperature is $\boldsymbol{T}(0)$. We will also refer to this term as the *cooling component.*
$(\boldsymbol{I} - \boldsymbol{e^{A \cdot t}})\boldsymbol{T^\infty}(\boldsymbol{B^{1,\psi^d}})$: Thermal component of z_1.
$(\boldsymbol{I} - \boldsymbol{e^{A \cdot t}})\boldsymbol{T^\infty}(\boldsymbol{B^{2,\psi^d}})$: Thermal component of z_2.
These three terms can be computed independently and then composed to get overall system temperature as shown in (13). Similarly, we can also compose in time. In general, if we have to evaluate the impact of n different executions, we can have $n + 1$ independent additive terms. The first term, as in (13), is the cooling component. The remaining n additive terms are the thermal components of the n executions. To compute the thermal component of an execution z, we can use (10) with the following assumptions:

- $\boldsymbol{T}(0) = \boldsymbol{T^A} = [0, \ldots, 0]^\intercal$
- $\psi(t)_i = \begin{cases} \psi^d & \text{if } \mathrm{st}_z \le t \le e_z \text{ and } i = \mathrm{cr}_z \\ 0 & \text{otherwise} \end{cases}$

Therefore:

$$\Theta(t, z) = \begin{cases} (\boldsymbol{I} - \boldsymbol{e^{A(t - \mathrm{st}_z)}})\boldsymbol{T^\infty}(\boldsymbol{B^{\mathrm{cr}_z,\psi^d}}) & \text{if } \mathrm{e}_z \ge t \ge \mathrm{st}_z \\ \boldsymbol{e^{A(t-e_z)}} \cdot \Theta(\mathrm{e}_z, z) & \text{if } t > \mathrm{e}_z \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Fig. 3 shows another example illustrating thermal composability in time. In this figure, ambient temperature is assumed to be zero and the thermal components of two executions z_1 ($\mathrm{st}_{z_1} = 1\mathrm{ms}, \mathrm{e}_{z_1} = 11\mathrm{ms}, \mathrm{cr}_{z_1} = 1$) and z_2 ($\mathrm{st}_{z_2} = 6\mathrm{ms}, \mathrm{e}_{z_2} = 16\mathrm{ms}, \mathrm{cr}_{z_2} = 2$) are evaluated. As shown in Fig. 3, the two thermal components can then be added to the cooling component to compute overall system temperature.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

5 THERMAL ISOLATION SERVERS

In this section we introduce TI Server, which is the scheduling construct used to provide thermal isolation/protection. TI Server are partitioned periodic resources with a static schedule. TI Server S_i is characterized by period P_i, utilization U_i, phase Φ_i (where $0 \leq \Phi_i \leq P_i(1-U_i)$), and the core on which S_i executes cr_i. We will refer to cr_i as S_i's *self core*. state(S_i, t) represents the state of TI Server S_i at time t and can be either *active* or *idle*. state(S_i, t) follows a fixed periodic schedule such that:

$$\text{state}(S_i, t) = \begin{cases} active & \text{if } k \cdot P_i + \Phi_i \leq t \leq P_i(k+U_i) + \Phi_i \\ idle & \text{otherwise} \end{cases}$$

where $k \in Z_0^+$ i.e., set of non-negative integers. Each server is assigned a set of sporadic tasks. When server state is *active*, Earliest Deadline First (EDF) or Fixed Priority (FP) is used to schedule individual tasks. Nothing is executed when server state is *idle*. Note that the schedule of server state is static and does not depend on the workload (tasks assigned to the server). Multiple servers can execute on the same core as long as their active times (time range where server state is active) do not overlap.

In order to provide thermal isolation, a TI Server is assigned a thermal budget, $\boldsymbol{\Lambda^i}$ which is the upper bound on temperature increase caused by all tasks executed by S_i. $\boldsymbol{\Lambda^i}$ is a function of P_i, U_i and cr_i. In this section, we will first compute $\boldsymbol{\Lambda^i}$ and analyze how changing P_i and U_i changes $\boldsymbol{\Lambda^i}$. We will then study how timing guarantees can be given using TI Servers and how multiple TI Servers can be executed together without violating thermal constraint. The proofs of all theorems are given in the appendix.

5.1 Thermal properties of TI Servers

Let $\boldsymbol{\Theta}(t, S_i)$ represent the thermal component of TI Server S_i i.e., temperature increase at time t due to all executions performed by S_i until t. To capture maximum workload, assume that S_i never runs out of tasks to execute when its state is *active*.

Theorem 5.1. *The thermal component at the end of active time of S_i with $P_i > 0$ converges to:*

$$\boldsymbol{\Theta}(\lim_{k\to\infty}(P_i(k+U_i)+\Phi_i), S_i) = (\boldsymbol{I} - \boldsymbol{e}^{\boldsymbol{A}\cdot P_i})^{-1}(\boldsymbol{I} - \boldsymbol{e}^{\boldsymbol{A}\cdot P_i \cdot U_i}) \cdot \boldsymbol{T}^{\infty}(\boldsymbol{B}^{\mathbf{cr_i},\boldsymbol{\psi^d}}) \tag{15}$$

Theorem 5.2. *The thermal component of S_i with $P_i \to 0$ converges to:*

$$\boldsymbol{\Theta}(\lim_{t\to\infty} t, S_i) = \boldsymbol{T}^{\infty}(\boldsymbol{B}^{\mathbf{cr_i},\boldsymbol{\psi^d}}) \cdot U_i \tag{16}$$

Theorem 5.3. *The upper bound on thermal component of S_i is given by:*

$$\Lambda^i_j = -A^{-1}_{j,\text{cr}_i} \cdot \Theta(t_e, S_i)_{\text{cr}_i} / (-A^{-1}_{\text{cr}_i,\text{cr}_i}) \tag{17}$$

$$\textit{where } \boldsymbol{\Theta}(t_e, S_i) = \begin{cases} \textit{R.H.S of (15)} & \textit{if } P_i > 0 \\ \textit{R.H.S of (16)} & \textit{if } P_i \to 0 \end{cases}$$

With the help of (17), we can analyze how period and utilization of a TI Server impact its self core thermal budget. This is plotted in Fig. 4 for S_i where $\text{cr}_\text{i} = 1$. As expected, Λ^i_1 increases with utilization, as higher active time to idle time ratio leads to higher value of thermal component. The rate of change of thermal budget is higher at low utilizations and approaches zero as we approach utilization 1. At utilization 1 the thermal budget approaches $T^{\infty}(\boldsymbol{B}^{\mathbf{1},\boldsymbol{\psi^d}})_1$ for all periods. On the other axis, as we increase the period, the thermal budget increases; as larger periods lead to longer contiguous active times. The initial rate of increase of thermal budget is higher compared to its value at higher periods. For large period values, all servers of non-zero utilization approach

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

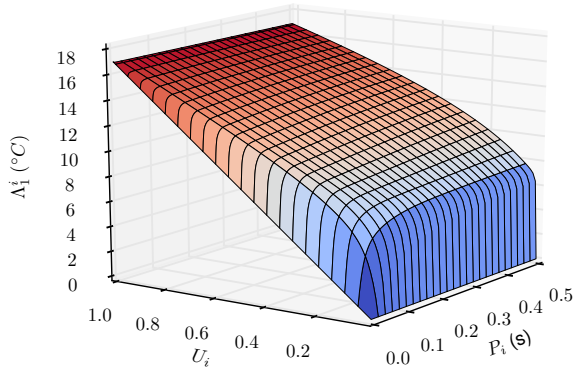
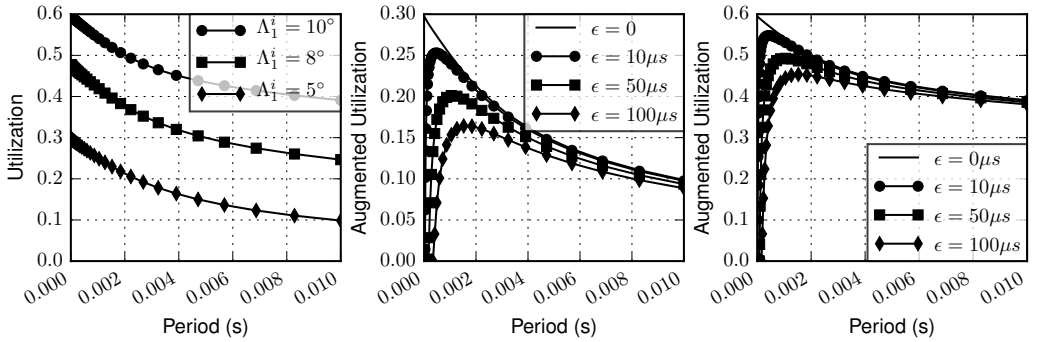


Fig. 4. Self core thermal budget of S_i where $\text{cr}_i = 1$, as a function of server period and utilization



(a) $\Lambda_1^i = \{5°, 8°, 10°\}C, \epsilon = 0\mu s$, (b) $\Lambda_1^i = 5°C, \epsilon = \{0, 10, 50, 100\}\mu s$ (c) $\Lambda_1^i = 10°C, \epsilon = \{0, 10, 50, 100\}\mu s$

Fig. 5. Period/Utilization trade-off when core 1 thermal budget for TI Server S_i executing on core 1 is fixed.

$T^\infty(\boldsymbol{B^{1,\psi^d}})_1$. We cannot discern this from Fig. 4 since we do not show periods higher than 0.5s (convergence is visible for periods $\approx$ 100s).

It is also interesting to see what the constant thermal budget design space looks like. Given a fixed value of self core thermal budget $\Lambda_{\text{cr}_i}^i$ of server S_i, following inequality has to be satisfied:

$$\Lambda_{\text{cr}_i}^i \geq \begin{cases} [(\boldsymbol{I} - \boldsymbol{e}^{\boldsymbol{A}\cdot P_i})^{-1}(\boldsymbol{I} - \boldsymbol{e}^{\boldsymbol{A}\cdot P_i \cdot U_i})\boldsymbol{T}^\infty(\boldsymbol{B^{\text{cr}_i,\psi^d}})]_{\text{cr}_i} & \text{if } P_i > 0 \\ T^\infty(\boldsymbol{B^{\text{cr}_i,\psi^d}})_{\text{cr}_i} \cdot U_i & \text{if } P_i \to 0 \end{cases} \tag{18}$$

For the $P_i > 0$ case, we have a continuous range of $(P_i,\ U_i)$ tuples that satisfy the minimum feasible value of $\Lambda_{\text{cr}_i}^i$ in (18).

Theorem 5.4. *Given a fixed self core thermal budget $\Lambda_{\text{cr}_i}^i$ of S_i, the highest utilization point occurs when $P_i \to 0$.*

Now we empirically look at the constant self core thermal budget design space. Fig. 5a shows the trade-off between server period and utilization. The server is run on core 1 and the thermal budget of core 1 is fixed. We present results for three different values of the fixed self core thermal budget $\{5, 8, 10\}$. For each of these self core thermal budgets, the thermal budgets of other cores are

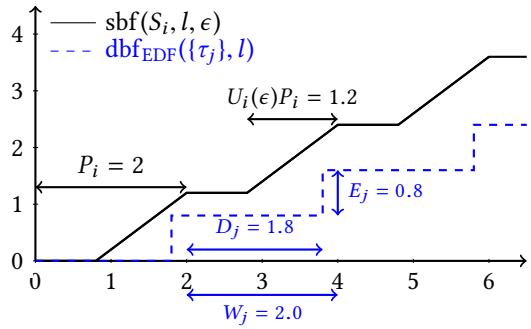


Fig. 6. SBF of server S_i $(P_i, U_i(\epsilon)) = (2.0, 0.6)$. DBF of EDF scheduled task-set $\{\tau_j\}$ where $(E_j, D_j, W_j) = (0.8, 1.9, 2.0)$

calculated using (17). As stated by Thm. 5.4 and illustrated by Fig. 5a, it is preferable to have the server period as low as possible. However, such a scheduling scheme is hardly practical since it incurs high number of preemptions. To take this into account, we introduce a server overhead ϵ. ϵ time is wasted when a server switches from idle to active state. The power dissipated during ϵ is ψ_a. Therefore, the server utilization which can be used to execute tasks becomes:

$$U_i(\epsilon) = \frac{\max(P_i \cdot U_i - \epsilon, 0)}{P_i} \tag{19}$$

We call $U_i(\epsilon)$ the *augmented utilization* of server S_i. When $\epsilon > 0$, the optimal server period is > 0. Fig. 5b and Fig. 5c show the trade-off between augmented utilization and period for different values of ϵ and Λ_1^i. In general, the optimal point (maximum utilization) moves to the right (higher period values) when ϵ is increased and moves to the left when self core budget is increased.

5.2 Timing guarantees

In this section, we cover how timing guarantees can be provided using TI Servers. As stated earlier, TI Servers execute tasks using EDF or FP in their active time slots. For such periodic resources, Shin et al. [20] derive schedulability tests based on Supply Bound Function (SBF) and Demand Bound Function (DBF). These functions have the following definitions:

Supply Bound Function: SBF lower bounds the amount of supplied execution within any time window of length l. The SBF for TI Server S_i is given by:

$$\text{sbf}(S_i, l, \epsilon) = \lfloor l/P_i \rfloor \cdot P_i \cdot U_i(\epsilon) + \max\{l - P_i(1 - U_i(\epsilon)) - \lfloor l/P_i \rfloor \cdot P_i, \quad 0\} \tag{20}$$

Note that the SBF expression in (20) is different from the SBF of a periodic resource given in [20]. This is because we assume a static server schedule; as opposed to the dynamic schedule assumed in [20].

Demand Bound Function: DBF upper bounds the minimum amount of compulsory execution within any time window of fixed length l. When EDF scheduling is used, DBF for a sporadic task-set Π can be computed as:

$$\text{dbf}_{\text{EDF}}(\Pi, l) = \sum_{\tau_j \in \Pi} \max\left\{\left(\left\lfloor (l - D_j)/W_j \right\rfloor + 1\right) \cdot E_j, \quad 0\right\} \tag{21}$$

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

When FP scheduling is used, a function similar to DBF is computed for each sporadic task $\tau_j \in \Pi$ as:

$$\text{dbf}^*_{\text{FP}}(\Pi, l, j) = E_j + \sum_{\tau_i \in \text{HP}_\Pi(j)} (\lceil l/W_i \rceil) \cdot E_i \tag{22}$$

where $\text{HP}_\Pi(j)$ is the set of all tasks in Π with priority greater than τ_j. Note that $\text{dbf}^*_{\text{FP}}(\Pi, l, j)$ is not strictly a demand bound function. This is because, its value does not characterize the compulsory execution requirement for all values of l. Fig. 6 plots sample SBF and DBF for a given TI Server and task-set respectively.

Theorem 5.5. *(Theorem 4.1 from [20]) Let Π_i denote of tasks assigned to TI Server S_i which uses EDF for scheduling. A necessary and sufficient condition for meeting deadlines of all tasks in Π_i is:*

$$\text{dbf}_{\text{EDF}}(\Pi_i, l) \leq \text{sbf}(S_i, l, \epsilon) \qquad \forall l \in [0,\ \text{LCM}_{\Pi_i}] \tag{23}$$

where LCM_{Π_i} is the least common multiple of the periods of all tasks in Π_i.

Theorem 5.6. *(Theorem 4.2 from [20]) Let Π_i denote set of tasks assigned to TI Server S_i which uses FP scheduling. A necessary and sufficient condition for meeting deadlines of all tasks in Π_i is:*

$$\forall \tau_j \in \Pi_i \ \ \exists\, l \in [0, D_j] \qquad \text{dbf}^*_{\text{FP}}(\Pi_i, l, j) \leq \text{sbf}(S_i, l, \epsilon) \tag{24}$$

Theorem 5.7. *Given two TI servers S_1 and S_2 with $U_1(\epsilon) \geq U_2(\epsilon)$ and $P_1 \leq P_2$ and $P_2/P_1 \in Z^+$, then $\text{sbf}(S_1, l, \epsilon) \geq \text{sbf}(S_2, l, \epsilon) \ \ \forall l$.*

The implication of Thm. 5.7 is that, keeping the augmented utilization same, low harmonic server periods are *always* better in-terms of schedulability. This means that, for a given self core thermal budget and ϵ, all harmonic server periods greater than the period that yields maximum utilization (see figures 5b and 5c), are sub-optimal.

5.3 Composing multiple servers

Theorem 5.8. *Given n TI Servers, the maximum temperature increase due to their execution is upper bounded by:*

$$\sum_{1 \leq i \leq n} \mathbf{\Lambda}^i \tag{25}$$

Theorem 5.9. *Suppose that $\boldsymbol{T}^{\Delta}$ is the thermal constraint, and the platform is only executing n TI Servers. In this setting, we have the following sufficient thermal feasibility condition:*

$$\left[\sum_{1 \leq i \leq n} \mathbf{\Lambda}^{\boldsymbol{i}} \right]_j \leq T^{\Delta}_j - [\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\textbf{idle}})]_j \qquad \forall j \in M \tag{26}$$

Theorems 5.8 and 5.9 are the direct result of thermal composability of TI Servers. This simplifies the application design process under thermal constraints. Effectively, it allows us to individually design TI Servers for subsets of tasks that constitute the entire application. Each TI Server can be designed relatively independently of other servers; where only (26) has to be checked to guarantee adherence to thermal constraint. The simplicity of the design process is also illustrated in the next section, where we present a heuristic for designing TI Servers; and in section 8 where we illustrate how TI Servers can be used to provide thermal protection to a mixed-critical application running on a hardware test-bed.

Note that TI Servers only cover the scheduling of independant tasks. If tasks have precedence constraints, then the timing feasibility results in this section can not be applied. However, this impacts the timing feasibility results and not the thermal results. In principle, an elaborate scheme could be conceived which formulates the scheduling problem as a MILP with task precedence and TI Server scheduling constraints.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

Variables:

$$\alpha_{i,j} = \begin{cases} 1, & \text{if task } \tau_i \text{ is assigned to core } j \\ 0, & \text{otherwise} \end{cases}$$

Θ_i = Thermal component of core i

CoreUtil_i = Utilization of core i

Objective:

$$\text{maximize} \quad \min_{1 \le i \le m} \left(T_i^{\Delta} - [\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\text{idle}})]_i - \Theta_i \right)$$

Constraints:

$$\sum_{1 \le j \le m} \alpha_{i,j} = 1 \qquad \forall \tau_i \in \Pi \tag{27}$$

$$\text{CoreUtil}_j = \sum_{\tau_i \in \Pi} \left(\frac{E_i}{W_i} \cdot \alpha_{i,j} \right) \qquad \forall j \in M \tag{28}$$

$$\text{CoreUtil}_j \le 1 \qquad \forall j \in M \tag{29}$$

$$\Theta_j = \frac{\psi^d}{C_{j,j}} \cdot \sum_{i \in M} -\boldsymbol{A}_{j,i}^{-1} \cdot \text{CoreUtil}_i \qquad \forall j \in M \tag{30}$$

Formulation 1. MILP for optimal thermal partitioning of sporadic tasks

6 DESIGNING THERMAL ISOLATION SERVERS

This section proposes a heuristic for designing TI Servers such that the peak temperature is minimized while adhering to timing constraints. Note that due to the thermal component based analysis used by TI Servers, the design process is relatively low complexity compared to exhaustively evaluating all combinations of scheduling options. The proposed heuristic has two stages: 1) Optimal partitioning of tasks to cores. 2) Optimal server configuration search.

6.1 Optimal task partitioning

In the partitioning phase, we assign tasks to cores *optimally* assuming a fluid system, using a MILP formulation. In Formulation 1, we assign tasks for cores such that the minimum temperature difference between threshold temperature and the maximum temperature of a given core is maximized. The constraints state that: (27) each task has to be assigned to exactly one core, (28) utilization of each core (CoreUtil) is the sum of utilization of tasks assigned to that core, (29) utilization of each core is ≤ 1 for timing feasibility, and (30) assignment of thermal component using (16) assuming TI Server period $\to 0$.

6.2 Optimal server configuration search

Next we employ a search strategy to find the server configurations that meet the timing requirements of the partitioned tasks. For the i^{th} core, let us suppose that the utilization after partitioning is U_i^* and the server preemption overhead is ϵ. We consider server periods in the range $[\frac{\epsilon}{1-U_i^*},\ P^{\max}]$, in steps of 0.01. $P^{\max}$ is specific to a given problem, and in our evaluations, was set statically to $2ms$. However, dynamic allocation of $P^{\max}$ based on self core thermal budget is also possible. For each period, we find the minimum TI Server utilization such that the schedulability condition of Thm. 5.5 in case of EDF or Thm. 5.6 in case of FP, is satisfied. This gives us a set of timing feasible TI Servers. Out of this set, we select the configuration which has the lowest self core thermal budget.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

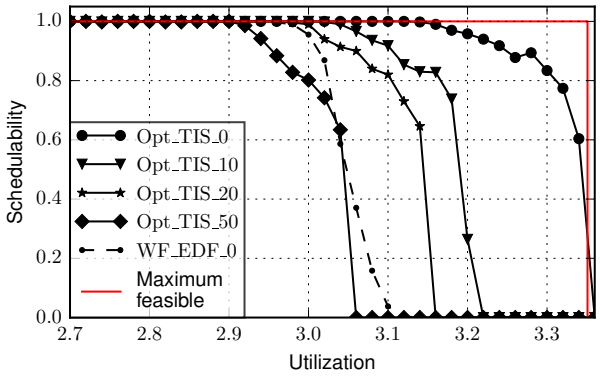


Fig. 7. Schedulability of TI Server with overheads $\{0\mu s, 10\mu s, 20\mu s, 50\mu s\}$ compared with EDF with worst-fit partitioning

7 EXPERIMENTAL EVALUATION

Apart from providing thermal isolation by construction, TI Servers can be used to reduce peak system temperature. In this section, we illustrate the efficacy of TI Servers in this context. The platform simulated in this section is a quad-core described in section 3.3. We simulate 80 evenly spaced utilization points in the range of [2.0, 3.6] (granularity of 0.02). For each utilization point, we simulate 500 synthetic periodic task-sets with implicit deadlines. We use UUniFast-Discard [8] to generate the synthetic task-sets where maximum periodic task utilization is set to 1. Harmonic periods between [10ms, 1280 ms] are assigned randomly to tasks. Following schemes are considered:

- WF_EDF_xx: Tasks partitioned using worst-fit bin packing. EDF is used for scheduling tasks on each core. xx specifies the preemption overhead in μs.
- Opt_TIS_xx: Tasks are partitioned using Formulation 1. Tasks are scheduled using TI Servers. xx is the server overhead (ϵ) in μs. EDF is used within server active time.

For each of these schemes, we consider a task-set thermally feasible if the maximum temperature across all cores is $\leq 70^\circ$ C. We exclude task-sets which are not computationally feasible (have a computation utilization > 1 on any core after partitioning). Therefore, the results presented in this section only illustrate thermal feasibility.

Fig. 7 compares the schedulability of TI Servers with various server overheads with EDF. The preemption overhead for EDF is assumed to be zero. The results illustrate that TI Servers significantly improve thermal schedulability as long as ϵ is of the order of a few tens of micro seconds. However, if ϵ is higher (depending on the operating system and the memory footprint of tasks), TI Server degrades thermal schedulability. We can see from Fig. 7 that for $\epsilon = 50\mu s$, TI Server performs worse compared to EDF. However, for $\epsilon = 10\mu s$ and $\epsilon = 20\mu s$, TI Server yields better thermal schedulability. We also plot the fluid case ($\epsilon = 0\mu s$) to illustrate the best possible achievable schedulability. For the simulated platform, all cores reach threshold temperature of 70°C if power dissipation of each core is 61.89 W. Assuming a fluid system , this translates into utilization of each core being $\frac{61.89-\psi^{\text{idle}}}{\psi^d} = 0.838$. Therefore, the maximum possible thermally feasible system utilization which can be achieved by *any* scheduling algorithm is $0.838 \times 4 = 3.3509$. As shown in Fig. 7, Opt_TIS_0 reaches close to this utilization. Note that for the TI Server results in Fig. 7, we only deem a task-set thermally feasible if the temperature upper bound due to execution of all servers is less than 70°C (see (26)). If actual peak temperature is used, TI Server would yield better schedulability. However,

Table 1. FMS parameters

Purpose	CL	Count	Period (ms)	Exec.Time (ms)
Sensor data acquisition	HI	5	200	10
Localization	HI	3	200	10
	HI	3	1000	50
	HI	1	5000	50
Flight-plan management	HI	4	1000	50
	LO	4	1000	50
Flight-plan computation	HI	2	1000	50
	HI	1	5000	750
	HI	1	5000	180
	HI	1	5000	150
	HI	1	5000	90
	HI	1	5000	75
Guidance	HI	1	200	10
Nearest Airport	LO	1	1000	50

the selling point of TI Servers is the ease of analysis/design. Having to simulate the entire schedule to determine thermal feasibility defeats this purpose; which is why thermal feasibility results have been based on temperature upper bounds.

8 GENERALIZING APPLICATION OF THERMAL ISOLATION SERVERS

TI Servers are general in the sense that they can be applied to various task models. To make this case, we illustrate how TI Servers can be applied to a mixed-critical Flight Management System (FMS). We also illustrate in this section how we can apply the theoretical concepts in practice.

8.1 Flight management system application

The FMS application is dual-critical in nature [11]. It comprises of critical tasks such as localization and guidance; and lower criticality tasks such as flight-plan management and a task that determines the nearest airport. The parameters of all tasks are given in Table 1. All tasks are assumed to be periodic with implicit deadlines. The goal here is to provide the HI criticality application thermal protection.

8.2 Platform and experimental setup

We emulate the FMS application on a hardware test-bed. For this purpose, we execute tasks with parameters given in Table 1 on a laptop platform (Lenovo Thinkpad T440p). The platform has Core i7-4700MQ quad-core processor. The operating frequency of all cores is fixed to 3.2 GHz and the fan speed set to maximum. Our hardware platform is running Ubuntu 16.04 with preempt-rt patch [10]. This patch makes the Linux kernel preempt-able, leading to near real-time performance. To execute the tasks, we augmented the HSF [21] framework by adding a new scheduling class for TI Server and adding the ability to log on-chip thermal sensors. Core 0 is reserved for HSF scheduling and temperature logging tasks. In all of the following results, temperature of core 0 is not plotted since it does execute any tasks of the FMS application. LO criticality application is executed on core 1 using EDF. HI criticality application is executed on core 2 and core 3 using TI Servers, where EDF is used within server active times. Thermal constraint is violated if the temperature of any core exceeds

70°C. We choose different scheduling algorithms for LO and HI criticality application, to show how TI Servers can work alongside other scheduling algorithms to provide thermal protection.

8.3 Computing available thermal budget:

To provide thermal protection to the HI criticality application, we need to compute the available thermal budget for execution of HI criticality application using TI Servers, and verify that the available budget is sufficient. To compute this budget, we perform the following calibration tests.

(1) Test1: Core 1 executing LO criticality application using EDF. Core 2 and Core 3 always idle.
(2) Test2: Core 1 executing LO criticality application using EDF. Core 2 and core 3 always active.
(3) Test3: Core 1 always idle. Core 2 and core 3 always active.

In each of these tests, we record the temperature of all cores. Each test is sufficiently long such that temperature profile is similar across hyperperiods[1]. $\boldsymbol{T}(t, \text{Testx})$ is used to represent the temperature profile of test $x \in \{1, 2, 3\}$. Furthermore, $\boldsymbol{PX}(\boldsymbol{T}(t, \text{Testx}))$ is a vector representing the X^{th} percentile temperature of each core e.g., $\boldsymbol{P}100(\boldsymbol{T}(t, \text{Test1}))$ represents the maximum temperature of each core in Test1. $\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\textbf{idle}}) = [36.8, 38.12, 38.6]^{\intercal}$ and is calculated by averaging several temperature values when each core is idle. Total available thermal budget of cores 1-3 (represented as $\boldsymbol{\Lambda}^{\textbf{total}}$) is computed as:

$$\boldsymbol{T}^{\Delta} - \begin{bmatrix} P99.9(\boldsymbol{T}(t, \text{Test1}))_1 \\ P99.9(\boldsymbol{T}(t, \text{Test2}))_2 - P99.9(\boldsymbol{T}(t, \text{Test3}))_2 + [\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\textbf{idle}})]_2 \\ P99.9(\boldsymbol{T}(t, \text{Test2}))_3 - P99.9(\boldsymbol{T}(t, \text{Test3}))_3 + [\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\textbf{idle}})]_3 \end{bmatrix} = \begin{bmatrix} 16.0 \\ 28.88 \\ 27.4 \end{bmatrix}$$

99.9 percentile values are used to account for outliers in temperature measurements.

8.4 Developing the thermal model

To apply TI Server scheme, we need to first develop thermal model of the platform. This is done through several thermal calibration tests. With these tests, we intend to capture the following effects:

- Transient temperature characteristics of each core.
- Steady state temperature characteristics of each core.
- Steady state temperature effect of a given core on every other core.

Note that we do not need to characterize transient thermal effect of a given core on other cores, since this characterization is not required by the TI Server scheme. To capture these effects, we execute calibration tests of the following three types:

(1) Periodic task with period of 10s and computation time of 5s running on core $i \in \{1, 2, 3\}$. All other cores idle. Test duration 60s.
(2) Core i active for 40 minutes where $i \in \{1, 2, 3\}$. All other cores idle. This is followed by all cores idle for 20 minutes.
(3) Cores (i, j) active for 40 minutes where $(i, j) \in \{(1, 2), (1, 3), (2, 3)\}$. The other core is idle. This is followed by all cores idle for 20 minutes.

In all tests, we record the temperature of each core. Core 0 was not thermally modeled.

Developing steady state model for task partitioning: Formulation 1, for partitioning tasks to cores, only requires the $-\boldsymbol{A}^{-1} \cdot \boldsymbol{C}^{-1} \cdot \psi^d$ value to estimate temperature using (30). By (7), $-A^{-1}_{i,j} \cdot \psi^d / C_{i,i}$ can be interpreted as the steady state temperature of core i when core j is dissipating ψ^d and ambient temperature is 0. Therefore, to estimate $\boldsymbol{A}^{-1} \cdot \boldsymbol{C}^{-1} \cdot \psi^d$, we can use calibration tests of type 2 and 3. Each of these calibration tests are performed 10 times and the average of these 10 runs is computed.

[1] hyperperiod is the least common multiple of the periods of all tasks in a given task-set

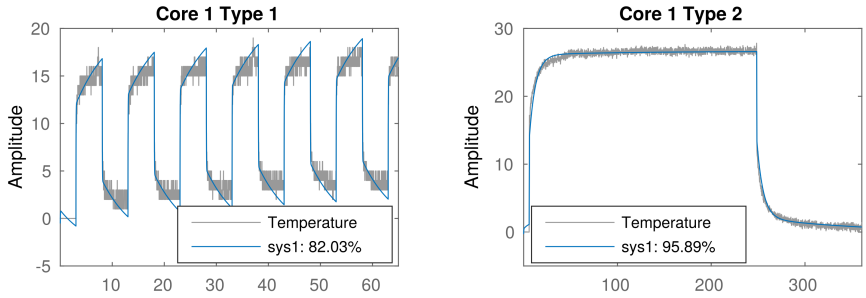


Fig. 8. Model for core 1. Goodness of fit = 82.03% for type 1 and 95.89% for type 2 calibration test.

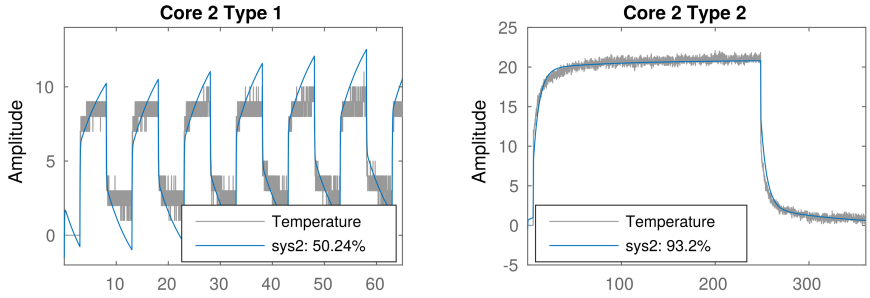


Fig. 9. Model for core 2. Goodness of fit = 50.24% for type 1 and 93.2% for type 2 calibration test.

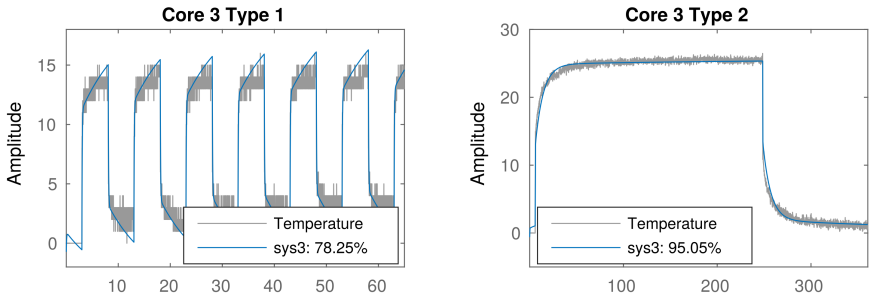


Fig. 10. Model for core 3. Goodness of fit = 78.25% for type 1 and 95.05% for type 2 calibration test.

This is done to mitigate the noise in temperature measurements. $\boldsymbol{T^c}(y)$ is used to represent 99.9^{th} percentile value of each core's temperature, in the averaged calibration trace where all core/s in y are active for 40 minutes. $-\boldsymbol{A}^{-1} \cdot \boldsymbol{C}^{-1} \cdot \psi^d$ can be evaluated as:

$$\begin{bmatrix} T^c(1)_1 - [\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\text{idle}})]_1 & T^c(1,2)_1 - T^c(1)_1 & T^c(1,3)_1 - T^c(1)_1 \\ T^c(1,2)_2 - T^c(2)_2 & T^c(2)_2 - [\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\text{idle}})]_2 & T^c(2,3)_2 - T^c(2)_2 \\ T^c(1,3)_3 - T^c(3)_3 & T^c(2,3)_3 - T^c(3)_3 & T^c(3)_3 - [\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\text{idle}})]_3 \end{bmatrix}$$

$$= \begin{bmatrix} 27.2 & 9.48 & 6.80 \\ 8.68 & 21.60 & 10.68 \\ 7.00 & 8.4 & 25.8 \end{bmatrix}$$

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

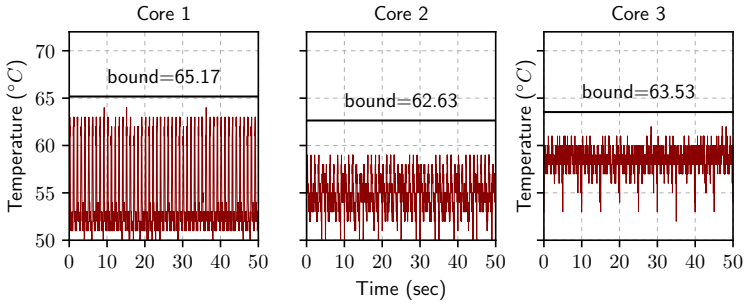


Fig. 11. System temperature. HI criticality tasks partitioned using Formulation 1 and scheduled using TI Server

Developing self-core model: To compute the maximum temperature increase caused due to the execution of a given TI Server on its self-core, we need a thermal model that accurately characterizes the transient and steady state behavior of the TI Server's self-core. To get this thermal model, we use the temperature traces of calibration tests of types 1 and 2. Following three steps are performed:

(1) From all calibration traces of types 1 and 2, we subtract idle temperature ($\boldsymbol{T}^{\infty}(\boldsymbol{B}^{\text{idle}})$). We refer to these traces as the normalized calibration traces.
(2) For each core $i \in \{1, 2, 3\}$, we construct a larger calibration trace by concatenating the calibration traces of types 1 and 2. After this step we get three calibration traces (one for each core) which characterize both transient and steady state behavior of each core. We will refer to these long traces as concatenated traces.
(3) The concatenated trace of core $i \in \{1, 2, 3\}$ is used to estimate the self-core temperature transfer function. The transfer function is estimated using Matlab's tfest function using 3 poles and 2 zeros[2].

Fig. 8, Fig. 9 and Fig. 10 show the modeling results for cores 1, 2 and 3 respectively. In these figures, all of the reported goodness of fit values have been computed using the normalized root mean square cost function (the default setting for Matlab's tfest function). As shown in the figures, the models for all cores have low error especially, when temperature reaches a steady state value (Type 2 calibration tests).

8.5 Server configuration and results:

We can now do thermal constrained task partitioning to assign the HI criticality tasks to core 2 and core 3. An additional constraint is added in Formulation 1 which prevents any HI criticality tasks from being assigned to core 1; since core 1 is reserved exclusively for LO criticality application. The utilization of core 2 and core 3 after partitioning is 0.678 and 0.431 respectively. In our simulations, we computed a safe bound on ϵ to be $150\mu s$. Based on these values, we chose the following server configurations:
$S_1 : P_1 = 10ms, U_1 = 0.693, \text{Core} = 2$
$S_2 : P_1 = 10ms, U_1 = 0.546, \text{Core} = 3$
This server configuration passes the timing feasibility test in Theorem 5.5 and leads to substantial reduction in peak system temperature. Fig. 11 shows the system temperature when Formulation 1 is used to partition the HI criticality tasks on Cores 2 and 3, and TI Server is used to schedule tasks. Fig. 12 shows the system temperature when the HI criticality tasks are partitioned on cores 2 and 3

[2]this configuration lead to low temperature estimation error.

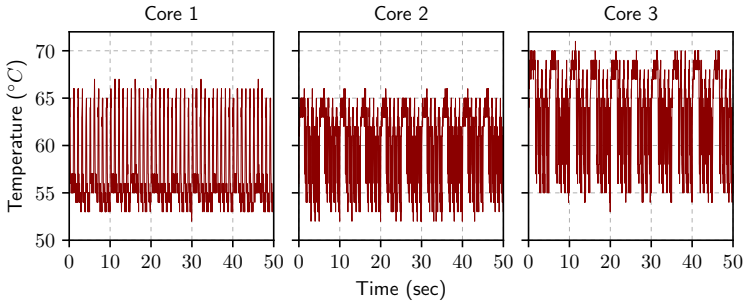


Fig. 12. System temperature. HI criticality tasks partitioned using worst-fit bin packing and scheduled using EDF

using worst-fit bin packing, and tasks are executed using EDF. The later case results in violation of the thermal constraint as core 3 reaches a peak temperature of $72°C$. For the TI Server based scheme, the temperature always remains below $64°C$. Fig. 11 also shows the upper bounds on peak temperature.

To compute temperature bounds, we simulate the server schedule using the temperature transfer function we evaluated in section 8.4 until a steady state is reached (temperature is same across different periods of TI Server). Λ_2^1 and Λ_3^2 are the self-core thermal budgets of S_1 and S_2, respectively. These can be computed directly from the thermal simulation of the corresponding server schedule. Non-self-core thermal budgets are computed as:

$$\Lambda_j^1 = \Lambda_2^1 \cdot A_{j,2}^{-1}/A_{2,2}^{-1} \quad j \in \{1,3\}$$
$$\Lambda_j^2 = \Lambda_3^2 \cdot A_{j,3}^{-1}/A_{3,3}^{-1} \quad j \in \{1,2\}$$

The upper-bound on peak temperature can be computed using $\boldsymbol{T}^{\Lambda} - \boldsymbol{\Lambda}^{\mathbf{total}} + \boldsymbol{\Lambda}^{\mathbf{1}} + \boldsymbol{\Lambda}^{\mathbf{2}} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is an error margin of 1°C which is added to account for the limited precision of the on-chip thermal sensors. As shown in the Fig. 11, temperature of each core remains below the analytically computed bound. This validates our theoretical analysis.

9 CONCLUSION

In this paper, we propose Thermal Isolation Servers (TI Servers) which can be used to upper bound peak temperature increase due to execution of a task-set in a multi-core system. By construction, TI Servers can be used to thermally isolate and protect tasks; a property which is particularly important for mixed-criticality systems. Furthermore, TI Servers are time and space composable, which simplifies their design process. We present theoretical analysis to prove the worst-case temperature bounds and also provide insights into designing TI Servers. Apart from providing thermal protection/isolation, TI Servers can also be used to reduce peak system temperature and consequently improve system schedulability, for a given thermal constraint. The proposed theoretical concepts are also validated on a hardware test-bed where thermal protection is provided to a mixed-critical application.

10 APPENDIX

Theorem 5.1 proof : Consider a TI Server S_i. The thermal component at the end of the first active time slot is given by $(\boldsymbol{I} - \boldsymbol{e}^{\boldsymbol{A}\cdot P_i \cdot U_i}) \cdot \boldsymbol{T}^{\infty}(\boldsymbol{B}^{\mathbf{cr}_i}, \boldsymbol{\psi}^{\boldsymbol{d}})$. At the end of second active time, the thermal component is given by: $(\boldsymbol{I} + \boldsymbol{e}^{\boldsymbol{A}\cdot P_i})\Big((\boldsymbol{I} - \boldsymbol{e}^{\boldsymbol{A}\cdot P_i \cdot U_i}) \cdot \boldsymbol{T}^{\infty}(\boldsymbol{B}^{\mathbf{cr}_i}, \boldsymbol{\psi}^{\boldsymbol{d}})\Big)$. In general the thermal component

at the end of k^{th} active time slot is given by:

$$\Theta(P_i(k+U_i)+\Phi_i, S_i) = \Big(\sum_{1\le z\le k} \boldsymbol{e^{(z-1)\cdot A\cdot P_i}}\Big)\Big((\boldsymbol{I}-\boldsymbol{e^{A\cdot P_i\cdot U_i}})\cdot \boldsymbol{T^{\infty}(B^{\mathrm{cr}_i,\psi^d})}\Big)$$

This is a geometric progression with initial term: $(\boldsymbol{I}-\boldsymbol{e^{A\cdot P_i\cdot U_i}})\cdot \boldsymbol{T^{\infty}(B^{\mathrm{cr}_i,\psi^d})}$ and common ratio $\boldsymbol{e^{A\cdot P_i}}$. Therefore, the thermal component at the end of server active time converges to a steady state value as we progress through the server periods. The value of this steady state thermal component is given by the infinity sum of this geometric progression which is given by (15). □

Theorem 5.2 proof : $P_i \to 0$ refers to a fluid system where the active and idle times of the server are infinitesimally small. This can be interpreted as S_i consuming a fraction of core cr_i's computation capability equal to U_i. The power dissipation of cr_i under a fluid schedule is constant at $U_i\cdot(\psi^{\text{active}})$. With this power dissipation, the steady state thermal component is given by (16). □

Theorem 5.3 proof: We prove the following two cases:

Case 1: $j=\mathrm{cr}_i$**:**. For an RC network which is used to thermally model the processor, the self core impulse response has a maximum at $t=0$ [17]. This implies that temperature increases with zero delay for self core. Therefore, for the self core, the maximum thermal component within any period always occurs at the end of server active time. Therefore, $\Lambda^i_{\mathrm{cr}_i}$ is set to $\Theta(\lim_{k\to\infty}(P_i(k+U_i)+\Phi_i), S_i)_{\mathrm{cr}_i}$ from (15) if $P_i>0$ or $\Theta(\lim_{t\to 0} t, S_i)_{\mathrm{cr}_i}$ from (16) in the fluid case.

Case 2: $j\neq\mathrm{cr}_i$**:** Non self core impulse response has a maximum at $t>0$ [17]. This implies that temperature increases with a delay in this case. Therefore, for non self core, the maximum thermal component occurs after the end of server active time. To compute the upper bound, we assume that the thermal component of self core (cr_i) remains indefinitely at its upper bound $\Lambda^i_{\mathrm{cr}_i}$ computed in the previous case ($j=\mathrm{cr}_i$) of this theorem. With this assumption we compute the steady state system temperature. This steady state temperature can be computed using $j\neq\mathrm{cr}_i$ case of (17). □

Theorem 5.4 proof: Thm. 4.4 of [3] proves that the maximum temperature due to execution of a an implicit deadline periodic task is minimized if it is executed in a *fluid* manner with execution rate equal to its utilization. Conversely, if we are given a fixed self core thermal budget, a fluid server ($P_i\to 0$) would yield the largest feasible utilization. □

Theorem 5.5 proof: This is a necessary and sufficient condition for timing feasibility under EDF scheduling as proved in Theorem 4.1 of [20].

Theorem 5.6 proof: Theorem 4.2 of [20] proves this necessary and sufficient condition for timing feasibility under Rate Monotonic (RM) scheduling. Note that although the stated proof in [20] is for RM scheduling, it is generally applicable for FP scheduling. This is because (22) gives an upper-bound on execution requirement of a given task τ_j. This is computed with the critical instant assumption i.e., τ_j experiences worst-case interference from all higher priority tasks; where priorities are assigned arbitrarily. If the condition in (24) is satisfied, then the critical instants of all tasks are schedulable; which is a necessary and sufficient condition for fixed priority scheduling.

Theorem 5.7 proof: We prove the following two cases:

Case 1 $P_1=P_2$ and $U_1(\epsilon)\geq U_2(\epsilon)$: In this case, it is clear that $\mathrm{sbf}(S_1,l,\epsilon)\geq\mathrm{sbf}(S_2,l,\epsilon)\ \ \forall t$ since all terms in (20) are non decreasing w.r.t. $U_i(\epsilon)$.

Case 2 $P_2=n\cdot P_1$ where $n\in Z^+$ and $U_1(\epsilon)=U_2(\epsilon)=U$: $\forall k\in Z_0^+$, in the interval $l\in[k\cdot P_2,\ (k+1)P_2]$, $\mathrm{sbf}(S_2,k\cdot P_2,\epsilon)$ increases only once by $P_1\cdot n\cdot U$ between $[P_2(k+1-U),\ P_2(k+1)]$ with a gradient of 1. Whereas, $\mathrm{sbf}(S_1,l,\epsilon)$, increases by $P_1\cdot U$, n times with a gradient of 1. Therefore, $\mathrm{sbf}(S_1,l,\epsilon)=\mathrm{sbf}(S_2,l,\epsilon)=k\cdot n\cdot P_1\cdot U$, when $l=k\cdot P_2$. For all other values of l, $\mathrm{sbf}(S_1,l,\epsilon)\geq\mathrm{sbf}(S_2,l,\epsilon)$, since the singular increase of $\mathrm{sbf}(S_2,k\cdot P_2,\epsilon)$ is towards the end of each P_2 interval.

Theorem 5.8 proof: Assuming that all TI Servers have tasks to execute when their state is *active*, the temperature increase caused by the n TI Servers is given by $\sum_{1\le i\le n}\Theta(t,S_i)$. We can compute a

bound on the maximum temperature increase for core j by making the conservative assumption that the maximum thermal component for core j occurs at the same time for all TI Servers. i.e.,

$$\max_t \left\{ \sum_{1 \le i \le n} \Theta(t, S_i)_j \right\} \le \sum_{1 \le i \le n} \max_t \left\{ \Theta(t, S_i)_j \right\}$$

By Thm. 5.3, we know that

$$\max_t \left\{ \Theta(t, S_i)_j \right\} \le \Lambda_j^i \implies \max_t \left\{ \sum_{1 \le i \le n} \Theta(t, S_i)_j \right\} \le \sum_{1 \le i \le n} \Lambda_j^i$$

□

Theorem 5.9 proof : This follows from the Thm.5.8. The condition states that for all cores $j \in M$, the net temperature increase due to execution of all TI Servers must not exceed the maximum allowed temperature increase $(T_j^{\Delta} - [T^{\infty}(B^{\text{idle}})]_j)$. □

REFERENCES

[1] M. Ahmed, N. Fisher, S. Wang, and P. Hettiarachchi. 2011. Minimizing peak temperature in embedded real-time systems via thermal-aware periodic resources. *Sustainable Computing: Informatics and Systems* 1, 3 (2011), 226–240.

[2] R. Ahmed, P. Huang, M. Millen, and L. Thiele. 2017. *On the design and application of thermal isolation servers.* Technical Report 368. ETH Zurich, Laboratory TIK.

[3] R. Ahmed, P. Ramanathan, and K.K. Saluja. 2016. Necessary and Sufficient Conditions for Thermal Schedulability of Periodic Real-Time Tasks Under Fluid Scheduling Model. *ACM-TECS* 15, 3 (2016), 49:1–49:26.

[4] A. Burns and R. Davis. 2013. Mixed criticality systems-a review. *Department of Computer Science, University of York, Tech. Rep* (2013).

[5] T. Chantem, R.P. Dick, and X.S. Hu. 2011. Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In *DATE*. 288–293.

[6] J.J. Chen, S. Wang, and L. Thiele. 2009. Proactive speed scheduling for real-time tasks under thermal constraints. In *RTAS*. IEEE, 141–150.

[7] J. Cui and D.L. Maskell. 2012. A Fast High-Level Event-Driven Thermal Estimator for Dynamic Thermal Aware Scheduling. *IEEE CAD ICS* 31, 6 (2012), 904–917.

[8] P. Emberson, R. Stafford, and R.I Davis. 2010. Techniques for the synthesis of multiprocessor tasksets. In *WATERS*. 6–11.

[9] N. Fisher, J.J. Chen, S.Wang, and L. Thiele. 2009. Thermal-Aware Global Real-Time Scheduling on Multicore Systems. In *RTAS*. 131–140.

[10] Luotao Fu and Schwebel Robert. 2017. Preempt-RT Patch. https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO. (2017), 5 pages. Accessed: 2017-04-05.

[11] G. Giannopoulou, N. Stoimenov, P. Huang, and L. Thiele. 2013. Scheduling of mixed-criticality applications on resource-sharing multicore systems. In *EMSOFT*. 1–15.

[12] P. Kumar and L. Thiele. 2011. Cool Shapers: Shaping Real-Time Tasks for Improved Thermal Guarantees. In *DAC*. 468–473.

[13] Y. Liu, R.P Dick, L. Shang, and H. Yang. 2007. Accurate Temperature-Dependent Integrated Circuit Leakage Power Estimation is Easy. In *DATE*. 1526–1531.

[14] C. Moler and C.V. Loan. 2003. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review* 45, 1 (2003), 3–49.

[15] S. Pagani, J.J Chen, M. Shafique, and J. Henkel. 2015. MatEx: Efficient transient and peak temperature computation for compact thermal models. In *DATE*. 1515–1520.

[16] SAFURE: 2015. Safety And Security By Design For Interconnected Mixed-Critical Cyber-Physical Systems. https://safure.eu/. (2015).

[17] L. Schor, I. Bacivarov, H. Yang, and L. Thiele. 2012. Worst-case temperature guarantees for real-time applications on multi-core systems. In *RTAS*. 87–96.

[18] L. Schor, H. Yang, I. Bacivarov, and L. Thiele. 2011. Thermal-Aware Task Assignment for Real-Time Applications on Multi-Core Systems. 294–313.

[19] L. Sha and et al. 2014. *Single Core Equivalent Virtual Machines for Hard Real-Time Computing on Multicore Processors.* Technical Report. University of Illinois at Urbana-Champaign.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue

[20] I. Shin and I. Lee. 2008. Compositional Real-time Scheduling Framework with Periodic Model. *ACM TECS* 7, 3 (2008), 30:1–30:39.

[21] L. Sigrist, G. Giannopoulou, P. Huang, A. Gomez, and L. Thiele. 2015. Mixed-criticality runtime mechanisms and evaluation on multicores. In *RTAS*. 194–206.

[22] K. Skadron, M.R Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. 2004. Temperature-aware microarchitecture: Modeling and implementation. *ACM TACO* 1, 1 (2004), 94–125.

[23] T. Ungerer and et al. 2010. Merasa: Multicore Execution of Hard Real-Time Applications Supporting Analyzability. *IEEE Micro* 30, 5 (2010), 66–75.

[24] S. Wang and R. Bettati. 2006. Delay analysis in temperature-constrained hard real-time systems with general task arrivals. In *RTSS*. IEEE, 323–334.

[25] Y. Xie and W. Hung. 2006. Temperature-Aware Task Allocation and Scheduling for Embedded Multiprocessor Systems-on-Chip (MPSoC) Design. *JVLSI* 45, 3 (2006), 177–189.

This article was presented in the International Conference on Embedded Software (EMSOFT) 2017 and appears as part of the ESWEEK-TECS special issue