IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 25, 2022, accepted March 15, 2022, date of publication March 28, 2022, date of current version April 1, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3162617

Task Migration Policy for Thermal-Aware Dynamic Performance Optimization in Many-Core Systems

BEHNAZ POURMOHSENI[1], STEFAN WILDERMANN[1], FEDOR SMIRNOV[2], PAUL E. MEYER[1], AND JÜRGEN TEICH[1], (Fellow, IEEE)

[1]Chair for Hardware/Software Co-Design, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91058 Erlangen, Germany
[2]Distributed and Parallel Systems Group, Institute of Computer Science, University of Innsbruck, 6020 Innsbruck, Austria

Corresponding authors: Behnaz Pourmohseni (behnaz.pourmohseni@fau.de) and Stefan Wildermann (stefan.wildermann@fau.de)

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project TRR89 Invasive Computing (project No. 146371743).

ABSTRACT The steady downsizing of semiconductor technology nodes in recent years has led to a rapid increase in the density of power consumption on chips which, in turn, renders *temperature* a major issue for many-core systems, adversely affecting their performance, reliability, leakage, cost, etc. In this context, *task migration* is a powerful technique that is widely used for controlling the temperature profile of many-core systems under dynamic workloads with the goal to improve their performance, utilization, reliability, etc. In this paper, we present a task migration policy for thermal-aware performance optimization in heterogeneous many-core systems. The proposed policy is developed based on an analytical and thermally safe power-budgeting scheme and uses Dynamic Voltage and Frequency Scaling (DVFS) for power and thermal management of the system. Our migration policy aims at maximizing the system's performance and, at the same time, proactively enforcing thermal safety using DVFS. To that end, it iteratively adapts the distribution of active cores in the system (through proper migration decisions) to maximize the thermally safe power budget of active cores and, thereby, enable them to operate on higher frequencies without violating their safe thermal threshold. Experimental results demonstrate that the proposed policy offers 2× higher performance gain in comparison to existing approaches which aim at greedily reducing the average, variance, or gradient of temperature as an indirect means to enhance performance.

INDEX TERMS Dynamic thermal management, heterogeneous, many-core systems, performance optimization, task migration.

I. INTRODUCTION

Over the past decades, the continuous increase in the compute-power requirements of embedded applications and the end of Dennard scaling [1] have led to the introduction of embedded multi/many-core systems. By integrating a large number of processing resources (henceforth called cores) on a single chip, many-core systems provide massive and scalable compute power which enables the concurrent execution of a large number of applications. To achieve compute-power scalability, many-core systems typically follow a tile-based organization of resources, where the cores and their necessary peripherals, e.g., caches, are clustered into a set of tiles which are interconnected via a Network-on-Chip (NoC), see e.g. [2]–[4]. In particular, *heterogeneous* many-core systems are becoming the de facto standard architectures for embedded systems; The variety of different types of cores available in a heterogeneous system enables catering to diverse processing demands of different processes and applications and, thereby, establish an efficient execution with desirable performance-energy trade-offs [5].

A large share of embedded systems manifest a *dynamic* execution environment where (i) the mix of concurrently executed applications may change unforeseeably, e.g., following user requests, and/or (ii) the applications exhibit dynamic workload and execution characteristics. To achieve high resource utilization, such systems rely on dynamic resource management schemes to dynamically allocate the

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Tang.

compute and communication resources necessary for executing each application at its launch time and, later, release these resources upon a termination request [6], [7]. To that end, these systems feature a so-called *Run-time Manager (RM)* which continuously monitors the system's state w.r.t. the deployed applications and availability of resources. Upon a launch request, the RM deploys the incoming application on the currently available resources using lightweight application-mapping heuristics whose compute overhead is low enough for online use. It may also adapt the deployment of running applications during their execution for, e.g., load balancing, improving energy efficiency, or evacuating failed resources.

While yielding massive processing power, many-core systems typically exhibit increased on-chip *temperature* due to their dense integration of resources and the resulting high *density of power consumption*[1] [8], [9]. In fact, due to the ongoing technology downsizing, many-core systems are strongly susceptible to hotspots which can jeopardize the thermal integrity of the system, accelerate aging, lead to transient faults, and even cause permanent system failure [9], [10]. This renders thermal concerns particularly crucial aspects of consideration during the design of many-core systems as well as the dynamic management of their resources at run time.

Dynamic systems often exhibit a dynamic temperature profile following the changes in the activity of resources. The temperature of each resource depends on its density of power consumption (as its local source of heat) and the temperature of its nearby regions (due to on-chip heat transfer). To keep the system in a thermally safe state, many-core systems employ *Dynamic Thermal Management (DTM)* mechanisms which continuously monitor the thermal state of the system and autonomously (typically at hardware level) counteract hotspots to maintain the peak temperature of the system below a critical threshold, see [11] for an overview. For this purpose, they either stall the overheated resources by means of clock/power gating and/or throttle them using DVFS. Evidently, while preserving thermal safety, DTM countermeasures incur performance degradation and may cause loss of processing progress. Moreover, depending on the system's thermal profile, its ambient temperature, and its packaging/cooling subsystem, the overheated regions may require some time to cool down before becoming operational again, resulting in an extended utilization penalty. To alleviate the performance and/or utilization penalty of DTM—and, thereby, maximize the system's performance [12] or utilization [13] under thermal constraints—thermal considerations are increasingly incorporated into many-core resource-management and application-mapping strategies.

In view of the dynamic workload and, hence, power and temperature profile of many-core systems, *task migration* and *DVFS* serve as the key mechanisms used for thermal-aware dynamic resource management. Using task migration, the RM is enabled to dynamically modify the deployment of running applications in the system and, thereby, adjust the chip's thermal profile. DVFS enables the RM to calibrate the power consumption of each active core for fine-grained thermal control. As a result, many thermal-aware resource management strategies employ temperature monitoring together with task migration and DVFS to reduce [14] or even prevent [15] the occurrence of thermal violations and, thereby, pursue the ultimate goal of optimizing the system's performance (and/or its utilization) by preventing situations that lead to the activation of DTM countermeasures.

When making resource-management decisions (e.g., deciding between multiple task migration options), it is typically computationally too expensive or even infeasible to adequately assess the performance impact of an option. Therefore, despite performance gain being their primary goal, thermal-aware resource-management strategies chiefly pursue an *indirect objective* which (i) can be attained using lightweight evaluations and decision-making heuristics while (ii) correlating well with performance so that its optimization entails performance gain. In this context, *thermal balance* is often viewed as a promising indirect objective to maximize performance under thermal constraints, since enhancing thermal balance enables operating the active cores on a higher frequency which, in turn, often translates into performance gain. Moreover, a migration policy for optimizing thermal balance can be established practically, since the thermal profile of the system can be derived through temperature measurement (if temperature sensors are available) or using lightweight temperature prediction methods, and migration decisions can be made on that basis using lightweight heuristics. As a result, a large share of thermal-aware resource-management strategies employ task migration to relocate tasks from hot to cold (often idle) cores (possibly coupled with DVFS to control power consumption) to maximize thermal balance by, e.g., minimizing the peak temperature of the chip, its temperature gradient, temperature variance, etc.

A. MOTIVATION

The main hypothesis of this work (which is supported by experimental evidence in Sections IV-A and V) is that thermal balance as an indirect objective towards performance optimization serves well only for *homogeneous* target platforms; In a homogeneous system, all cores have identical power-, thermal-, and performance inter-relations such that different cores that operate at the same temperature (by running at the same or comparable V/F level) also yield comparable performance. In such a setup, maximizing thermal balance enables increasing the average operating frequency of active cores (without thermal violation) and, thereby, optimize performance. This uniformity, however, does not hold for *heterogeneous* systems where core instances of different type exhibit different inter-relations between their power-, thermal-, and performance characteristics and, hence,

[1]The *density* of power consumption refers to the amount of power consumed per unit of die area which is proportional to the amount of heat generated as a consequence of this power consumption.

two cores of different types operating at the same temperature may exhibit substantially different performance profiles and/or even run at substantially different frequencies while sustaining their temperature. In fact, as will be shown later in Sections IV-A and V, migration options that yield maximal performance gain in a heterogeneous system often even lead to increased thermal imbalance in the system.

Besides their choice of indirect objective, the performance benefit of thermal-aware migration strategies depends also on their ability to prevent thermal violations. While existing migration strategies are able to reduce the frequency of thermal violations, they often cannot prevent violations completely, meaning that the system under their control may still undergo some violations and suffer the performance penalty of DTM counteractions, e.g., clock/power gating or throttling of overheated regions. On the other hand, migration strategies that are capable of preventing thermal violations chiefly rely on conservative temperature constraints which often result in under-utilization of the safe thermal headroom of the chip, e.g., by operating some cores at a V/F level that is indeed lower that the level they can maintain in a thermally safe way. Thus, they often cannot maximally exploit the performance potential of the system in the absence of violations.

B. CONTRIBUTION

As a remedy to the above concerns, this paper (i) promotes a new indirect objective for migration-based performance optimization which is suitable also for *heterogeneous* many-core systems and can ensure *thermal safety*. We then (ii) present a migration policy which optimizes this new objective using a lightweight heuristic, making it suitable for online use.

As for the new indirect objective, we promote the so-called *Thermally Safe Power Density (TSPD)* introduced in [16]. TSPD is a system-wide (uniform) power-density constraint that enforces thermal safety and suits both homogeneous and heterogeneous systems. Our proposition is based on two observations: Firstly, we experimentally demonstrate that, compared to thermal-balance optimization, TSPD optimization exhibits stronger correlation with performance gain in a heterogeneous system, rendering TSPD a superior indirect objective for thermal-aware migration policies targeting heterogeneous systems. Secondly, the only dynamic factor that affects the TSPD constraint of a system is the *distribution of active and idle cores*. Thus, proper migration decisions can enable maximizing TSPD. The increase in TSPD can, in turn, be exploited by DVFS so as to operate each active core at a higher V/F level without exceeding the TSPD constraint and, thereby, improve performance in a thermally safe manner.

Establishing an *optimal* migration policy that truly maximizes TSPD is not infeasible, since enumerating all migration options is computationally too expensive. As a remedy, we propose a migration policy which optimizes TSPD based on a heuristic approximation of the optimal approach which is lightweight and, hence, practical for online use. As a result, by maximizing the system's TSPD constraint and exploiting the resulting power-budget gain through DVFS, the proposed policy optimizes the system's performance while preventing thermal violations and their consequent DTM-induced performance penalty by construction. The experimental results in Section V demonstrate that, by optimizing TSPD, the proposed policy outperforms existing migration policies that aim at maximizing the thermal balance in the system as an indirect goal to maximize performance under thermal constraints.

C. ORGANIZATION

The remainder of this paper is organized as follows: Section II provides an overview of related work on thermal-aware resource management and task migration in many-core systems. Section III presents the system model considered in this work. In Section IV, the proposed thermal-aware task migration policy is presented. Experimental results are provided in Section V, before the paper is concluded in Section VI.

II. RELATED WORK

In dynamic many-core systems, whenever an application must be launched, *application-mapping* techniques are used to find a (near-)optimal mapping/schedule of the incoming application's tasks onto the (free) resources of the many-core platform with the goal to achieve maximized performance (and/or utilization, energy efficiency, reliability, etc.) while taking the chip's thermal constraints into consideration to avoid thermal violations. However, due to the NP-hard nature of the application-mapping problem [17], exact optimization approaches capable of finding the optimal mapping are computationally infeasible, limiting the practical choices of mapping approaches to lightweight heuristics that yield sub-optimal yet acceptable mappings. In such a setting, after an application is launched or terminated, revising the distribution of running applications in the system typically offers a significant potential for performance optimization which can be exploited by means of *task migration*. Therefore, thermal-aware *application-mapping* techniques and thermal-aware *task-migration* techniques have become the main cornerstones for thermal-aware dynamic resource management in multi/many-core systems. A large body of work exists on thermal-aware multi/many-core application mapping, e.g. [18]–[24]. However, as this paper focuses on the *task-migration* aspect of resource management, our review of related work in the following is focused on the state of the art in thermal-aware task migration in multi/many-core systems. An overview and taxonomy of multi/many-core application-mapping techniques is provided in [5].

Thermal-aware task migration policies[2] often exploit task relocation in conjunction with DVFS for thermal management. Here, relocating tasks between different cores enables coarse-grained adaptation of the system's power and thermal state while DVFS enables fine-grained and localized

[2]Note the distinction between migration *policy* and migration *mechanism*. Migration policies—which are the focus of this paper—make migration decisions, i.e., determine which migration to perform at which point in time. Migration mechanisms—which are outside the scope of this work—specify how a given migration (decided by the policy in use) is performed.

adjustments therein. A large share of existing migration policies, e.g., [14], [25]–[27], and the policy proposed in this paper exploit task relocation and DVFS in tandem.

Existing thermal-aware migration policies mostly aim at enhancing the thermal balance in the system as an indirect objective to enable improving performance under thermal constraints, e.g. [14], [26], [28]–[32]. They chiefly introduce a scheme to classify cores (based on their temperature) and/or classify tasks (based on their compute intensity) into temperature categories, e.g., hot, warm, or cold. The resulting classification is then used in a migration policy that either migrates the workload of hot cores to cold ones (core-level granularity), e.g. [26], [32]–[35], or migrates hot (cold) tasks to cold (hot) cores (task-level granularity), e.g. [29], [36].

Temperature-driven migration policies are either reactive or proactive. *Reactive policies*, e.g., [26], [27], [37], operate based on temperature *monitoring* and trigger migrations based on the current system temperature. Upon a thermal violation, they migrate the affected tasks from the hotspot(s) to cold regions for an immediate execution resumption. *Proactive policies*, e.g., [14], [15], [25], [31], [38], [39], operate based on *predicting* the future thermal profile of the system by projecting the temperature history thus far, e.g., using machine learning techniques [14], [40]. This enables them to proactively identify emerging hotspots and attempt to dissolve them before they result in thermal violations. The effectiveness of such policies is determined by the accuracy of their thermal predictions. Some works use error correction [14] and/or online model tuning [38] to enhance their prediction accuracy. However, as such predictions fundamentally rely on the thermal history of the chip, they generally cannot deliver a high accuracy in dynamic systems where the future thermal profile of the system correlates only weakly with its thermal history [29], [39].

To reduce the frequency of thermal violations or possibly prevent them, most migration policies trigger migrations based on a conservative temperature threshold (lower than the critical threshold at which DTM countermeasures are triggered) [14], [15], [26]. While alleviating or excluding one source of performance penalty, this may introduce another source of performance loss due to unnecessary task migrations and, in case of DVFS, operating resources at a lower V/F setting than what they can maintain without causing thermal violations. To address this issue, analytical thermal models can be used as in [16], [41] to derive accurate thermal predictions based on the power and thermal characteristics of the resources and the floorplan of the chip. While preventing thermal violations is beneficial in general, it is indeed crucial for real-time systems which strongly rely on timing predictability and, hence, may fail upon DTM activation [21].

The chip's floorplan and the architectural properties of its resources strongly affect the thermal characteristics of the system. A large share of existing thermal-aware migration policies, e.g., [15], [27]–[30], [41], are applicable to homogeneous architectures only as they do not capture the power, thermal, and performance dissimilarities between the different types of cores in a heterogeneous system. Moreover, some policies rely on specific assumptions about the chip's floorplan which restricts their general applicability. For instance, the policy in [41] employs a compute-intensive analysis to obtain optimal performance under thermal constraints. To simplify the analysis, however, it assumes a certain placement of cores and caches on the chip. Many of the policies that consider resource heterogeneity are restricted to specific target platforms. For instance, the policies in [14], [26], [42] are tailored to ARM's big.LITTLE architecture [43]. Finally, in spite of their effectiveness, some approaches, e.g., [12], [41], entail a relatively high compute overhead which prohibits their online use.

In summary, existing thermal-aware task migration policies share at least one of these properties: (i) They aim at maximizing the system's performance by pursuing indirect objectives which often only weakly correlate with performance optimization. (ii) They rely on temperature monitoring/prediction and, hence, either cannot prevent thermal violations, or use conservative temperature thresholds that may cause performance loss. (iii) They employ compute-intensive analyses and optimization processes which render them impractical for use at run time. (iv) They are applicable only to homogeneous systems or a specific choice of heterogeneous platforms. The thermal-aware migration policy proposed in this work addresses these concerns as it (i) uses an analytical thermal model of the system for an accurate prediction of its future thermal profile without requiring thermal sensors or online model calibration, (ii) is lightweight and, hence, suited for online use, (iii) aims at maximizing the power budget of cores, which in combination with DVFS, directly translates into performance gain, and (iv) is applicable to heterogeneous many-core systems without relying on restrictive assumptions about their architecture or floorplan.

III. SYSTEM MODEL

The task migration policy presented in this paper is applicable to any multi/many-core platform for which an RC thermal network is available, cf. [44]. In general, the target platform consists of a set of processing cores and a set of peripheral resources referred to as *uncores*. The latter includes interconnects, e.g., NoC and buses, and the memory subsystem, e.g., L2/L3 caches, main memory, and memory controllers. Processing cores can be homogeneous or heterogeneous. DVFS support is taken to be available for cores so that their V/F setting can be changed dynamically at run time. Note that the proposed approach is agnostic to the granularity of DVFS and, thus, is applicable to systems with core-, tile-, or multi-tile-level DVFS support. The following presents the power and thermal models considered in this work.

A. POWER MODEL

The power consumed by an active core is derived using Eq. (1) where the first summand calculates the dynamic power consumption p_{dy} of the core based on its switching

activity $\sigma \in [0, 1]$, effective capacitance C_{eff}, supply voltage V, and operating frequency f. The second summand in Eq. (1) derives the static power consumption p_{st} of the core where I denotes its leakage currents.

$$p = \sigma\, C_{\text{eff}}\, V^2 f + V\, I \tag{1}$$

Given the V/F setting (V, f) applied to a core, a safe upper-bound estimate of its power consumption is derived by (i) considering $\sigma = 1$ which yields the peak dynamic power[3] while (ii) deriving the leakage current I at the critical temperature T_{DTM} of the chip to obtain the peak static power, cf. [45], [46]. The critical temperature T_{DTM} denotes the temperature threshold at which DTM countermeasures are triggered to enforce thermal safety. The *density of power consumption* for a core is calculated as $\rho = p/\alpha$ where p denotes the power consumption derived by Eq. (1), and α denotes the area of that core. Contrarily to cores, each uncore is taken to operate at a fixed V/F setting and, hence, has a fixed power consumption.

B. THERMAL MODEL

RC thermal networks are widely used for thermal modeling of multi/many-core platforms [16], [44]. Given a platform with C cores and U uncores, the equivalent RC thermal network comprises $N \geq C + U$ thermal nodes, each associated with a thermal capacitance (capturing transient thermal effects) and connected to the other nodes via thermal conductances (modeling heat transfer). Sources of power consumption in the platform are modeled by current sources in the network, and the current flow in the network represents the heat flow in the platform. The voltage at each node in the network denotes the temperature of its equivalent node in the platform. The ambient temperature is reflected in the network by a thermal node with a constant temperature T_∞.

In this context, the steady-state temperature of thermal nodes can be derived using the system of equations in Eq. (2), cf. [16]. Here, vectors $\mathbf{T}_{N\times 1}$, $\mathbf{P}_{N\times 1}$, and $\mathbf{G}_{N\times 1}$ reflect the temperature, power consumption, and thermal conductance with the ambiance for the thermal nodes, respectively, and matrix $\mathbf{B}_{N\times N}$ comprises pairwise thermal conductances among the nodes in the network. Note that the entries of $\mathbf{B}$ and $\mathbf{G}$ are constant, platform-specific, and derived statically by measurement [47] or simulation [44], see also [16]. Thus, $T_\infty \mathbf{B}^{-1}\mathbf{G}$ is constant at a given ambient temperature T_∞.

$$\mathbf{T} = \mathbf{B}^{-1}\mathbf{P} + T_\infty \mathbf{B}^{-1}\mathbf{G} \tag{2}$$

The vector $\mathbf{P}$ of power consumption in Eq. (2) can be decomposed into constant and varying vectors as $\mathbf{P} = \mathbf{P}^{\text{core}} + \mathbf{P}^{\text{uncore}} + \mathbf{P}^{\text{other}}$. Here, $\mathbf{P}^{\text{core}}_{N\times 1}$ retains the power values for cores, $\mathbf{P}^{\text{uncore}}_{N\times 1}$ for uncores, and $\mathbf{P}^{\text{other}}_{N\times 1}$ for remaining nodes where $\mathbf{P}^{\text{other}} = \mathbf{0}$. Hence, Eq. (2) can be reformulated into Eq. (3) where the second summand given in parentheses is constant. Thus, the steady-state temperature of a node i can be derived using Eq. (4) where $b^{-1}_{i,j}$ and p_j denote entries of $\mathbf{B}^{-1}$ and $\mathbf{P}^{\text{core}}$, respectively, and $\hat{T}_i = \sum_{j=1}^{N} b^{-1}_{i,j}(p^{\text{uncore}}_j + T_\infty \cdot g_j)$ represents the constant temperature contribution to node i from uncores and the ambiance, corresponding to the second summand in the system of equations in Eq. (3).

$$\mathbf{T} = \mathbf{B}^{-1}\mathbf{P}^{\text{core}} + \left(\mathbf{B}^{-1}\mathbf{P}^{\text{uncore}} + T_\infty \mathbf{B}^{-1}\mathbf{G}\right) \tag{3}$$

$$T_i = \sum\nolimits_{j=1}^{N} b^{-1}_{i,j} \cdot p_j + \hat{T}_i \tag{4}$$

C. THERMAL-AWARE POWER BUDGETING

The migration policy proposed in this paper is based on the analytical power-budgeting concept presented in [16]. In the following, we present a brief summary of the power-budgeting concept and analysis in [16] which is used as the basis for the migration policy proposed in Section IV. This approach uses power budgeting as an abstraction of the thermal problem, empowered by the observation that the power consumption of cores is the sole varying factor contributing to the temperature of each node in the system, see Eq. (4). Given the thermal model of the system as presented in Section III-B and the set of cores that are currently active in the system, the approach in [16] derives a uniform power-density budget (constraint) for the active cores such that thermal safety is preserved, i.e., $T_i \leq T_{\text{DTM}}$ for each core/uncore i, as long as this budget is not exceeded by any of the given active cores. Note that considering a *power-density* budget (rather than a power budget) enables an efficient uniform power budgeting for *heterogeneous* platforms, taking into account the area- and power-consumption disparity between different core types and its impact on their thermal characteristics, cf. [16].

To reflect the activity of cores in the thermal model, a binary allocation vector $\mathbf{A}_{C\times 1}$ is introduced in [16] which indicates whether the core associated with index $i = 1, \ldots, C$ is active ($a_i = 1$) or idle ($a_i = 0$). Accordingly, Eq. (4) is reformulated into Eq. (5) where the first summand from Eq. (4) is decomposed into two parts capturing the impact of active and idle cores on the temperature of each node $i = 1, \ldots, N$, respectively.[4] For active cores (i.e., when $a_j = 1$), power consumption is expressed as the product of the power density ρ_j and the area α_j of the core, see the first summand in Eq. (5). For idle cores (i.e., when $a_j = 0$), the power consumption in the sleep/off state of the core is considered, denoted by p^{off}_j in Eq. (5).

$$T_i = \sum_{j=1}^{C} a_j \cdot b^{-1}_{i,\kappa_j} \cdot \rho_j \cdot \alpha_j + \sum_{j=1}^{C} (1 - a_j) \cdot b^{-1}_{i,\kappa_j} \cdot p^{\text{off}}_j + \hat{T}_i \tag{5}$$

Based on this model, a uniform, thermally safe power-density constraint ρ^* (upper bound) is derived in [16] for active cores by considering $\rho_j = \rho^*$ for all $j = 1, \ldots, C$

[3] Equivalent to the dynamic power under peak workload.

[4] For notation clarity, κ_j is used as a one-to-one translation of each core index j from the core index space ($j = 1, \ldots, C$) to the larger space of thermal indices ($1, \ldots, N$), cf. [16].

(which becomes effective only when $a_j = 1$) and, subsequently, identifying the maximum value of ρ^* such that $T_i \leq T_{\text{DTM}}$ for each core/uncore i. Equation (6) is, thus, used to derive this constraint.[5] Finally, the core power-density constraint ρ^* is examined and possibly refined to ensure that the overall power consumed by the chip does not exceed a predefined peak power constraint. In the rest of this paper, we refer to the constraint ρ^* as the *Thermally Safe Power Density (TSPD)* budget in the system. Note that the above description of the TSPD analysis is distilled to the parts necessary for the presentation of the proposed task migration policy. Please refer to [16] for a detailed description of the analysis.

$$\rho^*(\mathbf{A}) = \min_{i \in \mathbf{L}} \left\{ \frac{T_{\text{DTM}} - \hat{T}_i - \sum_{j=1}^{C} (1 - a_j) \cdot b_{i,\kappa_j}^{-1} \cdot p_j^{\text{off}}}{\sum_{j=1}^{C} a_j \cdot b_{i,\kappa_j}^{-1} \cdot \alpha_j} \right\} \tag{6}$$

IV. THERMAL-AWARE TASK MIGRATION

This section presents the proposed thermal-aware task migration policy which aims at maximizing the system's performance while guaranteeing the prevention of thermal emergencies. In this work, migration decisions are conducted in the form of a relocation of task(s) between an active core and an idle core (i.e., core-level migration). Moreover, to ensure binary and process-state compatibility, migrations are performed between source and destination cores of the same type. The proposed migration policy is developed based on the power-budgeting approach from [16] summarized in Section III-C. Given the allocation vector $\mathbf{A}_{C\times 1}$ which reflects the distribution of active and idle cores in the system, the proposed migration policy selects an active core index s ($1 \leq s \leq C \wedge a_s = 1$) and an idle core index d ($1 \leq d \leq C \wedge a_d = 0$) of the same core type, where s and d serve as the source and the destination cores of migration, respectively. We select s and d in such a way that a maximal increase in the TSPD budget of active cores is achieved by migrating the task(s) that are running on s to d such that s becomes idle and d becomes active after the migration. Subsequently, to exploit this increased budget, the V/F setting of each active core is adapted using DVFS to the maximum V/F level that respects the TSPD budget of the system under the post-migration distribution of active cores. This allows each core to deliver the maximal performance that it can maintain without creating a thermal violation.

A. INTUITION AND OBSERVATION

Intuitively, maximizing performance under thermal considerations lies in line with maximizing the thermal balance and, hence, minimizing the temperature gradient in the system. This can be achieved by migrating tasks from the hottest (active) core to the coldest (idle) core. Based on this or similar intuitions, several thermal-aware task migration policies have been introduced to date—referred to as *HotCold* approaches in the following—which migrate tasks between the hottest core and the coldest core, identified using measurement- and/or prediction-based temperature monitoring schemes.

While HotCold approaches perform well in reducing the temperature gradient (improving thermal balance) in the system and prove effective for performance optimization in homogeneous systems, we observe that *their migration choices often do not lead to a noticeable performance gain when applied for heterogeneous systems*. In the following, we showcase this in a motivational experiment for a heterogeneous many-core system with 52 cores of three different types.[6] For a large set of system utilization scenarios—each corresponding to a unique set of active cores in the system—we have performed thermal-aware migrations in the form of relocating the complete set of tasks running on one (active) core to an (idle) core of the same type. The following migration policies are considered for this purpose:

- **HotCold**: This policy migrates the task(s) running on the hottest active core to the coldest idle core. Then, it derives the TSPD budget for the new set of active cores as described in Section III-C and, subsequently, applies DVFS to adjust the V/F setting of active cores to the maximum level that respects the derived TSPD budget.
- **PerfOracle**: This policy embodies a performance oracle which performs exhaustive search to find the pair of active and idle cores for which the migration of task(s) from the former to the latter yields maximal performance gain. Hence, it performs a brute-force search to find the migration option that results in maximal performance gain. The gain here refers to the performance gain obtained by performing the migration, deriving the TSPD budget, and applying DVFS as described above.
- **PdOracle**: This policy is an oracle approach that performs exhaustive search to find and perform the migration option that yields maximal increase in the TSPD budget in the system. The obtained gain in the TSPD budget is exploited subsequently by applying DVFS as described above.

In the following experiment, we investigated the impact of the migrations performed by each of these policies on the performance of the system, its TSPD budget, and the standard deviation of temperature across the platform.[7] Throughput (in terms of the total number of application executions per second) is used to represent the system's performance where each core executes an instance of a multi-task streaming application indefinitely. Figure 1 provides the results for different levels of system utilization (% active cores). The results represent the average value among 1,000 considered

[5] For notation clarity, index vector $\mathbf{L}$ is used which comprises the indices of all core/uncore blocks in the larger index space of thermal nodes $(1, \ldots, N)$, cf. [16].

[6] This many-core platform is also used for our experimental evaluations in Section V. The floorplan and the thermal-, power-, and performance characteristics of the platform are provided in Section V-A.

[7] See Section V-B for details of the experiment setup and the performance evaluation scheme.

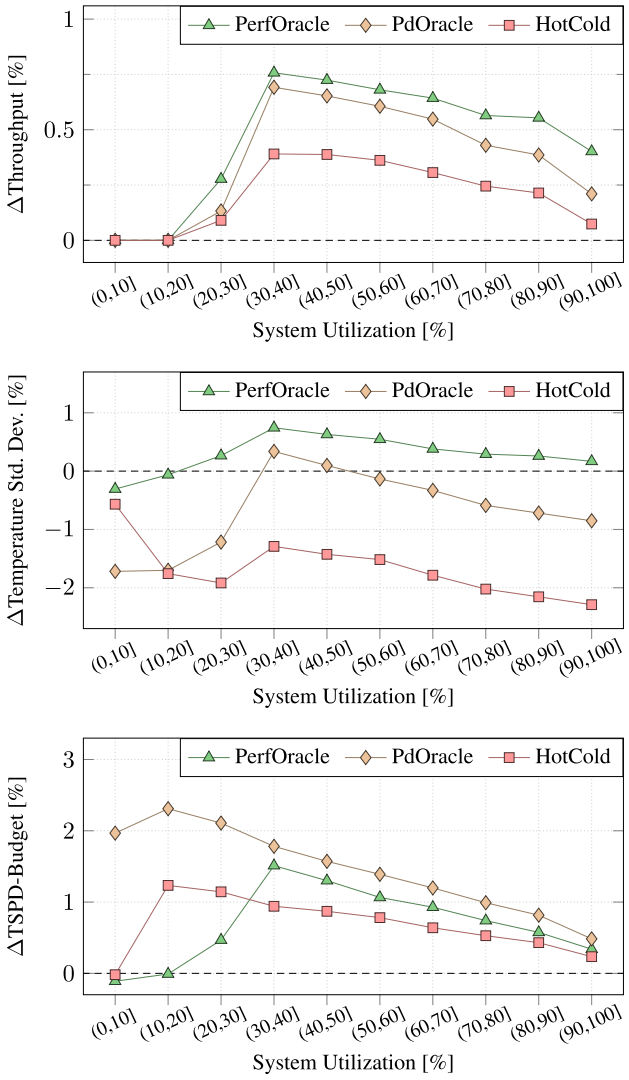

FIGURE 1. The impact of the three investigated task migration policies, namely, PerfOracle, PdOracle, and HotCold, on the throughput (top), standard deviation of temperature (middle), and TSPD budget (bottom) in the system. Each plot reflects the change observed in the respective property when performing a single migration. At each level of system utilization (% active cores), the reported results represent the average value among all investigated utilization samples corresponding to that level.

utilization samples for each utilization level. Note that different core types typically exhibit different throughput contributions due to their dissimilarities w.r.t. execution characteristics and maximum thermally safe V/F setting.

As shown, at low system utilization levels (specifically, up to 28%), the safe thermal headroom of the active cores is high enough to allow every core to operate on its maximum V/F level, so that no performance gain can be attained through migration (see Fig. 1-top). The benefits of task migration, thus, begin to appear at ≈30% utilization. As the system utilization grows, the impact of a migration on the performance or TSPD budget subsides. Evidently, the PerfOracle policy offers the highest increase in throughput (see Fig. 1-top) as it performs optimally w.r.t. performance gain. However, as shown in Fig. 1-middle, its migration decisions often even increase the thermal imbalance in the system. Contrarily, the HotCold policy performs best in enhancing thermal balance (see Fig. 1-middle) while offering the least gain in performance (see Fig. 1-top). In fact, we have observed many cases where HotCold even degrades performance. Such cases are not visible in Fig. 1 as it provides average results, but will be shown later in Section V. Both PerfOracle and HotCold policies improve the TSPD budget of active cores (see Fig. 1-bottom) for which PdOracle delivers optimal results. In terms of performance gain, PdOracle performs fairly comparable to PerfOracle while offering a middle point for thermal balance improvement compared to the other two policies.

A key insight attained by the above experiment is that *maximizing the gain in the TSPD budget correlates with optimizing the system's performance, whereas maximizing thermal balance does not necessarily offer a correlation therewith.* When performing several migrations consecutively to improve the system's performance even further, another important insight is attained (which is experimentally showcased in Section V): Although the PerfOracle policy delivers maximal performance gain in a *single* migration, due to its disregard for TSPD budget, it often drives the system into *local optima* w.r.t. performance after only a few migrations such that a subsequent migration that enhances the performance further cannot be found. In contrast, the PdOracle approach is able to maintain its TSPD gain for a considerably higher number of migrations which eventually result in a higher overall gain in the system's performance compared to what PerfOracle can achieve.

The above correlation observed between the optimization of performance and optimizing the TSPD budget when performing a single migration, on the one hand, and the superiority of the PdOracle policy in enhancing performance through a series of consecutive migrations, on the other hand, promotes the maximization of TSPD budget as a superior indirect objective for performance maximization under thermal constraints. Evidently, the PdOracle policy operates based on a brute-force search that is computationally too intensive to be performed online.[8] To address this issue, we propose a *lightweight* migration policy which exploits the intermediate results of the TSPD analysis to identify *near-optimal* migration options w.r.t. TSPD gain. This policy is presented in the following.

B. PROPOSED TASK MIGRATION POLICY

The proposed task migration policy conducts migration decisions with the objective to maximize the TSPD budget in the system. As TSPD is a function of the distribution of active cores in the system, we focus on *core-level* migrations

[8]The PdOracle policy exhibits a time complexity of $O(C^4 + C^3U)$ which, for a many-core system, is often too high for online use. This complexity results from the exhaustive $O(C^2)$ migration checks performed per migration decision, where each check requires a TSPD analysis with a time complexity of $O(CN)$ where $N \geq C + U$, cf.Section III-C.

where the complete set of tasks running on an active core (migration source) are relocated to an idle core (migration destination) so that the distribution of active cores is modified without exposing tasks of different applications to each other. To achieve a minimal run-time overhead, our policy exploits the intermediate results of the TSPD analysis to identify a proper choice for migration source and destination cores.

As detailed in Section III-C, the TSPD budget in the system is determined as the maximum power-density level that can be maintained by all active cores in the system without causing a thermal violation. This constraint is derived based on the current choice of active cores in the system, reflected by the previously introduced allocation vector $\mathbf{A}_{C\times 1}$ where $a_j = 1$ ($a_j = 0$) denotes that core j is active (idle). Given $\mathbf{A}$, for each resource $i \in \mathbf{L}$ in the system, the analysis derives the peak uniform TSPD constraint $\rho_i^*(\mathbf{A})$ as shown in Eq. (7), by assuming the situation where i reaches but not exceeds the DTM's threshold temperature (i.e., $T_i = T_{\text{DTM}}$). Then, the TSPD budget for the system is determined as the lowest value among all derived $\rho_i^*(\mathbf{A})$ constraints for different resources, i.e., $\rho^*(\mathbf{A}) = \min\{\rho_i^*(\mathbf{A}) \mid i \in \mathbf{L}\}$, cf. Eq. (6).

$$\rho_i^*(\mathbf{A}) = \frac{T_{\text{DTM}} - \hat{T}_i - \sum_{j=1}^{C}(1-a_j)\cdot b_{i,\kappa_j}^{-1}\cdot p_j^{\text{off}}}{\sum_{j=1}^{C} a_j \cdot b_{i,\kappa_j}^{-1}\cdot \alpha_j} \tag{7}$$

In our observations, the resource $s \in \mathbf{L}$ that restricts the TSPD budget is always an active core. Thus, a maximal increase in the system's TSPD budget can often be achieved by switching core s off. Complementary to that, when switching an idle core on, a minimal impact on the TSPD budget is often observed for the idle core d that exhibits the highest $\rho_d^*(\mathbf{A})$. Based on these observations, we establish that migrating tasks from s to d and, thereafter, switching s off and d on, can yield a maximal gain in the TSPD budget. Since a migration between s and d may not be possible, e.g., due to resource-type mismatch, we develop a heuristic migration policy that generalizes the observation above by using *sorted lists* of source/destination candidate cores $i \in \mathbf{L}$ w.r.t. $\rho_i^*(\mathbf{A})$. This extends the decision space of the proposed policy to all possible migrations between active and idle cores. Based on the core ranks in the sorted lists, the migration options will be considered in the approximate order of their impact on the TSPD budget as detailed below.

The decision procedure of the proposed task migration policy is elucidated in Algorithm 1. The policy takes as input (i) the allocation vector $\mathbf{A}_{C\times 1} = [a_i]$, reflecting whether each core i is active ($a_i = 1$) or idle ($a_i = 0$), and (ii) vector $\mathbf{R}_{C\times 1} = [\rho_i^*]$ which provides the power-density constraint $\rho_i^* \equiv \rho_i^*(\mathbf{A}) \in \mathbb{R}^+$ corresponding to each core i under allocation $\mathbf{A}$, derived using Eq. (7). The entries of $\mathbf{R}$ are intermediate results derived during the TSPD analysis of the system where the TSPD budget corresponds to $\min\{\mathbf{R}\}$. Based on these inputs, the policy (i) selects and performs TSPD-budget-maximizing task migrations, (ii) applies DVFS to maximize the V/F level of active cores under the updated TSPD budget, and (iii) returns as output a tuple of three entries: the updated allocation vector, the updated power-density vector, and the gain obtained in the TSPD budget.

Algorithm 1 Decision procedure of the proposed thermal-aware task migration policy.

```
1:  procedure checkRunMig(A = [a_i], R = [ρ*_i])
2:      S ← ⟨ i | a_i = 1 ∧ ∀i,j ∈ S : i ≺_S j → ρ*_i ≤ ρ*_j ⟩
3:      D ← ⟨ i | a_i = 0 ∧ ∀i,j ∈ D : i ≺_D j → ρ*_i ≥ ρ*_j ⟩
4:      types ← {}
5:      gain ← 0
6:      for s ∈ S do                                  ▷ source candidates
7:          if type(s) ∈ types then
8:              continue (Line 6)
9:          end if
10:         types ← types ∪ {type(s)}
11:         for d ∈ D do                              ▷ destination candidates
12:             if type(s) ≠ type(d) then
13:                 continue (Line 11)
14:             end if
15:             Ä ← A;  ä_s ← 0;  ä_d ← 1
16:             R̈ ← getTSPD(Ä)
17:             gain ← min{R̈} − min{R}
18:             if gain > 0 then                      ▷ migration option found
19:                 migrateTasks(s, d)
20:                 applyDVFS(min{R̈})
21:                 return (Ä, R̈, gain)
22:             end if
23:         end for
24:     end for
25:     return (A, R, 0)                              ▷ no beneficial migration found
26: end procedure
```

Given $\mathbf{A}$ and $\mathbf{R}$, the proposed policy first derives the sorted lists of source and destination candidate cores for migration decisions, sorted in the ascending and descending order of power-density constraints ρ_i^*, respectively, (lines 2 and 3). Here, the notations $i \underset{S}{\prec} j$ and $i \underset{D}{\prec} j$ reflect that core i precedes core j in the lists S and D, respectively. After deriving the lists, in lines 6–24, we iterate through the list of migration source candidates S (outer loop, see line 6) and destination candidates D (inner loop, see line 11). Lines 7–9 ensure that, for each type of core, only the first instance in the list of source candidates is investigated, as the first instance would offer the highest TSPD-budget gain if a migration for that core type would be possible.[9] Once a destination core $d \in D$ of the same type is found for the current source core $s \in S$, the

[9]Note that the migration options based on subsequent source candidates of the same core type are not considered as they are not viable. The reason here is that a migration option based on the first candidate may be unavailable only in the absence of an (idle) destination core of that core type. Hence, it becomes pointless to check further source candidates of the same core type.

post-migration allocation vector $\ddot{\mathbf{A}}$ is constructed (line 15). Subsequently, the vector $\ddot{\mathbf{R}}$ of power-density constraints corresponding to $\ddot{\mathbf{A}}$ is derived (line 16) to calculate the TSPD-budget gain corresponding to that migration (line 17). In case of a positive gain, the migration is performed (line 19) and, subsequently, DVFS is applied to adjust the V/F level of each active core to the maximum level it can maintain without exceeding the new TSPD budget $\min\{\ddot{\mathbf{R}}\}$ (line 20). The procedure then terminates by returning the updated allocation vector, the updated vector of power-density constraints, and the attained gain (line 21). If no feasible and beneficial migration has been identified, the procedure terminates in line 25. In terms of run-time compute overhead, the time complexity of the proposed policy stays in the same range as that of the TSPD analysis, i.e., $O(CN)$ (cf. Section III-C).[10]

At run time, the RM may evoke the proposed migration policy to enhance the system's performance after every change in the system's utilization, e.g., after launching or terminating applications. In case of a successful migration, denoted by the positive gain value returned by the procedure, subsequent migrations may be evoked to further improve the system's performance. As mentioned earlier, at low system utilization levels, the TSPD budget in the system is high enough to allow every active core to operate at its peak V/F setting so that performing migrations offers no performance gain even if the TSPD budget is increased. To prevent performing migrations in such cases, the procedure of the proposed policy may be adapted to first check whether the current TSPD budget is lower than the peak power density $\rho_{\max}$ among different types of cores at their peak V/F setting (which can be derived offline). If that is not the case, the procedure can be terminated as every active core can already operate at its peak V/F setting.

Note that task migration itself can impose a performance penalty due to the overhead of relocating tasks, possible temporary suspension of their execution, cache warm-up overheads, etc. The magnitude of this penalty chiefly depends on the frequency of migrations and the memory scheme of the system (since the memory scheme dictates the volume/amount of data which needs to be relocated during the migration process).[11] In our investigations, the gain in TSPD budget (and, hence, performance) saturates with only a few consecutive migrations, keeping the performance impact of migrations fairly negligible. In general, however, to incorporate the impact of migration overhead in the decision process, line 18 of the procedure may be adapted to trigger a migration only if its gain exceeds a minimal threshold. The threshold can be determined according to the metrics above and the current system utilization to ensure that triggered migrations always yield reasonable performance gain.

[10]Computing the sorted lists of cores (lines 2 and 3) introduces a time complexity of $O(C \log C)$. Within the nested loops, the TSPD analysis (line 16) is the most compute intensive part of the procedure with a time complexity of $O(CN)$ where $N \geq C + U$. The core-type checks (lines 7 and 12), however, restrict the invocation of this analysis to at most once per core type. Hence, given T core types in the system, the proposed policy exhibits a total time complexity of $O(TCN)$. This, however, simplifies to $O(CN)$ given that $T \leq 4$ in a typical heterogeneous many-core chip.

[11]A discussion of migration overhead is provided later in Section V-F.

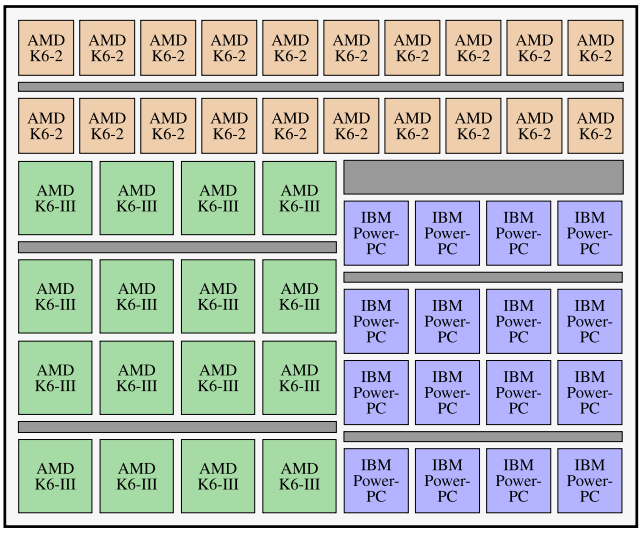

FIGURE 2. Floorplan of the many-core platform used in the experiments. Processing cores are denoted by squares where the core types are reflected by color and noted on the cores. Communication resources and shared memories are contained in the gray regions.

V. EXPERIMENTS

This section presents the results of a series of experiments performed to assess the effectiveness of the proposed thermal-aware migration policy in enhancing the TSPD budget and, thereby, the performance of the system.

A. ARCHITECTURE AND WORKLOAD

For the experiments, a heterogeneous many-core architecture is considered which comprises a total of 52 cores from three core types (AMD K6-III, AMD K6-2, and IBM PowerPC), hence $C = 52$. The floorplan of the chip is illustrated in Fig. 2. The specifications and performance characteristics of the three core types are provided by the Embedded System Synthesis Benchmarks Suite (E3S) [48] for a 180 nm technology size. To adhere to the current process technology node size, these measures are projected to a 10 nm technology using the technology scaling factors provided by [44] which are derived based on [49], [50]. The resulting power and performance characteristics of the three core types are given in Table 1. DVFS support is envisioned on a per-core granularity which enables adjusting the V/F level of each core with frequency steps of 50 MHz with the minimum V/F level of 0.7 V at 1 GHz for all cores. In terms of thermal characteristics, the DTM threshold temperature of the chip is set to $T_{\text{DTM}} = 80°\text{C}$, and an ambient temperature of $T_\infty = 45°\text{C}$ is considered. The RC thermal network of the floorplan has been derived using the HotSpot simulator [44] with the die characteristics listed in Table 2. This architecture is identical to the experimental platform used in [21].

In terms of workload, we consider a streaming automotive application from [48] consisting of $\Omega = 18$ tasks.

TABLE 1. Estimated power- and performance-related characteristics of the processing cores used in the experiments, adopted from [48] (provided for a 180 nm node size) which are projected here to a 10 nm process technology.

core type	α [mm^2]	C_{eff} [nF]	$f_{\max}$ [GHz]	$V\|_{f_{\max}}$ [V]	$(p_{\text{dy}}\|_{f_{\max}}, p_{\text{st}})$ [W]
IBM PowerPC	4.4	0.29	2.0	1.4	(1.1, 0.3)
AMD K6-III	5.6	1.73	4.2	1.1	(8.8, 2.2)
AMD K6-2	3.4	1.23	3.0	1.2	(5.3, 1.3)

TABLE 2. Temperature-related platform characteristics considered by the HotSpot simulator [44] (default configuration) to derive the thermal model of the platform used in the experiments.

property	value
chip thickness [mm]	0.15
heat sink dimensions (W×L) [mm^2]	60 × 60
heat sink thickness [mm]	6.9
heat spreader dimensions (W×L) [mm^2]	30 × 30
heat spreader thickness [mm]	1.0
silicon thermal conductivity [W/mK]	100
silicon specific heat [J/m^{3}K]	$1.75 \cdot 10^6$
heat sink convection capacitance [J/K]	140.4
heat sink convection resistance [K/W]	0.1
heat sink/spreader thermal conductivity [W/mK]	400
heat sink/spreader specific heat [J/m^{3}K]	$3.55 \cdot 10^6$
interface material specific heat [J/m^{3}K]	$4 \cdot 10^6$
interface material thickness [um]	20
interface material thermal conductivity [W/mK]	4.0

The execution time of each task on each of the three core types at its peak frequency is provided by [48] for a 180 nm technology. We scale these measures to a 10 nm technology inversely proportional to the technology scaling factor of the peak frequency of each core type. For clarity of description in the following, the resulting measures are taken to be summarized in an execution-time matrix $\mathbf{X}_{\Omega\times C} = [\chi_{i,j}]$ where $\chi_{i,j}$ denotes the execution time of task i when executed on core j at its peak frequency at a 10 nm technology node. $\mathbf{X}_{\Omega\times C}$ is used in our evaluation of the system's performance as detailed in the following section.

B. EXPERIMENTS SETUP

In the experiments, we investigate the impact of thermal-aware task migration policies on increasing the TSPD budget and the performance of the system when coupled with DVFS to maximally exploit the TSPD budget at different system utilization levels. To that end, we have considered 50 system utilization levels with the number of active cores ranging from $n = 2$ to $n = 51$ (as the level with only one active core and the one with all cores being active are out of the scope for core-level migration decisions). For each utilization level n, we have randomly generated 200 utilization samples where each sample corresponds to a system state with n randomly selected active cores and $(52 - n)$ idle cores. Thus, a total of $50 \times 200 = 10{,}000$ utilization samples are considered in the experiments. In each sample, each active core is taken to execute one instance of the 18-task automotive application from E3S [48]. Each application instance processes an (infinite) stream of input data in successive execution iterations. This is a common operation mode in the embedded domain, e.g., for signal processing applications. The investigated application deployment scheme results in a total of $18 \times n$ tasks being deployed in the system when n cores are active. The tasks deployed on each core are scheduled with a First-In First-Out (FIFO) scheduling policy.[12]

The key performance indicator used for the evaluation of the migration policies in our experiments is the *steady-state throughput* of the system, denoting the system throughput that is observed after the transient effects, e.g., due to cache warm-up or task relocation, have passed. For a given system utilization sample, the steady-state throughput is quantified analytically using Eq. (8) in terms of the total number of application iterations completed per second. As given in Eq. (8), the system throughput is derived based on (i) the allocation vector $\mathbf{A}_{C\times 1}$ which specifies whether each core i is active ($a_i = 1$) or inactive ($a_i = 0$) and (ii) the frequency vector $\mathbf{F}_{C\times 1}$ which specifies the current frequency f_i applied to each core i. The execution time of each task is taken to scale inversely proportional to the frequency of the core executing that task.

$$\text{Throughput}(\mathbf{A}, \mathbf{F}) = \sum_{i=1}^{C} a_i \cdot \frac{f_i}{f_{\max_i} \cdot \sum_{j=1}^{\Omega} \chi_{j,i}} \tag{8}$$

C. INVESTIGATED MIGRATION POLICIES

For each of the utilization samples generated as described above, we investigate the TSPD-budget and performance gain that can be attained using each of the following migration policies:

- **Proposed** which represents the proposed migration policy presented in Section IV-B,
- **PerfOracle** which represents the exhaustive-search-based oracle policy w.r.t. performance gain as introduced in Section IV-A,
- **PdOracle** which represents the exhaustive-search-based oracle policy w.r.t. TSPD-budget gain as introduced in Section IV-A, and
- **HotCold** which represents the class of policies that migrate tasks from hot (active) to cold (idle) cores.

Among these policies, PerfOracle and PdOracle are not practical for use at run time in a real system due their high computational complexity. In our experiments, they only serve as references (optimal policies) when considering performance and TSPD budget, respectively, as the policy's optimization objective.

[12]Note that the decision-making procedure of the proposed migration policy is agnostic to the choice of scheduling policy applied to each core and the number and choices of tasks running on each active core. Its decision-making procedure depends only on the activity state of cores, i.e., active or idle (reflected by input $\mathbf{A} = [a_i]$ in Algorithm 1) and the intermediate results of the TSPD analysis computed based on the activity state of cores (reflected by input $\mathbf{R} = [\rho_i^*]$ in Algorithm 1).

For each system utilization sample, each migration policy performs core-level migrations for as long as it envisions a subsequent migration to be of advantage w.r.t. its pursued objective. More precisely, PerfOracle successively identifies and performs migrations that yield the maximum performance gain and, eventually, terminates when further performance gain cannot be attained by a subsequent migration. PdOracle performs migrations that yield the maximum TSPD-budget gain and terminates if no further gain can be attained through a subsequent migration. HotCold performs migrations successively so long as temperature differences exist between active and idle cores of each type. The termination policy of the proposed approach is similar to that of PdOracle, see Section IV-B for details. An upper bound of 15 successive migrations is considered per utilization sample. Most policies, however, terminate before reaching this limit.

D. RESULTS: SINGLE MIGRATION

To assess the capability of the proposed policy in enhancing the TSPD budget and performance of the system, we have measured the gain attained in these qualities by each of the investigated policies when performing a single migration, namely, their first migration for each utilization sample. For different system utilization levels, Fig. 3 provides the results as an average among all 1,000 considered samples corresponding to each utilization level.

Figure 3-top reflects the TSPD-budget gain attained by the investigated policies relative to the gain yielded by PdOracle which performs optimally in maximizing the TSPD budget. As demonstrated, the proposed policy achieves TSPD-budget gains comparable to that of the PdOracle policy while introducing only a fraction of the compute overhead imposed by that (see the computational complexities reported in Footnotes 8 and 10). On average among all utilization levels, the TSPD-budget gain of the proposed, PerfOracle, and HotCold policies lie within the 92%, 47%, and 46% range of the optimal gain delivered by PdOracle, respectively. This demonstrates the effectiveness of the proposed policy in optimizing the TSPD budget of the system. Among all policies, HotCold exhibits the least impact on the TSPD budget of the system which, as discussed in the following, results in poor performance gain for such policies that aim at reducing temperature gradients greedily.

Figure 3-bottom provides the performance gain attained by each investigated policy relative to the gain yielded by PerfOracle which performs optimally in maximizing performance. As shown, the maximal TSPD-budget gain of PdOracle enables this policy to attain a decent performance gain which lies within the 79% range of the optimal performance gain delivered by PerfOracle. The performance gain yielded by the proposed policy closely follows that of PdOracle and lies within the 71% range of the optimal gain. The performance gain of the HotCold policies lies within the 45% of the optimal gain, rendering this policy the least effective approach in optimizing the system's performance. Note that, when performing a single migration, the PerfOracle policy

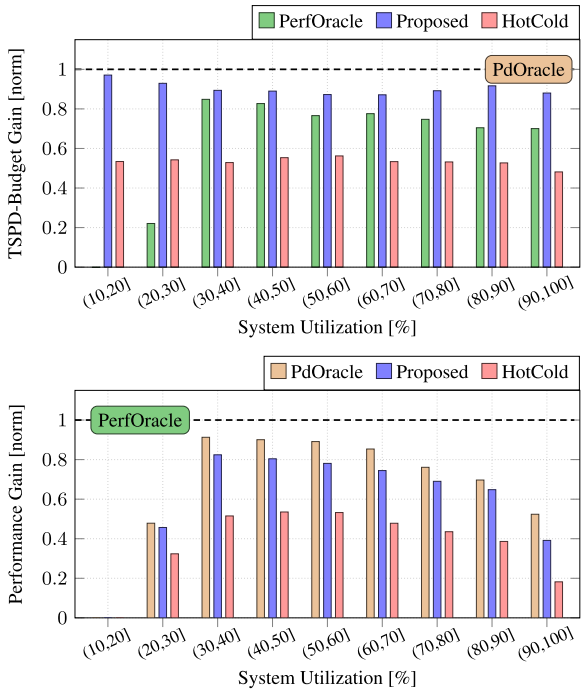

FIGURE 3. The average gain in TSPD-budget (top) and performance (bottom) attained by each migration policy when performing a single migration. The results are reported for different system utilization levels (% active cores), relative to the gain delivered by the optimal policy for the respective metric, namely, PdOracle for TSPD-budget gain and PerfOracle for performance gain.

performs optimally in maximizing the performance gain. Therefore, the performance gain reported in Fig. 3 could be taken as a quantitative measure of how well the goal pursued by each policy correlates with maximizing the system's performance under thermal constraints.

E. RESULTS: MULTIPLE MIGRATIONS

By performing multiple migrations successively, the investigated policies adapt the distribution of active cores in the system gradually to maximally enhance the system's TSPD budget and/or performance. Figure 4 provides the obtained overall gain results when the policies perform task migrations in succession for as long as they deem a subsequent migration beneficial w.r.t. their pursued goal as detailed in Section V-C (yet, not exceeding 15 migrations).

Figure 4-top provides the TSPD-budget gain attained by the policies relative to the gain yielded by PdOracle which performs best in maximizing the TSPD budget in the system.[13] As demonstrated, the proposed policy achieves a significant gain in TSPD budget compared to PerfOracle

[13]Note that, while PdOracle performs optimally in maximizing the TSPD-budget gain in a single migration, it may yield a sub-optimal gain when performing $x \geq 2$ successive migrations even though it outperforms the other policies. The reason here is that the search space of PdOracle is restricted to a single swap of active and idle cores per migration. As it cannot capture the TSPD-budget impact of multiple swaps at once, it may not find the globally optimal sequence of migrations if that sequence contains migration choices that are not necessarily optimal in yielding maximal TSPD-budget gain in their respective migration step.

and HotCold. In fact, on average among all utilization levels, the TSPD-budget gain of the proposed, PerfOracle, and HotCold policies lie within the 70%, 26%, and 29% range of the gain delivered by PdOracle, respectively.

Figure 4-middle reflects the performance gain attained by each policy relative to the gain yielded by PerfOracle. Interestingly, although PerfOracle performs optimally in maximizing performance in a single migration decision, it almost always exhibits an inferior performance gain when performing several migrations in sequence. The reason here is that PerfOracle disregards thermal aspects altogether in its migration decisions. Hence, it often drives the system into local performance optima within only 1–2 migrations so that subsequent migrations cannot offer any further gain in performance. This is also reflected in Fig. 4-bottom which provides the average number of migrations performed by each policy before termination. On the other hand, PdOracle and the proposed policy outperform PerfOracle by 66% and 14%, respectively, in optimizing the system's performance through a series of migrations, as given in Fig. 4-middle. Among all policies, HotCold offers the lowest gain in performance. The observations above advocate for TSPD-budget gain as the superior objective to be pursued in order to optimize performance under thermal constraints.

Recall that the two oracle policies are used only as references, since they are not viable for online use due to their immense compute overhead. Among the two practical policies, namely, Proposed and HotCold, the proposed policy achieves an average performance gain which is over 2× higher than what HotCold delivers. Moreover, as suggested by Fig. 4-bottom, the proposed policy obtains this gain through a considerably lower number of migrations compared to HotCold. On average, the proposed policy terminates after 2–3 migrations. However, as achieving a uniform thermal distribution in the system is fairly unlikely, HotCold never reached its termination criterion in our experiments, so that it was terminated by the enforced upper bound of 15 migrations. Moreover, we have frequently observed scenarios in which HotCold gets trapped in an oscillation loop of migrating tasks between one and the same pair of cores infinitely. In our experiments, the overhead of migrations were taken to be negligible. However, when the overhead accumulates over the large (or even infinite) number of migrations triggered by HotCold, it can outweigh the attained performance gain. Note that the number of migrations triggered by HotCold can be alleviated by considering a temperature-gradient threshold for triggering migrations and identifying oscillation cases to prevent revisiting an already visited utilization scenario. Such remedies, however, cannot improve the performance gain of HotCold beyond the measures reported above.

To obtain insight into the absolute performance and TSPD gain attained by each policy, Fig. 5 provides the distribution of the respective gains. Considering average-case system utilization ranges of 30–70%, among the two oracle policies,

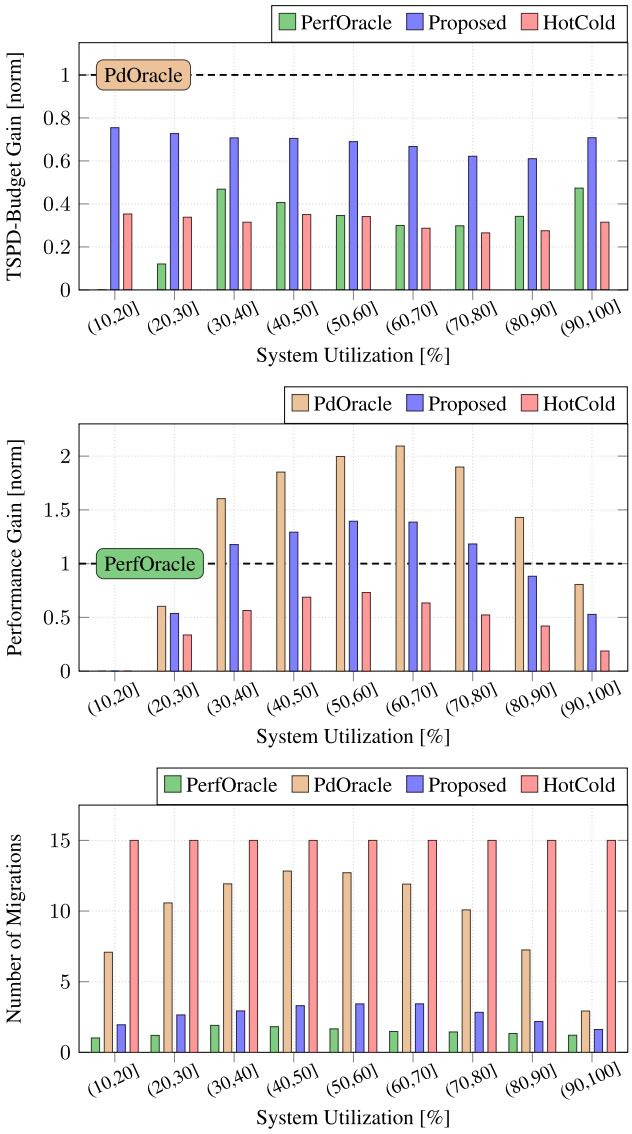

FIGURE 4. The average gain in TSPD-budget (top) and performance (middle) achieved by each migration policy when performing a series of successive migrations. The results are reported for different system utilization levels (% active cores), relative to the gain delivered by the policy which performs optimally in that respect when performing a single migration, namely, PdOracle for TSPD-budget gain and PerfOracle for performance gain. The bottom plot provides the average number of migrations performed by each policy before termination (with an upper bound of 15 migrations).

PdOracle achieves an average TSPD-budget gain of 4.4% (up to 10.5%) and an average performance gain of 1.5% (up to 5.2%), as illustrated in Fig. 5-top and Fig. 5-bottom, respectively. PerfOracle, on the other hand, exhibits inferior results in both aspects, achieving an average TSPD-budget gain of 1.5% (up to 7.3%) and an average performance gain of 0.9% (up to 4.2%). Contrarily to that, the proposed policy yields an average TSPD-budget gain of 3.1% (up to 9.4%) whereas the average gain delivered by HotCold is restricted to 1.4% (up to 9.2%). Notably, at each utilization level, the HotCold policy even reduces the TSPD budget of the system

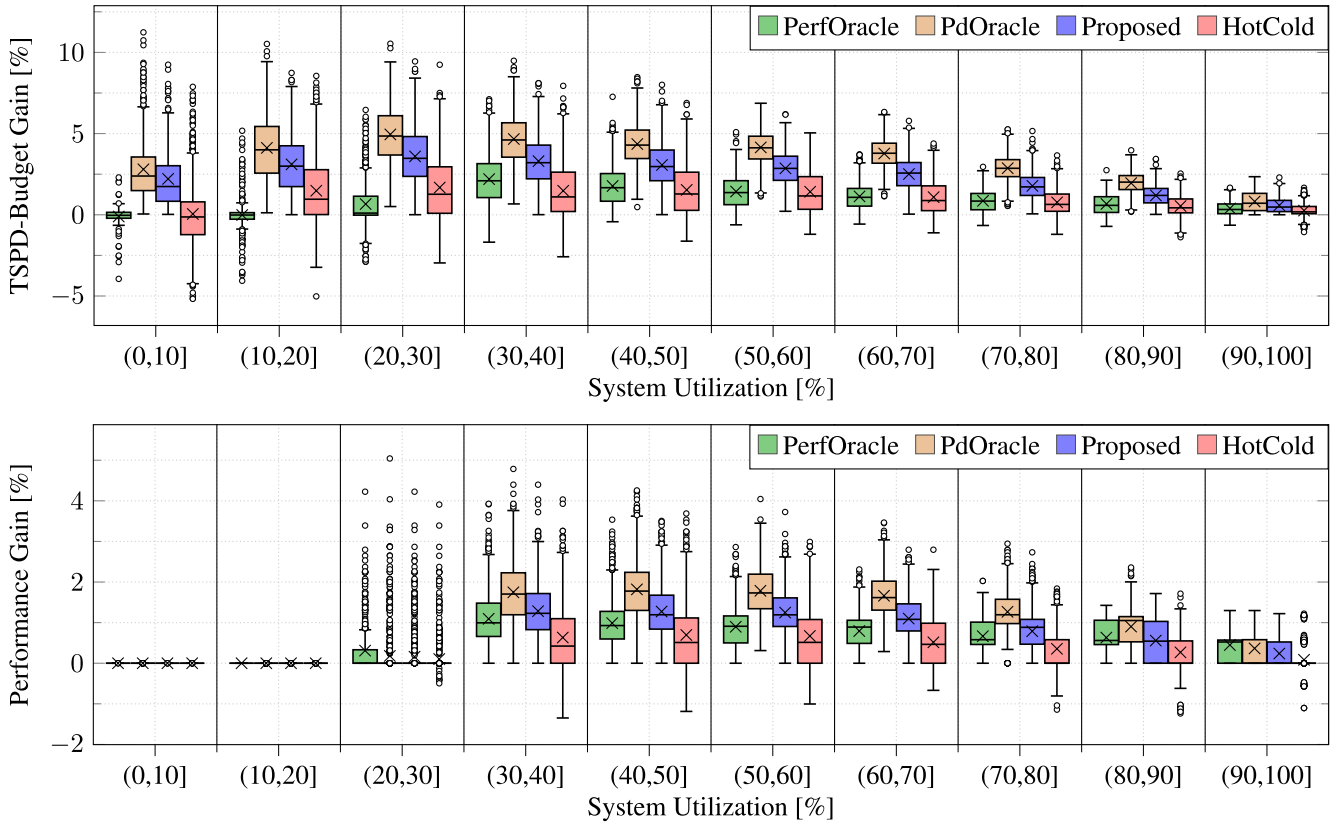

FIGURE 5. The distribution of the TSPD-budget gain (top) and performance gain (bottom) attained by each migration policy when performing a series of migrations in succession. The cross mark in each box represents the average value.

in nearly 25% of samples (see the bottom whiskers[14]). This also leads to comparable observations in terms of performance where HotCold incurs performance degradation in nearly 25% of the samples as shown in Fig. 5-bottom. In fact, HotCold exhibits an average performance gain of 0.5% (up to 4.0%) across the typical utilization range of 30–70%. On the other hand, the TSPD-budget gain attained by the proposed policy enables it to yield an average performance gain of 1% (up to 4.4%) over the same range of utilization levels, hence, offering a 2× higher performance gain compared to HotCold.

F. DISCUSSION AND OUTLOOK

In our experiments, the performance impact of *transient effects* of task migration, e.g., the relocation overhead or cache warm-up, is not reflected as we evaluate performance in terms of the *steady-state* throughput. The impact of these effects is typically insignificant in systems with a moderate *degree of dynamism* (i.e., the rate at which new applications are deployed or running ones are terminated). In highly dynamic systems, however, the performance impact of such effects may become significant. The degree of imposed overhead there depends on several factors, e.g., the frequency of migrations, the migration mechanism in use which specifies *how* a given migration is performed, and the memory scheme of the system (shared or distributed) which specifies how much data need to be relocated between the source and destination cores. Reflecting the absolute overheads in the reported performance-gain results would restrict the expressiveness of our evaluations to the degree of dynamism, choice of migration mechanism, and memory scheme assumed for the target system and, hence, render the results irrelevant for systems deviating from that choice. Therefore, we instead report the *number of migrations* performed by each policy to provide an abstract indicator of the transient performance overhead of each policy relative to the others. Note that, since the proposed policy relies on far fewer migrations compared to HotCold (see Fig. 4-middle) and, thus, imposes a far smaller migration overhead, the performance-gain gap between the proposed policy and HotCold is expected to grow even larger than the measures reported in Section V-E if migration overheads are accounted for.

The experimental results give evidence that, for heterogeneous target systems, a task migration policy oriented towards maximizing TSPD offers higher potential for performance improvement compared to a migration policy that aims at maximizing thermal balance. The proposed migration policy manifests a lightweight heuristic for TSPD-oriented

[14]In a box plot, the bottom whisker reflects the range of samples between the minimal value (lower end of the whisker) and the first quartile (lower end of the box). Thus, 25% of samples fall into the range of the bottom whisker.

migration decisions. Among the possible directions that the proposed policy could be extended in the future, two directions appear particularly promising, namely, extension towards other migration patterns and extension towards systems with QoS requirements). In the following we briefly discuss these direction.

The proposed migration policy pursues maximizing TSPD through migrations that relocate the complete workload of an active core to an idle core of the same type. A possible direction for future work is to extend this decision space to consider other migration patterns for changing the distribution of active cores in the system, e.g., migrating tasks between cores of different types, merging the workload of two active cores, or offloading parts of the workload of an active core to an idle core.

The main focus of this paper has been on the *average-case performance* of the system which is a central goal for best-effort systems. The proposed migration policy can be adopted, however, in a real-time and/or mixed-criticality context as well, since it fulfills a crucial prerequisite of such systems, namely, the prohibition of thermal violations, as pointed out in Section II. Nonetheless, incorporating it into such systems requires careful evaluation and filtering of the migration options to ensure seamless satisfaction of the system's QoS constraints, for instance, by means of an online migration timing analysis and admission check in the case of applications with real-time constraints (see, e.g. [51], for details on such techniques). Extending its decision criteria for such systems and investigating its effectiveness in such a setting, therefore, presents another direction for future work.

VI. CONCLUSION

This paper presented a task migration policy for thermal-aware performance optimization in heterogeneous many-core systems. Based on the observation that performance optimization under thermal constraints correlates with maximizing the Thermally Safe Power Density (TSPD) budget in the system, the proposed policy aims at optimizing the TSPD budget and uses Dynamic Voltage and Frequency Scaling (DVFS) to exploit the power-density budget of active cores to maximize the system's performance. The proposed policy is developed based on an analytical power-budgeting scheme that enforces thermal safety. Hence, by using this analysis coupled with DVFS, it eliminates thermal emergencies by construction. Experimental results demonstrated the superiority of the proposed policy in performance optimization under thermal constraints when compared to existing policies which aim at greedily reducing the average, variance, or gradient of temperature as an indirect means to enhance performance.

REFERENCES

[1] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. Solid-State Circuits*, vol. SSC-9, no. 5, pp. 256–268, Oct. 1974.

[2] *Tile Processor Architecture Overview for the TILE-Gx Series*, Tilera Corp., San Jose, CA, USA, 2012.

[3] B. Bohnenstiehl, A. Stillmaker, J. Pimentel, T. Andreas, B. Liu, A. Tran, E. Adeagbo, and B. Baas, "A 5.8 pJ/Op 115 billion ops/sec, to 1.78 trillion ops/sec 32 nm 1000-processor array," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Jun. 2016, pp. 1–2.

[4] B. D. de Dinechin, R. Ayrignac, P.-E. Beaucamps, P. Couvert, B. Ganne, P. G. de Massas, F. Jacquet, S. Jones, N. M. Chaisemartin, F. Riss, and T. Strudel, "A clustered manycore processor architecture for embedded and accelerated applications," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2013, pp. 1–6.

[5] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proc. Annu. Design Automat. Conf. (DAC)*, May 2013, pp. 1–10.

[6] A. K. Singh, P. Dziurzanski, H. R. Mendis, and L. S. Indrusiak, "A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems," *ACM Comput. Surveys*, vol. 50, no. 2, pp. 1–40, Mar. 2018.

[7] B. Pourmohseni, M. Glaß, J. Henkel, H. Khdr, M. Rapp, V. Richthammer, T. Schwarzer, F. Smirnov, J. Spieck, J. Teich, A. Weichslgartner, and S. Wildermann, "Hybrid application mapping for composable many-core systems: Overview and future perspective," *J. Low Power Electron. Appl.*, vol. 10, no. 4, p. 38, Nov. 2020.

[8] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2011, pp. 365–376.

[9] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives," in *Proc. Annu. Design Automat. Conf. (DAC)*, Jun. 2014, pp. 1–6.

[10] H. Amrouch, B. Khaleghi, A. Gerstlauer, and J. Henkel, "Reliability-aware design to suppress aging," in *Proc. Annu. Design Automat. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[11] A. K. Singh, S. Dey, K. McDonald-Maier, K. R. Basireddy, G. V. Merrett, and B. M. Al-Hashimi, "Dynamic energy and thermal management of multi-core mobile platforms: A survey," *IEEE Des. Test. Comput.*, vol. 37, no. 5, pp. 25–33, Oct. 2020.

[12] H. Khdr, S. Pagani, M. Shafique, and J. Henkel, "Thermal constrained resource management for mixed ILP-TLP workloads in dark silicon chips," in *Proc. Annu. Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[13] Y. Lee, H. S. Chwa, K. G. Shin, and S. Wang, "Thermal-aware resource management for embedded real-time systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2857–2868, Nov. 2018.

[14] E. W. Wachter, C. de Bellefroid, K. R. Basireddy, A. K. Singh, B. M. Al-Hashimi, and G. Merrett, "Predictive thermal management for energy-efficient execution of concurrent applications on heterogeneous multicores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 6, pp. 1404–1415, Jun. 2019.

[15] H. Khdr, T. Ebi, M. Shafique, H. Amrouch, and J. H. Karlsruhe, "MDTM: Multi-objective dynamic thermal management for on-chip systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6.

[16] S. Pagani, H. Khdr, J.-J. Chen, M. Shafique, M. Li, and J. Henkel, "Thermal safe power (TSP): Efficient power budgeting for heterogeneous manycore systems in dark silicon," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 147–162, Jan. 2017.

[17] T. Blickle, J. Teich, and L. Thiele, "System-level synthesis using evolutionary algorithms," *Design Autom. Embedded Syst.*, vol. 3, no. 1, pp. 23–58, 1998.

[18] K. Manna, P. Mukherjee, S. Chattopadhyay, and I. Sengupta, "Thermal-aware application mapping strategy for Network-on-Chip based system design," *IEEE Trans. Comput.*, vol. 67, no. 4, pp. 528–542, Apr. 2018.

[19] H. Khdr, S. Pagani, E. Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel, "Power density-aware resource management for heterogeneous tiled multicores," *IEEE Trans. Comput.*, vol. 66, no. 3, pp. 488–501, Mar. 2017.

[20] S. Liu, J. Zhang, Q. Wu, and Q. Qiu, "Thermal-aware job allocation and scheduling for three dimensional chip multiprocessor," in *Proc. Int. Symp. Quality Electronic Design (ISQED)*, Mar. 2010, pp. 390–398.

[21] B. Pourmohseni, F. Smirnov, H. Khdr, S. Wildermann, J. Teich, and J. Henkel, "Thermally composable hybrid application mapping for real-time applications in heterogeneous many-core systems," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2019, pp. 220–232.

[22] M. Rapp, M. Sagi, A. Pathania, A. Herkersdorf, and J. Henkel, "Power- and cache-aware task mapping with dynamic power budgeting for many-cores," *IEEE Trans. Comput.*, vol. 69, no. 1, pp. 1–13, Jan. 2019.

[23] M. Ansari, J. Saber-Latibari, M. Pasandideh, and A. Ejlali, "Simultaneous management of peak-power and reliability in heterogeneous multicore embedded systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 623–633, Mar. 2020.

[24] J. Saber-Latibari, M. Ansari, P. Gohari-Nazari, S. Yari-Karin, A. M. H. Monazzah, and A. Ejlali, "READY: Reliability- and deadline-aware power-budgeting for heterogeneous multicore systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 4, pp. 646–654, Apr. 2021.

[25] H. Wang, M. Zhang, S. X.-D. Tan, C. Zhang, Y. Yuan, K. Huang, and Z. Zhang, "New power budgeting and thermal management scheme for multi-core systems in dark silicon," in *Proc. 29th IEEE Int. Syst.-on-Chip Conf. (SOCC)*, Sep. 2016, pp. 344–349.

[26] Y. G. Kim, M. Kim, J. Kong, and S. W. Chung, "An adaptive thermal management framework for heterogeneous multi-core processors," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 894–906, Jun. 2020.

[27] M. S. Mohammed, A. A. M. Al-Kubati, N. Paraman, A. A.-H. Ab Rahman, and M. N. Marsono, "DTaPO: Dynamic thermal-aware performance optimization for dark silicon many-core systems," *Electronics*, vol. 9, no. 11, p. 1980, Nov. 2020.

[28] A. Merkel, F. Bellosa, and A. Weissel, "Event-driven thermal management in SMP systems," in *Proc. 2nd Workshop Temperature-Aware Comput. Syst. (TACS)*, 2005, pp. 1–10.

[29] Y. Ge, P. Malani, and Q. Qiu, "Distributed task migration for thermal management in many-core systems," in *Proc. 47th Design Autom. Conf. (DAC)*, 2010, pp. 579–584.

[30] A. K. Coskun, J. L. Ayala, D. Atienza, T. S. Rosing, and Y. Leblebici, "Dynamic thermal management in 3D multicore architectures," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2009, pp. 1410–1415.

[31] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Proactive temperature management in MPSoCs," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2008, pp. 165–170.

[32] F. Mulas, M. Pittau, M. Buttu, S. Carta, A. Acquaviva, L. Benini, D. Atienza, and G. De Micheli, "Thermal balancing policy for streaming computing on multiprocessor architectures," in *Proc. Design, Autom. Test Eur.*, Mar. 2008, pp. 734–739.

[33] D. Cuesta, J. Ayala, J. Hidalgo, D. Atienza, A. Acquaviva, and E. Macii, "Adaptive task migration policies for thermal control in MPSoCs," in *Proc. IEEE Annu. Symp. VLSI (ISVLSI)*. Dordrecht, The Netherlands: Springer, 2011, pp. 83–115.

[34] X. Wang, A. K. Singh, and S. Wen, "Exploiting dark cores for performance optimization via patterning for many-core chips in the dark silicon era," in *Proc. 12th IEEE/ACM Int. Symp. Netw.-on-Chip (NOCS)*, Oct. 2018, pp. 1–8.

[35] M. Prakash Gupta, M. Cho, S. Mukhopadhyay, and S. Kumar, "Thermal investigation into power multiplexing for homogeneous many-core processors," *J. Heat Transf.*, vol. 134, no. 6, Jun. 2012.

[36] J. Choi, C.-Y. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose, "Thermal-aware task scheduling at the system software level," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 2007, pp. 213–218.

[37] M. Gomaa, M. D. Powell, and T. N. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system," *ACM SIGOPS Operating Syst. Rev.*, vol. 38, no. 5, pp. 260–270, Dec. 2004.

[38] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *Proc. Annu. Design Autom. Conf. (DAC)*, 2008, pp. 734–739.

[39] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Utilizing predictors for efficient thermal management in multiprocessor SoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1503–1516, Oct. 2009.

[40] K. Zhang, A. Guliani, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman, "Machine learning-based temperature prediction for runtime thermal management across system components," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 2, pp. 405–419, Feb. 2018.

[41] V. Hanumaiah, S. Vrudhula, and K. S. Chatha, "Performance optimal online DVFS and task migration techniques for thermally constrained multi-core processors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 11, pp. 1677–1690, Nov. 2011.

[42] R. Kumar, A. Sachan, A. Gogoi, and B. Ghoshal, "Application phase behavior-guided thermal management of embedded platforms," *IEEE Embedded Syst. Lett.*, vol. 12, no. 4, pp. 121–124, Dec. 2020.

[43] P. Greenhalgh, "Big.LITTLE processing with arm cortex-a15 & cortex-a7," ARM, Cambridge, U.K., White Paper, 2011.

[44] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.

[45] N. H. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. London, U.K.: Pearson, 2015.

[46] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, Dec. 2003.

[47] T. J. Eguia, S. X.-D. Tan, R. Shen, E. H. Pacheco, and M. Tirumala, "General behavioral thermal modeling and characterization for multi-core microprocessor design," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2010, pp. 1136–1141.

[48] R. Dick. (2010). *Embedded System Synthesis Benchmarks Suite (E3S)*. [Online]. Available: http://ziyang.eecs.umich.edu/ dickrp/e3sdd/

[49] (2019). *International Technology Roadmap for Semiconductors (ITRS)*. [Online]. Available: http://www.itrs.net/

[50] R. Borkar, M. Bohr, and S. Jourdan, "Advancing Moore's law in 2014," Intel, Santa Clara, CA, USA, 2014. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/presentation/advancing-moores-law-in-2014-presentation.pdf

[51] B. Pourmohseni, F. Smirnov, S. Wildermann, and J. Teich, "Real-time task migration for dynamic resource management in many-core systems," in *Proc. Workshop Next Gener. Real-Time Embedded Syst. (NG-RES)*, vol. 77. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020, pp. 5:1–5:14.

BEHNAZ POURMOHSENI received the B.S. and M.S. degrees in electrical engineering from Shahid Beheshti University, Iran, in 2011 and 2013, respectively, and the Ph.D. degree in computer science from Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany, in 2021. She is currently a Research Engineer at the Corporate Research Center, Robert Bosch GmbH, Germany. Her research interests include system-level design automation, performance analysis, and resource management for dynamic distributed systems with real-time requirements, in particular, embedded multi/many-core systems.

STEFAN WILDERMANN received the Diploma and Ph.D. (Dr.-Ing.) degrees in computer science Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany, in 2006 and 2012, respectively. He leads the Reconfigurable Computing Group, Computer Science Department, FAU. Since then, he has been a Research Assistant, a Lecturer, and a Group Leader with the Chair of Hardware/Software Co-Design, FAU. His current research interests include reconfigurable computing and system-level design automation for embedded systems.

FEDOR SMIRNOV was born in Udomlja, Russia, in 1989. He received the B.S. and M.S. degrees in mechatronics and the Ph.D. degree in computer science from Friedrich-Alexander Universität Erlangen-Nürnberg (FAU), in 2014 and 2019. Since January 2020, he has been a Postdoctoral Researcher at the Computer Science Faculty, University of Innsbruck. His research interests include multi-objective constrained optimization problems and the design automation of distributed systems, with a particular emphasis on cloud-edge and automotive applications.

PAUL E. MEYER was born in Feuchtwangen, Bavaria, Germany, in 1996. He received the B.S. degree in communication and information technology from Friedrich-Alexander Universität Erlangen-Nürnberg (FAU), Erlangen, Germany, in 2020. From 2019 to 2021, he was a Student at the Fraunhofer Gesellschaft IIS, Erlangen. Since 2021, he has been a Software Developer at sepp.med gmbh, Röttenbach, Germany.

JÜRGEN TEICH (Fellow, IEEE) received the M.S. (Dipl.-Ing.) degree (Hons.) from the University of Kaiserslautern, Germany, in 1989, and the Ph.D. (Dr.-Ing.) degree *(summa cum laude)* from the University of Saarland, Saarbrücken, Germany, in 1993. From 1998 to 2002, he was a Full Professor with the Electrical Engineering and Information Technology Department, University of Paderborn, Germany. He is with Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany, where he is directing the Chair for Hardware/Software Co-Design, since 2003. Since 2010, he has been a Principal Coordinator of the Transregional Research Center 89 ''Invasive Computing'' funded by the German Research Foundation (DFG). He has edited two text books on *Hardware/Software Co-Design* and a Handbook on *Hardware/Software Co-Design* (Springer). His current research interests include electronic design automation of embedded systems with emphasis on hardware/software co-design, reconfigurable computing, and multi-core systems. He is a member of the Academia Europaea, the Academy of Europe, the National Academy of Science and Engineering (acatech), and the German Society of Humboldtians. He serves on the Editorial Board of journals, including *ACM Transactions on Design Automation of Electronic Systems*, *ACM Transactions on Embedded Computing Systems*, and *IEEE Design and Test* magazine. He has organized various ACM/IEEE Conferences/Symposium as a Program Chair, including CODES+ISSS'07, FPL'08, ASAP'10, and DATE'16. He was the Vice General Chair of DATE 2018 and the General Chair of DATE 2019.