The Journal of Systems and Software 133 (2017) 1–16

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms☆

Junlong Zhou[a,b], Kun Cao[a], Peijin Cong[a], Tongquan Wei[a,*], Mingsong Chen[c], Gongxuan Zhang[b], Jianming Yan[d], Yue Ma[e]

[a] Department of Computer Science and Technology, East China Normal University, Shanghai, 200062, China
[b] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, 210094, China
[c] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, 200062, China
[d] Meituan.com Corporation, Beijing, 100102, China
[e] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, 46656, USA

ARTICLE INFO

Article history:
Received 21 January 2017
Revised 20 July 2017
Accepted 22 July 2017
Available online 29 July 2017

Keywords:
Makespan minimization
Reliability
Temperature
Task assignment and scheduling

ABSTRACT

We study the problem of scheduling tasks onto a heterogeneous multi-core processor platform for makespan minimization, where each cluster on the platform has a probability of failure governed by an exponential law and the processor platform has a thermal constraint specified by a peak temperature threshold. The goal of our work is to design algorithms that optimize makespan under the constraints of reliability and temperature. We first provide a mixed-integer linear programming (MILP) formulation for assigning and scheduling independent tasks with reliability and temperature constraints on the heterogeneous platform to minimize the makespan. However, MILP takes exponential time to finish. We then propose a two-stage heuristic that determines the assignment, replication, operating frequency, and execution order of tasks to minimize the makespan while satisfying the real-time, reliability, and temperature constraints based on the analysis of the effects of task assignment on makespan, reliability, and temperature. We finally carry out extensive simulation experiments to validate our proposed MILP formulation and two-stage heuristic. Simulation results demonstrate that the proposed MILP formulation can achieve the best performance in reducing makespan among all the methods used in the comparison. The results also show that the proposed two-stage heuristic has a close performance as the representative existing approach ESTS and a better performance when compared to the representative existing approach RBSA, in terms of reducing makespan. In addition, the proposed two-stage heuristic has the highest feasibility as compared to RBSA and ESTS.

1. Introduction

With the rapid advance in semiconductor manufacturing technology and the ever increasing demand for high performance, multi-core processors have replaced single-core processors to become the main design paradigm for modern processors (Vajda, 2011). In the meantime, parallel computing that executes multiple operations or tasks on different processors simultaneously is adopted to satisfy the growing computational requirements. Parallel computing on multi-core processors improves system performance in terms of latency or throughput by minimizing task execution makespan, i.e., the latest completion time of all tasks on processors. However, this strategy results in excessive power densities (Culler et al., 1999). Higher power densities elevate chip temperature, which in turn downgrades system reliability and reduces energy efficiency due to increased leakage power (Kim et al., 2003). Although dynamic voltage and frequency scaling (DVFS) technique can be utilized for thermal control and energy saving, scaling down the speed of a processor increases the rate of cosmic ray-induced transient faults (Zhu et al., 2004). The probability of system failure due to transient fault increases exponentially, and this probability cannot be neglected in modern complex heterogeneous multi-core processor systems. Therefore, system performance, chip temperature and cosmic ray-induced transient faults interplay and need to be jointly investigated. In this paper, we focus on designing mechanisms that minimize task execution makespan under constraints of thermal and reliability target.

☆ This work was partially supported by Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000), Natural Science Foundation of China (Grant No. 61672230), and ECNU Outstanding Doctoral Dissertation Cultivation Plan of Action (Grant No. PY2015047).

* Corresponding author.
E-mail address: tqwei@cs.ecnu.edu.cn (T. Wei).

http://dx.doi.org/10.1016/j.jss.2017.07.032
0164-1212/

Makespan minimization problem has been a subject of continuing interest for researchers and practitioners during the past decades (Rajendran and Ziegler, 2004; Zhang et al., 2016; Li et al., 2014; Zheng and Sakellariou, 2013; Albers and Hellwig, 2016). Rajendran and Ziegler (2004) investigated the problem of permutation flowshop scheduling with the goal of minimizing the makespan, and proposed two ant-colony optimization based algorithms to solve the problem. Zhang et al. (2016) studied the problem of distributed workload dissemination for makespan minimization in disruption tolerant networks, and designed a centralized polynomial-time disseminating algorithm based on the shortest delay tree. A heuristic algorithm (Li et al., 2014) that specifically considers the stochastic characteristic of task execution time is presented to achieve a balance between schedule length (i.e., makespan) and energy consumption. A novel Monte Carlo based DAG scheduling approach (Zheng and Sakellariou, 2013) is developed to generate a static schedule that can minimize the expected makespan without incurring a prohibitively high time overhead. Unlike the static approaches (Rajendran and Ziegler, 2004; Zhang et al., 2016; Li et al., 2014; Zheng and Sakellariou, 2013) proposed for makespan minimization, Albers and Hellwig (2016) introduced an online algorithm that can dynamically minimize makespan with parallel schedules. However, none of the above work considers reliability.

Reliability is of utmost importance in multi-core 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 (Zhou et al., 2016a; Wei et al., 2012). However, makespan-aware scheduling techniques themselves do not address tolerance for system failure due to soft errors. Therefore, reliability issues need to be specifically dealt with in addition to the execution time or makespan. Considerable research efforts have been devoted to jointly handling makespan and reliability issues (Dongarra et al., 2007; Wang et al., 2011; Assayad et al., 2011; Aupy et al., 2012). Dongarra et al. (2007) studied the problem of scheduling task graphs on a set of heterogeneous resources with the bi-objective of minimizing makespan and maximizing reliability, and proposed an approach that can help the user choose a suitable makespan/reliability trade-off. Wang et al. (2011) improved the traditional genetic algorithm and developed a look-ahead genetic algorithm to optimize both the makespan and reliability of a workflow application in distributed computing environments. Assayad et al. (2011) presented an off-line scheduling heuristic to jointly optimize task schedule length, system reliability, and power consumption. Specifically, the heuristic uses active replication to minimize makespan and ensure system reliability, and employs DVFS to reduce power consumption. Aupy et al. (2012) solved the problem of energy minimization under the constraints of a prescribed bound on makespan and a reliability threshold by determining which task to re-execute and at which speed each execution of a task should be operated.

Note that all of the aforementioned methods ignore the effects of elevated operating temperature caused by the soaring increase in system power density on system. That is, a system will fall into the predicament of functional incorrectness, low reliability and even hardware failures if the operating temperature exceeds a certain threshold (Zhou et al., 2016b). Therefore, thermal management to avoid temperature-induced failures is also a significant and pressing research issue in modern computer systems, especially for embedded systems with limited cooling techniques. As far as we know, little investigation has been conducted in the literature on thermal management for makespan-aware multi-core systems. Recently, Hanumaiah and Vrudhula (2012) developed a temperature-aware DVFS-based approach for multi-core systems to optimize the makespan while satisfying timing and thermal constraints. They also provided a theoretical basis and analytical relations between speed, voltage, power, and temperature. However, reliability is not taken into account.

In this paper, we focus on designing a makespan-aware task scheduling scheme for heterogeneous multi-core processor systems under the reliability and temperature constraints. The scheme generates a makespan-optimum schedule that meets the design requirements by wisely determining the task assignment, the operating frequency of assigned tasks, the number of replicas for every task, and the execution order of tasks on the core. The major contributions of this paper are summarized as follows.

- We presented a MILP formulation for assigning and scheduling independent tasks with reliability and temperature constraints on a heterogeneous multi-core platform to minimize the makespan.
- We analyzed the effects of task assignment on makespan, reliability, and temperature to guide the design of our task assignment and scheduling heuristic.
- We proposed a static two-stage heuristic that first determines the assignment and replication of tasks for minimizing makespan under the constraint of task reliability, then determines the schedule of assigned tasks on the cores to satisfy the real-time and temperature constraint.
- We conducted extensive simulation experiments to validate the proposed MILP formulation and heuristic. Simulation results have demonstrated the efficacy of the proposed MILP formulation and heuristic.

The rest of the paper is organized as follows. Section 2 introduces the system model and problem definition. Section 3 describes the MILP formulation and Section 4 presents the proposed task assignment and scheduling scheme. Section 5 validates the effectiveness of the proposed scheme and Section 6 discusses the novelties of this paper when comparing with our previous works. Concluding remarks are given in Section 7.

2. System model and problem definition

This section first presents system models including the application model, the fault and reliability model, the architecture and execution model, and the temperature model, then describes the reliability and temperature constrained task assignment and scheduling problem for makespan minimization.

2.1. Application model

Consider a bag-of-tasks (BoT) application model that has been widely adopted in the literature (Braun et al., 2001; Gutierrez-Garcia and Sim, 2013; Cirne et al., 2003) and assumes tasks in the application are atomic, independent, and heterogeneous. Such applications can be found in the fields of astronomy, bioinformatics, and high energy physics. Typical examples are data mining, tomographic reconstructions, fractal calculations, and Monte Carlo simulations (Li et al., 2014). Tasks are independent in the sense that there is no precedence or communication among tasks. Due to the various safety demand for tasks, different tasks have different reliability requirements (Huang et al., 2014b). In addition, task executions must be finished before the deadline, which is commonly regarded as an QoS parameter (Li et al., 2014; Kim et al., 2007; Assayad et al., 2004; Netto and Buyya, 2009), especially for service-oriented systems. In such service-oriented systems, the system designer needs to take into account the concerns of both users and service provider. That is, from the perspective of users, all the services (tasks) need to be finished before the deadline such that the QoS requirement of users in terms of real time can be satisfied; from the perspective of service provider, the services need to be

completed as soon as possible such that more services can be provided (in other words, more profits can be gained).

Suppose a BoT application that consists of n tasks is denoted by $\mathcal{B} = \{\tau_1, \tau_2, \cdots, \tau_n\}$, and the characteristic of every task is described by a quadruple τ_i: $\{\mu_i, r_i, wc_i, D\}$ $(1 \le i \le n)$. μ_i (ranging in (0,1]) is the activity factor of task τ_i, which relates to task power consumption and is utilized to capture how intensively functional units are being used (Huang et al., 2014a). r_i is the lowest reliability requirement of task τ_i. In other words, the reliability level of task τ_i should be no less than r_i. wc_i denotes the worst-case execution time of task τ_i in cycles and D is the common deadline.

2.2. Fault and 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. Unlike permanent fault (resulting in hard error), transient fault is independent of temperature and would not lead to the breakdown of hardware devices. It is indispensable for many safety-related systems to have the capacity of providing a reliable execution in the presence of soft errors. Soft errors are typically modeled using an exponential distribution with an average arrival rate λ, which represents the expected number of failures that occur per second (Zhou and Wei, 2015; Zhao et al., 2009; Ejlali et al., 2012; Casas et al., 2017). It has been shown that the average rate λ highly depends on the processor frequency and can be modeled as

$$\lambda(f) = \lambda_0 10^{\frac{d(1-f)}{1-f_{\min}}}, \tag{1}$$

where λ_0 is the average fault rate at processor maximal frequency $f_{\max}$, and d (>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 and given the fault arrival rate $\lambda(f)$, then the reliability of task τ_i running at frequency f is expressed as

$$R_i(f) = e^{-\lambda(f)\frac{wc_i}{f}}, \tag{2}$$

where $\frac{wc_i}{f}$ is the worst-case execution time of τ_i at frequency f.

The replication technique has been widely used in improving system reliability due to transient faults. In this paper, we consider systems that use replication to tolerate up to one transient fault since single-fault-tolerance is a common assumption (Aminzadeh and Ejlali, 2011). Given a task τ_i with γ_i replicated tasks working at the same frequency, the new reliability is then given by

$$R_i^{\gamma_i}(f) = 1 - (1 - R_i(f))^{\gamma_i}. \tag{3}$$

Ideally, different replicas of the same task can execute at different frequencies. However, obtaining the frequency assignment for all the replicas of all the tasks will be computationally prohibitive as the increase in the number of replicas (Haque et al., 2016). Therefore, to reduce the computational overhead, uniform frequency is adopted for all replicas of a given task.

Note that there is a lower bound on the number of replicas needed to achieve the requirement of a certain reliability level. In addition, the more replicas used, the higher reliability level achieved. Based on Eq. (3), the minimum number γ_i of replicas required to achieve the target reliability level r_i of task τ_i at a given frequency level f can be easily determined, that is,

$$\gamma_i \ge \left\lceil \frac{\log(1 - r_i)}{\log(1 - R_i(f))} \right\rceil. \tag{4}$$

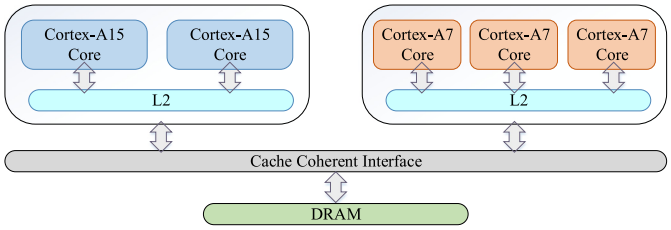


Fig. 1. ARM big.LITTLE heterogeneous multi-core architecture.

2.3. Architecture and execution model

Heterogeneous multi-core architectures, such as ARM big.LITTLE (i.e., Cortex-A15+Cortex-A7), TI DaVinci DM6000 (i.e., ARM9+DSP), and Xilinx Zynq7000 (i.e., Cortex-A9+FPGA), are systems that gain performance or energy efficiency not just by adding the same type of processors, but by adding dissimilar coprocessors to handle particular tasks. As the example of ARM big.LITTLE shown in Fig. 1, the architecture is composed of pairs of high-performance cores (i.e., Cortex-A15) and low-power cores (i.e., Cortex-A7) (ARM, 2013). The heterogeneous multi-core platform used in this work is based on a generalization of the ARM big.LITTLE heterogeneous multi-core, which has been widely adopted in the literature (Tan et al., 2015; Ma et al., 2017; Kriebel et al., 2014; Wang et al., 2017). NvidiaÕs variable symmetric multiprocessing (vSMP) also falls into this category Nvidia. The platform $\mathcal{P}$ consists of m clusters $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m$, which are heterogeneous in the sense that they are diverse in processing capabilities and fault arrival rates. Each cluster $\mathcal{C}_k$ $(1 \le k \le m)$ consists of a set of homogeneous cores, which are DVFS-enabled and equipped with a set of discrete voltage/frequency pairs $(v_{k,1}, f_{k,1}), \cdots, (v_{k,\iota}, f_{k,\iota}), \cdots, (v_{k,\ell_k}, f_{k,\ell_k})$, where ℓ_k is the number of voltage/frequency levels supported by cluster $\mathcal{C}_k$ and $1 \le \iota \le \ell_k$ holds. We assume that $v_{k,\min} = v_{k,1} \le v_{k,2} \le \cdots \le v_{k,\ell_k} = v_{k,\max}$ and $f_{k,\min} = f_{k,1} \le f_{k,2} \le \cdots \le f_{k,\ell_k} = f_{k,\max}$ hold for the sake of easy presentation. All the cores in a cluster need to run at the same frequency level.

As mentioned above, replication is used to tolerate transient faults. We assume that the original task and its replicas need to be executed in the same cluster, which is based on the following considerations. First, replication is a typical example of fault-tolerant technique explicit output comparison (EOC) that relies on explicit redundancy/replication: executing the same task multiple times and comparing their outputs. Second, using EOC improves the systems capability to tolerate soft errors, but also introduces significant timing overhead (i.e., timing overhead of error detection or output comparison) and may be detrimental to meeting real-time constraints. Third, the output comparison cost when executing the original task and its replicas in the same cluster is much smaller than that when executing in the different clusters. Benefiting from the identical operation pattern of homogeneous cores in the same cluster, this assumption can also guarantee the uniform frequency setting for the original task and its replicas. Note that the original task and its replicas should be executed on different cores to support the detection and recovery of faults. Therefore, for any cluster on the platform, we select one core from the cluster as the primary core to execute the original tasks and the rest cores as the secondary cores to execute the replicas. In addition, a secondary core is only allowed to accommodate one replica of a original task. Under this setting, the number of replicas for a task may exceed the number of secondary cores in theory. However, the number of replicas needed for a task is typically small in practice. Thus the setting is reasonable. For example, a task with a low

reliability of 0.7 could immediately have a high reliability of 0.91 if a replica is equipped.

2.4. Temperature model

Assume that there is negligible or no heat transfer among processing units and among other different units. This assumption is widely made for thermal-aware scheduling (Quan and Chaturvedi, 2010; Saha et al., 2012; Huang et al., 2014a; Zhou and Wei, 2015; Zhou et al., 2016c). Based on the assumption, a lumped RC thermal model HotSpot proposed by Skadron et al. (2004) is adopted to characterize the chip thermal profiles, which is expressed by the following system of ordinary differential equation.

$$\mathbf{C}\mathbf{T}' + \mathbf{G}\mathbf{T} = \mathbf{P} + \mathbf{G}T_{\text{amb}}. \tag{5}$$

This thermal model divides the chip into a number of thermal elements. Heat capacities of these elements are captured in matrix **C**. Thermal conductance values are captured in matrix **G**. In this equation, **T** is the temperature vector, $\mathbf{T}'$ is the first order derivative of the temperature, **P** contains the power consumption on all thermal nodes, and T_{amb} is the die's ambient temperature. In the steady-state, Eq. (5) becomes

$$\mathbf{G}\mathbf{T}_{\text{steady}} = \mathbf{P} + \mathbf{G}T_{\text{amb}}, \tag{6}$$

where $\mathbf{T}_{\text{steady}}$ contains the steady-state temperatures on all thermal nodes.

The accuracy of the adopted HotSpot model in terms of temperature prediction has been investigated. Huang et al. (2006) compared the temperature sensor readings from real platforms with values obtained from the corresponding HotSpot model. The results show that the temperatures predicted by the HotSpot model differs by less than 0.2 °C on average from those obtained from the sensors. Furthermore, they also compared the transient and steady state temperature measurements with results from the HotSpot model (Huang et al., 2004). The results show that the average absolute error for transient temperatures and steady state temperatures are 2.26% and 1.46%, respectively. As indicated from these results, HotSpot is an accurate thermal model thus has been widely used in the literature (Saha et al., 2012; Huang et al., 2014a; Quan and Chaturvedi, 2010; Jayaseelan and Mitra, 2008; Zhou and Wei, 2015; Zhou et al., 2016b; Ma et al., 2017; Chen et al., 2017).

2.5. Problem definition

Given the platform $\mathcal{P}$ consisting of m clusters, and the BoT application $\mathcal{B}$ containing n independent tasks, determine an assignment of tasks to clusters and a schedule of tasks on cores such that the task reliability and peak temperature constraints and real-time deadlines are satisfied and the makespan is minimized.

3. MILP formulation

In this section, we present our approach to solving the problem described above, and discuss limitations of the MILP-based approach at the end of the section. We define the following binary variables for the sake of easy representation.

$$A_{i,k,\iota} = \begin{cases} 1 & \text{if task } \tau_i \text{ is assigned to cluster } \mathcal{C}_k \\ & \text{and executed at frequency } f_{k,\iota}, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

$$S_{i,j} = \begin{cases} 1 & \text{if task } \tau_i \text{ starts before task } \tau_j, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

3.1. Objective

Let $t_{\text{start}}(\tau_i)$ and $t_{\text{finish}}(\tau_i)$ denote the start time and finish time of task τ_i, respectively. Then we have

$$t_{\text{finish}}(\tau_i) = t_{\text{start}}(\tau_i) + \sum_{k=1}^{m}\sum_{\iota=1}^{\ell_k} \frac{wc_i}{f_{k,\iota}} A_{i,k,\iota}. \tag{9}$$

As pointed out in Section 1, the makespan is defined as the latest completion time of all tasks and replicas on cores. Since the replicas on the secondary cores are operating the same pattern as the n original tasks on the primary cores, the makespan of the whole system can be calculated as

$$t_{\text{finish}} = \max_{i=1,2,\cdots,n} t_{\text{finish}}(\tau_i). \tag{10}$$

Consequently, the objective function of the MILP is expressed as

$$\min t_{\text{finish}}, \text{ where } t_{\text{finish}} = \max_{i=1,2,\cdots,n} t_{\text{finish}}(\tau_i). \tag{11}$$

3.2. Constraints

To guarantee that every task in the application $\mathcal{B}$ can be feasibly scheduled, the following constraints must be satisfied.

1). Every task τ_i is assigned to exactly one cluster and executed at one frequency level.

$$\sum_{k=1}^{m}\sum_{\iota=1}^{\ell_k} A_{i,k,\iota} = 1, \quad \forall i = 1, 2, \cdots, n. \tag{12}$$

2). Every task τ_i meets its deadline.

$$t_{\text{finish}}(\tau_i) \le D, \quad \forall i = 1, 2, \cdots, n. \tag{13}$$

3). Every task τ_i satisfies its reliability requirement.

$$R_{i,k,\iota} A_{i,k,\iota} \ge r_i, \ \forall i = 1, 2, \cdots, n, \tag{14}$$

where $R_{i,k,\iota}$ is the reliability achieved by task τ_i with replication when the task is assigned to cluster $\mathcal{C}_k$ and executed at frequency $f_{k,\iota}$. The reliability $R_{i,k,\iota}$ can be derived using Eq. (3).

4). The system peak temperature is below the temperature limit. To avoid temperature-induced failures, the peak temperature of clusters should be below a temperature limit (threshold) $T_{\max}$. The value of $T_{\max}$ is in general specified based on system design requirements. Let $T_{\text{peak}}(\mathcal{C}_k)$ denote the peak temperature of cluster $\mathcal{C}_k$, which is given by

$$T_{\text{peak}}(\mathcal{C}_k) = \max : \{T(t) \mid \forall t \in [0, t_{\text{finish}}(\mathcal{C}_k)]\}.$$

Here $T(t)$ is the instantaneous temperature during time interval $[0, t_{\text{finish}}(\mathcal{C}_k)]$ and can be obtained by Eq. (5). Then let T_{peak} denote the on-chip peak temperature, which can be calculated as

$$T_{\text{peak}} = \max_{k=1,2,\cdots,m} T_{\text{peak}}(\mathcal{C}_k). \tag{15}$$

5). Tasks have no overlapping executions in the same cluster. For any tasks τ_i and τ_j in the application $\mathcal{B}$ and assigned to the same cluster $\mathcal{C}_k$ (actually the primary core of cluster $\mathcal{C}_k$), the following inequalities need to be satisfied in order to avoid their overlapping executions.

$$\forall \tau_i, \tau_j \in \mathcal{B}$$

$$S_{i,j} + S_{j,i} \ge 0, \tag{16}$$

$$S_{i,j} + S_{j,i} \le 1, \tag{17}$$

$$t_{\text{start}}(\tau_i) \le t_{\text{start}}(\tau_j) + (1 - S_{i,j}) \cdot \Delta \cdot D, \tag{18}$$

$$t_{\text{start}}(\tau_j) \le t_{\text{start}}(\tau_i) + S_{i,j} \cdot \Delta \cdot D, \tag{19}$$

$$\forall \tau_i, \tau_j \in \mathcal{B}, \quad \iota, \bar{\iota} \in [1, \ell_k], \quad k \in [1, m],$$
$$t_{\text{finish}}(\tau_i) \le t_{\text{start}}(\tau_j) + (3 - A_{i,k,\iota} - A_{j,k,\bar{\iota}} - S_{i,j}) \cdot \Delta \cdot D, \tag{20}$$

$$t_{\text{finish}}(\tau_j) \le t_{\text{start}}(\tau_i) + (2 - A_{i,k,\iota} - A_{j,k,\bar{\iota}} + S_{i,j}) \cdot \Delta \cdot D, \tag{21}$$

where Δ is a constant number equal to or larger than 1. Eq. (18) states that task τ_i must start before task τ_j if $S_{i,j} = 1$. Eq. (20) guarantees that task τ_i finishes before task τ_j starts if tasks τ_i and τ_j are executed in the same cluster and task τ_i start before task τ_j. Similar conditions hold for Eqs. (19) and (21). Note that the replicas would naturally meet these constraints if the original tasks satisfy, which is due to the same operation pattern of the original tasks and replicas. Therefore, the constraints of replicas are not included in the formulation.

3.3. Limitation of MILP-based approach

In fact, the problem modeled in this section is a problem of determining the assignment of tasks to clusters, the operating frequency of all assigned tasks, the execution order of tasks on cores, and the number of replicas for every task. Considering that a given set of n tasks can be partitioned into can be partitioned into m subsets, the assigned tasks on the core can have $n!$ execution orders in extreme case that all tasks are assigned to the same core, every task can have ℓ operating frequencies at most, and the number of replicas for n tasks all need to be determined, the complexity of the modeled problem is $O(m^n \cdot n! \cdot \ell^n \cdot n)$, where $\ell = \max\{\ell_1, \ell_2, \cdots, \ell_m\}$ is the maximum of frequency levels supported by clusters. It is clear that the studied problem is a combinatorial optimization problem that is NP-hard (Korte et al., 2002). The target of the problem is to find the optimum solution from all feasible solutions. For systems with small number of clusters/cores and applications with small number of tasks, the studied problem can be optimally solved using an MILP solver. However, for systems of a larger granularity, the MILP solver cannot be used to efficiently solve the problem. Thus it is necessary to develop a polynomial-time heuristic to schedule tasks to cores.

4. Makespan-aware task assignment and scheduling under reliability and temperature constraints

The objective of this work is to generate a makespan-optimum schedule without violating the reliability and thermal design constraints. We first propose an MILP-based approach to obtain the optimum schedule, as described above. We then consider the limitation of the MILP-based approach for systems of a larger granularity and design a polynomial-time task assignment and scheduling heuristic. The heuristic is carried out in two steps. In the first step, the assignments of tasks to clusters for minimizing makespan and the replication of the assigned tasks for satisfying reliability requirements are determined. In the second step, the decisions on how to execute the assigned tasks (in which sequence and using which frequency) on cores for meeting the peak temperature constraint and task deadline are made. The heuristic is developed based on some observations from analyzing the effects of task assignment on makespan, reliability, and temperature. Therefore, this section first analyzes the effects of task assignment on makespan, reliability, and temperature, then show the overview of the proposed two-stage heuristic, and finally presents the details of the task assignment and scheduling algorithms.

4.1. Effects of task assignment on makespan, reliability, and temperature

Effects of Task Assignment on Makespan: Fig. 2 shows that different task-to-cluster assignments result in different makespan. As shown in the figure, tasks $\tau_1 - \tau_6$ are assigned to clusters $\mathcal{C}_1 - \mathcal{C}_3$, which operate at normalized frequencies $f = 1.0$, 0.8, and 0.6, respectively. The number of cores in every cluster is assumed to be one for simplicity. The execution time of tasks $\tau_1 - \tau_6$ operating at normalized frequency $f = 1.0$ are 1, 2, 2, 5, 6, and 8 time units, respectively. Fig. 2(a) presents an example task assignment with makespan of 31.7, which arranges tasks τ_1, τ_2 to cluster $\mathcal{C}_1$, task τ_3 to cluster $\mathcal{C}_2$, and tasks τ_4, τ_5, τ_6 to cluster $\mathcal{C}_3$. Fig. 2(b) presents an example task assignment with makespan of 10, which arranges tasks τ_2, τ_6 to cluster $\mathcal{C}_1$, tasks τ_3, τ_5 to cluster $\mathcal{C}_2$, and tasks τ_1, τ_4 to cluster $\mathcal{C}_3$. Compared to Fig. 2(b), the assignment presented in Fig. 2(a) can reduce the makespan by 68.5%. Therefore, we can readily make an observation that the task-to-cluster assignment has significant effect on makespan.

Effects of Task Assignment on Reliability: Unlike the assignment of tasks to homogeneous processors, the assignment of tasks to heterogeneous processors should consider the difference between soft error rates (SER) of processors. However, the previous works either focus on homogeneous processors or ignore the SER difference of heterogeneous processors. Therefore, we first introduce how to calculate the SER of processors, then investigate the SER difference of simulated heterogeneous processors using Monte Carlo simulation.

Fig. 3 shows that the failure rate of a processor can be estimated as the sum of failure rates of components that constitute the processor, where AVF_ζ is the architecture vulnerability factor of component ζ and takes the value in the range (0, 1] (Li et al., 2007). The AVF expresses the probability that a visible failure will occur, given a raw error event in a component. Clearly, before calculating the processor failure rate, we need to derive the failure rates of components. The following model can be utilized to predict neutron-induced[1] SER at component level (Hazucha and Svensson, 2000):

$$\text{SER}_{\text{Comp}} = \text{Const} \times \text{Flux} \times \text{Area} \times e^{-\frac{Q_{\text{crit}}}{Q_{\text{coll}}}}, \tag{22}$$

where Const is a parameter depending on the process technology, Flux is the flux of neutrons at the specific location (latitude and longitude), Area is the area of the circuit sensitive to soft errors, Q_{crit} is the critical charge, and Q_{coll} is the charge collection efficiency. Using Eq. (22), the SER of components 6T SRAM cell, 8T SRAM cell, Latch, and NAND2 can be calculated, and it also has been shown that the SER of four components built with the same technology are close when they are running at the same environment and parameters like voltage and temperature (Riera et al., 2016).

To investigate the effect of task assignment on reliability, we build two simulated processors and compare their SER. Similar to the literature (Riera et al., 2016), one simulated processor is assumed to be made up of 6T SRAM cell, Latch, and NAND2 while the other simulated processor is assumed to be made up of 8T SRAM cell, Latch, and NAND2. For each of the two simulated processors, the respective proportions of three components are denoted by α_1, α_2, α_3 and hold for $\alpha_1 + \alpha_2 + \alpha_3 = 1$, and the respective architecture vulnerability factors of three components are

[1] Soft errors are radiation induced failures that are mainly produced by two types of sources: alpha particles from the packaging and neutrons from the atmosphere. Alpha particles are already well known and can be mitigated by changing the packaging materials of the chip, while neutron strikes produce soft errors that are difficult to detect and have a high impact on the reliability (Riera et al., 2016). Therefore, we focus on the soft error rates due to neutrons.

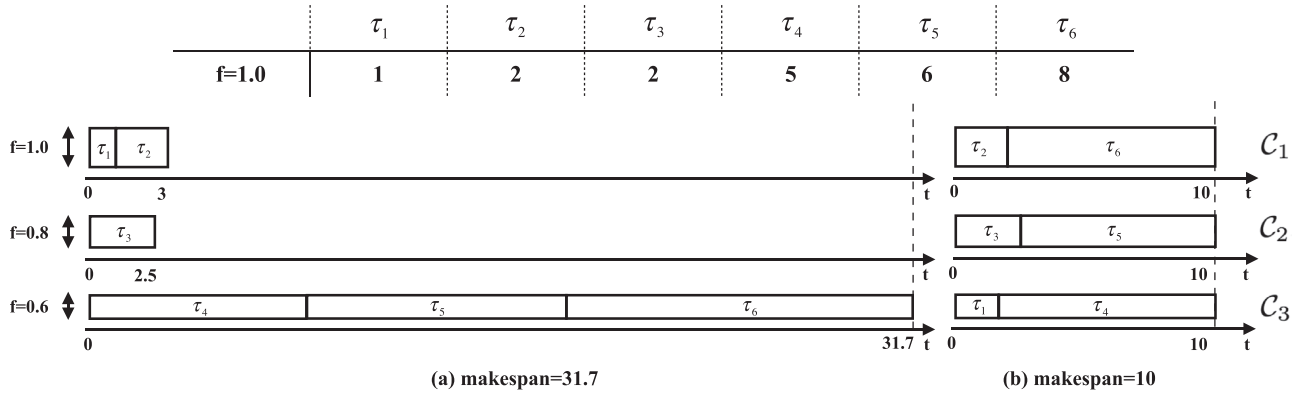


Fig. 2. Two examples of assigning tasks $\tau_1 - \tau_6$ to clusters $\mathcal{C}_1 - \mathcal{C}_3$.

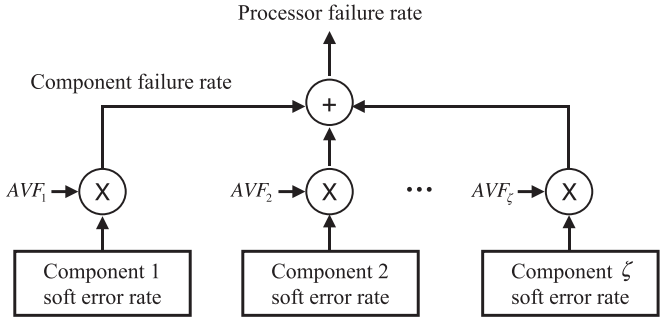


Fig. 3. The calculation of processor failure rate (Li et al., 2007).

denoted by AVF_1, AVF_2, AVF_3 and take the value from interval of (0, 1]. Without loss of generality, Monte Carlo simulation is used to produce samples by randomly setting the value of α_1, α_2, α_3 and AVF_1, AVF_2, AVF_3. Two produced processors with the same α_1, α_2, α_3, AVF_1, AVF_2, AVF_3 constitute one sample of Monte Carlo simulation. We take 10,000 Monte Carlo samples to compare the SER of simulated processors, which are calculated by the method presented in Fig. 3 and based on the SER of 6T SRAM cell, 8T SRAM cell, Latch, and NAND2 derived in the literature (Riera et al., 2016).

Fig. 4 shows the SER of two simulated processors and Fig. 5 plots the ratio of SER variation to processor SER, where each data in the figures is averaged over 100 Monte Carlo samples. It is clear in the figure that the SER of two simulated processors are close and all in the range of $[4.04 \times 10^{-5}, 5.07 \times 10^{-5}]$. In addition, both the ratio of SER variation to SER of simulated processor 1 and the ratio of SER variation to SER of simulated processor 2 are low, which are below 3.75% and 3.58%, respectively. These results indicate that the SER of heterogeneous processors still be similar if they are built with the same technology and working at the same environment and parameters, thus we make an observation that the assignment of tasks to clusters on the same chip has neglectable impact on neutron-related reliability.

Effects of Task Assignment on Temperature: It has been shown that the processor temperature profiles and peak temperature strongly depends on task execution order and operating frequency, both of which are determined in the scheduling step of our proposed scheme (Zhou and Wei, 2015; Jayaseelan and Mitra, 2008; Zhou et al., 2016c).

4.2. Overview of our two-stage scheme

Based on the above analysis that

- makespan minimization strongly depends on task assignment,
- effect of task assignment on reliability is neglectable,
- task temperature profiles and peak temperature mainly rely on task execution order and frequency,

and the following consideration that

- reliability constraint can be ensured by replication,
- temperature profiles of processors can be improved by wisely determining task execution order and operating frequency after task assignment,

the proposed two-stage scheme operates as follows. In stage 1, the scheme first assigns the tasks to the primary core of clusters in a makespan-optimal manner that enables the resultant schedule length of clusters are equal or nearly equal, and calculates the number of replicas required to achieve the target reliability level for every assigned task. During the calculation of the number of replicas required for a given task, its operating frequency is assumed to be the maximal frequency of its assigned cluster. This is based on the consideration that high frequency results in short execution time and low fault arrival rate, which in turn lead to early completion time and high reliability. Afterwards, the scheme determines the execution order of tasks assigned to the primary cores of clusters using RM scheduling (Liu and Layland, 1973), and creates replicas on the secondary cores in order to guarantee the reliability constraint.

In stage 2, the scheme checks the real-time and peak temperature constraint of tasks. If all tasks can be finished before the deadline and their peak temperature is below the temperature limit, the task schedule with optimized makespan is reported. If the real-time constraint of tasks cannot be satisfied, the scheme exits, which means that the input BoT application cannot be feasibly scheduled on the platform. If the real-time constraint is met but the peak temperature constraint is violated, the scheme then utilizes thermal-aware task sequencing to reduce temperature without incurring increase in makespan. Task sequencing is a technique that classifies tasks into hot/cool tasks and alternates the execution of hot and cool tasks (Zhou and Wei, 2015; Jayaseelan and Mitra, 2008). After the thermal profiles are improved by task sequencing, the scheme verifies the peak temperature constraint again. If the constraint is met, the scheme outputs the derived task schedule. Otherwise, the available slack is exploited to reduce the temperature by using frequency scaling that specifically scales down the frequency of hot tasks. It is worth noting that the task reliability still can be maintained by using more replicas when the task operating frequency is scaled. This is because the same target reliability can be achieved using a small number of replicas running at high frequencies or a large number of replicas running at low frequencies, as indicated in Eqs. (2) and (3).

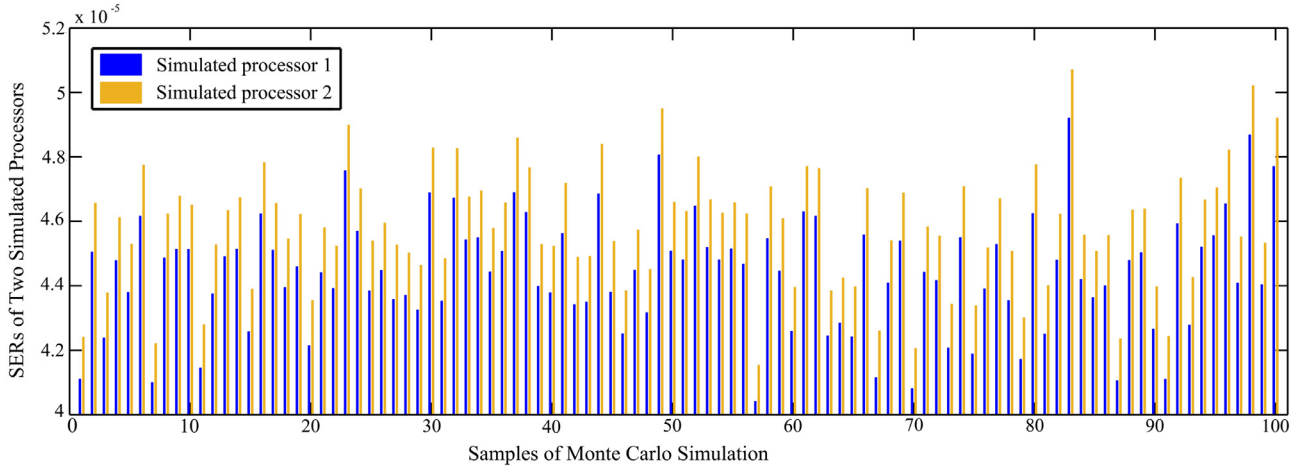


Fig. 4. The SER of two simulated processors.

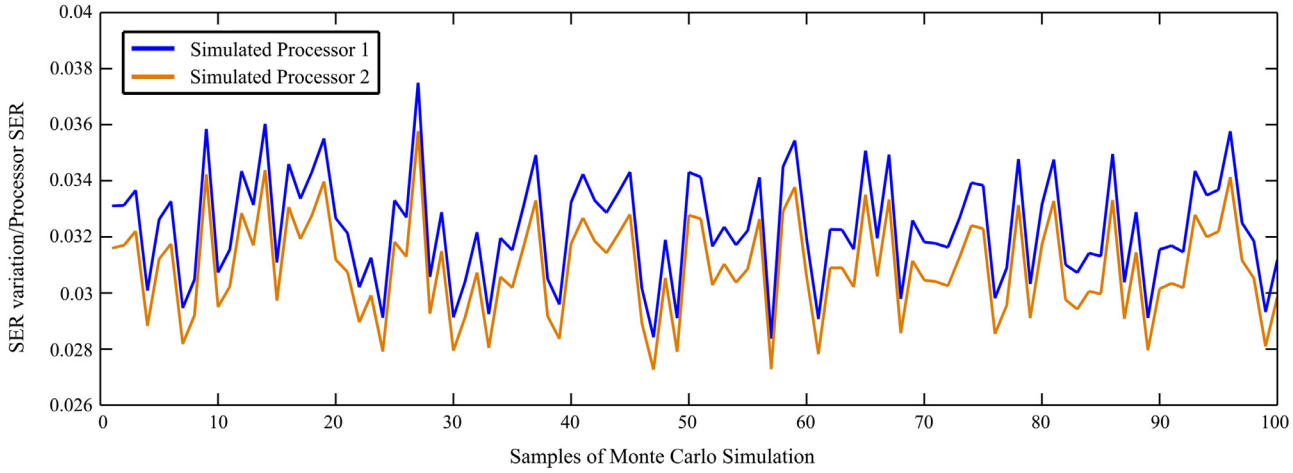


Fig. 5. The ratio of SER variation to processor SER.

Through the two stages, the assignment of tasks to clusters and the scheduling of tasks on the primary cores are derived. In particular, due to the same operation pattern of original tasks and replicas, the scheduling of replicas on the secondary cores can be also derived once the scheduling of their original tasks is generated. The overview of our proposed two-stage scheme is shown in Fig. 6 and the corresponding pseudo-code is described in Algorithm 1.

Before introducing the details of the algorithms of our scheme, we discuss the novelty of the scheme as below.

- Unlike the previous methods (Dongarra et al., 2007; Braun et al., 2001; Saha et al., 2012; Zhou et al., 2016c) ignoring the fact that the assignment of tasks to heterogeneous processors may affect makespan, reliability, and temperature, our proposed two-stage scheme is designed based on the observations from analyzing the effects of task assignment on the concerned subjects.
- The thermal-aware task sequencing technique adopted in the scheme has been developed in our previous work (Zhou and Wei, 2015). It is an improved version of the approach (Jayaseelan and Mitra, 2008) and can achieve a lower peak temperature by alternating the execution of tasks in the hot-cool order.
- Unlike the previous frequency scaling approaches (Wei et al., 2012; Hanumaiah and Vrudhula, 2012; Zhou and Wei, 2015; Jayaseelan and Mitra, 2008) that scale the operating frequency of all tasks either for reducing energy consumption or improving thermal profiles, the proposed scheme only scales the operating frequency of hot tasks with two considerations. First, the violation of peak temperature constraint is mostly incurred during the execution of hot tasks. Second, less operations of task frequency scaling result in earlier task completion time and less task replications, thus less sacrifice of makespan optimality.

4.3. Algorithms of our two-stage scheme

The objective of our two-stage scheme is to generate a makespan optimum schedule of independent tasks without violating the reliability and thermal design constraints. As introduced in Section 4.2, the first stage of the scheme determines the task assignment and replication strategy with the purpose of minimizing makespan under the constraint of task reliability, and the second stage of the scheme determines the schedule of assigned tasks on the core to satisfy the real-time and temperature constraint. The pseudo-code of our scheme is given in Algorithm 1.

Algorithm 1 takes the BoT application $\mathcal{B}$, the platform $\mathcal{P}$, and the temperature limit $T_{\max}$ as input. Based on the consideration that the makespan is minimized if the schedule length of all clusters are equal or close, the algorithm first calculates the total workload W_{tot} of n input tasks and computes the workloads assigned to m clusters that result in a minimum makespan using $W_{k,\text{opt}} = W_{\text{tot}} \times \frac{f_{k,\max}}{f_{1,\max}+f_{2,\max}+\cdots+f_{m,\max}}$. The algorithm then groups the n tasks into m sets according to the computed workloads in a first-fit manner and assigns the m task sets to clusters. Afterwards, the

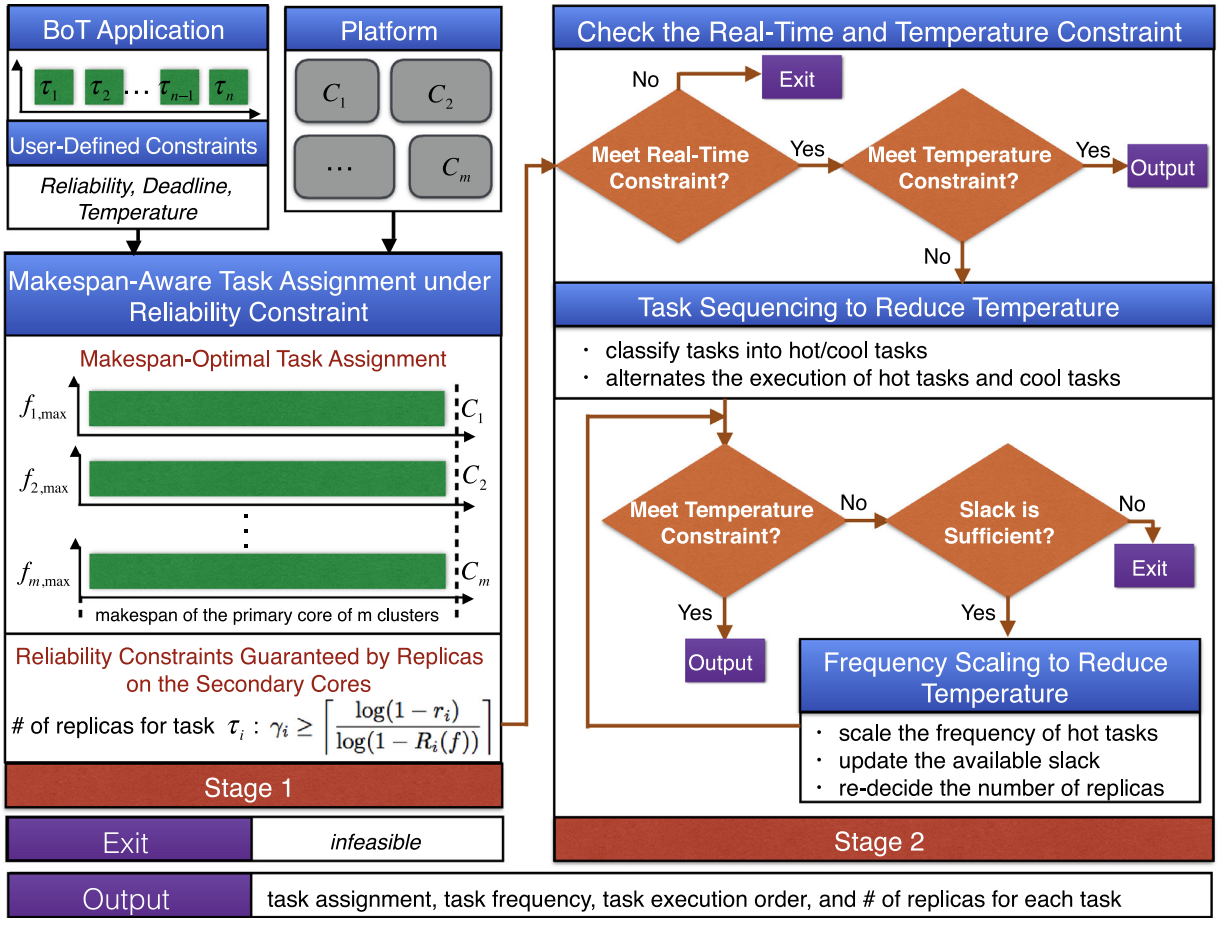


Fig. 6. The overview of our proposed two-stage scheme.

algorithm sets the operating frequency of individual tasks to the maximal frequencies of their respective assigned clusters and calculates the number of replicas for individual tasks under their respective reliability constraints using Eq. (4). Since all the tasks and their replicas are executed at the maximal frequencies supported by their assigned clusters that lead to shortest execution time, the tasks in the application are considered to be infeasibly scheduled if a task violates its real-time constraint.

The algorithm derives the peak temperature using Eq. (15) to verify if the current schedule can satisfy the thermal constraint. Specifically, if the peak temperature T_{peak} is higher than the predefined temperature limit $T_{\max}$, the temperature of assigned tasks in clusters are then reduced by using the thermal-aware task sequencing technique, the details of which are given in Algorithm 2. Otherwise, the algorithm exits and outputs the current schedule with optimized makespan. However, the peak temperature may not be controlled below the temperature limit due to the maximal frequencies adopted. Therefore, the algorithm trades the optimality of makespan for further reducing the temperature of tasks by using the thermal-aware frequency scaling technique, the details of which are given in Algorithm 3.

Unlike traditional cooling solutions, the thermal-aware task sequencing technique can reduce the peak temperature by exploiting the thermal characteristics of tasks without degrading task reliability and incurring increase in makespan. The technique is based on the observation (Jayaseelan and Mitra, 2008) that the execution order of a hot task and a cool task has non-negligible impact on peak temperature, and the final temperature of tasks executing in the hot-cool order is lower than that of tasks executing in the cool-hot order. Therefore, we propose a static task sequencing scheme based on hot-cool pairing to improve the temperature profiles of tasks in the clusters. The pseudo-code of the proposed thermal-aware task sequencing is given in Algorithm 2.

Before showing the details of Algorithm 2, we first define T_{start} as the start temperature of a hot task assuming the task ends its execution at the maximum temperature limit $T_{\max}$ and T_{end} as the end temperature of a cool task assuming the task starts its execution at the ambient temperature T_{amb}. The T_{start} and T_{end} are key characteristics of hot and cool tasks, respectively. A lower T_{start} of a task indicates that the task is hotter and a lower T_{end} of a task indicates that the task is cooler. Tasks in the hot queue $Q_{k,\text{hot}}$ are sorted in the increasing order of T_{start} and tasks in the cool queue $Q_{k,\text{cool}}$ are sorted in the increasing order of T_{end}.

Given these, Algorithm 2 can maintain a hot queue $Q_{k,\text{hot}}$ with the hottest task at the head and a cool queue $Q_{k,\text{cool}}$ with the coolest task at the head, and takes a target queue $Q_{k,\text{tar}}$ and the set $\mathcal{T}_k$ of tasks assigned to cluster $\mathcal{C}_k$ as input. The algorithm first initializes the target queue $Q_{k,\text{tar}}$ by pushing all the assigned tasks of cluster $\mathcal{C}_k$ into the queue. It then classifies a task τ_i into hot or cool task category based on the steady state temperature $T_{\text{steady}}(\tau_i)$ of the task and inserts the task into the corresponding queue. More specifically, if $T_{\text{steady}}(\tau_i) \geq T_{\max}$, task τ_i is considered as a hot task and inserted into the hot queue $Q_{k,\text{hot}}$. Otherwise, the task is a cool task and inserted into the cool queue $Q_{k,\text{cool}}$. After obtaining the hot queue and cool queue, the algorithm begins the iterative paring and sequencing of tasks. In each round of the iteration, if neither Q_{hot} nor Q_{cool} are empty, the algorithm pairs the task at the head of Q_{hot} and the task at the head of Q_{cool} in the order of

Algorithm 1: Makespan minimization under the constraints of reliability, real-time, and temperature.

Input: i) BoT application, $\mathcal{B} = \{\tau_1, \tau_2, \cdots, \tau_n\}$;
ii) platform, $\mathcal{P} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m\}$;
iii) temperature limit, $T_{\max}$;
Output: i) the task assignment, $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_m$;
ii) task operating frequency, $f(\tau_1), f(\tau_2), \cdots, f(\tau_n)$;
iii) task execution order in $Q_{1,\text{tar}}, Q_{2,\text{tar}}, \cdots, Q_{m,\text{tar}}$;
iv) the number of replicas for tasks, $\gamma_1, \gamma_2, \cdots, \gamma_n$;

calculate the total workload W_{tot} of n tasks using $W_{\text{tot}} = wc_1 + wc_2 + \cdots + wc_n$;
for $k = 1$ to m **do**
  calculate the desired workload of cluster $\mathcal{C}_k$ that leads to the minimized makespan using $W_{k,\text{opt}} = W_{\text{tot}} \times \frac{f_{k,\max}}{f_{1,\max}+f_{2,\max}+\cdots+f_{m,\max}}$;
end
while $\mathcal{B} \neq \varnothing$ and $k \leq m$ **do**
  initialize the set $\mathcal{T}_k$ of tasks assigned to the primary core of cluster $\mathcal{C}_k$ and the workload W_k of cluster $\mathcal{C}_k$ using $\mathcal{T}_k = \varnothing$ and $W_k = 0$;
  for $i = 1$ to $\mathcal{B}$.size() **do**
    if $W_k + wc_i \leq W_{k,\text{opt}}$ **then**
      $\mathcal{T}_k = \mathcal{T}_k + \tau_i$;
      $\mathcal{B} = \mathcal{B} - \tau_i$;
      $W_k = W_k + wc_i$;
    end
  end
  $k = k + 1$;
end
for $k = 1$ to m **do**
  for $i = 1$ to $\mathcal{T}_k$.*size()* **do**
    set the operating frequency of τ_i by $f(\tau_i) = f_{k,\max}$;
    derive the number γ_i of replicas for task τ_i to satisfy the reliability constraint r_i using Eq. (4);
    if $t_{\text{finish}}(\tau_i) > D$ **then**
      exit with *infeasible schedule*;
    end
  end
end
derive the peak temperature T_{peak} using Eq. (15);
if $T_{\text{peak}} > T_{\max}$ **then**
  for $k = 1$ to m **do**
    reduce the temperature of assigned tasks in set$\mathcal{T}_k$ by task sequencing given in Algorithm 2;
  end
  derive the peak temperature T_{peak} using Eq. (15);
  if $T_{\text{peak}} > T_{\max}$ **then**
    for $k = 1$ to m **do**
      reduce the temperature of assigned tasks in set $\mathcal{T}_k$ by frequency scaling given in Algorithm˜3;
    end
  end
  else
    exit with **Output**;
  end
end
else
  exit with **Output**;
end

Algorithm 2: Thermal-aware task sequencing.

Input: i) set $\mathcal{T}_k$ of tasks assigned to cluster $\mathcal{C}_k$;
ii) maintain a hot queue $Q_{k,\text{hot}}$ with the hottest task at the head, a cool queue $Q_{k,\text{cool}}$ with the coolest task at the head, and a target queue $Q_{k,\text{tar}}$;
Output: target queue $Q_{k,\text{tar}}$

move all tasks in set $\mathcal{T}_k$ into the target queue $Q_{k,\text{tar}}$;
for $i = 1$ to $Q_{k,\text{tar}}$.size() **do**
  classify τ_i into hot (cool) task based on $T_{\text{steady}}(\tau_i)$;
  derive $T_{\text{start}}(\tau_i)$ if hot and $T_{\text{end}}(\tau_i)$ if cool;
  insert τ_i into hot/cool queue $Q_{k,\text{hot}}/Q_{k,\text{cool}}$;
end
while $Q_{k,\text{hot}} \neq$ NULL or $Q_{k,\text{cool}} \neq$ NULL **do**
  if $Q_{k,\text{hot}} \neq$ NULL and $Q_{k,\text{cool}} \neq$ NULL **then**
    pair tasks at the head of $Q_{k,\text{hot}}$ and $Q_{k,\text{cool}}$;
    sequence the two tasks in the newly formed pair in the order of hot-cool;
    push the pair into the target queue $Q_{k,\text{tar}}$;
    update the $Q_{k,\text{hot}}$ and $Q_{k,\text{cool}}$;
  end
  else
    append tasks in non-empty queue to $Q_{k,\text{tar}}$;
  end
end

Algorithm 3: Thermal-aware frequency scaling.

Input: i) set $\mathcal{T}_k$ of tasks assigned to cluster $\mathcal{C}_k$;
ii) frequency set $\{f_{k,1}, f_{k,2}, \cdots, f_{k,\ell_k}\}$ supported by cluster $\mathcal{C}_k$, where $f_{k,\min} = f_{k,1}$ and $f_{k,\max} = f_{k,\ell_k}$;
Output: operating frequency of tasks in set $\mathcal{T}_k$

derive the initial available slack $S_{k,\text{ava}}$ using $S_{k,\text{ava}} = D - \sum_{i=1}^{\mathcal{T}_k.\text{size}()} \frac{wc_i}{f_{k,\max}}$;
for $i = 1$ to $\mathcal{T}_k$.size() **do**
  if task τ_i is hot **then**
    $\zeta = \ell_k$;
    while $S_{k,\text{ava}} > 0$ **do**
      if $\frac{wc_i}{f_{k,\zeta-1}} - \frac{wc_i}{f_{k,\zeta}} \leq S_{k,\text{ava}}$ **then**
        scale the operating frequency of task τ_i using $f(\tau_i) = f_{k,\zeta-1}$;
        update the available slack using $S_{k,\text{ava}} = S_{k,\text{ava}} - (\frac{wc_i}{f_{k,\zeta-1}} - \frac{wc_i}{f_{k,\zeta}})$;
        re-decide the number γ_i of replicas for task τ_i to meet reliability constraint r_i by Eq. (4);
        $\zeta = \zeta - 1$;
        if $\zeta == 1$ **then**
          break;
        end
      end
    end
  end
end

hot-cool, and pushes the pair into the target queue Q_{tar}. The Q_{hot} and Q_{cool} are hence updated. Otherwise, the algorithm appends the tasks in the non-empty queue to the tail of the target queue.

As pointed out earlier, task sequencing may not be able to guarantee that the peak temperature is below the temperature limit since all tasks are executed at the maximum frequencies of their respectively assigned clusters. To handle this situation, the

Table 1
Parameters of the real-world benchmarks.

Application	Description	Expected Task Execution Time	Standard Deviation	# of Tasks in the Application
toast	GSM speech encoder	3.286	1.27	100
madplay	MP3 audio decoder	2.253	1.01	130
tmndec	H.263 video decoder	2.799	0.762	90
mpegplay	MPEG video decoder	2.036	0.722	140

optimality of makespan needs to be traded for a lower peak temperature by scaling down the operating frequencies of hot tasks. The pseudo-code of the proposed thermal-aware frequency scaling is given in Algorithm 3. The inputs of the algorithm are the set $\mathcal{T}_k$ of tasks assigned to cluster $\mathcal{C}_k$ and the frequency set supported by the cluster. The algorithm operates as follows. It first derives the initial available slack for the input tasks by $S_{k,\text{ava}} = D - \sum_{i=1}^{\mathcal{T}_k.\text{size}()} \frac{wc_i}{f_{k,\max}}$. It then utilizes the available slack in a greedy way to scale down the operating frequencies of hot tasks in set $\mathcal{T}_k$. If the slack demand required to scale down the frequency is not greater than the available slack, i.e., $\frac{wc_i}{f_{k,\zeta-1}} - \frac{wc_i}{f_{k,\zeta}} \leq S_{k,\text{ava}}$, the task operating frequency is then scaled by $f(\tau_i) = f_{k,\zeta-1}$. The available slack $S_{k,\text{ava}}$ is hence updated to $S_{k,\text{ava}} - (\frac{wc_i}{f_{k,\zeta-1}} - \frac{wc_i}{f_{k,\zeta}})$ and the number γ_i of replicas for task τ_i to meet the reliability constraint r_i is re-determined by Eq. (4). This process repeats until all the hot tasks in set $\mathcal{T}_k$ are examined once. The time complexity of Algorithms 1–3 are $O(nm)$, $O(n)$, and $O(n\ell)$, where n is the number of tasks in the application $\mathcal{B}$, m is the number of clusters in the platform $\mathcal{P}$, and $\ell = \max\{\ell_1, \ell_2, \cdots, \ell_m\}$ is the maximum of frequency levels supported by clusters.

5. Evaluation

In this section, we first describe the simulation setups for validation, then present and analyze the simulation results.

5.1. Simulation setups

Extensive simulation experiments were carried out to validate the effectiveness of the proposed schemes. We generated the synthetic BoT applications by a random task generator to verify the proposed schemes. The expected value and standard deviation of every task worst-case execution cycles are specified in the intervals of $[4 \times 10^8, 8 \times 10^9]$ and $[2 \times 10^8, 2 \times 10^9]$, respectively. The common deadline is set to 1.5 times of the total task worst-case execution cycles. Eleven task sets (applications) are created in this way and the size of every task set is increased from 100 to 600, with a step increase of 50. We also utilized real-world multimedia applications (Li et al., 2014) toast, madplay, tmndec, and mpegplay to validate the proposed schemes. The parameters of tasks in the four practical applications, including the expected value and standard deviation of task execution time and the number of tasks in each application, are shown in Table 1.

The same settings of reliability and temperature constraints are adopted for the synthetic applications and real-world benchmarks. That is, the reliability requirement of tasks are specified in the interval of [0.7, 0.999] (Huang et al., 2014b). The ambient temperature and maximal temperature limit are set to 40 °C and 70 °C, respectively. The simulated processor is modeled based on the Versatile Express Development Platform (Tan et al., 2015) that includes a prototype version of the ARM big.LITTLE chip containing 3 Cortex A7 cores and 2 Cortex A15 cores. The average arrival rates of Cortex A7 and Cortex A15 operating at the maximal frequency are assumed to be 4×10^{-6} and 7×10^{-6}, respectively. The frequency of Cortex A7 core is varied from 1 GHz to 1.2 GHz and the frequency of Cortex A15 core is varied from 1.4 GHz to 2.5 GHz.

It is exceedingly difficult to find the related works that have the same objective (i.e., makespan minimization) with the consideration of same constraints (i.e., deadline, reliability, and peak temperature limit). Therefore, we compared our proposed schemes (i.e., the MILP approach and the two-stage heuristic) with two representative existing approaches RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014) in the simulations. Because the two approaches have the most similar concerns and do the most similar work with our proposed scheme among the previous literature as far as we know. The MILP formulation is described in Section 3. The two-stage heuristic attempts to achieve the makespan minimization under the constraints of reliability and peak temperature by exploiting the techniques of makespan-optimum task assignment, reliability-aware task replication, thermal-aware task sequencing and frequency scaling. RBSA is a list scheduling heuristics (Assayad et al., 2004) that aims to minimize the schedule length (i.e., makespan) as well as maximize the system reliability based on a bi-criteria compromise function. It utilizes a parameter of the compromise function to control the weight of two objectives, in order to satisfy the reliability or the schedule length requirements. ESTS is a stochastic task scheduling algorithm (Li et al., 2014) that takes into account the variation of task execution time in different task instances. It can maximize the probability of minimizing makespan under the task deadline and energy consumption budget constraints. All the algorithms were implemented in C++, and the simulations were performed on a machine with Intel Dual-Core 3.0 GHz processor and 8GB memory. For the sake of fair comparison, the same simulation setups are adopted for our proposed heuristic and methods RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014).

5.2. MILP formulation performance

In this set of experiments, we used CPLEX with AMPL to solve instances of the MILP formulation in Section 3 for the optimal makespan under the reliability and thermal constraints. As discussed in Section 3.3, the MILP solver cannot be used to efficiently solve the problem for systems of a larger granularity. Typically, MILP is able to produce the optimal makespan for small applications, and for most of the applications, MILP cannot find optimal solutions in several hours. Therefore, unlike the simulation setups above, we limited the size of task sets to be blow 100 for the MILP experiments, and terminated the MILP solver after 12 hours and used the best results that solver had. Since neither of RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014) used in the comparison considers the temperature and reliability constraints simultaneously, the solutions generated by the two methods may violate the constraints. However, the focus of this set of experiments is to verify the performance of MILP in optimizing makespan under the reliability and thermal constraints. Thus we removed these invalid results of RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014) in the comparison.

Table 2 shows the comparison of makespan of eleven task sets achieved by the MILP approach, the proposed two-stage heuristic, RBSA (Assayad et al., 2004), and ESTS (Li et al., 2014). The "MS" column represents the makespan of different approaches. The "Reduc" column represents the makespan reduction of MILP, the

Table 2
Makespan of eleven task sets achieved by the MILP approach, the proposed two-stage heuristic, RBSA (Assayad et al., 2004), and ESTS (Li et al., 2014).

Synthetic	Random	MILP		Proposed-Heuristic		RBSA		ESTS	
Task Set	MS	MS	Reduc	MS	Reduc	MS	Reduc	MS	Reduc
Set 1	132.751	94.978	28.5%	101.693	23.4%	107.649	18.9%	110.864	16.5%
Set 2	167.243	131.224	21.5%	131.702	21.3%	150.460	10.0%	144.905	13.4%
Set 3	153.056	106.646	30.3%	112.126	26.7%	129.681	15.3%	129.681	15.3%
Set 4	113.525	82.945	26.9%	87.944	22.5%	98.823	13.0%	93.036	18.0%
Set 5	101.583	81.304	20.0%	85.164	16.2%	92.985	8.5%	96.306	5.2%
Set 6	88.003	69.211	21.4%	71.476	18.8%	86.957	1.2%	81.023	7.9%
Set 7	123.297	102.988	16.5%	99.288	19.5%	114.944	6.8%	108.782	11.8%
Set 8	109.038	79.612	27.0%	89.161	18.2%	97.206	10.9%	95.881	12.1%
Set 9	133.655	95.439	28.6%	100.251	25.0%	112.045	16.2%	118.235	11.5%
Set 10	98.494	75.610	23.2%	77.269	21.5%	84.351	14.4%	81.728	17.0%
Set 11	92.600	70.428	23.9%	71.065	23.3%	82.012	11.4%	87.945	5.0%
Avg.	119.386	90.035	24.6%	93.376	21.8%	105.19	11.9%	104.399	12.6%

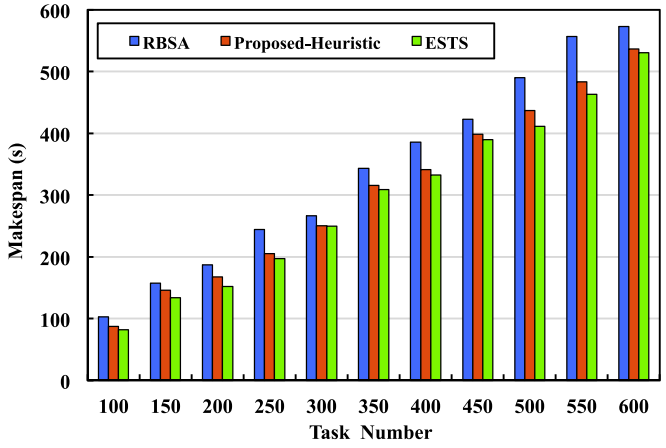


Fig. 7. Makespan of eleven task sets achieved by methods RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic.

proposed heuristic, RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014) over Random, which is a baseline method that generates the assignment, operating frequency, execution order, and replicas of tasks at random. As the simulation results in the table, the average makespan reduction achieved by MILP, the proposed heuristic, RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014) are 24.6%, 21.8%, 11.9%, and 12.6%, respectively. Moreover, MILP almost can maximize the reduction of makespan for all task sets when compared to the proposed heuristic, RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014). However, the MILP solver always finds the solutions in several hours while heuristic algorithms in several minutes.

5.3. Performance of the two-stage heuristic

Two sets of simulation experiments are implemented to validate the effectiveness of the proposed two-stage heuristic in terms of reducing makespan, improving feasibility, and controlling peak temperature. In the first set of simulations, synthetic BoT applications were generated by a random task generator while in the second set of simulations, real-world multimedia applications were utilized. The simulation results are described below in detail.

5.3.1. Simulation results of synthetic tasks

Fig. 7 compares the makespan of eleven task sets achieved by methods RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic. Apparently, the makespan achieved by the three algorithms are all growing with the increase of the size of task set. As shown in the figure, the average makespan of eleven task sets achieved by RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic are 339.3s, 295.6s, and 306.4s, respectively. The results indicate that the makespan of the proposed heuristic is smaller than that of RBSA (Assayad et al., 2004). For example, the proposed scheme can reduce the makespan of the forth task set by 16.03% as compared to RBSA (Assayad et al., 2004). This benefits from the makespan-aware task assignment in the proposed heuristic. The results also show that the makespan of the proposed heuristic is close to that of ESTS (Li et al., 2014). The reason why ESTS (Li et al., 2014) outperforms the proposed heuristic is that ESTS (Li et al., 2014) ignores the reliability and temperature constraints, which are antagonistic to the objective of reducing makespan.

Table 3 compares the feasibility of eleven task sets achieved by methods RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic. From the results in the table, we can easily find that the proposed heuristic achieves the highest feasibility as compared to RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014), no matter when considering all the constraints, or either only considering the temperature constraint or only considering the reliability constraint. For instance, when all the constraints are taken into account, the proposed heuristic can improve the feasibility by up to 28.1% and 33.8% as compared to RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014), respectively. The better performance with respect to feasibility achieved by the proposed heuristic is due to the effectiveness of the adopted temperature and reliability-aware techniques.

Fig. 8 compares the peak temperature of eleven task sets achieved by methods RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic. It has been shown in the figure that the proposed heuristic has the lowest peak temperature among the three approaches. For example, the proposed heuristic can lower the peak temperature of the fifth task set by 8.22 °C and 5.01 °C as compared to RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014), respectively. The reduction of peak temperature benefits from the thermal-aware task sequencing and frequency scaling adopted in the proposed heuristic, the effectiveness of which in thermal management has been demonstrated in our previous work (Zhou and Wei, 2015) thus is not discussed in this paper.

5.3.2. Simulation results of real-world benchmarks

Fig. 9 compares the makespan of four benchmarks toast, madplay, tmndec, and mpegplay achieved by RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic. Similar to the results shown in Fig. 7, the makespan of the four benchmarks achieved by the proposed scheme is smaller (up to 11.75%) than that of RBSA (Assayad et al., 2004), which is due to the

Table 3
Feasibility of eleven task sets achieved by methods RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic.

# of tasks	Feasibility when considering all the constraints			Feasibility when considering the temperature constraint			Feasibility when considering the reliability constraint		
	RBSA	Proposed-Heuristic	ESTS	RBSA	Proposed-Heuristic	ESTS	RBSA	Proposed-Heuristic	ESTS
100	100%	100%	98.0%	100%	100%	100%	100%	100%	98.0%
150	100%	100%	97.3%	100%	100%	100%	100%	100%	97.3%
200	100%	100%	96.5%	100%	100%	100%	100%	100%	96.5%
250	100%	100%	96.0%	100%	100%	100%	100%	100%	96.0%
300	100%	100%	95.0%	100%	100%	100%	100%	100%	95.0%
350	100%	100%	94.6%	100%	100%	100%	100%	100%	94.6%
400	100%	100%	93.0%	100%	100%	100%	100%	100%	93.0%
450	90.2%	100%	88.0%	90.2%	100%	92.0%	100%	100%	92.4%
500	87.0%	100%	82.0%	87.0%	100%	86.2%	100%	100%	93.0%
550	73.5%	100%	72.7%	73.5%	100%	76.2%	100%	100%	92.4%
600	71.9%	100%	66.2%	71.9%	100%	70.8%	100%	100%	90.9%

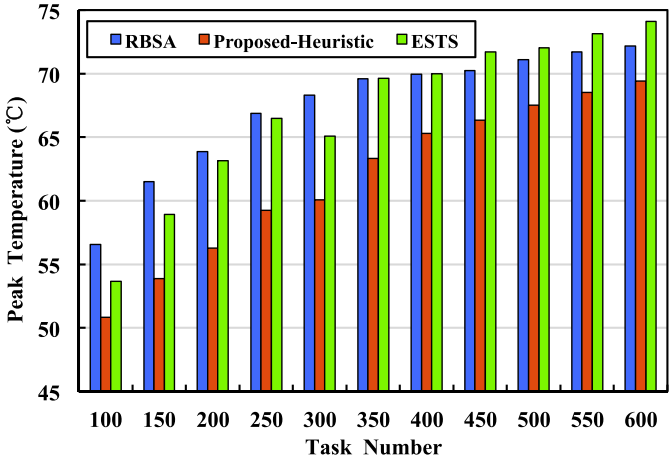


Fig. 8. Peak temperature of eleven task sets achieved by RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic.

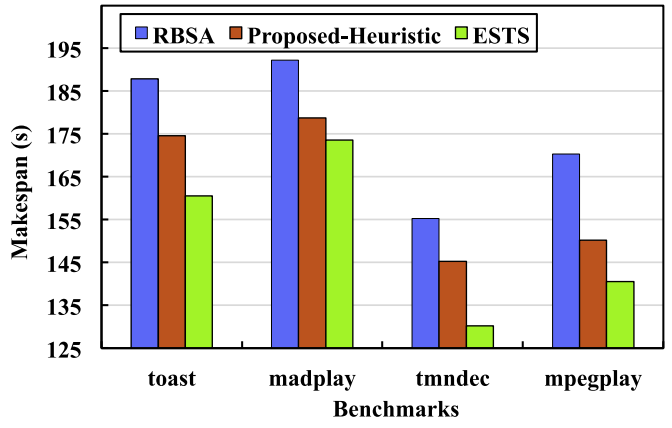


Fig. 9. Makespan of toast, madplay, tmndec, and mpegplay achieved by RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic.

effectiveness of the proposed makespan-aware task assignment heuristics. In addition, the makespan of the four benchmarks achieved by the proposed heuristic is larger than that of ESTS (Li et al., 2014), which is due to the trade of the optimality of makespan made in the proposed heuristic for guaranteeing the temperature and reliability constraints. Table 4 compares the feasibility of the four benchmarks in terms of satisfying temperature and reliability constraints achieved by RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic. As expected, the feasibility of the proposed scheme is the highest among the three

Table 4
Feasibility of toast, madplay, tmndec, and mpegplay achieved by RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic.

Benchmarks	RBSA	Proposed-Heuristic	ESTS
toast	86.0%	100%	78.0%
madplay	92.4%	100%	86.5%
tmndec	90.0%	100%	78.0%
mpegplay	85.6%	100%	71.5%

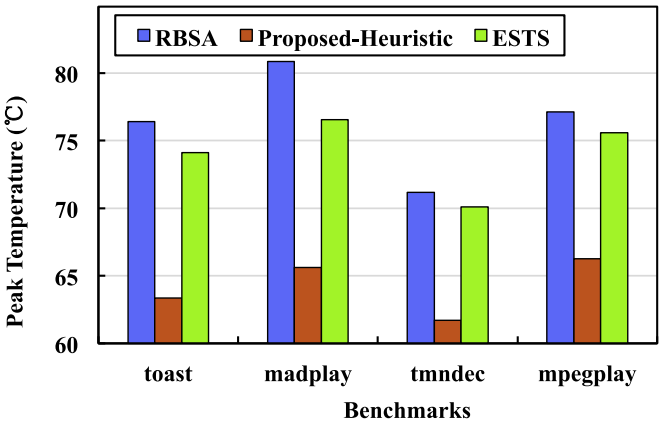


Fig. 10. Peak temperature of toast, madplay, tmndec, and mpegplay achieved by RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic.

algorithms. Taking the benchmark mpegplay as an example, the feasibility can be improved by 14.4% and 28.5% using the proposed heuristic as compared to RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014), respectively. Fig. 10 compares the peak temperature of the four benchmarks achieved by RBSA (Assayad et al., 2004), ESTS (Li et al., 2014), and the proposed heuristic. The results in the figure clearly demonstrate that the peak temperature of the four benchmarks can be greatly reduced by the thermal management techniques used in the proposed heuristic. More specifically, the peak temperature of the four benchmarks can be lowered by up to 15.2 °C and 10.9 °C using the proposed heuristic as compared to RBSA (Assayad et al., 2004) and ESTS (Li et al., 2014), respectively.

6. Discussion

This section discusses the novelties of this paper when comparing with our previous works (Zhou et al., 2016a; Wei et al., 2012; Zhou et al., 2016b; Zhou and Wei, 2015; Zhou et al., 2016c). Traditional reliability-aware task scheduling mechanisms can improve or maintain the system reliability by using techniques such as

rollback recovery, replication, and frequency elevation, which either tolerate the occurred transient faults or lower the fault arrival rate. However, these techniques would inevitably have the side effects of elevated energy consumption, increased makespan (thus reduced throughput), and higher temperature (thus shortened device lifetime). It is most desirable if we can work out a new task scheduling mechanism that is able to handle all these issues together. But unfortunately, this cannot be realized due to the considerations below.

1. Energy minimization and makespan minimization are two opposite objectives, thus cannot be optimized simultaneously. More specifically, if the operating frequencies of tasks are scaled to reduce energy consumption, makespan is increased because of the delayed task completions. Similarly, if high frequencies are used to execute tasks for reducing makespan, energy consumption would be inevitably increased.
2. The research on temperature can be conducted from two aspects. On one hand, when the system thermal profile is not very bad to cause hardware failures, temperature is typically considered as a constraint and the focus of research is on optimizing other objectives such as energy and makespan. In other words, system peak temperature cannot exceed a safe threshold. On the other hand, when the system thermal profile is very bad to cause hardware failures, the focus of research is on reducing the peak temperature to the utmost in the purpose of avoiding thermal emergency (and hence hardware failures).

Based on the first consideration, we concentrate on optimizing energy consumption in our published paper (Wei et al., 2012; Zhou et al., 2016b; Zhou and Wei, 2015) while optimizing makespan in this work. Based on the second consideration, we concentrate on controlling system peak temperature below a safe threshold and optimizing energy consumption or makespan in our published paper (Wei et al., 2012; Zhou et al., 2016b; Zhou and Wei, 2015) and this work, whereas we concentrate on minimizing system peak temperature in our published paper (Zhou et al., 2016c). Unlike this work and our published paper (Wei et al., 2012; Zhou et al., 2016b; Zhou and Wei, 2015; Zhou et al., 2016c), two types of faults are jointly handled and the soft-error reliability and lifetime reliability are balanced such that system availability is maximized (Zhou et al., 2016a).

7. Conclusion

In this paper, we presented an assignment and scheduling technique that uses a mixed-integer linear program solver to optimize the makespan under the constraints of deadline, reliability, and peak temperature. To efficiently solve this NP-hard assignment and scheduling problem, we also proposed a task assignment and scheduling heuristic in which the assignment, replication, operating frequency, and execution order of tasks are determined. The heuristic is developed based on the analysis of the effects of task assignment on makespan, reliability, and temperature.

Extensive simulations were performed to validate the proposed MILP formulation and the proposed heuristic. The results of experiments for verifying the effectiveness of the proposed MILP formulation in reducing makespan show that MILP has the best performance. To be specific, the average makespan reduction achieved by MILP, the proposed heuristic, RBSA and ESTS over baseline method Random are 24.6%, 21.8%, 11.9%, and 12.6%, respectively. The results of experiments for verifying the effectiveness of the proposed heuristic show that the heuristic can reduce the makespan by up to 16.03% as compared to RBSA, improves the feasibility by up to 33.8% and lowers the peak temperature by up to 15.2 °C as compared to RBSA and ESTS.

References

Albers, S., Hellwig, M., 2016. Online makespan minimization with parallel schedules. Algorithmica 1–29.

Aminzadeh, S., Ejlali, A., 2011. A comparative study of system-level energy management methods for fault-tolerant hard real-time systems. IEEE Trans. Comput. 60 (9), 1288–1299.

ARM, big.LITTLE technology: the future of mobile. [Online]. Available: https://www.arm.com/files/pdf/big_LITTLE_Technology_the_Future_of_Mobile.pdf, 2013.

Assayad, I., Girault, A., Kalla, H., 2004. A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints. In: Proceedings of International Conference on Dependable Systems and Networks, pp. 347–356.

Assayad, I., Girault, A., Kalla, H., 2011. Tradeoff exploration between reliability power consumption and execution time. In: Proceedings of International Conference on Computer Safety, Reliability and Security, pp. 437–451.

Aupy, G., Benoit, A., Robert, Y., 2012. Energy-aware scheduling under reliability and makespan constraints. In: Proceedings of International Conference on High Performance Computing, pp. 1–10.

Braun, T., Hensgen, D., Freund, R., Siegel, H., et al., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput. 61 (6), 810–837.

Casas, I., Taheri, J., Ranjan, R., Wang, L., Zomaya, A., 2017. A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems. Future Gen. Comput. Syst. 74, 168–178.

Chen, X., Huang, X., Xiang, Y., Zhang, D., Ranjan, R., Liao, C., 2017. A CPS framework based perturbation constrained buffer planning approach in VLSI design. J. Parallel Distrib. Comput. 103, 3–10.

Cirne, W., Brasileiro, F., Sauve, J., Andrade, N., Paranhos, D., Santos-Neto, E., Medeiros, R., 2003. Grid computing for bag of tasks applications. In: Proceedings of International Conference on E-Commerce, E-Business and E-Government.

Culler, D., Singh, J., Gupta, A., 1999. Parallel computer architecture: a hardware/software approach.

Dongarra, J., Jeannot, E., Saule, E., Shi, Z., 2007. Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. In: Proceedings of ACM Symposium on Parallel Algorithms and Architectures, pp. 280–288.

Ejlali, A., Al-Hashimi, B., Eles, P., 2012. Low-energy stand by sparing for hard real-time systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 31 (3), 329–342.

Gutierrez-Garcia, J., Sim, K., 2013. A family of heuristics for agent-based elastic cloud bag-of-tasks concurrent scheduling. Future Gen. Comput. Syst. 29 (7), 1682–1699.

Hanumaiah, V., Vrudhula, S., 2012. Temperature-aware DVFS for hard real-time applications on multicore processors. IEEE Trans. Comput. 61 (10), 1484–1494.

Haque, M., Aydin, H., Zhu, D., 2016. On reliability management of energy-aware real-time systems through task replication. IEEE Transactions on Parallel and Distributed Systems.

Hazucha, P., Svensson, C., 2000. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. IEEE Trans Nucl Sci 47 (6), 2586–2594.

Huang, H., Chaturvedi, V., Quan, G., Fan, J., Qiu, M., 2014a. Throughput maximization for periodic real-time systems under the maximal temperature constraint. ACM Trans. Embedded Comput. Syst. 13 (2s), 1–22.

Huang, P., Yang, H., Thiele, L., 2014b. On the scheduling of fault-tolerant mixed-criticality systems. In: Proceedings of International Conference on Design Automation, pp. 1–6.

Huang, W., Ghosh, S., Velusamy, S., Sankaranarayanan, K., Skadron, K., Stan, M., 2006. Hotspot: a compact thermal modeling methodology for early-stage VLSI design. IEEE Trans. Very Large Scale Integ. (VLSI) Syst. 14 (5), 501–513.

Huang, W., Stan, M., Skadron, K., Sankaranarayanan, K., Ghosh, S., Velusaml, S., 2004. Compact thermal modeling for temperature-aware design. In: Proceedings of International Conference on Design Automation, pp. 878–883.

Jayaseelan, R., Mitra, T., 2008. Temperature aware task sequencing and voltage scaling. In: Proceedings of International Conference on Computer-Aided Design, pp. 618–623.

Kim, K., Buyya, R., Kim, J., 2007. Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. In: Proceedings of the International Symposium on Cluster Computing and the Grid, pp. 541–548.

Kim, N., Austin, T., Baauw, D., Mudge, T., Flautner, K., Hu, J., Irwin, M., Kandemir, M., Narayanan, V., 2003. Leakage current: Moore's law meets static power. Computer 36 (12), 68–75.

Korte, B., Vygen, J., Korye, B., Vygen, J., 2002. Combinatorial Optimization. Berlin. Germany: Springer.

Kriebel, F., Rehman, S., Sun, D., Shafique, M., Henkel, J., 2014. ASER: adaptive soft error resilience for reliability-heterogeneous processors in the dark silicon era. In: Proceedings of International Conference on Design Automation, pp. 1–6.

Li, K., Tang, X., Li, K., 2014. Energy-efficient stochastic task scheduling on heterogeneous computing systems. IEEE Trans. Parallel Distrib. Syst. 25 (11), 2867–2876.

Li, X., P. Bose, S.V., River, J., 2007. Architecture-level soft error analysis: examining the limits of common assumptions. In: Proceedings of International Conference on Dependable Systems and Networks, pp. 266–275.

Liu, C., Layland, J., 1973. Scheduling algorithms for multiprogramming in a hard real-time environment. J. ACM 20 (1), 46–61.

Ma, Y., Chantem, T., Dick, R., Wang, S., Hu, X., 2017. An on-line framework for improving reliability of real-time systems on big-little type MPSoCs. In: Proceedings of Design, Automation & Test in Europe.

Netto, M., Buyya, R., 2009. Offer-based scheduling of deadline-constrained bag-of-tasks applications for utility computing systems. In: Proceedings of the International Symposium on Parallel & Distributed Processing, pp. 1–11.

Nvidia, Variable SMP (4-plus-1tm) a multi-core CPU architecture for low power and high performance. [Online]. Available: https://www.nvidia.com/content/PDF/tegra_white_papers.

Quan, G., Chaturvedi, V., 2010. Feasibility analysis for temperature constraint hard real-time periodic tasks. IEEE Trans. Ind. Inf. 6 (3), 329–339.

Rajendran, C., Ziegler, H., 2004. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. Eur J Oper Res 155 (2), 426–438.

Riera, M., Canal, R., Abella, J., Gonzalez, A., 2016. A detailed methodology to compute soft error rates in advanced technologies. In: Proceedings of Design, Automation & Test in Europe, pp. 212–222.

Saha, S., Lu, Y., Deogun, J., 2012. Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems. In: Proceedings of International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 41–50.

Skadron, K., Stan, M., Sankaranarayanan, K., Huang, W., Velusamy, S., Tarjan, D., 2004. Temperature-aware microarchitecture: modeling and implementation. ACM Trans. Archit. Code Optim. 1 (1), 94–125.

Tan, C., Muthukaruppan, T., Mitra, T., Ju, L., 2015. Approximation-aware scheduling on heterogeneous multi-core architectures. In: Proceedings of Asia and South Pacific Design Automation Conference, pp. 618–623.

Vajda, A., 2011. Multi-Core and Many-Core Processor Architectures. Springer US, pp. 9–43.

Wang, X., Yeo, C., Buyya, R., Su, J., 2011. Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. Future Gen. Comput. Syst. 27 (8), 1124–1134.

Wang, Y., Jiang, J., Zhang, H., Dong, X., Wang, L., Ranjan, R., Zomaya, A., 2017. A scalable parallel algorithm for atmospheric general circulation models on a multi–core cluster. Future Gen. Comput. Syst. 72, 1–10.

Wei, T., Mishra, P., Wu, K., Zhou, J., 2012. Quasi-static fault-tolerant scheduling schemes for energy-efficient hard real-time systems. J. Syst. Software 85 (6), 1386–1399.

Zhang, S., Wu, J., Lu, S., 2016. Distributed workload dissemination for makespan minimization in disruption tolerant networks. IEEE Trans. Mob. Comput. 15 (7), 1661–1673.

Zhao, B., Aydin, H., Zhu, D., 2009. Enhanced Reliability-aware Power Management through Shared Recovery Technique. In: Proceedings of International Conference on Computer-Aided Design, pp. 63–70.

Zheng, W., Sakellariou, R., 2013. Stochastic DAG scheduling using a monte carlo approach. J. Parallel Distrib. Comput. 73 (12), 1673–1689.

Zhou, J., Hu, X., Ma, Y., Wei, T., 2016a. Balancing lifetime and soft-error reliability to improve system availability. In: Proceedings of Asia and South Pacific Design Automation Conference, pp. 685–690.

Zhou, J., Wei, T., 2015. Stochastic thermal-aware real-time task scheduling with considerations of soft errors. J. Syst. Softw. 102, 123–133.

Zhou, J., Wei, T., Chen, M., Yan, J., Hu, X., Ma, Y., 2016b. Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoc systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 35 (8), 1269–1282.

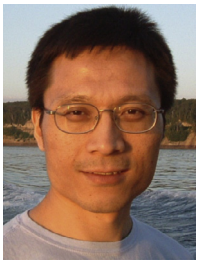
Zhou, J., Yan, J., Chen, J., Wei, T., 2016c. Peak temperature minimization via task allocation and splitting for heterogeneous MPSoc real-time systems. J. Signal Process. Syst. 84, 111–121.

Zhu, D., Melhem, R., Mosse, D., 2004. The effects of energy management on reliability in real-time embedded systems. In: Proceedings of International Conference on Computer-Aided Design, pp. 35–40.

Junlong Zhou received the Ph.D. degree in Computer Science from East China Normal University, Shanghai, China, in 2017. He was a Research Visitor 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, cyber physical systems, and cloud computing. He has published a dozen of papers in referred journals and conferences. 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. He is a member of the IEEE.

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 high performance computing, multiprocessor systems-on-chip and cyber physical systems.

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.



Tongquan Wei received the Ph.D. degree in electrical engineering from Michigan Technological University, Houghton, MI, USA, in 2009. He is currently an Associate Professor with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interests include real-time embedded systems, green and reliable computing, parallel and distributed systems, and cloud computing. Dr. Wei has been a Regional Editor for the Journal of Circuits, Systems, and Computers since 2012. He served as a Guest Editor for several special sections of the IEEE Transactions on Industrial Informatics and ACM Transactions on Embedded Computing Systems. He is a member of the IEEE.

Mingsong Chen received the B.S. and M.E. degrees from the 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, FL, USA, in 2010. He is currently a full Professor with the Department of Embedded Software and Systems, East China Normal University, Shanghai, China. His current research interests include design automation of cyber-physical systems, formal verification techniques, and mobile cloud computing. He is a member of the IEEE.

Gongxuan Zhang received the BEng degree in Computing from Tianjin University and the MEng and PhD degrees in Computer Application from the Nanjing University of Science and Technology. Also, he was a Senior Visiting Scholar in Royal Melbourne Institute of Technology from 2001.9 to 2002.3. Since 1991, he has been with the Nanjing University of Science and Technology, where he is currently a professor in the School of Computer Science and Engineering. He is a senior member of the IEEE.

Jianming Yan received the master's degree from the Department of Computer Science and Technology, East China Normal University, Shanghai, China, in 2016. He is currently a senior software engineer with Meituan.com Corporation, Beijing, China. His research interests include task allocation and scheduling techniques in heterogeneous real-time MPSoC systems.

Yue Ma received the bachelor's degree from Chengdu University of Technology, Chengdu, China, in 2010. He received the master's degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013. He is currently working toward the Ph.D. degree with the University of Notre Dame, Notre Dame, IN, USA. His current research interests are in the areas of high performance computing, multiprocessor systems-on-chip and cyber physical systems. He has authored or coauthored over 20 research papers in the related areas. He is a student member of the IEEE.