Affinity-Driven Modeling and Scheduling for Makespan Optimization in Heterogeneous Multiprocessor Systems

Kun Cao, Junlong Zhou, Peijin Cong, Liying Li,
Tongquan Wei, Mingsong Chen, Shiyan Hu, and X. Sharon Hu

***Abstract*—With the advent of heterogeneous multiprocessor architectures, efficient scheduling for high performance has been of significant importance. However, joint considerations of reliability, temperature, and stochastic characteristics of precedence-constrained tasks for performance optimization make task scheduling particularly challenging. In this paper, we tackle this challenge by using an affinity (i.e., probability) -driven task allocation and scheduling approach that decouples schedule lengths and thermal profiles of processors. Specifically, we separately model the affinity of a task for processors with respect to schedule lengths and the affinity of a task for processors with regard to chip thermal profiles considering task reliability and stochastic characteristics of task execution time and inter-task communication time. Subsequently, we combine the two types of affinities, and design a scheduling heuristic that assigns a task to the processor with the highest joint affinity. Extensive simulations based on randomly generated stochastic and real-world applications are performed to validate the effectiveness of the proposed approach. Experiment results show that the proposed scheme can reduce the system makespan by up to 30.1% without violating the temperature and reliability constraints compared to benchmarking methods.**

***Index Terms*—Affinity-driven modeling, stochastic dependent tasks, scheduling, makespan, reliability, temperature.**

I. Introduction

DUE to the advance of technology scaling and ever increasing demand for performance, multiprocessors have replaced uniprocessors to become the main design paradigm for current and future processors [1], [2]. For multiprocessor scheduling, a typical target is to minimize the overall length of time required to execute the applications on processors, namely makespan. Most of multiprocessor scheduling strategies are designed based on worst-case execution times of tasks. However, the uncertainty in task execution times are not considered. Since tasks in real-world usually contain conditional instructions and/or operations, different inputs of such a task may result in varying execution times [3]–[6]. Given this, some novel stochastic scheduling algorithms have been investigated and developed. Tang et al. [3] presented a stochastic earliest finish time scheduling scheme to schedule tasks with random processing time and communication time for makespan minimization. Li et al. [4] developed a heuristic energy-aware stochastic task scheduling algorithm to jointly optimize makespan and energy consumption of heterogeneous computing systems. A stochastic dynamic level scheduling algorithm was also presented by Li et al. [5] to tackle the problem of minimizing a parallel application's expected schedule length (i.e., makespan) on heterogeneous cluster systems. Zheng et al. [6] presented a Monte Carlo based scheduling approach to minimize the expected value of makespan for dependent tasks with stochastic execution times.

Reliability is another key concern in multiprocessor scheduling. This is because the susceptibility of modern processors to soft errors is dramatically increasing with the relentless scaling of feature size and operating voltage [7]. Considerable research efforts have been devoted to jointly handling makespan and reliability issues [8]–[11]. Girault et al. [8] proposed a scheduling algorithm that enables users to choose a tradeoff between makespan and reliability. Sathappan et al. [9] developed a modified genetic algorithm to improve system reliability and makespan. Wang et al. [10] presented a look-ahead genetic algorithm to optimize both makespan and reliability of a workflow application in distributed computing environments. Aupy et al. [11] addressed the problem of realizing energy minimization under the constraints of a prescribed bound on makespan and reliability. However, all the above works assume that task execution times are fixed or determined, and the uncertainty in task execution times is not taken into account.

With the ever continued technology scaling, the chip power density has increased exponentially, which in turn leads to the elevated chip temperature. A system will fall into the predicament of functional incorrectness, low reliability and even permanent damage if the operating temperature exceeds a certain threshold [12]. Thus, thermal management has been a significant and pressing research issue in computing systems. Numerous thermal management techniques such as dynamic voltage-frequency scaling (DVFS) [13], scheduling priority adjustments [14], task migration [15], and task splitting [16] have been proposed to obtain high-performance computing

K. Cao, P. Cong, L. Li, and T. Wei are with the Department of Computer Science and Technology, East China Normal University, Shanghai 200062, China. M. Chen is with the Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200241, China. T. Wei and M. Chen are the co-corresponding authors: tqwei@cs.ecnu.edu.cn, mschen@sei.ecnu.edu.cn.

J. Zhou is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.

S. Hu is with the Department of Electrical and Computer Engineering, Michigan Technological University, MI 49931, USA.

X. Sharon Hu is with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46656, USA.

This work was partially supported by Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000) and Natural Science Foundation of China (Grant No. 61672230).

capability while maintaining the peak temperature of chips below a specified temperature limit. However, thermal-aware designs themselves based on these techniques cannot guarantee a dependable system operation since reliability is not taken as a design constraint.

To the best of our knowledge, no research effort has been devoted to optimizing makespan with joint considerations of stochastic task execution times, reliability, and temperature. In [3]–[6], it has been shown that stochastic algorithms achieve better performance in terms of makespan than the algorithms that use worst-case execution times of tasks. Since makespan minimization is a typical optimization goal while both reliability and temperature issues should be carefully addressed in the design of heterogeneous multiprocessor systems, it is necessary to optimize makespan by jointly considering stochastic task execution times, reliability, and temperature. In this paper, we explore this optimization problem and propose an effective affinity-driven task scheduling scheme. The major contributions of this paper are summarized as follows.

- We separately model the affinity of a task for processors with respect to schedule lengths and the affinity of a task for processors with regard to chip thermal profiles. Task reliability and stochastic characteristics of task execution time and inter-task communication time are considered.
- We derive a joint affinity of a task for processors, and propose an affinity-driven task scheduling heuristic for makespan optimization under the reliability and temperature constraints. Task precedence constraints are taken into account in the scheduling heuristic.
- We conduct extensive simulation experiments based on real-world and randomly generated stochastic applications. The results show that the proposed scheme can reduce up to 30.1% makespan without violating the temperature and reliability constraints.

The rest of the paper is organized as follows. Section II introduces the system models. Section III formulates the problem definition. Section IV models the affinity of a task for processors, and Section V describes the proposed affinity-driven scheduling heuristic. The effectiveness of the proposed scheme is verified by simulation in Section VI and concluding remarks are given in Section VII.

II. System Models

A. Architecture Model

We consider a loosely coupled heterogeneous multiprocessor system like a cluster of machines [4], [5], [17], [18]. Each machine is characterized by a processor and thus the system can be modeled as M heterogeneous processors $\Theta = \{\Theta_1, \Theta_2, \cdots, \Theta_M\}$. Every processor $\Theta_k \in \Theta$ $(1 \leq k \leq M)$ is equipped with ℓ_k discrete supply voltage and operating frequency pairs denoted by $\{(v_{k,1}, f_{k,1}), (v_{k,2}, f_{k,2}), \cdots, (v_{k,\iota}, f_{k,\iota}), \cdots, (v_{k,\ell_k}, f_{k,\ell_k})\}$ $(1 \leq \iota \leq \ell_k)$. For the sake of easy presentation, $v_{k,min} = v_{k,1} \leq v_{k,2} \leq \cdots \leq v_{k,\ell_k} = v_{k,max} = 1.0$ and $f_{k,min} = f_{k,1} \leq f_{k,2} \leq \cdots \leq f_{k,\ell_k} = f_{k,max} = 1.0$ hold, where $(v_{k,min}, f_{k,min})$ is the minimal supply voltage and operating frequency pair whereas $(v_{k,max}, f_{k,max})$ is the maximal supply voltage and operating frequency pair. Every processor Θ_k supports one sleep mode and ℓ_k active modes that are characterized by the supply voltage and operating frequency pair $(v_{k,\iota}, f_{k,\iota})$. Tasks can be only executed in the active mode and the processor is idle when it is in sleep mode. The data transfer rate of inter-processor is considered to be fixed and constant while the communication cost of intra-processor is assumed to be negligible [18].

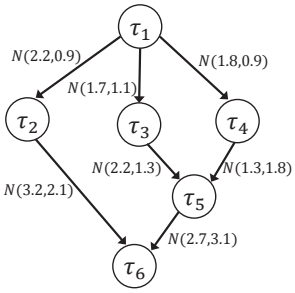

Fig. 1: An example of a stochastic parallel application.

B. Application Model

Assume that stochastic parallel applications are to execute on the target system. Such an application consists of precedence constrained tasks and can be represented by a directed acyclic graph (DAG) $G = (V, E)$. $V = \{\tau_1, \cdots, \tau_i, \cdots, \tau_N\}$ denotes the set of N tasks. E is the set of edges corresponding to the precedence constraint between tasks. For the edge $(\tau_i, \tau_j) \in E$, task τ_i is a direct predecessor of task τ_j, which means task τ_j cannot start to execute until task τ_i finishes its execution. The characteristic of task τ_i is described by a tuple $\tau_i : \{\mu_i, E[t_{i,k,\iota}], Var[t_{i,k,\iota}], E[c_{i,j}], Var[c_{i,j}], RG\}$. μ_i is the activity factor of task τ_i, which demonstrates the heterogenous nature of tasks [16]. $t_{i,k,\iota}$ is the fault-free execution time of task τ_i executing on processor Θ_k at frequency $f_{k,\iota}$. $E[t_{i,k,\iota}]$ and $Var[t_{i,k,\iota}]$ are the expected value and variance of $t_{i,k,\iota}$, respectively. $E[c_{i,j}]$ and $Var[c_{i,j}]$ are the expected value and variance of communication time $c_{i,j}$ between task τ_i and task τ_j, respectively. RG is the reliability goal of an application and it is also the reliability goal of each task in the application [19].

Since tasks usually contain conditional instructions and/or operations, different inputs of such a task result in varying execution times that approximately follow normal distributions [4], [5]. Therefore, we use the notation $t_{i,k,\iota} \sim N(E[t_{i,k,\iota}], Var[t_{i,k,\iota}])$ to denote the normal distribution of task execution time $t_{i,k,\iota}$. The communication time $c_{i,j}$ between task τ_i and task τ_j is equal to zero if the two tasks are assigned to the same processor, and follows normal distribution otherwise [4]. Similarly, the probability distribution of communication time $c_{i,j}$ can be represented by $c_{i,j} \sim N(E[c_{i,j}], Var[c_{i,j}])$. Fig. 1 shows an example of a stochastic parallel application. The normal distributions of communication times among tasks are given in the figure, and the normal distributions of task execution times on two heterogeneous processors are listed in Table I. All the values in Fig. 1 and Table I are generated by using the tool of Task Graphs for Free (TGFF) [20] which will be detailed later in Section VI-A.

TABLE I: The normal distribution of execution time $t_{i,k,\iota}$.

	TI DSP processor		Intel Xton processor		
Task	$t_{i,1,1}$	$t_{i,1,2}$	$t_{i,2,1}$	$t_{i,2,2}$	$t_{i,2,3}$
τ_1	$N(4.1, 2.4)$	$N(2.6, 1.0)$	$N(4.6, 0.9)$	$N(3.3, 1.7)$	$N(2.2, 1.6)$
τ_2	$N(2.9, 1.8)$	$N(2.0, 1.3)$	$N(3.1, 1.1)$	$N(2.2, 0.9)$	$N(1.5, 0.4)$
τ_3	$N(3.5, 2.1)$	$N(2.5, 2.4)$	$N(4.0, 3.2)$	$N(2.9, 1.4)$	$N(1.9, 1.6)$
τ_4	$N(4.7, 2.3)$	$N(3.0, 1.7)$	$N(5.0, 4.1)$	$N(3.6, 2.6)$	$N(2.6, 1.4)$
τ_5	$N(4.4, 1.7)$	$N(2.8, 2.1)$	$N(4.8, 2.0)$	$N(3.5, 2.7)$	$N(2.5, 2.2)$
τ_6	$N(3.8, 1.3)$	$N(2.4, 0.6)$	$N(4.3, 1.1)$	$N(3.0, 0.7)$	$N(2.1, 1.5)$

C. Reliability Model

Transient fault (resulting in soft error) is a type of failure that appears for a short time and then disappears without damage to the device, and is caused by electromagnetic interference or cosmic radiation. Exponential distribution is typically utilized to model soft errors. The average fault rate of processor Θ_k at operating frequency $f_{k,\iota}$ is modeled as [21]

$$\lambda(f_{k,\iota}) = \lambda_{k,max} 10^{\frac{d_k(1-f_{k,\iota})}{1-f_{k,1}}}, \quad (1)$$

where $\lambda_{k,max}$ is the average fault rate at processor maximal frequency $f_{k,max}$, and d_k (> 0) is a hardware specific factor indicating the sensitivity of fault rates to frequency scaling.

The reliability of a task is defined as the probability of its successful execution without the occurrence of soft errors, and can be determined by the exponential failure law. Therefore, using the exponential distribution assumption, the reliability of task τ_i running at frequency $f_{k,\iota}$ without considering precedence constraints is expressed as [21]

$$R'(\tau_i, f_{k,\iota}) = e^{-\lambda(f_{k,\iota})t_{i,k,\iota}}. \quad (2)$$

When considering precedence constraints between tasks, the reliability of task τ_i is then given by

$$R(\tau_i, f_{k,\iota}) = R(\tau_{m_i}, \tau_{p_i}, \tau_{q_i}, \cdots, \tau_{s_i}, \tau_{n_i}) \cdot R'(\tau_i, f_{k,\iota}). \quad (3)$$

In Eq. (3), $R(\tau_{m_i}, \tau_{p_i}, \tau_{q_i}, \cdots, \tau_{s_i}, \tau_{n_i})$ indicates the probability that all the direct predecessors $\tau_{m_i}, \tau_{p_i}, \tau_{q_i}, \cdots, \tau_{s_i}, \tau_{n_i}$ of task τ_i are successfully executed, and it is calculated as

$$\begin{aligned} R(\tau_{m_i},\tau_{p_i}, \tau_{q_i}, \cdots, \tau_{s_i}, \tau_{n_i}) = R(\tau_{m_i}) R(\tau_{p_i}|\tau_{m_i}) \cdot \\ R(\tau_{q_i}|\tau_{m_i}, \tau_{p_i}) \cdots R(\tau_{n_i}|\tau_{m_i}, \tau_{p_i}, \tau_{q_i}, \cdots, \tau_{s_i}). \end{aligned} \quad (4)$$

Reliability $R(\tau_{m_i})$, omitting the operating frequency of task τ_{m_i}, denotes the probability of successful execution of task τ_{m_i}, and reliability $R(\tau_{n_i}|\tau_{m_i}, \tau_{p_i}, \tau_{q_i}, \cdots, \tau_{s_i})$ represents the conditional probability of successful execution of task τ_{n_i} when tasks $\tau_{m_i}, \tau_{p_i}, \tau_{q_i}, \cdots, \tau_{s_i}$ are all successfully executed.

We adopt the task replication technique to provide fault-tolerance, since it has been widely used in improving system reliability due to transient faults. Furthermore, we consider systems that use replication to tolerate up to one transient fault since single-fault-tolerance is a common assumption [2]. In addition, uniform processor and frequency is adopted for all replicas of a given task to reduce the computational overhead. To check the occurrence of soft errors, an acceptance test [22] is performed at the end of execution of each task replica. If the acceptance test indicates no error, then the output of the task replica is committed; otherwise, it is discarded. Since the time overhead of acceptance test is much smaller than task execution time [22], we assume that the time overhead for fault detection does not increase task execution time.

D. Power Model

The power consumption of a CMOS-based processor can be modeled as the sum of static power consumption P^{sta} and dynamic power consumption P^{dyn} [16]. P^{sta} is consumed by the leakage current required to maintain basic state of circuits. The static power consumption at the supply voltage and frequency pair $(v_{k,\iota}, f_{k,\iota})$ is given by [16]

$$P^{sta}_{k,\iota} = \vartheta_k v_{k,\iota} + \eta_k v_{k,\iota} T_k(t), \quad (5)$$

where ϑ_k and η_k are non-negative architecture dependent constants of processor Θ_k, and $T_k(t)$ is the operating temperature of processor Θ_k at time instance t.

The dynamic power consumption of a processor is only related to processor switching activity when executing tasks and can be formulated as a function of supply voltage and operating frequency. The dynamic power consumption when executing task τ_i at the supply voltage and operating frequency pair $(v_{k,\iota}, f_{k,\iota})$ is calculated as [16]

$$P^{dyn}_{i,k,\iota} = \mu_i \rho_k v^2_{k,\iota} f_{k,\iota}, \quad (6)$$

where μ_i is the active factor of task τ_i and ρ_k is a non-negative constant depending on the architecture of processor Θ_k.

E. Temperature Model

An accurate and practical temperature model is needed to accurately characterize the thermal behavior of a task. For a loosely coupled heterogeneous multiprocessor system like a cluster of machines, the heat transfer among multiple machines has shown to be negligible [17], [23]. Given this, we adopt a heat-independent RC thermal model widely utilized for thermal-aware task scheduling to capture thermal profiles. The RC thermal model is expressed by the following system of ordinary differential equation [13], [24]

$$C\frac{dT(t)}{dt} = P(t) - \frac{T(t) - T^{amb}}{R}, \quad (7)$$

where $P(t)$ is the power consumption at time instance t, and it can be obtained using Eqs. (5)–(6). R and C are thermal resistance and capacitance, respectively and they are hardware dependent constants. T^{amb} is the ambient temperature.

Let T^{amb}_k, R_k, and C_k represent the ambient temperature, thermal resistance, and thermal capacitance of processor Θ_k, respectively. Then, for a given time interval $[t_0, t_0 + t_{i,k,\iota}]$, if the initial temperature is $T^{init}_{i,k,\iota}$, the ending temperature of executing task τ_i on processor Θ_k at the supply voltage and frequency pair $(v_{k,\iota}, f_{k,\iota})$, denoted as $T^{end}_{i,k,\iota}$, can be derived by solving Eq. (7). The derivation of ending temperature $T^{end}_{i,k,\iota}$ is described in Eqs. (8)–(10) [24]. In Eq. (10), $T^{std}_{i,k,\iota}$ is the steady state temperature of task τ_i when executing on processor Θ_k at the supply voltage and frequency pair $(v_{k,\iota}, f_{k,\iota})$. The steady state temperature of task τ_i is the temperature that will be reached if infinite number of instances of task τ_i execute continuously on the processor.

III. Problem Definition

The system makespan can be defined as the overall length of time required to complete the tasks of applications on pro-

$$\frac{dT_k(t)}{dt} = \frac{T_k^{amb} + \vartheta_k R_k v_{k,\iota} + \eta_k R_k v_{k,\iota} T_k(t) + \mu_i \rho_k R_k v_{k,\iota}^2 f_{k,\iota}}{R_k C_k} - \frac{T_k(t)}{R_k C_k} \tag{8}$$

$$\int_{t_0}^{t_0+t_{i,k,\iota}} dt = \int_{T_{i,k,\iota}^{init}}^{T_{i,k,\iota}^{end}} \frac{dT_k(t)}{\frac{T_k^{amb} + \vartheta_k R_k v_{k,\iota} + \mu_i \rho_k R_k v_{k,\iota}^2 f_{k,\iota}}{R_k C_k} - (\frac{1-\eta_k R_k v_{k,\iota}}{R_k C_k}) T_k(t)} \tag{9}$$

$$\begin{cases} T_{i,k,\iota}^{end} = T_{i,k,\iota}^{std} - (T_{i,k,\iota}^{std} - T_{i,k,\iota}^{init}) e^{-(\frac{1-\eta_k R_k v_{k,\iota}}{R_k C_k}) t_{i,k,\iota}} \\ T_{i,k,\iota}^{std} = \frac{T_k^{amb} + \vartheta_k R_k v_{k,\iota} + \mu_i \rho_k R_k v_{k,\iota}^2 f_{k,\iota}}{1-\eta_k R_k v_{k,\iota}} \end{cases} \tag{10}$$

cessors. Let $MS(k)$ be the schedule length of tasks executing on processor Θ_k, the makespan of the whole system is then the maximum of schedule length of processors, that is,

$$MS = \max\{MS(k) | k \in [1, 2, \cdots, M]\}. \tag{11}$$

For the DAG $G = (V, E)$ of an application, the precedence constraints among tasks should be satisfied. Let $E[t_f(i)]$ and $E[t_s(j)]$ denote the expected finish execution time of task τ_i, and the expected start execution time of task τ_j, respectively. A binary variable $\Delta_{i,j}$ is utilized to indicate the precedence constraint between task τ_i and task τ_j. If task τ_i precedes task τ_j, $\Delta_{i,j}$ takes the value of 1, otherwise takes the value of 0. Task τ_i precedes task τ_j meaning that there exists a path from task τ_i to task τ_j.

To avoid temperature-induced failures, the peak temperature of processors should be below a temperature limit (threshold) T^{max}. The value of T^{max} is in general specified based on system design requirements. Let T_k^{peak} denote the peak temperature of processor Θ_k, then the on-chip peak temperature, denoted by T^{peak}, can be calculated as

$$T^{peak} = \max\{T_k^{peak} | k \in [1, 2, \cdots, M]\}. \tag{12}$$

The system is deemed to be safe when the peak temperature T^{peak} does not exceed the threshold temperature T^{max}. In addition to the temperature constraint, all tasks in an application should achieve their reliability goals, and maintain the precedence constraints. Considering the above design constraints, the task allocation and scheduling problem to minimize the expected value $E[MS]$ of makespan for an M-processor system can be formulated into the below form.

$$\text{Minimize: } E[MS] \tag{13}$$

$$\text{Subject to: } TS^{peak} \leq T^{max} \tag{14}$$

$$RG \leq RS(\tau_i) \tag{15}$$

$$E[(t_f(i) + c_{i,j})] \cdot \Delta_{i,j} \leq E[t_s(j)] \tag{16}$$

$$\forall i, j \in [1, 2, \cdots, N]$$

TS^{peak} is the estimated value of T^{peak}, and $RS(\tau_i)$ is the estimated reliability for task τ_i. Eq. (14) is the temperature constraint, which prevents the peak temperature from exceeding the temperature limit. Eq. (15) is the reliability constraint, which means the reliability goal of every task should be satisfied. Eq. (16) is the precedence constraint, which suggests task τ_j cannot start its execution until task τ_i ends if task τ_i precedes task τ_j.

IV. Modeling The Affinity of A Task

In this section, we model the affinity of a task for processors considering reliability, temperature, and stochastic characteristics of task execution time and inter-task communication time. First, task replication technique is adopted to provide the required level of fault-tolerance in Section IV-A. Then, Section IV-B models the affinity of a task for processors with respect to schedule lengths, and Section IV-C models the affinity of a task for processors with regard to thermal profiles. Section IV-D finally shows the way to calculate the processor sleep time utilized in the modeling of the two affinities.

A. Calculating Task Replication Number

Since task execution time $t_{i,k,\iota}$ is normally distributed, the reliability $R'(\tau_i, f_{k,\iota})$ without precedence constraints in Eq. (2) follows a log-normal distribution [25]

$$R'(\tau_i, f_{k,\iota}) \sim ln[N(P(\tau_i, f_{k,\iota}), Q(\tau_i, f_{k,\iota}))]. \tag{17}$$

where $P(\tau_i, f_{k,\iota}) = -\lambda(f_{k,\iota}) E[t_{i,k,\iota}]$ and $Q(\tau_i, f_{k,\iota}) = \lambda^2(f_{k,\iota}) Var[t_{i,k,\iota}]$. The expected value and variance of reliability $R'(\tau_i, f_{k,\iota})$ are then given by

$$\begin{cases} E[R'(\tau_i, f_{k,\iota})] = e^{P(\tau_i, f_{k,\iota}) + \frac{1}{2} Q(\tau_i, f_{k,\iota})} \\ Var[R'(\tau_i, f_{k,\iota})] = E^2[R'(\tau_i, f_{k,\iota})](e^{Q(\tau_i, f_{k,\iota})} - 1) \end{cases}. \tag{18}$$

During the step of deriving the minimum number of replicas of task τ_i, the minimum replicated number of its direct predecessors has already been determined in previous steps. That is, $R(\tau_{m_i}, \cdots, \tau_{n_i})$ in Eq. (3) is a known value at that moment. Therefore, the expected value and variance of reliability $R(\tau_i, f_{k,\iota})$ with precedence constraints can be calculated by Eq. (19) and Eq. (20), respectively.

$$E[R(\tau_i, f_{k,\iota})] = R(\tau_{m_i}, \cdots, \tau_{n_i}) \cdot E[R'(\tau_i, f_{k,\iota})] \tag{19}$$

$$Var[R(\tau_i, f_{k,\iota})] = R^2(\tau_{m_i}, \cdots, \tau_{n_i}) \cdot Var[R'(\tau_i, f_{k,\iota})] \tag{20}$$

$R(\tau_i, f_{k,\iota})$ can be estimated by

$$RS(\tau_i, f_{k,\iota}) \approx E[R(\tau_i, f_{k,\iota})] + \Lambda_{i,k,\iota} \sqrt{Var[R(\tau_i, f_{k,\iota})]}, \tag{21}$$

where $\Lambda_{i,k,\iota}$ is a constant number. Given this, the reliability of τ_i considering γ_i replica execution is thus estimated by

$$RS(\tau_i, f_{k,\iota}, \gamma_i) = 1 - (1 - RS(\tau_{i_1}, f_{k,\iota})) \cdot (1 - RS(\tau_{i_2}, f_{k,\iota})) \cdots (1 - RS(\tau_{i_{\gamma_i}}, f_{k,\iota})), \tag{22}$$

TABLE II: An example of replica parameter table: execution times of task replicas in multimedia application mpegplay.

Task	Frequency	1st Replica	2nd Replica	3rd Replica	$\cdots$	1st–2nd Replica	1st–3rd Replica	$\cdots$
τ_1	f_1	$N(32.6, 22.8)$	$N(30.2, 19.7)$	$N(28.3, 16.7)$	$\cdots$	$N(62.8, 35.7)$	$N(91.1, 46.8)$	$\cdots$
	f_2	$N(18.9, 10.2)$	$N(18.1, 9.9)$	$N(17.9, 11.5)$	$\cdots$	$N(37.0, 14.1)$	$N(54.9, 26.3)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
τ_{30}	f_1	$N(46.4, 25.3)$	$N(40.5, 22.0)$	$N(38.8, 27.7)$	$\cdots$	$N(86.9, 42.7)$	$N(125.7, 66.1)$	$\cdots$
	f_2	$N(32.1, 16.8)$	$N(26.9, 12.3)$	$N(29.6, 18.1)$	$\cdots$	$N(59.0, 30.2)$	$N(88.6, 50.4)$	$\cdots$

and the minimum replica number γ_i required to achieve the target reliability goal RG for task τ_i can be easily derived by

$$RS(\tau_i, f_{k,\iota}, \gamma_i) \geq RG. \tag{23}$$

$RS(\tau_{i_{\gamma_i}}, f_{k,\iota})$ is the estimated reliability of γ_ith replica $\tau_{i_{\gamma_i}}$ of task τ_i executing at frequency $f_{k,\iota}$. The execution time distribution $N(E[t_{i_1,k,\iota}], Var[t_{i_1,k,\iota}])$, $N(E[t_{i_2,k,\iota}], Var[t_{i_2,k,\iota}])$, $\cdots$, $N(E[t_{i_{\gamma_i},k,\iota}], Var[t_{i_{\gamma_i},k,\iota}])$ of replica τ_{i_1}, τ_{i_2}, $\cdots$, $\tau_{i_{\gamma_i}}$ of task τ_i can be obtained from the replica parameter table. TABLE II is an example of replica parameter table, which illustrates execution times of task replicas in multimedia application mpegplay consisting of 30 tasks running on a TI DSP processor. The TI DSP processor supports two supply voltage and operating frequency pairs: one is (0.98V, 2.0GHz) and the other is (1.42V, 3.0GHz) [4]. In TABLE II, the columns "1st replica", "2nd replica", and "3rd replica" present execution time distributions of the first replica, second replica, and third replica of tasks, respectively. The columns "1st–2nd replica" and "1st–3rd replica" indicate the time distributions of consecutively executing the first and second replica, and the first, second, and third replica of tasks, respectively. This replica parameter table is obtained by using the technique of profiling task off-line traces [4], [26], [27]. Using this technique, we can get the replica parameter table of tasks executing on the target heterogeneous multiprocessor system. Therefore, reliability $RS(\tau_{i_1}, f_{k,\iota})$, $RS(\tau_{i_2}, f_{k,\iota})$, $\cdots$, $RS(\tau_{i_{\gamma_i}}, f_{k,\iota})$ can be readily calculated by using Eq. (21). Once the minimum replica number γ_i of task τ_i executing at frequency $f_{k,\iota}$ is derived by using Eq. (23), we can obtain the distribution of execution time $t_{i,k,\iota}$ from the replica parameter table when replicas from τ_{i_1} to $\tau_{i_{\gamma_i}}$ of task τ_i are sequentially executed to fulfill task reliability constraint.

B. Modeling the Affinity with Respect to Schedule Lengths

The affinity $A_{k,w}$ of task τ_i for processor Θ_k with respect to schedule lengths is defined as

$$A_{k,w} = 1 - \frac{MS(k)}{\max_{1\leq k\leq M} MS(k)}. \tag{24}$$

The schedule length $MS(k)$ of processor Θ_k is given by

$$MS(k) = \begin{cases} \omega(i,k) + t'^{slp}_{i,k,\iota} + t_{i,k,\iota} & \tau_i \in Q_k \\ MS(k) & \text{otherwise} \end{cases}. \tag{25}$$

where $\omega(i,k) = \max\{MS(k), CF(i,k)\}$. From Eqs. (24)–(25) we can see that the smaller the schedule length $MS(k)$ is, the higher the affinity $A_{k,w}$ is. In Eq. (25), $\omega(i,k) + t'^{slp}_{i,k,\iota}$ is the start execution time of task τ_i if task τ_i is assigned to processor Θ_k. Q_k is the task queue on processor Θ_k. $t'^{slp}_{i,k,\iota}$ is the estimated value of sleep time $t^{slp}_{i,k,\iota}$ that is utilized to prevent the peak temperature of processor Θ_k from exceeding temperature limit (The way to calculate sleep time $t^{slp}_{i,k,\iota}$ and estimated sleep time $t'^{slp}_{i,k,\iota}$ will be detailedly described in Section IV-D.). $CF(i,k)$ is the communication finish time between task τ_i and its direct predecessors when task τ_i is assigned to processor Θ_k, and it is given by

$$CF(i,k) = \max_{\tau_m \in pre(\tau_i)} (t_f(m) + c_{m,i}), \tag{26}$$

where $pre(\tau_i)$ denotes the set of tasks that are direct predecessors of task τ_i. We below present a theorem which shows that the finish execution time of each task is normally distributed.

Theorem 1: *For any task DAG $G = (V, E)$ where both task execution time and inter-task communication time are normally distributed, the finish execution time of each task is normally distributed as well.*

Proof: We use the approach of structural induction to prove this theorem. First, for the induction basis, the finish execution time $t_f(1)$ of entry task τ_1 is given by $t_f(1) = t'^{slp}_{1,k,\iota} + t_{1,k,\iota}$. Since the execution time $t_{1,k,\iota}$ is normally distributed and $t'^{slp}_{1,k,\iota}$ is a determined value, it is easy to see that their sum $t'^{slp}_{1,k,\iota} + t_{1,k,\iota}$ is normally distributed [28]. Next, for the induction, we consider the allocation of task τ_i $(2 \leq i \leq N)$ to processor Θ_k. The induction hypothesis is that the finish execution times of all tasks before assigning task τ_i to processor Θ_k are normally distributed. That is, the finish execution times of task $\tau_2, \tau_3, \cdots, \tau_{i-1}$ are normally distributed. Since both the finish execution time $t_f(m)$ and communication time $c_{m,i}$ in Eq. (26) are normally distributed, $\max_{\tau_m \in pre(\tau_i)}(t_f(m) + c_{m,i})$ follows a normal distribution and its distribution can be derived by using Clark's equations which calculate the expected value and variance of the greatest of multiple normally distributed random variables [29]. Take the case where task τ_i has two direct predecessors $\tau_{m'}$ and $\tau_{n'}$ as an example. According to Clark's equations [29], the expected value and variance of communication finish time $CF(i,k)$ are given by Eq. (27). In a similar way, Clark's equations can be used recursively to derive expected value and variance of the greatest of multiple (≥ 3) normally distributed random variables. Before task τ_i is assigned to processor Θ_k, the current schedule length $MS(k)$ is normally distributed. This is because the current schedule length $MS(k)$ is the finish execution time of the last task assigned to processor Θ_k. Therefore, the item $\omega(i,k)$ defined in Eq. (25) is normally distributed, and its expected value and variance can be derived by using Clark's equations in a same way as illustrated in Eq. (27). Since $\omega(i,k)$ and $t_{i,k,\iota}$ are normally distributed as analyzed above, and $t'^{slp}_{i,k,\iota}$ is the estimated value of sleep time, their sum $\omega(i,k) + t_{i,k,\iota} + t'^{slp}_{i,k,\iota}$ in Eq. (25) follows

$$\begin{cases} E[CF(i,k)] = \{E[t_f(m')] + E[c_{m',i}]\}\Phi(b) + \{E[t_f(n')] + E[c_{n',i}]\}\Phi(-b) + a\varphi(b) \\ Var[CF(i,k)] = (\{E[t_f(m')] + E[c_{m',i}]\}^2 + \{Var[t_f(m')] + Var[c_{m',i}]\})\Phi(b) + \\ \qquad (\{E[t_f(n')] + E[c_{n',i}]\}^2 + \{Var[t_f(n')] + Var[c_{n',i}]\})\Phi(-b) + \\ \qquad (\{E[t_f(m')] + E[c_{m',i}]\} + \{E[t_f(n')] + E[c_{n',i}]\})a\varphi(b) - E^2[CF(i,k)] \end{cases} \tag{27}$$

$$\text{where} \begin{cases} a & = \sqrt{\{Var[t_f(m')] + Var[c_{m',i}]\} + \{Var[t_f(n')] + Var[c_{n',i}]\}} \\ b & = (\{E[t_f(m')] + E[c_{m',i}]\} - \{E[t_f(n')] + E[c_{n',i}]\})/a \\ \varphi(x) & = (2\pi)^{-1/2} e^{-x^2/2} \\ \Phi(x) & = \int_{-\infty}^{x} \varphi(t)dt \end{cases}$$

a normal distribution with the expected value and variance given below [28].

$$\begin{cases} E[\omega(i,k) + t_{i,k,\iota} + t'^{slp}_{i,k,\iota}] = E[\omega(i,k)] + E[t_{i,k,\iota}] + t'^{slp}_{i,k,\iota} \\ Var[\omega(i,k) + t_{i,k,\iota} + t'^{slp}_{i,k,\iota}] = Var[\omega(i,k)] + Var[t_{i,k,\iota}] \end{cases} \tag{28}$$

Thus, we finally conclude that the finish execution time $t_f(i)$ of task τ_i is normally distributed. The theorem is proved. ■

According to Theorem 1 and its proof, we can see that $MS(k)$ and $\max_{1\leq k\leq M} MS(k)$ in Eq. (24) are two correlated normal random variables. Let X_1 represent $MS(k)$, X_2 denote $\max_{1\leq k\leq M} MS(k)$, and X denote the ratio X_1/X_2. Based on the work [30], the cumulative distribution function $F(x)$ of X can be expressed as

$$F(x) \approx \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{E[X_2]x - E[X_1]}{\sqrt{Var[X_1]Var[X_2]}S(x)}} e^{-\frac{1}{2}t^2} dt, \tag{29}$$

where $S(x) = (\frac{x^2}{Var[X_1]} - \frac{2x}{Var[X_2]} + \frac{1}{Var[X_2]})^{\frac{1}{2}}$. It is clear that for a given value x_0, $F(x_0)$ is a determined value rather than a random variable. Therefore, using $F(x_0)$ and Eq. (24), the affinity $A_{k,w}$ is readily derived.

C. Modeling the Affinity with Regard to Thermal Profiles

We select the ending temperature of a task to characterize its thermal profiles since the peak temperature of a task execution is reached at its start/ending time instant [31]. Given this, the affinity $A_{k,t}$ of task τ_i for processor Θ_k with regard to thermal profiles is derived as

$$A_{k,t} = \begin{cases} 1 - \frac{TS^{end}_{i,k,\iota}}{T^{max}} & TS^{end}_{i,k,\iota} < T^{max} \\ 0 & \text{otherwise} \end{cases}, \tag{30}$$

where $TS^{end}_{i,k,\iota}$ is the estimated value of ending temperature $T^{end}_{i,k,\iota}$ when task τ_i is executed on processor Θ_k at frequency $f_{k,\iota}$. From the above equation we can see that the lower the estimated value $TS^{end}_{i,k,\iota}$ is, the higher the affinity $A_{k,t}$ is.

Since the execution time of task τ_i follows a normal distribution, the ending temperature $T^{end}_{i,k,\iota}$ is a random variable and it has been given by Eq. (10). According to the definition of log-normal distribution, we have the following log-normal distribution about temperature [25]

$$\frac{T^{end}_{i,k,\iota} - T^{std}_{i,k,\iota}}{T^{init}_{i,k,\iota} - T^{std}_{i,k,\iota}} \sim lnN(\zeta_{k,\iota} E[t_{i,k,\iota}], \zeta^2_{k,\iota} Var[t_{i,k,\iota}]), \tag{31}$$

where $\zeta_{k,\iota} = -(\frac{1-\eta_k R_k v_{k,\iota}}{R_k C_k})$. For simplicity, we use the notation $\widetilde{T}^{end}_{i,k,\iota}$ to represent $\frac{T^{end}_{i,k,\iota} - T^{std}_{i,k,\iota}}{T^{init}_{i,k,\iota} - T^{std}_{i,k,\iota}}$, thus Eq. (31) becomes

$$\widetilde{T}^{end}_{i,k,\iota} \sim lnN(\zeta_{k,\iota} E[t_{i,k,\iota}], \zeta^2_{k,\iota} Var[t_{i,k,\iota}]). \tag{32}$$

According to the manipulation of log-normal distribution, the expected value and variance of $\widetilde{T}^{end}_{i,k,\iota}$ are given by [25]

$$\begin{cases} E[\widetilde{T}^{end}_{i,k,\iota}] = e^{\zeta_{k,\iota} E[t_{i,k,\iota}] + \frac{1}{2}\zeta^2_{k,\iota} Var[t_{i,k,\iota}]} \\ Var[\widetilde{T}^{end}_{i,k,\iota}] = E^2[\widetilde{T}^{end}_{i,k,\iota}](e^{\zeta^2_{k,\iota} Var[t_{i,k,\iota}]} - 1) \end{cases}. \tag{33}$$

The expected value and variance of ending temperature $T^{end}_{i,k,\iota}$ are hence expressed as

$$\begin{cases} E[T^{end}_{i,k,\iota}] = (T^{init}_{i,k,\iota} - T^{std}_{i,k,\iota}) E[\widetilde{T}^{end}_{i,k,\iota}] + T^{std}_{i,k,\iota} \\ Var[T^{end}_{i,k,\iota}] = (T^{init}_{i,k,\iota} - T^{std}_{i,k,\iota})^2 Var[\widetilde{T}^{end}_{i,k,\iota}] \end{cases}. \tag{34}$$

Based on Eq. (34), the ending temperature $T^{end}_{i,k,\iota}$ can be estimated by

$$TS^{end}_{i,k,\iota} \approx E[T^{end}_{i,k,\iota}] + \Lambda^{end}_{i,k,\iota} \sqrt{Var[T^{end}_{i,k,\iota}]}, \tag{35}$$

where $\Lambda^{end}_{i,k,\iota}$ is a constant number.

Due to precedence constraints, task execution may not be continuous on a processor. This leads to the fact that the ending temperature of a task may not be the initial temperature of its successive task executed on the same processor. However, the initial temperature $T^{init}_{i,k,\iota}$ of task τ_i is a determined value in Eq. (34). To estimate the ending temperature $T^{end}_{i,k,\iota}$ of task τ_i, it should firstly determine the initial temperature $T^{init}_{i,k,\iota}$. When task τ_i is assigned to processor Θ_k, task τ_i can be executed if and only if i) processor Θ_k is in idle mode, ii) the communication between task τ_i and its predecessors is finished, and iii) the temperature constraint is not violated. Therefore, this idle period $p(i,k)$ is calculated as

$$p(i,k) = \omega(i,k) - MS(k) + t'^{slp}_{i,k,\iota}, \tag{36}$$

where both $\omega(i,k)$ and $MS(k)$ are normally distributed as analyzed before. The expected value and variance of idle

period $p(i,k)$ can be given by [28]

$$\begin{cases} E[p(i,k)] = E[\omega(i,k)] - E[MS(k)] + t'^{slp}_{i,k,\iota} \\ Var[p(i,k)] = Var[\omega(i,k)] + Var[MS(k)] \\ \qquad\qquad -2Cov(\omega(i,k), MS(k)) \end{cases}, \quad (37)$$

where $Cov(\omega(i,k), MS(k))$ is the covariance between $\omega(i,k)$ and $MS(k)$, and it can be calculated as

$$Cov(\omega(i,k), MS(k)) = E[\omega(i,k) \cdot MS(k)] - E[\omega(i,k)] \cdot E[MS(k)]. \quad (38)$$

However, since $\omega(i,k) \cdot MS(k)$ does not follow an existing probability distribution, it is difficult to directly derive $E[\omega(i,k) \cdot MS(k)]$. Consider the fact that we are only interested in getting an approximate affinity of task assignment, we ignore the calculation of $Cov(\omega(i,k), MS(k))$ for simplicity. Therefore, the variance $Var[p(i,k)]$ is given by

$$Var[p(i,k)] = Var[\omega(i,k)] + Var[MS(k)]. \quad (39)$$

The probability distribution of idle period $p(i,k)$ is expressed as $p(i,k) \sim N(E[p(i,k)], Var[p(i,k)])$ due to the fact that both $\omega(i,k)$ and $MS(k)$ are normally distributed [28], and this idle period can be treated as an idle task to lower the on-chip temperature. Given this, the ending temperature of this idle task can be easily estimated by using Eq. (35), and the estimated temperature is also the initial temperature of task τ_i.

D. Calculating Processor Sleep Time

Sleep mode distribution [16] can be utilized to calculate the minimal time for a processor to stay in sleep mode while incurring minimum impact on system makespan. The sleep mode distribution is mainly achieved by two steps. It first derives the safe temperature $T^{safe}_{i,k,\iota}$ of task τ_i under the peak temperature constraint, then calculates the sleep period $t^{slp}_{i,k,\iota}$ that the processor needs to stay in the sleep mode to reduce temperature from $T^{init}_{i,k,\iota}$ to $T^{safe}_{i,k,\iota}$. The safe temperature $T^{safe}_{i,k,\iota}$ of task τ_i is defined as the threshold of the initial temperature of task τ_i. If processor Θ_k continues to execute task τ_i when the initial temperature $T^{init}_{i,k,\iota}$ is higher than the safe temperature $T^{safe}_{i,k,\iota}$, the temperature constraint would be definitely violated. The derivation of safe temperature for a task with determined execution time has been thoroughly explored in [16], and we suggest readers to refer to [16] for more details.

However, the sleep mode distribution ignores the stochastic characteristics of tasks. We develop the sleep mode distribution to make it suitable for stochastic task scheduling. Since execution time $t_{i,k,\iota}$ is normally distributed, the expected value and variance of safe temperature $T^{safe}_{i,k,\iota}$ can be calculated as [28]

$$\begin{cases} E[T^{safe}_{i,k,\iota}] = T^{std}_{i,k,\iota} - (T^{std}_{i,k,\iota} - T^{max})/E[\widetilde{T}^{end}_{i,k,\iota}] \\ Var[T^{safe}_{i,k,\iota}] = \sqrt{|T^{max} - T^{std}_{i,k,\iota}|} Var[\widetilde{T}^{end}_{i,k,\iota}] / e^{-\frac{4E[t_{i,k,\iota}]}{R_k C_k}} \end{cases}. \quad (40)$$

The estimated safe temperature $TS^{safe}_{i,k,\iota}$ is hence derived as

$$TS^{safe}_{i,k,\iota} \approx E[T^{safe}_{i,k,\iota}] + \Lambda^{safe}_{i,k,\iota} \sqrt{Var[T^{safe}_{i,k,\iota}]}, \quad (41)$$

where $\Lambda^{safe}_{i,k,\iota}$ is a constant number. Once the estimated safe temperature $TS^{safe}_{i,k,\iota}$ is derived, we can calculate how long the processor needs to stay in sleep mode to reduce the temperature from $T^{init}_{i,k,\iota}$ to $TS^{safe}_{i,k,\iota}$. The sleep period $t^{slp}_{i,k,\iota}$ of processor Θ_k can be estimated by

$$t'^{slp}_{i,k,\iota} = (-R_k C_k) ln(TS^{safe}_{i,k,\iota} / T^{init}_{i,k,\iota}). \quad (42)$$

V. Affinity-Driven Task Scheduling Heuristic

The proposed affinity-driven scheduling heuristic operates as follows. For every task, the weighted affinities of assigning the task to all processors are first calculated based on the schedule lengths and thermal profiles of processors. Then, the processor with highest weighted affinity is chosen to execute the task. The weighted affinity of a task for a processor is defined as follows. Let A_k represent the weighted affinity of task τ_i for processor Θ_k, and it is given by

$$A_k = \alpha \times A_{k,w} + (1 - \alpha) \times A_{k,t}, \quad (43)$$

where $A_{k,w}$ is the affinity determined by schedule lengths, whereas $A_{k,t}$ is the affinity determined by thermal profiles. α (in the range $[0,1]$) is the weighting parameter, and it is introduced to combine affinity $A_{k,w}$ and affinity $A_{k,t}$.

The proposed task scheduling heuristic is implemented by Algorithm 1. Inputs of Algorithm 1 are the weighting parameter, replica parameter table, peak temperature limit, task DAG, and processor set. Line 1 initializes task assignment, task operating frequency, sleep time, and task replicated number. Line 2 constructs a queue Q of tasks in topological order. Lines 3-31 iteratively determine the most suitable processor and frequency for all tasks. In each round of iteration, line 4 derives the minimum replicated number of task τ_i executing on every processor's frequency. Line 5 gets task τ_i's execution time distribution with consideration of replica execution from the replica parameter table. Line 6 sets the operating frequency $f(\tau_i)$ of task τ_i to the maximal frequency $f_{1,max}$ of processor Θ_1, and initializes $t'^{slp}_{i,1}$ and $f_1(\tau_i)$. The derivation of the ending temperature $TS^{end}_{i,1,max}$ of task τ_i is implemented in line 7. If $TS^{end}_{i,1,max} > T^{max}$ holds, Algorithm 2 is called to reduce the temperature of task τ_i (lines 8-10). Line 11 calculates the affinity $A_1(\tau_i)$ of allocating task τ_i to processor Θ_1 with operating frequency $f_1(\tau_i)$. Line 12 initializes A_{max} and $flag$, where $flag$ indicates the index of processor that generates the maximal affinity A_{max}. Line 13 sets the sleep time t'^{slp}_{i} to $t'^{slp}_{i,1}$, and replicated number γ_i to $\gamma_{i,flag,f_1(\tau_i)}$ which is the minimum replicated number of task τ_i executing on processor Θ_{flag} at operating frequency $f_1(\tau_i)$. Lines 14-27 iteratively update the most suitable processor for task τ_i from processor Θ_2 to processor Θ_M. In each round of iteration, line 15 sets $f_k(\tau_i)$ to maximal operating frequency $f_{k,max}$ of processor Θ_k, and sleep time $t'^{slp}_{i,k}$ to 0. Line 16 derives the ending temperature $TS^{end}_{i,k,max}$ of task τ_i when executing at frequency $f_k(\tau_i)$. If the ending temperature $TS^{end}_{i,k,max}$ exceeds the temperature limit T^{max}, lines 17-19 call Algorithm 2 to reduce the temperature of task τ_i. Lines 20-21 calculate affinity $A_{k,t}(\tau_i)$ and affinity $A_{k,w}(\tau_i)$ when task τ_i is to execute at operating frequency $f_k(\tau_i)$. Based on the two affinities, the

Algorithm 1: Affinity-driven task scheduling heuristic

Input: i) task DAG of a parallel application; ii) replica parameter table; iii) processor set $\Theta = \{\Theta_1, \Theta_2, \cdots, \Theta_M\}$; iv) weighting parameter α; v) peak temperature limit T^{max};
Output: i) task assignment $\{Q_1, Q_2, \cdots, Q_M\}$; ii) task operating frequency $\{f(\tau_1), f(\tau_2), \cdots, f(\tau_N)\}$; iii) sleep time $\{t'^{slp}_1, t'^{slp}_2, \cdots, t'^{slp}_N\}$; iv) task replicated number $\{\gamma_1, \gamma_2, \cdots, \gamma_N\}$;

1 $\{Q_1, Q_2, \cdots, Q_M\} \leftarrow \{\emptyset, \emptyset, \cdots, \emptyset\}$, $\{f(\tau_1), f(\tau_2), \cdots, f(\tau_N)\} \leftarrow \{0, 0, \cdots, 0\}$, $\{t'^{slp}_1, t'^{slp}_2, \cdots, t'^{slp}_N\} \leftarrow \{0, 0, \cdots, 0\}$, $\{\gamma_1, \gamma_2, \cdots, \gamma_N\} \leftarrow \{0, 0, \cdots, 0\}$;
2 construct a queue $Q = \{\tau_1, \tau_2, \cdots, \tau_N\}$ of tasks in topological order;
3 **for** $i \leftarrow 1$ *to* N **do**
4 determine minimum replicated number of task τ_i executing on every processor's frequency using Eq. (23);
5 obtain execution time distribution of task τ_i considering replica execution from the replica parameter table;
6 $t'^{slp}_{i,1} \leftarrow 0$, $f(\tau_i) \leftarrow f_{1,max}$, $f_1(\tau_i) \leftarrow f_{1,max}$; // The operating frequency $f(\tau_i)$ of task τ_i is set to the maximal frequency of processor Θ_1.
7 derive $TS^{end}_{i,1,max}$ of task τ_i using Eq. (35);
8 **if** $TS^{end}_{i,1,max} > T^{max}$ **then**
9 $k \leftarrow 1$, call Algorithm 2 to reduce the temperature of task τ_i;
10 **end**
11 compute the affinity $A_1(\tau_i)$ of allocating task τ_i to processor Θ_1 with operating frequency $f_1(\tau_i)$ using Eq. (43);
12 $A_{max} \leftarrow A_1(\tau_i)$, $flag \leftarrow 1$; // $flag$ indicates the index of processor generating the maximal affinity A_{max}.
13 $t'^{slp}_i \leftarrow t'^{slp}_{i,1}$, $\gamma_i \leftarrow \gamma_{i,flag,f_1(\tau_i)}$; // $\gamma_{i,flag,f_1(\tau_i)}$ is the minimum replicated number of task τ_i executing on processor Θ_{flag} at operating frequency $f_1(\tau_i)$. It is obtained by line 4 of Algorithm 1.
14 **for** $k \leftarrow 2$ *to* M **do**
15 $t'^{slp}_{i,k} \leftarrow 0$, $f_k(\tau_i) \leftarrow f_{k,max}$;
16 derive $TS^{end}_{i,k,max}$ of task τ_i when executing at frequency $f_k(\tau_i)$ using Eq. (35);
17 **if** $TS^{end}_{i,k,max} > T^{max}$ **then**
18 call Algorithm 2 to reduce the temperature of task τ_i;
19 **end**
20 derive the affinity $A_{k,t}(\tau_i)$ on frequency $f_k(\tau_i)$ determined by thermal profiles using Eq. (30);
21 obtain the affinity $A_{k,w}(\tau_i)$ on frequency $f_k(\tau_i)$ determined by schedule length using Eq. (24);
22 calculate the weighted affinity $A_k(\tau_i)$ of allocating task τ_i to processor Θ_k with operating frequency $f_k(\tau_i)$ using Eq. (43);
23 **if** $A_{max} \leq A_k(\tau_i)$ **then**
24 $A_{max} \leftarrow A_k(\tau_i)$, $flag \leftarrow k$; $f(\tau_i) \leftarrow f_k(\tau_i)$;
25 $t'^{slp}_i \leftarrow t'^{slp}_{i,k}$, $\gamma_i \leftarrow \gamma_{i,flag,f_k(\tau_i)}$;
26 **end**
27 **end**
28 assign task τ_i to processor Θ_{flag};
29 update Q and Q_{flag} by $Q \leftarrow Q - \tau_i$, $Q_{flag} \leftarrow Q_{flag} + \tau_i$;
30 update the schedule length $MS(flag)$ using Eq. (25);
31 **end**
32 derive expected value $E[MS]$ of system makespan MS defined in Eq. (11) using Clark's equations [29];
33 exit with **Output**;

weighted affinity $A_k(\tau_i)$ of allocating task τ_i to processor Θ_k with operating frequency $f_k(\tau_i)$ is obtained in line 22. Lines 23-26 update the processor Θ_{flag} that generates the maximal allocation affinity A_{max}, and operating frequency $f(\tau_i)$, sleep time t'^{slp}_i and the minimum replicated number γ_i. Lines 28-30 assign task τ_i to processor Θ_{flag}, and update the task queue Q, task subqueue Q_k and schedule length $MS(flag)$. Repeat this process until all the tasks are assigned to processors (lines 3-31). The expected value of system makespan is derived in line 32, and the algorithm exits with **Output** in line 33.

Algorithm 2: Frequency scaling with stochastic sleep mode distribution

Input: i) task τ_i; ii) operating frequency $f_k(\tau_i)$;
Output: i) updated operating frequency $f_k(\tau_i)$; ii) sleep time $t'^{slp}_{i,k}$;

1 $\iota = \ell_k$; // The highest frequency of processor Θ_k is f_{k,ℓ_k}.
2 **while** $TS^{end}_{i,k,\iota} > T^{max}$ **do**
3 **if** $\iota \neq 1$ **then**
4 $\iota \leftarrow \iota - 1$;
5 $f_k(\tau_i) \leftarrow f_{k,\iota}$; // Frequency scaling is utilized.
6 $t'^{slp}_{i,k} \leftarrow 0$;
7 update the ending temperature $TS^{end}_{i,k,\iota}$ of using Eq. (35);
8 **end**
9 **else**
10 derive $E[T^{safe}_{i,k,1}]$ and $Var[T^{safe}_{i,k,1}]$ using Eq. (40);
11 estimate safe temperature $TS^{safe}_{i,k,1}$ using Eq. (41);
12 calculate estimated sleep time $t'^{slp}_{i,k,1}$ using Eq. (42);
13 $t'^{slp}_{i,k} \leftarrow t'^{slp}_{i,k,1}$; // Stochastic sleep mode distribution is adopted when frequency scaling fails to work.
14 **end**
15 **end**

Algorithm 2 describes our proposed thermal management scheme. As shown in the algorithm, when the estimated ending temperature of a task exceeds the temperature limit, frequency scaling is firstly utilized to adjust the operating frequency of this task such that a lower on-chip peak temperature is achieved. When frequency scaling fails to guarantee that the temperature constraint is not violated due to the bad thermal profiles, the stochastic sleep mode distribution is then adopted. To be specific, when the estimated ending temperature TS^{end}_{i,k,ℓ_k} of task τ_i executing on processor Θ_k at maximal frequency f_{k,ℓ_k} exceeds the temperature limit T^{max} (i.e., $TS^{end}_{i,k,\ell_k} > T^{max}$), frequency scaling is first utilized to find the maximal ι ($\iota \geq 1$) such that the temperature constraint $TS^{end}_{i,k,\iota} \leq T^{max}$ is met. However, when ι is scaled down to 1, the estimated ending temperature $TS^{end}_{i,k,1}$ of task τ_i executing on processor Θ_k at minimal frequency $f_{k,1}$ may be still higher than T^{max} (i.e., $TS^{end}_{i,k,1} > T^{max}$). To prevent the temperature constraint from being violated, the stochastic sleep mode distribution is then adopted. Using Eq. (42), the stochastic sleep mode distribution calculates the estimated sleep period $t'^{slp}_{i,k,1}$ that processor Θ_k needs to stay in the sleep mode to reduce temperature from initial temperature $T^{init}_{i,k,1}$ to safe temperature $TS^{safe}_{i,k,1}$ before task τ_i execution. After $t'^{slp}_{i,k,1}$ time, task τ_i is immediately executed at the minimal frequency $f_{k,1}$. According to the definition of safe temperature, when the initial temperature $T^{init}_{i,k,1}$ is scaled down to the safe temperature $TS^{safe}_{i,k,1}$ and the execution of task τ_i starts immediately, the temperature constraint will definitely not be violated.

During task execution at runtime, note that it is sufficient to successfully complete only one replica of any task. Therefore, when one replica of task completes and no fault is detected by using the acceptance test approach [22], we cancel other replicas of that task to lower processor temperature. Thanks to the benefits of canceling these redundant replicas of task τ_i executed on processor Θ_k, once i) processor Θ_k is in idle mode, ii) the input data of task τ_{i+1} to be executed next on processor Θ_k is ready, and iii) the thermal modeling tool

HotSpot [32] utilized in our simulations monitors that the current temperature of processor Θ_k is not higher than task τ_{i+1}'s calculated initial temperature derived by Algorithm 1, task τ_{i+1} starts its execution immediately instead of performing its execution after all these redundant replicas of task τ_i are executed.

VI. Evaluations

A. Experimental Settings

A heterogeneous multiprocessor system [4] composed of five machines is adopted in our experimental simulations. The five machines are equipped with an Intel Core Duo processor supporting 4 frequencies, an Intel Xton processor supporting 3 frequencies, an AMD Athlon processor supporting 3 frequencies, a TI DSP processor supporting 2 frequencies, and a SPARC64 processor supporting 2 frequencies, respectively. All the parameters of processor frequencies can be found in [4]. The average fault arrival rate at the maximal voltage supply and operating frequency of a processor is randomly selected from the interval $[10^{-8}, 10^{-7}]$, and the parameter d of a processor indicating the sensitivity of fault rates to frequency scaling is randomly selected from the interval $[1, 10]$ [22]. The tool HotSpot [32] is utilized to capture thermal profiles. The average of thermal resistance and thermal capacitance of processors are assumed to be 0.80 K/W and 340 J/K, respectively [23], and the variance of the thermal resistance and thermal capacitance of a processor are deemed to be 0.5 and 50.0, respectively.

Two sets of simulation experiments are carried out. In the first set of simulations, DAGs of synthetic applications are generated by using the tool of Task Graphs for Free (TGFF) [20]. Five synthetic applications are tested, and the number of tasks in applications is in the range between 100 and 300, in steps of 50. The task activity factors μ are uniformly distributed in the interval $[0.4, 1]$ [16]. The expected value and variance of every replica's execution time and inter-task communication time in an application are specified in the intervals of $[1, 5]$ and $(0, 5]$ respectively, according to the processor's capacity [4], [5]. The execution times of multiple replicas of the same task are considered to follow the same normal distribution. In the second set of simulations, real-world multimedia applications [4] mpegplay, madplay, tmndec, and toast, are utilized to validate the proposed scheme. The DAGs of the four applications are generated by using the DAG generation tool GGen [38]. The replica parameter tables and inter-task communication times of the four applications are obtained by profiling task off-line traces [4], [26], [27]. The same settings of reliability and temperature constraints are adopted for the synthetic applications and real-world benchmarks. That is, the reliability requirements of applications are specified in the interval of [0.85, 0.99999] [22]. The ambient temperature T^{amb} of each processor is set to 30°C. The temperature limit T^{max} takes the values of 70°C and 80°C. The weighting parameter α takes values of 0.3 and 0.8. We use PRSD1 to denote our proposed approach with $\alpha = 0.3$, and PRSD2 to represent our proposed method with $\alpha = 0.8$.

In the two sets of simulations, we compare makespan, peak temperature, reliability, and feasibility achieved by the proposed scheme with that of existing benchmarking schemes. We perform 1000 experiments to obtain the average of the simulation data. The benchmarking schemes for comparison are described as below.

- **LPRE** [36] is an Integer Linear Programming (ILP) based framework to maximize system reliability and minimize energy consumption. Task replication technique is utilized to provide fault-tolerance while the stochastic task execution times and temperature constraint are ignored.
- **SDLS** [5] is a stochastic dynamic level scheduling scheme that aims to minimize the expected system makespan, but it doesn't take the reliability and temperature constraints into account.
- **DDME** [37] is an approach to reduce system makespan and energy consumption. However, it ignores the uncertainty in task execution times, and constraints of temperature and reliability.
- **WPMP** [34] also introduces a weighting parameter α to balance makespan and peak temperature as our proposed scheme. However, it ignores both the stochastic behavior and reliability requirements of tasks. We use WPMP1 and WPMP2 to denote the algorithm under the setting of $\alpha = 0.3$, and $\alpha = 0.8$, respectively.
- **RMSR** [19] is a replication-based scheduling algorithm which aims at maximizing system reliability. It doesn't consider the temperature constraint of the system and the stochastic behavior of task execution times.
- **TASA** [33] is a scheduling algorithm for peak temperature reduction with stochastic workloads. Reliability requirements of tasks are ignored in this algorithm.
- **TARS** [35] is a rotation scheduling approach for peak temperature minimization. The stochastic behavior of task execution times and reliability requirements of tasks are not taken into consideration in this approach.

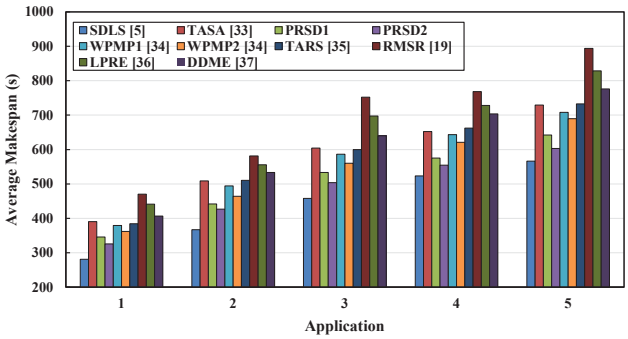

Fig. 2: Average makespan of synthetic applications under $T^{max} = 70$°C.

B. Results for Synthetic Applications

1) Comparison under $T^{max} = 70$°C): Fig. 2 demonstrates the average makespan of synthetic applications under $T^{max} = 70$°C. As shown in the figure, the average makespan achieved by the proposed scheme is much lower than that of the benchmarking schemes TASA, WPMP1, WPMP2, TARS, RMSR, LPRE, and DDME for the five synthetic applications

TABLE III: Average reliability of synthetic applications under $T^{max} = 70°C$ ($\overline{RG}$: Average Reliability Goal).

App.	$\overline{RG}$	SDLS [5]	TASA [33]	PRSD1	PRSD2	WPMP1 [34]	WPMP2 [34]	TARS [35]	RMSR [19]	LPRE [36]	DDME [37]
1	0.935	0.730	0.500	0.950	0.945	0.670	0.620	0.515	0.987	0.965	0.600
2	0.947	0.620	0.425	0.959	0.953	0.573	0.520	0.438	0.995	0.972	0.505
3	0.943	0.525	0.356	0.967	0.959	0.455	0.417	0.367	0.993	0.985	0.383
4	0.936	0.455	0.325	0.955	0.949	0.400	0.378	0.335	0.997	0.978	0.350
5	0.932	0.375	0.295	0.952	0.943	0.356	0.320	0.305	0.996	0.983	0.300
Avg.	0.939	0.541	0.380	0.957	0.950	0.491	0.451	0.392	0.994	0.977	0.428

TABLE IV: Feasibility of synthetic applications under $T^{max} = 70°C$.

App.	SDLS [5]	TASA [33]	PRSD1	PRSD2	WPMP1 [34]	WPMP2 [34]	TARS [35]	RMSR [19]	LPRE [36]	DDME [37]
1	50.1%	25.5%	100.0%	100.0%	50.7%	55.2%	26.3%	100.0%	100.0%	46.2%
2	35.8%	20.1%	100.0%	100.0%	40.2%	43.7%	20.7%	45.1%	50.1%	35.6%
3	26.2%	15.2%	100.0%	100.0%	25.5%	32.3%	15.6%	32.3%	35.5%	22.3%
4	20.5%	9.8%	100.0%	100.0%	20.7%	27.8%	10.1%	25.2%	29.7%	20.2%
5	15.1%	5.0%	100.0%	100.0%	15.3%	22.1%	5.1%	17.6%	19.2%	15.2%
Avg.	29.5%	15.1%	100.0%	100.0%	30.5%	36.2%	15.6%	44.0%	46.9	27.9%

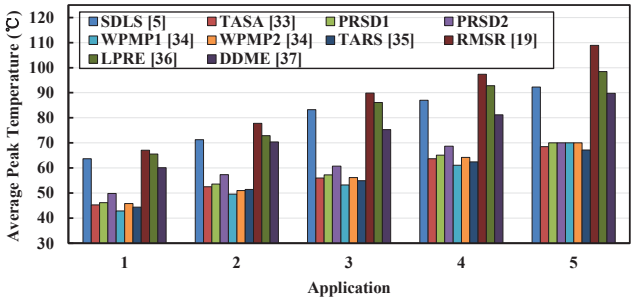

Fig. 3: Average peak temperature of synthetic applications under $T^{max} = 70°C$.

under test. For instance, for application 4, compared to TASA, WPMP1, WPMP2, TARS, RMSR, LPRE, and DDME, the proposed method PRSD1 reduces the average makespan by 11.8%, 10.6%, 7.4%, 13.1%, 25.1%, 21.0%, and 18.2%, respectively; the proposed method PRSD2 reduces the average makespan by 14.9%, 13.8%, 10.7%, 16.3%, 27.8%, 23.8%, and 21.2%, respectively. Fig. 3 plots the average peak temperature of synthetic applications under $T^{max} = 70°C$. It has been shown in the figure that the proposed scheme can always satisfy the temperature constraint. In addition, the proposed scheme can achieve a better average peak temperature reduction. Take application 1 as an example, the proposed approach PRSD1 reduces the average peak temperature by 27.4%, 31.1%, 29.6% and 23.1%, and the proposed approach PRSD2 reduces the average peak temperature by 21.7%, 25.6%, 24.0%, and 17.0% compared to SDLS, RMSR, LPRE, and DDME, respectively.

Table III compares the average reliability of synthetic applications under $T^{max} = 70°C$. The reliability of an application is calculated as the ratio of the sum of task reliability to the number of tasks in the application. As shown in the table, the average reliability achieved by our proposed scheme is always higher than the average reliability goal. Meanwhile, the table also reveals that the average reliability achieved by our proposed scheme is second to that of benchmarking methods RMSR and LPRE. Table IV lists the feasibility of synthetic applications under $T^{max} = 70°C$. The feasibility of an application is calculated as the ratio of the number of application instances that can be feasibly scheduled under reliability and temperature constraints to the total number of application instances (i.e., 1000). From the results shown in the table, we can easily find that the proposed scheme achieves 100% feasibility.

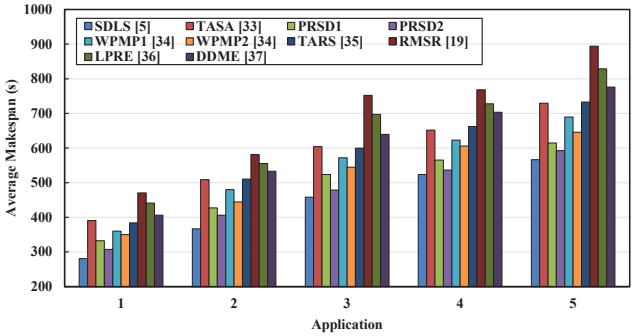

Fig. 4: Average makespan of synthetic applications under $T^{max} = 80°C$.

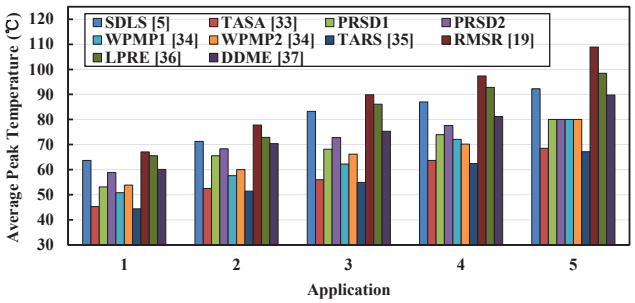

Fig. 5: Average peak temperature of synthetic applications under $T^{max} = 80°C$.

2) Comparison under $T^{max} = 80°C$): Fig. 4 demonstrates the average makespan of synthetic applications under $T^{max} = 80°C$. We can see from the figure that our proposed scheme can achieve better performance in terms of makespan. For instance, for application 4, the proposed method PRSD1 reduces the average makespan by 13.3%, 9.2%, 6.7%, 14.7%, 26.4%, 22.4%, and 19.7%, and the proposed method PRSD2 reduces the average makespan by

TABLE V: Average reliability of synthetic applications under $T^{max} = 80°C$ ($\overline{RG}$: Average Reliability Goal).

App.	$\overline{RG}$	SDLS [5]	TASA [33]	PRSD1	PRSD2	WPMP1 [34]	WPMP2 [34]	TARS [35]	RMSR [19]	LPRE [36]	DDME [37]
1	0.935	0.730	0.500	0.955	0.950	0.681	0.639	0.515	0.987	0.965	0.600
2	0.947	0.620	0.425	0.964	0.961	0.583	0.536	0.438	0.995	0.972	0.505
3	0.943	0.525	0.356	0.972	0.964	0.463	0.430	0.367	0.993	0.985	0.383
4	0.936	0.455	0.325	0.961	0.955	0.407	0.389	0.335	0.997	0.978	0.350
5	0.932	0.375	0.295	0.958	0.950	0.363	0.330	0.305	0.996	0.983	0.300
Avg.	0.939	0.541	0.380	0.962	0.956	0.499	0.465	0.392	0.994	0.977	0.428

TABLE VI: Feasibility of synthetic applications under $T^{max} = 80°C$.

App.	SDLS [5]	TASA [33]	PRSD1	PRSD2	WPMP1 [34]	WPMP2 [34]	TARS [35]	RMSR [19]	LPRE [36]	DDME [37]
1	50.1%	25.5%	100.0%	100.0%	52.2%	56.1%	26.3%	100.0%	100.0%	46.2%
2	35.8%	20.1%	100.0%	100.0%	41.4%	44.4%	20.7%	100.0%	100.0%	35.6%
3	26.2%	15.2%	100.0%	100.0%	26.3%	32.8%	15.6%	32.3%	35.5%	22.3%
4	20.5%	9.8%	100.0%	100.0%	21.2%	28.3%	10.1%	25.2%	29.7%	20.2%
5	15.1%	5.0%	100.0%	100.0%	15.7%	22.5%	5.1%	17.6%	19.2%	15.2%
Avg.	29.5%	15.1%	100.0%	100.0%	31.4%	36.8%	15.6%	55.0%	56.9%	27.9%

17.7%, 13.8%, 11.4%, 19.0%, 30.1%, 26.3%, and 23.7% compared to TASA, WPMP1, WPMP2, TARS, RMSR, LPRE, and DDME, respectively. Meanwhile, Fig. 4 also demonstrates that the makespan achieved by our proposed scheme is inferior to that of SDLS. The average peak temperature achieved by various schemes under $T^{max} = 80°C$ is compared in Fig. 5. As shown in the figure that the peak temperature achieved by our proposed scheme is always below the temperature limit 80°C. In addition, the proposed scheme can achieve a better average peak temperature reduction. Take application 1 as an example, the proposed method PRSD1 reduces the average peak temperature by 16.6%, 20.8%, 19.0% and 11.6%, and the proposed method PRSD2 reduces the average peak temperature by 7.5%, 12.2%, 10.3%, and 2.1% compared to SDLS, RMSR, LPRE, and DDME, respectively.

Table V presents the average reliability of synthetic applications under $T^{max} = 80°C$. From the table, we can see that the average reliability achieved by our proposed scheme is always higher than the average reliability goal for the five applications under test. Table VI compares the feasibility of synthetic applications achieved by benchmarking methods and the proposed scheme under $T^{max} = 80°C$. We can see from the table that the proposed scheme achieves better feasibility.

From the above analyses in Section VI-B1 and VI-B2, we can draw the below observations.

- SDLS outperforms our proposed scheme in terms of makespan, but it achieves a higher average peak temperature. The reason is that SDLS ignores the reliability and temperature constraints, which are antagonistic to the objective of reducing makespan.
- Our proposed scheme achieves a better tradeoff between makespan and peak temperature by introducing a weighting parameter α. Therefore, the scheduler can adjust the α to meet varying system design requirements.
- Our proposed scheme achieves a higher average peak temperature compared to WPMP, TASA, and TARS. This is because the three schemes fail to consider the stochastic characteristics of tasks, which results in unnecessary idle time to lower the peak temperature.
- Our proposed scheme is inferior to RMSR and LPRE in terms of average reliability. This is because our scheme is designed to meet reliability constraints rather than maximize reliability.
- Our proposed scheme achieves better feasibility, which is due to the effectiveness of the adopted temperature- and reliability-aware techniques.

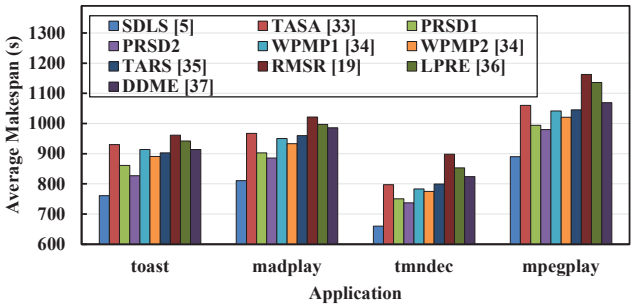

Fig. 6: Average makespan of real-world applications under $T^{max} = 70°C$.

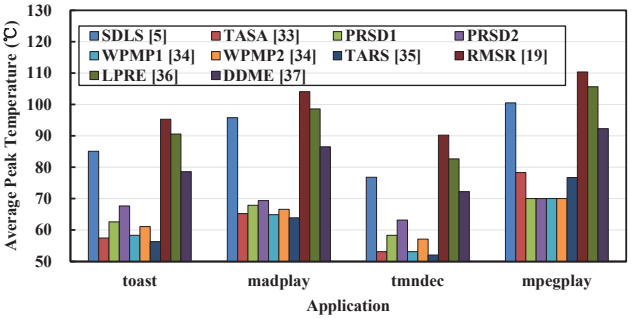

Fig. 7: Average peak temperature of real-world applications under $T^{max} = 70°C$.

C. Results for Real-World Benchmarks

1) Comparison under $T^{max} = 70°C$): We also validate the effectiveness of our proposed scheme for real-world applications. Fig. 6 plots the average makespan of four benchmarks toast, madplay, tmndec, and mepegplay under $T^{max} = 70°C$. Similar to the results shown in Section VI-B1, the makespan achieved by the proposed scheme is smaller (up to 18.2%)

TABLE VII: Average reliability of real-world applications under $T^{max} = 70°C$ ($\overline{RG}$: Average Reliability Goal).

App.	$\overline{RG}$	SDLS [5]	TASA [33]	PRSD1	PRSD2	WPMP1 [34]	WPMP2 [34]	TARS [35]	RMSR [19]	LPRE [36]	DDME [37]
toast	0.929	0.702	0.455	0.943	0.933	0.645	0.609	0.469	0.996	0.982	0.502
madplay	0.935	0.603	0.389	0.955	0.941	0.553	0.504	0.400	0.989	0.974	0.421
tmndec	0.944	0.625	0.402	0.962	0.950	0.578	0.526	0.414	0.997	0.989	0.452
mpegplay	0.930	0.502	0.355	0.946	0.938	0.455	0.400	0.366	0.999	0.997	0.389
Avg.	0.935	0.608	0.400	0.952	0.941	0.558	0.510	0.412	0.995	0.986	0.441

TABLE VIII: Feasibility of real-world applications under $T^{max} = 70°C$.

App.	SDLS [5]	TASA [33]	PRSD1	PRSD2	WPMP1 [34]	WPMP2 [34]	TARS [35]	RMSR [19]	LPRE [36]	DDME [37]
toast	27.6%	19.8%	100.0%	100.0%	30.5%	30.8%	20.0%	40.2%	41.2%	20.3%
madplay	25.9%	16.2%	100.0%	100.0%	26.1%	26.4%	16.4%	35.3%	35.5%	16.9%
tmndec	28.1%	17.1%	100.0%	100.0%	27.2%	27.4%	17.3%	37.2%	38.1%	17.6%
mpegplay	20.5%	13.2%	100.0%	100.0%	21.3%	21.5%	13.4%	28.9%	29.9%	14.1%
Avg.	25.5%	16.6%	100.0%	100.0%	26.3%	26.5%	16.8%	35.4%	36.2%	17.2%

TABLE IX: Average reliability of real-world applications under $T^{max} = 80°C$ ($\overline{RG}$: Average Reliability Goal).

App.	$\overline{RG}$	SDLS [5]	TASA [33]	PRSD1	PRSD2	WPMP1 [34]	WPMP2 [34]	TARS [35]	RMSR [19]	LPRE [36]	DDME [37]
toast	0.929	0.702	0.455	0.962	0.942	0.661	0.627	0.469	0.996	0.982	0.502
madplay	0.935	0.603	0.389	0.966	0.955	0.570	0.518	0.400	0.989	0.974	0.421
tmndec	0.944	0.625	0.402	0.971	0.959	0.594	0.542	0.414	0.997	0.989	0.452
mpegplay	0.930	0.502	0.355	0.948	0.944	0.469	0.410	0.366	0.999	0.997	0.389
Avg.	0.935	0.608	0.400	0.962	0.950	0.573	0.524	0.412	0.995	0.986	0.441

TABLE X: Feasibility of real-world applications under $T^{max} = 80°C$.

App.	SDLS [5]	TASA [33]	PRSD1	PRSD2	WPMP1 [34]	WPMP2 [34]	TARS [35]	RMSR [19]	LPRE [36]	DDME [37]
toast	27.6%	19.8%	100.0%	100.0%	30.8%	31.1%	20.0%	40.2%	41.2%	20.3%
madplay	25.9%	16.2%	100.0%	100.0%	26.4%	26.6%	16.4%	35.3%	35.5%	16.9%
tmndec	28.1%	17.1%	100.0%	100.0%	27.5%	27.7%	17.3%	37.2%	38.1%	17.6%
mpegplay	20.5%	13.2%	100.0%	100.0%	21.5%	21.7%	13.4%	28.9%	29.9%	14.1%
Avg.	25.5%	16.6%	100.0%	100.0%	26.5%	26.8%	16.8%	35.4%	36.2%	17.2%

than that of TASA, WPMP1, WPMP2, TARS, RMSR, LPRE, and DDME. Fig. 7 compares the average peak temperature achieved by benchmarking schemes and the proposed scheme. As shown in the figure, our proposed scheme can prevent the temperature limit (i.e., $T^{max} = 70°C$) from being violated for four benchmarks. Similar to the results shown in Section VI-B1, Table VII reveals that our proposed scheme achieves better (up to 182%) average reliability improvement than benchmarking schemes SDLS, TASA, WPMP1, WPMP2, TARS, and DDME. Table VIII compares the feasibility achieved by bencharking schemes and the proposed scheme under $T^{max} = 70°C$. We can see from the table that the feasibility achieved by the proposed scheme is 100%, which is the highest value among these algorithms.

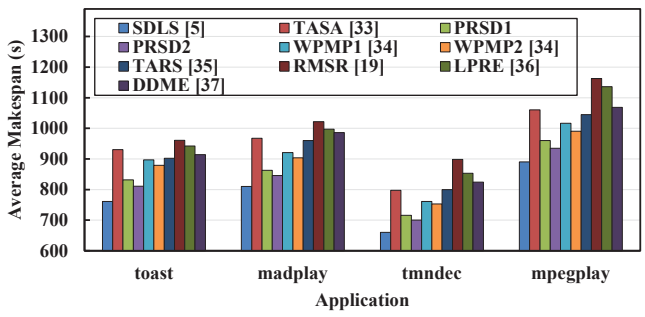

Fig. 8: Average makespan of real-world applications under $T^{max} = 80°C$.

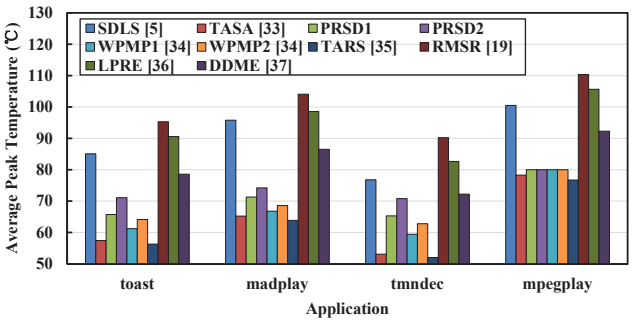

Fig. 9: Average peak temperature of real-world applications under $T^{max} = 80°C$.

2) Comparison under $T^{max} = 80°C$): The average makespan of the four benchmarks under $T^{max} = 80°C$ is presented in Fig. 8. As shown in the figure, the average makespan achieved by the proposed scheme can be reduced by up to 22.1%. Fig. 9 plots the average peak temperature achieved by benchmarking schemes and the proposed scheme. The results in the figure show that the peak temperature of the four benchmarks can be lowered by up to 27.5% using the proposed scheme. Table IX shows the average reliability of the four real-world applications under $T^{max} = 80°C$. As shown in the table, the average reliability achieved by our proposed approaches PRSD1 and PRSD2 is higher than the average reliability goal. Table X lists the feasibility achieved by bencharking schemes and the proposed scheme under

$T^{max} = 80°C$. As shown in the table, the feasibility achieved by the proposed scheme is the highest among these schemes.

VII. Conclusions

In this paper, we study the problem of makespan optimization jointly considering reliability, temperature, and stochastic characteristics of precedence-constrained tasks. We first model two types of affinities: one is the affinity of a task for processors with respect to schedule lengths and the other is the affinity of a task for processors with regard to chip thermal profiles. The task reliability, inter-task precedence, and stochastic characteristics of task execution time and communication time are taken into account throughout the modeling process. After the two types of affinities are modeled, we then join together them by introducing a weighting parameter. We finally propose an affinity-driven task scheduling heuristic that assigns a task to the processor with the highest combined affinities. Extensive simulations have been performed to validate the effectiveness of the proposed scheme. Simulation results show that the proposed scheme can achieve up to 30.1% reduction in makespan without violating temperature and reliability constraints compared to benchmarking schemes.

References

[1] J. Chang and G. Sohi, "Cooperative cache partitioning for chip multiprocessors," *in Proceedings of the ACM International Conference on Supercomputing*, pp. 402–412, 2014.

[2] J. Zhou, K. Cao, P. Pong, T. Wei, M. Chen, G. Zhang, J. Yan, and Y. Ma, "Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms," *Journal of Systems and Software*, vol. 133, pp. 1–16, 2017.

[3] X. Tang, K. Li, G. Liao, K. Fang, and F. Wu, "A stochastic scheduling algorithm for precedence constrained tasks on grid," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1083–1091, 2011.

[4] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2867–2876, 2014.

[5] K. Li, X. Tang, B. Veeravalli, and K. Li, "Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 191–204, 2015.

[6] W. Zheng and R. Sakellariou, "Stochastic DAG scheduling using a Monte Carlo approach," *Journal of Parallel and Distributed Computing*, vol. 73, no. 12, pp. 1673–1689, 2013.

[7] J. Zhou, T. Wei, M. Chen, J. Yan, X. Hu, and Y. Ma, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 8, pp. 1269–1282, 2016.

[8] A. Girault, E. Saule, and D. Trystram, "Reliability versus performance for critical applications," *Journal of Parallel and Distributed Computing*, vol. 69, no. 3, pp. 326–336, 2009.

[9] O. Sathappan, P. Chitra, P. Venkatesh, and M. Prabhu, "Modified genetic algorithm for multiobjective task scheduling on heterogeneous computing system," *International Journal of Information Technology, Communications and Convergence*, vol. 1, no. 2, pp. 146–158, 2011.

[10] X. Wang, C. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1124–1134, 2011.

[11] G. Aupy, A. Benoit, and Y. Robert, "Energy-aware scheduling under reliability and makespan constraints," *in Proceedings of the IEEE International Conference on High Performance Computing*, pp. 1–10, 2012.

[12] A. Chavan, M. Alghamdi, X. Jiang, X. Qin, M. Qiu, M. Jiang, and J. Zhang, "TIGER: Thermal-aware file assignment in storage clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 558–573, 2016.

[13] K. Skadron, "Hybrid architectural dynamic thermal management," *in Proceedings of the IEEE Design, Automation and Test in Europe Conference and Exhibition*, pp. 10–15, 2004.

[14] A. Kumar, S. Li, L. Peh, and N. Jha, "HybDTM: a coordinated hardware-software approach for dynamic thermal management," *in Proceedings of the ACM Design Automation Conference*, pp. 548–553, 2006.

[15] Y. Ge, P. Malani, and Q. Qiu, "Distributed task migration for thermal management in many-core systems," *in Proceedings of the ACM Design Automation Conference*, pp. 579–584, 2010.

[16] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint," *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 2s, pp. 1–22, 2014.

[17] B. Barrefors, Y. Lu, S. Saha, B. Al-Hashimi, and G. Merrett, "A novel thermal-constrained energy-aware partitioning algorithm for heterogeneous multiprocessor real-time systems," *in Proceedings of the IEEE International Performance Computing and Communications Conference*, pp. 1–8, 2014.

[18] D. Li and J. Wu, "Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 810–823, 2015.

[19] S. Wang, K. Li, J. Mei, J. Xiao, and K. Li, "A reliability-aware task scheduling algorithm based on replication on heterogeneous computing systems," *Journal of Grid Computing*, vol. 15, no. 1, pp. 23–39, 2017.

[20] R. Dick, D. Rhodes, and W. Wolf, "TGFF: task graphs for free," *in Proceedings of the IEEE International Workshop on Hardware/Software Codesign*, pp. 97–101, 1998.

[21] A. Ejlali, B. Al-Hashimi, and P. Eles, "Low-energy standby-sparing for hard real-time systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 3, pp. 329–342, 2012.

[22] M. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 813–825, 2017.

[23] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature constraint hard real-time periodic tasks," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 329–339, 2010.

[24] S. Saha, Y. Lu, and J. Deogun, "Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems," *in Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 41–50, 2012.

[25] N. Johnson, S. Kotz, and N. Balakrishnan, "Lognormal distributions," *Continuous Univariate Distributions*, vol. 1, pp. 601–606, 1994.

[26] C. Lin, Y. Syu, C. Chang, J. Wu, P. Liu, P. Cheng, and W. Hsu, "Energy-efficient task scheduling for multi-core platforms with per-core DVFS," *Journal of Parallel and Distributed Computing*, vol. 86, pp. 71–81, 2015.

[27] F. Doray and M. Dagenais, "Diagnosing performance variations by comparing multi-level execution traces," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 462–474, 2017.

[28] G. McPherson, "Statistics in scientific investigation: its basis, application, and interpretation," *Springer Science & Business Media*, 2013.

[29] C. Clark, "The greatest of a finite set of random variables," *Operations Research*, vol. 9, no. 2, pp. 145–162, 1961.

[30] D. Hinkley, "On the ratio of two correlated normal random variables," *Biometrika*, vol. 56, no. 3, pp. 635–639, 1969.

[31] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1668–1681, 2012.

[32] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, 2004.

[33] S. Liu and M. Qiu, "Thermal-aware scheduling for peak temperature reduction with stochastic workloads," *in Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 59–62, 2010.

[34] K. Cao, J. Zhou, M. Yin, T. Wei, and M. Chen, "Static thermal-aware task assignment and scheduling for makespan minimization in heterogeneous real-time MPSoCs," *in Proceedings of the IEEE International Symposium on System and Software Reliability*, pp. 111–118, 2016.

[35] J. Li, M. Qiu, J. Niu, L. Yang, Y. Zhu, and Z. Ming, "Thermal-aware task scheduling in 3d chip multiprocessor with real-time constrained workloads," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 2, pp. 760–766, 2013.

[36] S. Tosun, "Energy-and reliability-aware task scheduling onto heterogeneous MPSoC architectures," *The Journal of Supercomputing*, vol. 62, no. 1, pp. 265–289, 2012.

[37] J. Singh, A. Gujral, H. Singh, J. Singh, and N. Auluck, "Energy aware scheduling on heterogeneous multiprocessors with DVFS and duplication," *in Proceedings of the IEEE International Conference on*

Parallel and Distributed Computing, Applications and Technologies, pp. 105–112, 2016.

[38] D. Cordeiro, G. Mounie, S. Perarnau, and D. Trystram, "Random graph generation for scheduling simulations," *in Proceedings of the ACM International Conference on Simulation Tools and Techniques*, 2010.

Kun Cao is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China.

His current research interests are in the areas of 3D ICs, high performance computing, heterogeneous multiprocessor systems, and cyber physical systems. He received the Reviewer Award from Journal of Circuits, Systems, and Computers, in 2016.

Junlong Zhou (S'15-M'17) received the Ph.D. degree in Computer Science from East China Normal University, Shanghai, China, in 2017. He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, during 2014-2015. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include real-time embedded systems, cloud computing, and cyber physical systems. Dr. Zhou is an Active Reviewer of several international journals, including IEEE Transactions on Computers, IEEE Transactions on CAD of Integrated Circuits and Systems, and IEEE Transactions on Industrial Informatics. He received the Reviewer Award from Journal of Circuits, Systems, and Computers, in 2016. Dr. Zhou has been an Associate Editor for the Journal of Circuits, Systems, and Computers since 2017.

Peijin Cong received the B.S. degree from the Department of Computer Science and Technology, East China Normal University, shanghai, China, in 2016. She is currently pursuing the master degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. Her current research interest is in the area of power management in mobile devices.

Liying Li received the B.S. degree from the Department of Computer Science and Technology, East China Normal University, shanghai, China, in 2017. She is currently pursuing the master degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. Her current research interests are in the areas of cyber physical systems and IoT resource management.

Tongquan Wei (M'11) received his Ph.D. degree in Electrical Engineering from Michigan Technological University in 2009. He is currently an Associate Professor in the Department of Computer Science and Technology at the East China Normal University. His research interests are in the areas of Internet of Things, real-time embedded systems, green and reliable computing, parallel and distributed systems, and cloud computing. He serves as a Regional Editor for Journal of Circuits, Systems, and Computers since 2012. He is a member of the IEEE.

Mingsong Chen (S'08-M'11) received the B.S. and M.E. degrees from Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006 respectively, and the Ph.D. degree in Computer Engineering from the University of Florida, Gainesville, in 2010. He is currently a full Professor with the Department of Embedded Software and Systems of East China Normal University. His research interests are in the area of design automation of cyber-physical systems, formal verification techniques and mobile cloud computing. He is a member of the IEEE.

Shiyan Hu (SM'10) received his Ph.D. in Computer Engineering from Texas A&M University in 2008. He is an Associate Professor at Michigan Tech. where he is Director of Center for Cyber-Physical Systems and Associate Director of Institute of Computer and Cyber systems. He has been a Visiting Professor at IBM Research (Austin) in 2010, and a Visiting Associate Professor at Stanford University from 2015 to 2016. His research interests include Cyber-Physical Systems, Cybersecurity, Computer-Aided Design of VLSI Circuits, and Embedded Systems, where he has published more than 100 refereed papers.

He is an ACM Distinguished Speaker, an IEEE Computer Society Distinguished Visitor, an invited participant for U.S. National Academy of Engineering Frontiers of Engineering Symposium, a recipient of National Science Foundation (NSF) CAREER Award, a recipient of ACM SIGDA Richard Newton DAC Scholarship (as the faculty advisor), and a recipient of JSPS Faculty Invitation Fellowship. He is the Chair for IEEE Technical Committee on Cyber-Physical Systems. He serves as an Associate Editor for IEEE Transactions on Computer-Aided Design, IEEE Transactions on Industrial Informatics, and IEEE Transactions on Circuits and Systems. He is also a Guest Editor for 7 IEEE/ACM Transactions such as IEEE Transactions on Computers and IEEE Transactions on Computer-Aided Design. He has served as chairs, TPC chairs, TPC track chairs and TPC members for numerous conferences.

Xiaobo Sharon Hu (S'85-M'89-SM'02-F'16) received the B.S. degree from Tianjin University, Tianjin, China, the M.S. degree from the New York University Tandon School of Engineering, Brooklyn, NY, USA, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA. She is currently a Professor with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA. She has authored more than 250 papers in the related areas. Her current research interests include real-time embedded systems, low-power system design, and computing with emerging technologies.