IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 July 2023, accepted 3 August 2023, date of publication 7 August 2023, date of current version 16 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3302703

RESEARCH ARTICLE

Federated Scheduling Optimization Scheme for Typed Tasks With Power Constraints in Heterogeneous Multicore Processor Architectures

XIAOHONG WEN[1], GUOJIN LIU[1], DEJIAN LI[2], (Member, IEEE), YANTAO YU[1], (Member, IEEE), HAISEN ZHAO[3], AND TIANCONG HUANG[1]

[1]School of Microelectronics and Communication Engineering, Chongqing University, Chongqing 400044, China
[2]Beijing Smartchip Microelectronics Technology Company Ltd., Beijing 100192, China
[3]State Grid Jinzhong Electric Power Supply Company, Jinzhong 030600, China

Corresponding author: Yantao Yu (yantaoyu@cqu.edu.cn)

This work was supported in part by the 2020 State Grid Corporation of China Science and Technology Program under Grant 5700-202041395A-0-0-00.

ABSTRACT In heterogeneous multicore processor architectures, it is a critical concern to optimize the performance of typed tasks (real-time and non-real-time tasks) under limited power consumption. In this paper, we propose a power-constrained federated scheduling optimization scheme for typed tasks based on both global and local scheduling systems. The global scheduling system adopts a federated scheduling strategy to split the typed task stream into multiple real-time and non-real-time sub-streams. The local scheduling systems are constructed as a set of M/M/c/m queueing systems with heterogeneous multicore processors considering task priority and finite queueing capacity, avoiding task blocking and resource wastage through finite cache and parallel execution of two types of task sub-flows. To meet the deadline constraints of real-time tasks, a real-time task queueing model with strong preemption priority is constructed, and a stable and efficient real-time scheduling algorithm based on sequential quadratic programming block homotopy is proposed, which can ensure the optimal distribution of real-time tasks while maintaining load balancing and schedulability. For non-real-time tasks, we propose a dichotomous search-based scheduling algorithm to minimize the average response time of non-real-time tasks under strong preemption constraints on real-time tasks, providing a theoretical analysis proof of the optimal processor speed configurations for heterogeneous multiprocessor systems under power constraints. Simulation results demonstrate that factors such as processor size, system capacity, available power, and task arrival rate have a significant impact on the system performance, and that the proposed scheme enables optimal load distribution for typed tasks with power constraints in multiprocessor queueing systems.

INDEX TERMS Federated scheduling, heterogeneous multicore processor architecture, M/M/c/m queueing system, non-real-time tasks, real-time tasks.

I. INTRODUCTION

A. MOTIVATION

Heterogeneous multicore processors have gained widespread adoption in recent years for mobile electronics, such as smartphones, automated robots, and wearable devices, owing to their superior computational performance and parallel task processing capabilities [1], [2], [3]. However, due to the sluggish progress in battery technology, mobile devices are constrained by strict power consumption requirements while aiming to enhance performance [4], [5], [6]. Conventional performance enhancement techniques predominantly rely on

The associate editor coordinating the review of this manuscript and approving it for publication was Thomas Canhao Xu.

raising the core frequency to boost performance, which in turn results in increased power consumption. Therefore, maximizing performance in heterogeneous multicore processors with limited power consumption is a critical issue.

The performance of heterogeneous multicore systems can be optimized by load balancing [7]. Various related approaches have been proposed, such as game theory [8], [9] and queueing theory [10]. Queueing-based load balancing methods are commonly employed in scenarios with a large number of task requests characterized by uncertain arrival times and task sizes. Several queueing models, such as $M/M/1/\infty$ [11], [12], $M/G/1/\infty$ [13], $M/M/K/\infty$ [14], $G/G/K/\infty$ [15], have been used by researchers to depict the stochastic behavior of tasks. However, most reported studies ideally assume the processor has an infinite queueing capacity without any limit, it may not be practical in resource-constrained heterogeneous multiprocessor systems. Moreover, it may result in congestion, overload and resource exhaustion as the number of tasks increases, and the system performance would be degraded in resource-constrained systems. Hence, it is essential to judiciously control the queueing system capacity to enhance the system performance and avoid wasting system resources.

In multicore processor queueing systems, to further improve system performance, many researchers typically perform task scheduling based on task type. Some studies focus on a single task type [12], [16], while others examine multiple types of tasks [11], [17], [18]. Most scheduling systems use a preloading policy for multiple types of tasks, which considers a specific task type as preloaded and studies load scheduling policies for the other types of tasks. Indeed, in resource-constrained multiprocessor systems, preloading strategies may lead to problems such as poor resource utilization, and limited system flexibility. In contrast, federated scheduling can flexibly allocate processor resources based on the characteristics of each task type and actual system requirements, effectively avoiding idle or overloaded resources. Therefore, federated scheduling for typed tasks has become a current research trend [19], [20].

The widespread adoption of real-time applications in heterogeneous multiprocessor systems has facilitated the categorization of typed tasks into two primary types: real-time and non-real-time tasks. Extensive research has been devoted to real-time task scheduling, covering areas such as schedulability analysis [21], reliable scheduling [22], and energy-efficient scheduling [23], [24]. Unfortunately, most of the existing literature primarily concentrates on the presence of real-time tasks, disregarding the resource competition posed by non-real-time tasks in resource-constrained multiprocessor systems. This oversight could cause real-time tasks to fail to meet their schedulability requirements when both task types coexist. Therefore, for both real-time and non-real-time task sets in resource-constrained multiprocessor systems, it is expected that efficient federated scheduling strategies can be introduced to rationally allocate processor resources and maximize the scheduling performance for non-real-time tasks while guaranteeing the schedulability for real-time tasks.

B. OUR CONTRIBUTIONS

In this paper, a federated scheduling optimization scheme for real-time and non-real-time tasks in heterogeneous multicore architectures is proposed to optimize the performance under limited power consumption. For typed tasks, queueing models that consider task priority and finite queueing capacity are constructed to avoid system blockage and resource wastage by effectively controlling the system queueing capacity. Furthermore, for system power constraints and real-time task deadlines, a federated scheduling strategy is employed to efficiently allocate tasks and ensure load balancing to achieve flexible allocation of system resources, and enhance the overall scheduling performance of the system.

The main contributions of this paper are summarized as follows.

- A hybrid task federated scheduling framework, which consists of a global scheduling system and multiple local scheduling systems, is proposed in this paper. Based on this framework, a real-time task queueing model with strong preemptive priority and a non-real-time task queueing model with low priority are constructed to characterize the stochastic behavior of typed tasks. Then, a federated scheduling optimization problem is formulated for both real-time and non-real-time tasks with consideration of system power and real-time task deadlines.
- To satisfy the real-time characteristics for real-time tasks, a Bounded-Constrained Augmented Lagrange Sequence Quadratic Programming Block Homotopy (BCAL-SQP-BHom) algorithm is used to iteratively update the load balancing strategy, and minimizes the maximum response time of real-time tasks while meeting the schedulability requirements. The algorithm utilizes a blocking technique to partition the scheduling problem into multiple sub-problems, which reduces the complexity of the problem-solving and improves the computational efficiency.
- To enhance the scheduling performance for non-real-time tasks, a Lagrangian-based mathematical approach to proposed to simplify the solution of the problem, and theoretical analysis of optimal speed configurations of processors in heterogeneous multiprocessor systems under power constraints is provided, and a dichotomous search algorithm is introduced to minimize the average response time of non-real-time tasks under the constraint of the strong preemption for real-time tasks.

The rest of this paper is organized as follows: The related work is reviewed in Section II. Section III outlines the system model and problem formulation. The federated scheduling optimization scheme for the typed tasks under power constraints is described in Section IV, and a detailed analysis of the simulation results of the proposed scheme is provided in Section V. Finally, the conclusions are drawn in Section VI.

II. RELATED WORK

In this section, the performance optimization or power optimization for heterogeneous multicore processor systems is reviewed from two perspectives: typed tasks federated scheduling and queueing system. TABLE 1 summarizes the related literature.

TABLE 1. Summary of related research.

Reference	Environment	Model	Task Type	Scheme(s)
Typed Tasks Federated Scheduling				
Han *et al.* [20]	Heterogeneous computing	DAG	Heavy tasks and light tasks	Federated scheduling for typed tasks
Ramegowda *et al.* [25]	Heterogeneous computing	DAG	Periodic and aperiodic tasks	Power-saving scheduling for typed tasks
Goubaa *et al.* [26]	Heterogeneous computing	DAG	Periodic and aperiodic tasks	Real-Time scheduling for typed tasks
Chang *et al* [27]	Heterogeneous computing	DAG	Multiple types of particular tasks	Real-Time scheduling for typed tasks
Chang *et al.* [28]	Heterogeneous computing	DAG	Multiple types of particular tasks	Exact calculation of WCRT for typed tasks
Queueing System				
Li *et al.* [16]	Data center	$G/G/1/\infty$	Unified tasks	Optimal server speed setting
He *et al.* [29]	Mobile edge computing	$M/M/c/\infty$	Offloadable tasks	Optimal server configuration and placement
Li *et al.* [30]	Cloud computing	$M/G/1/\infty$	Multiple types of applications	Optimal load distribution and server speed setting
Huang *et al.* [31]	Heterogeneous computing	$M/G/1/\infty$	Dedicated tasks and general tasks	Optimal load distribution and server speed setting
Huang *et al.* [11][17]	Heterogeneous computing	$M/M/1/\infty$ with three kinds of priorities	Dedicated tasks and general tasks	Optimal load balancing and power allocation
Our work	Heterogeneous computing	$M/M/c/m$ with a preemption priority	Real-time tasks and non-real-time tasks	Optimal load distribution and load balancing

A. TYPED TASKS FEDERATED SCHEDULING

Recently, there has been an increase in research on federated scheduling of typed tasks for heterogeneous multicore processors [20], [23], [24], [25], [26], [27], [28]. In [20], Han et al. proposed federated scheduling methods for typed Directed Acyclic Graph (DAG) tasks, in which tasks were classified into heavy and light tasks. Each heavy task was executed exclusively on multiple dedicated processors, and the light tasks were considered as sequential sporadic tasks and handled by the remaining processors. Moulik et al. suggested a low-overhead multi-stage heuristic strategy for energy-aware scheduling based on Dynamic Voltage and Frequency Scaling (DVFS) for a set of real-time periodic tasks on a heterogeneous multicore platform [23], which, however, suffers from high context switching/migration overheads. They further designed an improved strategy [24], which employs a half-partitioning approach to generate fewer inter-core task migrations. Ramegowda and Lin [25] extended the classical well-known DVFS technique cycle-conserving algorithm to handle typed task sets (periodic and nonperiodic tasks) and implement power-efficient task scheduling on Free Real-Time Operating Systems (FreeRTOS). In [26], Goubaa et al. addressed the real-time scheduling for multicore systems powered by renewable energy, in which periodic and aperiodic tasks were statically mapped to the kernel and were not allowed to migrate. Chang et al. investigated the real-time scheduling of typed DAG tasks on heterogeneous multicore platforms, a scheduling method that utilizes a criticality assignment policy was presented to assign different criticalities for each vertex [27], and a satisfiability modulo theories-based approach was proposed to accurately analyze the Worst-Case Response Time (WCRT) of the typed task [28]. It is clear that while there have been numerous excellent studies on the federated scheduling of typed tasks, most of them employ DAG to characterize typed tasks with explicit dependencies and static scheduling properties, while ignoring the randomness or uncertainty of typed tasks, which is more common in heterogeneous multiprocessor systems.

B. QUEUEING SYSTEM

A growing number of existing studies had used queueing systems to model the system performance of heterogeneous multicore processors with system power constraints [11], [16], [17], [29], [30], [31]. Li et al. [16] investigated power-constrained performance optimization and performance-constrained power optimization for data centers with multiple heterogeneous and arbitrary servers with $G/G/1/\infty$ queueing systems by setting the optimal speed of servers. He et al. [29] studied the federated optimization problem of edge server configuration (treating each edge server as a $M/M/c/\infty$ queueing model) in an edge computing environment to minimize operational expenses with system performance at a predetermined level. It was evident that literature [16] and [29] focused on studying one type of task, but the diversity of tasks was ignored. Li et al. [30] explored

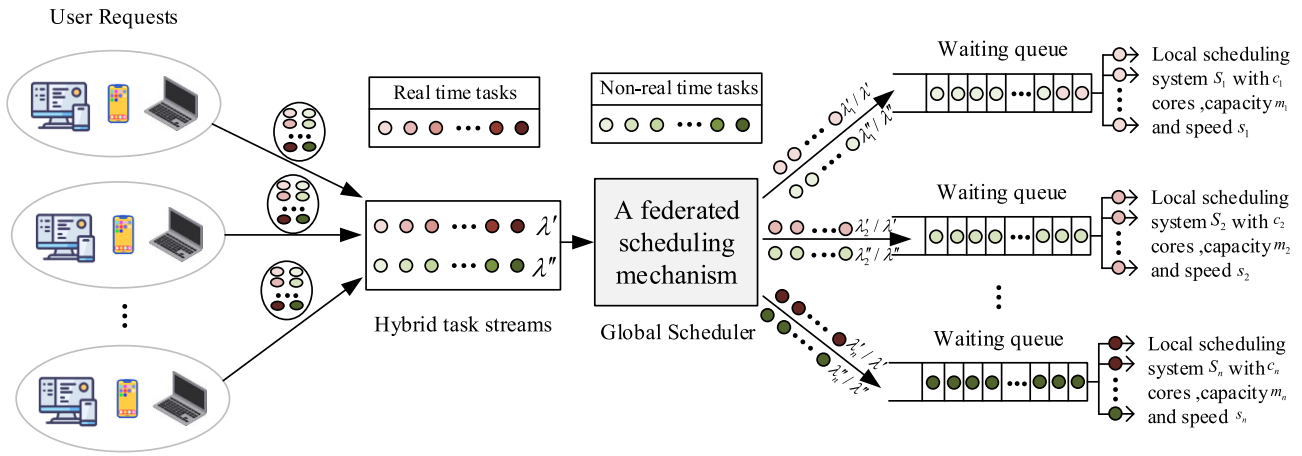

FIGURE 1. The proposed hybrid task federated scheduling framework.

techniques for variable and task-type dependent server speed management by treating each server as a $M/G/1/\infty$ queueing system with mixed task classes, and the optimal load distribution and optimal server speed settings were found to minimize the average task response time. In [31], Huang et al. minimized the average response time of general tasks on heterogeneous processors (treated as $M/G/1/\infty$ queueing systems) through optimal load distribution and optimal speed settings, where each processor had its own preloaded dedicated tasks and the processors had different queueing rules for scheduling dedicated and general tasks. Similar work can be found in [11] and [17]. Previous studies have primarily concentrated on load distribution for a certain class of tasks in heterogeneous multiprocessor queueing systems, usually assuming the ideal scenario of unlimited caching for all tasks. Unfortunately, this assumption may result in wasted system resources and performance degradation in resource-limited heterogeneous multiprocessor systems.

Different from the aforementioned studies, a federated scheduling scheme for typed tasks (real-time and non-real-time tasks) is proposed to optimize the system performance in multiple heterogeneous multiprocessors with power constraints and limited queueing capacity by considering various impact factors such as system capacity, available power, load balancing, and deadline of real-time tasks. The overall system performance is optimized by designing capacity-limited queueing systems and employing a federated scheduling strategy for typed tasks to effectively manage the task queue length and properly allocate processor resources. The proposed strategy can flexibly perform task scheduling based on queue state, task characteristics, and task priority to accommodate changes in the stochastic behavior of typed tasks.

III. SYSTEM MODEL

A hybrid task federated scheduling framework consisting of a global scheduling system and multiple local scheduling systems, shown in Figure 1, is constructed to optimize the performance of typed tasks in multiple heterogeneous multiprocessors with power constraints. Since the task arrival times and sizes are stochastic and uncertain, queueing theory is utilized to model the system. Suppose that the scheduling objects of the global scheduling system are n heterogeneous local scheduling systems. Each local scheduling system S_i with c_i homogeneous processors and processor speed s_i has queueing capacity of m_i, where $1 \leq i \leq n$. Given the multicore parallelism requirements and system resource constraints of heterogeneous systems, the local scheduling system in this paper is modeled as a $M/M/c/m$ queueing system with the number of processor cores as c and the system queueing capacity as m, which is elaborated as follows.

The global scheduling system accepts a real-time task Poisson stream with arrival rate λ' (measured by tasks per second), i.e., the arrival interval is an independent and identically distributed exponential random variable with mean $1/\lambda'$. The global scheduling system divides it into n sub-streams $\lambda'_1, \lambda'_2, \ldots, \lambda'_n$, which are assigned to n local scheduling systems $S_1, S_2, \ldots, S_n$, where $\lambda' = \lambda'_1 + \lambda'_2 + \cdots + \lambda'_n$. Similarly, there is a Poisson stream of non-real-time tasks with arrival rate λ''. It is split into n sub-streams $\lambda''_1, \lambda''_2, \ldots, \lambda''_n$, and the i-th sub-stream is assigned to the local scheduling system S_i, where $1 \leq i \leq n$, $\lambda'' = \lambda''_1 + \lambda''_2 + \cdots + \lambda''_n$. Thus, the task flow to the global scheduling system is a combinatorial Poisson flow with arrival rate $\lambda' + \lambda''$.

The average task size of r (measured by the number of instructions executed) is an exponential random variable with the mean value of $\bar{r}$, and the c_i processors of the local scheduling system S_i have the same execution speed s_i (giga instructions per second). Thus, the task execution time of each core in the local scheduling system S_i is an exponential random variable and can be depicted as $x_i = r/s_i$ with mean value of $\bar{x}_i = \bar{r}/s_i$.

The local system S_i, with limited caching capability, can accommodate up to m_i task sub-streams in the queue. Due to the real-time nature of real-time tasks, real-time tasks are assigned a higher priority than non-real-time tasks, with a strong preemption priority. This means that real-time tasks can interrupt and delay the execution of non-real-time tasks, while tasks within the same class follow a first-come,

first-served rule. For simplicity, the mathematical notations are summarized in TABLE 2.

A. REAL-TIME TASKS MODEL

The average service rate per core in the local scheduling system S_i can be denoted as $u_i = 1/\bar{x}_i = s_i/\bar{r}$. If there are k tasks (executing or waiting to be executed), the total service rate of the local scheduling system S_i is

$$u_{i,k} = \begin{cases} ku_i, & 0 \le k < c_i \\ c_i u_i, & u_i \le k \le m_i. \end{cases} \tag{1}$$

As real-time tasks have strong preemption priority, the core utilization of real-time tasks is only related to the arrival rate λ_i', and can be described as

$$\rho_i' = \frac{\lambda_i'}{c_i u_i} = \frac{\lambda_i' \bar{x}_i}{c_i} = \frac{\lambda_i' \bar{r}}{c_i s_i}, \tag{2}$$

which is the service intensity or load level of real-time tasks in the local scheduling system S_i.

At this point, the probability that there are k real-time tasks in the local scheduling system S_i is

$$p_{i,k}' = \begin{cases} \frac{(c_i\rho_i')^k}{k!} p_{i,0}', & 0 \le k < c_i \\ \frac{c_i^{c_i}(\rho_i')^k}{c_i!} p_{i,0}', & c_i \le k \le m_i, \end{cases} \tag{3}$$

where

$$p_{i,0}' = \left(\sum_{k=0}^{c_i-1} \frac{(c_i\rho_i')^k}{k!} + \frac{(c_i\rho_i')^{c_i}}{c_i!} \times \frac{1}{1-p_i'}\right)^{-1}. \tag{4}$$

The average number of cores processing real-time tasks in system S_i can be expressed as

$$\begin{aligned} Ls_i' &= \sum_{k=0}^{c_i-1} k p_{i,k}' + c_i \sum_{k=c_i}^{m_i} p_{i,k}' \\ &= c_i\rho_i'(1 - p_{i,m_i}'). \end{aligned} \tag{5}$$

Also, the average queue length for real-time tasks is

$$\begin{aligned} Lq_i' &= \sum_{k=c_i}^{m_i} (k - c_i) p_{i,k}' \\ &= \frac{(c_i\rho_i')^{c_i}\rho_i' p_{i,0}'}{c_i!} \left(\frac{1-(\rho_i')^{m_i-c_i+1}}{1-\rho_i'}\right)' \\ &= \frac{(c_i\rho_i')^{c_i}\rho_i' p_{i,0}'}{c_i!(1-\rho_i')^2} (1 - (m_i - c_i + 1)(\rho_i')^{m_i-c_i} \\ &\quad + (m_i - c_i)(\rho_i')^{m_i-c_i+1}). \end{aligned} \tag{6}$$

Then the average number of real-time tasks in system S_i can be written as

$$L_i' = Ls_i' + Lq_i'. \tag{7}$$

Applying Little's result, we get the average response time of real-time tasks in the system S_i as

$$T_i' = \frac{L_i'}{\lambda_i'} = \frac{Ls_i' + Lq_i'}{\lambda_i'}. \tag{8}$$

TABLE 2. Mathematical notations.

Symbol	Definition
	Inputs
S_i	Local scheduling system i , $1 \le i \le n$.
c_i	Number of processors of S_i
m_i	System capacity of S_i
P_i^*	Static power of S_i
α_i	Power consumption exponent of S_i
T_{dead}	Given deadlines for real-time tasks
$\tilde{P}$	Given system power
λ	Total arrival rate of typed tasks
λ'	Arrival rate of real-time tasks
λ''	Arrival rate of non-real-time tasks
$\bar{r}$	Average task size
	Variables
s_i	Processor speed of S_i
λ_i'	Arrival rate of real-time to S_i
λ_i''	Arrival rate of non-real-time to S_i
	Intermediate variables
$\bar{x}_i$	Average task execution time in S_i
u_i	Average service rate of the processor in S_i
$u_{i,k}$	Total service rate when local scheduling system S_i is in state k
ρ_i'	Service intensity of real-time tasks in S_i
$p_{i,k}'$	The probability of k real-time tasks existing in S_i
Ls_i'	Average number of cores processing real-time tasks in S_i
Lq_i'	Average queue length of real-time tasks in S_i
L_i'	Average number of real-time tasks in S_i
T_i'	Average response time of real-time tasks in S_i
T'	Average response time of real-time tasks in the global scheduling system
$T'_{\max}$	Maximum response time of real-time tasks in the global scheduling system
ρ_i''	Service intensity of typed tasks in S_i
$p_{i,k}''$	The probability of k typed tasks existing in S_i
T_i''	Average response time of non-real-time tasks in S_i
T_i	Average response time of typed tasks in S_i
T''	Average response time of non-real-time tasks in the global scheduling system
P_i	Average power consumption of S_i

B. NON-REAL-TIME TASKS MODEL

Both real-time and non-real-time tasks in the local scheduling system S_i are assumed to follow a negative exponential distribution. The average response time T_i of each typed task can be calculated according to the formula of the $M/M/c/m$ queueing model with arrival rate $\lambda_i' + \lambda_i''$. According to (5), (6), and (7), we get

$$\begin{aligned} T_i &= \frac{L_i}{\lambda_i' + \lambda_i''} \\ &= \frac{\sum_{k=0}^{c_i-1} k p_{i,k}'' + c_i \sum_{k=c_i}^{m_i} p_{i,k}'' + \sum_{k=c_i}^{m_i} (k-c_i) p_{i,k}''}{\lambda_i' + \lambda_i''}, \end{aligned} \tag{9}$$

where the probability that there are k tasks in system S_i is

$$p_{i,k}'' = \begin{cases} \frac{(c_i \rho_i'')^k}{k!} p_{i,0}'', & 0 \le k < c_i \\ \frac{c_i^{c_i} (\rho_i'')^k}{c_i!} p_{i,0}'', & c_i \le k \le m_i, \end{cases}$$

where

$$\rho_i'' = \frac{\lambda_i' + \lambda_i''}{c_i u_i} = \frac{\bar{x}_i}{c_i}(\lambda_i' + \lambda_i'') = \frac{\bar{r}}{c_i s_i}(\lambda_i' + \lambda_i''),$$

and

$$p_{i,0}'' = \left(\sum_{k=0}^{c_i-1} \frac{(c_i \rho_i'')^k}{k!} + \frac{(c_i \rho_i'')^{c_i}}{c_i!} \times \frac{1}{1-\rho_i''}\right)^{-1}.$$

Let T_i'' denote the average response time of non-real-time tasks in system S_i, according to $(\lambda_i' + \lambda_i'')T_i = \lambda_i' T_i' + \lambda_i'' T_i''$, we have

$$\begin{aligned} T_i'' &= \left(\frac{\lambda_i' + \lambda_i''}{\lambda_i''}\right) T_i - \frac{\lambda_i'}{\lambda_i''} T_i' \\ &= \frac{1}{\lambda_i''}(L_i - L_i'). \end{aligned} \tag{10}$$

C. POWER MODEL

The power consumption of the processor can be mainly divided into dynamic and static power consumption, with the dynamic power consumption being dominant. It is assumed that $P = \kappa C V^2 f$ denotes the processor dynamic power consumption, where κ is the activity factor, C is the load capacitance, V is the supply voltage, and f is the clock frequency [32]. Since $s \propto f$, where s is the processor speed and $f \propto V^{\varsigma}$, where $0 < \varsigma \le 1$ [33]. From this, we can obtain the processor power consumption $P \propto s^{\alpha}$, where $\alpha = 1 + 2/\varsigma \ge 3$. For the sake of discussion, the dynamic and static power consumption of a processor core in system S_i, are denoted as $s_i^{\alpha_i}$ and P_i^*, respectively. Thus, the power model of a processor with speed s_i can be expressed as $s_i^{\alpha_i} + P_i^*$. In this paper, we design a variable frequency power model in which the processor only consumes static power P_i^* when it is idle. In system S_i, ρ_i'' denotes the average percentage of time per unit time that a processor is executing a task, and then $1 - \rho_i''$ represents the average percentage of time that a processor is idle. From this, the average power consumption per unit time in system S_i with c_i processor cores can be modeled as

$$\begin{aligned} P_i &= c_i(\rho_i''(s_i^{\alpha_i} + P_i^*) + (1-\rho_i'')P_i^*) \\ &= (\lambda_i' + \lambda_i'')\bar{r} s_i^{\alpha_i - 1} + c_i P_i^*. \end{aligned} \tag{11}$$

D. PROBLEM FORMULATION

According to (8), we can obtain the average response time of real-time tasks for n heterogeneous local scheduling systems in the global scheduling system as

$$\begin{aligned} T' &= \frac{\lambda_1'}{\lambda'} T_1' + \frac{\lambda_2'}{\lambda'} T_2' + \cdots + \frac{\lambda_n'}{\lambda'} T_n' \\ &= \frac{1}{\lambda'} \sum_{i=1}^{n} L_i'. \end{aligned} \tag{12}$$

Given the time limit constraint of real-time tasks and the parallel processing characteristics of the computing system, we take the maximum value of the average response time of real-time tasks in system S_i as its maximum response time, i.e., according to (8) we have

$$T'_{\max} = \max_{i \in N} T_i', \tag{13}$$

where $i \in N = \{1, 2, \ldots, n\}$.

Assuming that the deadline of the real-time task is T_{dead}, we need to ensure that its maximum response time does not exceed the deadline to satisfy the schedulability of the real-time task, namely

$$T'_{\max} \le T_{dead}. \tag{14}$$

Moreover, according to (10), the average response time of non-real-time tasks for n heterogeneous local scheduling systems $S_1, S_2, \ldots, S_n$ in the global scheduling system is

$$\begin{aligned} T'' &= \frac{\lambda_1''}{\lambda''} T_1'' + \frac{\lambda_2''}{\lambda''} T_2'' + \cdots + \frac{\lambda_n''}{\lambda''} T_n'' \\ &= \frac{1}{\lambda''} \sum_{i=1}^{n} (L_i - L_i'). \end{aligned} \tag{15}$$

Note that the above T'' can be viewed as a function of the real-time task load distribution, the non-real-time task load distribution, and the processor core speed, that is, $T''(\lambda_1', \lambda_2', \ldots, \lambda_n', \lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n)$.

Similarly, the average power consumption $P(\lambda_1', \lambda_2', \ldots, \lambda_n', \lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n)$ of n heterogeneous local scheduling systems is also a function of the real-time task load distribution, the non-real-time task load distribution, and the processor core speed. Under the variable frequency power model, we have

$$\begin{aligned} &P(\lambda_1', \lambda_2', \ldots, \lambda_n', \lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n) \\ &= \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} ((\lambda_i' + \lambda_i'')\bar{r} s_i^{\alpha_i - 1} + c_i P_i^*) \\ &= \sum_{i=1}^{n} (\lambda_i' + \lambda_i'')\bar{r} s_i^{\alpha_i - 1} + \sum_{i=1}^{n} c_i P_i^*. \end{aligned} \tag{16}$$

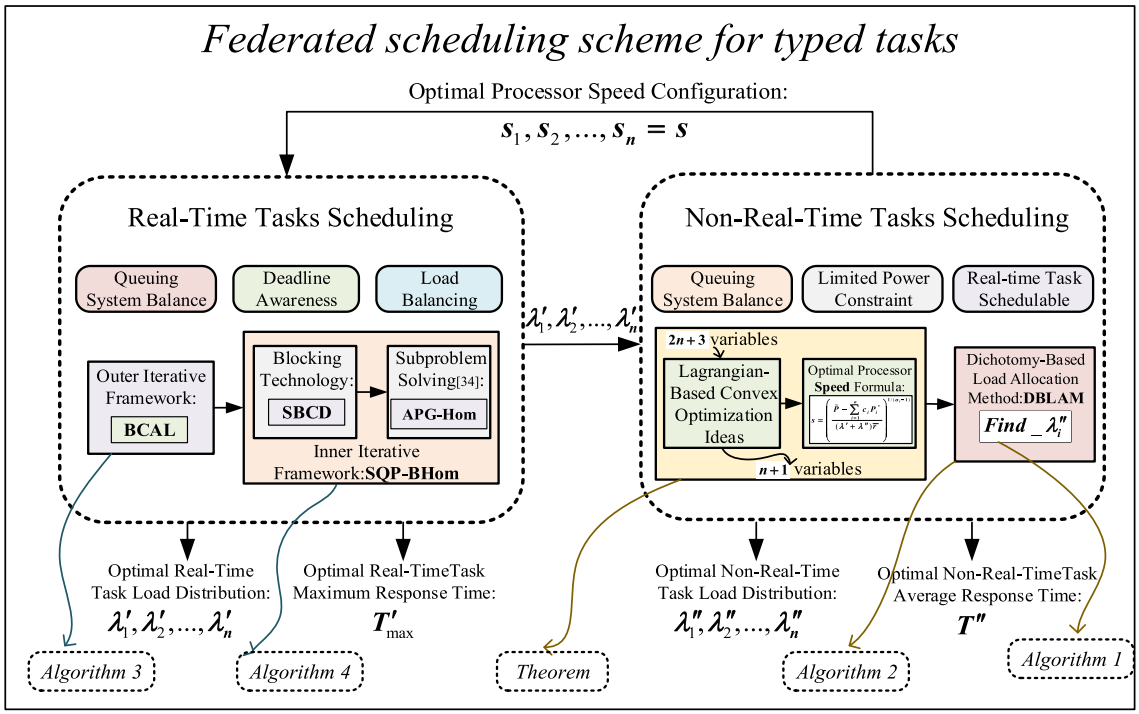

FIGURE 2. Global illustration of a federated scheduling scheme for typed tasks.

Here, we formulate the federated scheduling optimization problem for typed tasks. Under the constraints of multicore processor power consumption and real-time task deadlines, we analyze the schedulability of real-time tasks and optimize the average response time of non-real-time tasks to ensure optimal load distribution and real-time scheduling of tasks. Therefore, our objective problem can be formulated as follows: given n multicore local scheduling systems, the number of system processor cores $c_1, c_2, \ldots, c_n$, the system capacity $m_1, m_2, \ldots, m_n$, the static power consumption $P_1^*, P_2^*, \ldots, P_n^*$, the average task size $\bar{r}$, the real-time task arrival rate λ', the non-real-time task arrival rate λ'', the real-time task deadline T_{dead}, and the available power $\tilde{P}$, find the real-time task load distribution $\lambda_1', \lambda_2', \ldots, \lambda_n'$, the non-real-time task load distribution $\lambda_1'', \lambda_2'', \ldots, \lambda_n''$, and the processor speed $s_1, s_2, \ldots, s_n$ on the local scheduling systems, such that the average response time of the non-real-time tasks is minimized under power constraints, while the maximum response time of the real-time tasks is satisfied without exceeding the deadline. That is, the mathematical formula is stated as follows:

$$
\begin{aligned}
P1\colon\ & \min\ T''(\lambda_1', \lambda_2', \ldots, \lambda_n', \lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n) \\
& s.t.\ C1\colon \rho_i'' < 1 \\
& \quad C2\colon \lambda_1' + \lambda_2' + \cdots + \lambda_n' = \lambda' \\
& \quad C3\colon \lambda_1'' + \lambda_2'' + \cdots + \lambda_n'' = \lambda'' \\
& \quad C4\colon T'_{\max}(\lambda_1', \lambda_2', \ldots, \lambda_n', s_1, s_2, \ldots, s_n) \le T_{dead} \\
& \quad C5\colon P(\lambda_1', \lambda_2', \ldots, \lambda_n', \lambda_1'', \lambda_2'', \ldots, \\
& \qquad \lambda_n'', s_1, s_2, \ldots, s_n) \le \tilde{P}, \qquad (17)
\end{aligned}
$$

where the constraint $C1$ means that the system is in equilibrium, that is, $\lambda_i'' < c_i s_i/\bar{r} - \lambda_i'$, where $1 \le i \le n$. $C2$ denotes the total arrival rate constraint for real-time tasks. $C3$ is the total arrival rate constraint for non-real-time tasks. $C4$ indicates that the maximum response time of real-time tasks does not exceed the deadline, that is, the time limit constraint for real-time tasks, where $T'_{\max}$ is a function of the real-time task distribution $\lambda_1', \lambda_2', \ldots, \lambda_n'$ and the processor speed $s_1, s_2, \ldots, s_n$. $C5$ implies that the system's average power consumption does not exceed the available power, i.e., the system power constraint.

IV. TYPED TASKS FEDERATED SCHEDULING OPTIMIZATION SCHEME

A federated scheduling optimization scheme for typed tasks with power constraints is adapted to the global scheduling system to solve the optimization problem $P1$ represented by (17). The scheme comprises two parts: real-time task scheduling and non-real-time task scheduling, as illustrated in Figure 2.

In real-time task scheduling, a real-time scheduling algorithm based on block homotopy of sequential quadratic programming is utilized to optimize the load distribution of real-time tasks and minimize the maximum response time to satisfy their schedulability requirements, considering the queueing system balance, deadline awareness, and load balancing constraints. The algorithm consists of two parts, the outer iterative framework (Algorithm 3) adopts the Bounded Constrained Augmented Lagrangian method (BCAL), while the inner iterative framework (Algorithm 4) employs the Sequence Quadratic Programming Block Homo-

topy (SQP-BHom). The inner iterative framework incorporates the Stochastic Block Coordinate Descent method (SBCD) and the Accelerated Proximal Gradient Homotopy algorithm (APG-Hom) to partition and solve the scheduling subproblem, respectively. The details of the algorithm are discussed in Section IV-B.

Non-real-time task scheduling is based on real-time task scheduling, taking into consideration constraints such as queueing system balance, finite system power, and real-time task schedulability. It presents the optimal processor speed allocation formula (Theorem) based on Lagrange's mathematical approach, which simplifies the original problem from $2n+3$ variables to $n+1$ variables. Furthermore, we propose a Dichotomy-Based Load Allocation Method, DBLAM (Algorithm 1, Algorithm 2), that minimizes the average response time while determining the optimal non-real-time load distribution. The details of this approach are elaborated in Section IV-A.

Moreover, in real-time task scheduling, the processor speed is dynamically adjusted based on the optimal processor speed configuration, while in non-real-time task scheduling, the optimal scheduling of non-real-time tasks is implemented based on the optimal real-time task load distribution. It can be seen that the two types of scheduling are mutually constrained and form a tightly coupled federated scheduling system.

In Section IV-A, it is assumed that the optimal real-time task load distribution $\lambda_1', \lambda_2', \ldots, \lambda_n'$ is known. It then derives the formulation and proves the theorem for optimal processor speed configurations in heterogeneous multiprocessor systems with power constraints, and presents the algorithm flow for non-real-time task scheduling. In Section IV-B, the optimal processor speed from Section IV-A is used to perform the speed configuration and solve for the optimal real-time task load distribution under the deadline constraint. The flow of algorithms for real-time task scheduling is also presented. Finally, the real-time task load distribution $\lambda_1', \lambda_2', \ldots, \lambda_n'$, obtained from the solution in Section IV-B, is used as the known input in Section IV-A to perform federated scheduling optimization for both real-time and non-real-time tasks under the system power constraint.

A. NON-REAL-TIME TASKS SCHEDULING UNDER POWER CONSTRAINTS

Since the real-time task arrival rate $\lambda_1', \lambda_2', \ldots, \lambda_n'$ is assumed to be a known variable, its associated constraint $C2$ can be first disregarded and $C4$ can be converted to

$$G(s_1, s_2, \ldots, s_n) + \beta - T_{dead} = 0, \tag{18}$$

where $G(s_1, s_2, \ldots, s_n) = \max_{i \in N} T_i'$, and $0 \le \beta < T_{dead}$, β can be solved in Section IV-B, so we can treat it as a known variable.

It is found that the optimization problem $P1$ is still a multivariate, multi-constrained optimization problem, and can be rewritten as

$$\begin{aligned} P2: \quad & \min\ T''(\lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n) \\ & s.t.\ C6: \lambda_i'' < c_i s_i / \bar{r} - \lambda_i' \\ & \quad C7: F(\lambda_1'', \lambda_2'', \ldots, \lambda_n'') - \lambda'' = 0 \\ & \quad C8: G(s_1, s_2, \ldots, s_n) + \beta - T_{dead} = 0 \\ & \quad C9: P(\lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n) - \tilde{P} = 0, \end{aligned} \tag{19}$$

where the constraints $C6$, $C7$, $C8$, $C9$ correspond to variants of $C1$, $C3$, $C4$, $C5$, respectively, and $F(\lambda_1'', \lambda_2'', \ldots, \lambda_n'') = \lambda_1'' + \lambda_2'' + \cdots + \lambda_n''$ in $C7$. Moreover, considering the limiting case of the problem, it is assumed that the power consumption in $C5$ is exactly equal to the available power, that is $P(\lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n) - \tilde{P} = 0$ in $C9$. It is well known that the above optimization problem can be solved by using the classical Lagrange multiplier method [35], but it introduces a nonlinear system of equations with $2n+3$ variables, i.e., $\lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n$, and three Lagrange multipliers, leading to difficulties in obtaining joint closed-form solutions. To this end, this paper carefully analyzes the structure of the optimizing problem, delves into the relationship between processor core speed and task load distribution, and then derives the optimal processor speed configuration for heterogeneous multiprocessor systems under power constraints. The details are as follows.

***Theorem**: For the non-real-time tasks scheduling model under power constraints, the average response time T'' for non-real-time tasks on n multiprocessors is minimized when all processors have the same speed. That is, $s_1 = s_2 = \cdots = s_n = s$, where*

$$s = \left(\frac{1}{(\lambda' + \lambda'')\bar{r}}\left(\tilde{P} - \sum_{i=1}^{n} c_i P_i^*\right)\right)^{1/(\alpha_i - 1)}$$

The proof of the theorem will be shown in Appendix A.

Based on this theorem, we bring (A7) from Appendix A into (A8) to get:

$$\frac{\partial T''}{\partial \lambda_i''} = \frac{\bar{r}}{\lambda'' c_i s_i} \cdot \frac{\partial L_i}{\partial \rho_i''} = \left(1 - \frac{1}{\alpha_i}\right)\phi, \tag{20}$$

for all $1 \le i \le n$. From the above, we reduce the $2n+3$ variables to $n+1$ variables, that is, $\lambda_1'', \lambda_2'', \ldots, \lambda_n''$ and a Lagrange multiplier ϕ, by mathematical derivation. Next, two binary-based task allocation algorithms are used to solve the load distribution $\lambda_1'', \lambda_2'', \ldots, \lambda_n''$, the Lagrange multiplier ϕ, and the minimum average response time T''. The procedure for Algorithm 1 is shown below.

The initial value of ub in line 2 of Algorithm 1 is set by the constraint $C6$. The judgment condition in line 6 depends on (20). Next, the dichotomy-based load allocation method is used to solve ϕ, $\lambda_1'', \lambda_2'', \ldots, \lambda_n''$ and T''. The procedure is shown in Algorithm 2.

Algorithm 1 *Find_*$\lambda_i''(c_i, m_i, \lambda_i', \lambda'', \bar{r}, s, \phi)$

Input: c_i, m_i, λ_i', for all $1 \le i \le n, \lambda'', \bar{r}, s, \phi$.
Output: λ_i''.
$lb \leftarrow 0$;
$ub \leftarrow (c_i s - \lambda_i')/\bar{r}$;
while $(ub - lb > \varepsilon)$ **do**
  $middle \leftarrow (lb + ub)/2$;
  $\lambda_i'' \leftarrow middle$;
  if $\frac{\bar{r}}{\lambda'' c_i s_i} \cdot \frac{\partial L_i}{\partial \rho_i''} < (1 - \frac{1}{\alpha_i})\phi$ **then**
    $lb \leftarrow middle$;
  else
    $ub \leftarrow middle$;
end if
end while
$\lambda_i'' \leftarrow (lb + ub)/2$;
return λ_i''.

B. REAL-TIME TASKS SCHEDULING UNDER TIME CONSTRAINTS

In this subsection, we address the optimization of real-time task scheduling in a global scheduling system to ensure the schedulability of real-time tasks while minimizing their maximum response time. To this end, we configure the processor speed based on the optimal speed from Section IV-A, and convert the constraints $C1$, $C2$ and $C4$ in (17) into a nonlinear min max optimization problem [36], namely

$$\begin{aligned} P3: \quad & \min T'_{\max}(\lambda_1', \lambda_2', \ldots, \lambda_n') \\ & = \max\{T_1'(\lambda_1'), T_2'(\lambda_2'), \ldots, T_n'(\lambda_n')\} \\ s.t.\ & C10: \rho_i' < 1 \\ & C11: F'(\lambda_1', \lambda_2', \ldots, \lambda_n') = \lambda' \\ & C12: T'_{\max}(\lambda_1', \lambda_2', \ldots, \lambda_n') \le T_{dead}, \end{aligned} \tag{21}$$

where the constraint $C10$ is a variant of $C1$, which represents the equilibrium state constraints of real-time tasks in the system, i.e., $\lambda_i' < c_i s_i/\bar{r}$, where $1 \le i \le n$. $C11$ is a variant of $C2$ with $F'(\lambda_1', \lambda_2', \ldots, \lambda_n') = \lambda_1' + \lambda_2' + \cdots + \lambda_n'$. $C12$ is a variant of $C4$ and the processor speed of the system S_i has been calculated by Theorem, so $T'_{\max}$ in $C12$ is only a function of the load distribution $\lambda_1', \lambda_2', \ldots, \lambda_n'$. The above problem is a multivariate and multi-constrained nonlinear min max problem, and unbalanced load distribution may occur if this problem is solved directly.

For this reason, we introduce the load balancing effective degree σ to measure the real-time task processing time difference among the heterogeneous systems, and it is denoted as

$$\sigma = [\sum_{i=1}^{n} (T_i' - T')^2/n]^{1/2}, \tag{22}$$

where σ denotes the variance of the processing time of each heterogeneous system, and a smaller value of σ means that the processing time of the processors in the system is more balanced, and the overall performance of the processors is

Algorithm 2 Dichotomy-Based Load Allocation Method (DBLAM)

Input: $n, c_i, m_i, \lambda_i', P_i^*$, for all $1 \le i \le n, \bar{r}, \tilde{P}, \lambda', \lambda''$.
Output: $\phi, s, \lambda_1'', \lambda_2'', \ldots, \lambda_n'', T''$.
Initialize core speed s based on **Theorem**.
$\phi \leftarrow$ a small value.
do
  Initialize F to 0, and set ϕ to grow by a factor of 1.25.
  for $(i \leftarrow 1; i \le n; i++)$ **do**
  Calculating $\lambda_i'' \leftarrow$ *Find_*$\lambda_i''(c_i, m_i, \lambda_i', \lambda'', \bar{r}, s, \phi)$, according to Algorithm 1.
  Calculating the total task arrival rate F cumulatively.
  end for
while $(F < \lambda'')$;
Initialize lower bound lb to 0, upper bound ub to ϕ.
while $(ub - lb > \varepsilon)$ **do**
  Initialize F to 0, and set $middle$ to the average of lb and ub.
  for $(i \leftarrow 1; i \le n; i++)$ **do**
  Calculating $\lambda_i'' \leftarrow$ *Find_*$\lambda_i''(c_i, m_i, \lambda_i', \lambda'', \bar{r}, s, middle)$, according to Algorithm 1.
  Assigning the cumulative value of λ'' to F.
  end for
  if $F < \lambda''$ **then**
    $lb \leftarrow middle$;
  else
    $ub \leftarrow middle$;
end if
end while
Set ϕ to the average of lb and ub.
for $(i \leftarrow 1; i \le n; i++)$ **do**
  Calculating the optimal load distribution, $\lambda_i'' \leftarrow$ *Find_*$\lambda_i''(c_i, m_i, \lambda_i', \lambda'', \bar{r}, s, \phi)$, according to Algorithm 1.
end for
Initialize minimum average response time T'' to 0.
for $(i \leftarrow 1; i \le n; i++)$ **do**
  Calculating $T'' \leftarrow T'' + (L_i - L_i')/\lambda''$, according to (15).
end for
return $\lambda_1'', \lambda_2'', \ldots, \lambda_n'', T''$.

better. Thus, we introduce the load balancing constraint based on the original optimization objective, that is

$$\begin{aligned} P4: \quad & \min T'_{\max}(\lambda_1', \lambda_2', \ldots, \lambda_n') \\ & = \max\{T_1'(\lambda_1'), T_2'(\lambda_2'), \ldots, T_n'(\lambda_n')\} \\ s.t.\ & C10: \lambda_i' < c_i s_i/\bar{r} \\ & C11: F'(\lambda_1', \lambda_2', \ldots, \lambda_n') = \lambda' \\ & C12: T'_{\max}(\lambda_1', \lambda_2', \ldots, \lambda_n') \le T_{dead} \\ & C13: \sigma \le \varepsilon, \end{aligned} \tag{23}$$

where ε is the load balancing factor, which is in the range of $0 \le \varepsilon \le 10^{-2}$. Subsequently, a Bounded-Constrained

Augmented Lagrange Sequence Quadratic Programming Block Homotopy (BCAL-SQP-BHom) algorithm is used to solve (23), which enables real-time scheduling and load balancing for real-time tasks.

Nonlinear constrained minmax optimization problems are typically addressed using the penalty function approach. This involves adding the degree of constraint violation as a penalty term to the objective function, thereby converting the original problem into an unconstrained optimization problem. However, to obtain an optimal solution to the original problem, the penalty factor must converge to ∞. Unfortunately, this approach can result in the penalty function becoming pathological and eventually lead to computational challenges. In this paper, we propose a solution to overcome the pathological issues caused by excessively large penalty factors. Specifically, we construct an augmented Lagrangian function for problem $P4$ as follows:

$$\begin{aligned}&\mathcal{L}_A(\lambda, \gamma, y; \pi)\\&= T'_{\max}(\lambda) - [\gamma_1(F'(\lambda) - \lambda') + \gamma_2(T'_{\max}(\lambda) - T_{dead} + y_1^2)\\&\quad + \gamma_3(\sigma - \varepsilon + y_2^2)] + \frac{\pi}{2}[(F'(\lambda) - \lambda')^2\\&\quad + (T'_{\max}(\lambda) - T_{dead} + y_1^2)^2 + (\sigma - \varepsilon + y_2^2)^2]. \qquad (24)\end{aligned}$$

The (24) introduces slack variables to convert inequality constraints into equation constraints, where $\mathcal{L}_A$ represents the Lagrange function, $\lambda = [\lambda'_1\ \lambda'_2\ \cdots\ \lambda'_n]^T$ denotes the real-time task load distribution vector, $\gamma = [\gamma_1\ \gamma_2\ \gamma_3]^T$ indicates the Lagrange multipliers, $y = [y_1\ y_2]^T$ means slack variables, and π is the penalty factor. According to the literature [37], it is known that the slack variables can be eliminated by transformation, i.e.,

$$y_1^2 = \frac{1}{\pi}\max\{0, \gamma_2 - \pi(T'_{\max}(\lambda) - T_{dead})\},$$

and

$$y_2^2 = \frac{1}{\pi}\max\{0, \gamma_3 - \pi(\sigma(\lambda) - \varepsilon)\}.$$

Thus, the change of the slack variable y will not be described in detail in the following article. Meanwhile, for the convenience of analysis and calculation, we record $h(\lambda)$ as

$$[(F'(\lambda) - \lambda')(T'_{\max}(\lambda) - T_{dead} + y_1^2)(\sigma - \varepsilon + y_2^2)]^T,$$

which is the combined vector of constraint conditions $C11$, $C12$ and $C13$, then (24) is expressed as:

$$\mathcal{L}_A(\lambda, \gamma; \pi) = T'_{\max}(\lambda) - \gamma^T h(\lambda) + \frac{\pi}{2}||h(\lambda)||^2. \qquad (25)$$

Next, we perform an iterative solution using the Augmented Lagrangian Method (ALM), given λ^0, γ^0, π^0, and the iterative form as follows:

$$\begin{aligned}\lambda^{k+1} &= \arg\min\{\mathcal{L}_A(\lambda, \gamma^k; \pi^k) | 0 < \lambda < \mathrm{I}\}\\ \gamma^{k+1} &= \gamma^k - \pi^k h^{k+1}, \qquad (26)\end{aligned}$$

where $0 < \lambda < \mathrm{I}$ is the bound constraint, which is a variant of the constraint $C10$, and $\mathrm{I} = [\frac{c_1 s_1}{\tilde{r}}\ \frac{c_2 s_2}{\tilde{r}}\ \cdots\ \frac{c_n s_n}{\tilde{r}}]^T$ denotes the upper bound of the constraint. The Bounded Constraint Augmented Lagrangian Approach (BCAL) is shown in Algorithm 3.

Algorithm 3 Bounded Constraint Augmented Lagrangian Method (BCAL)

Input: $\partial > 0, \lambda^0 \in \mathcal{R}^n, \gamma^0 \in \mathcal{R}^3, \pi^0 > 0, \eta^0 = 1/(\pi^0)^{0.1}, \partial^0 = 1/\pi^0, k = 0.$
Output: $\lambda^*, T'_{\max}(\lambda^*)$.
1: Use the **SQP-BHom** method to solve the bounded constrained minimization problem in (26) to obtain the solution λ^{k+1}; otherwise, go to step 2.
2: Update Lagrange Multiplier: $\gamma^{k+1} = \gamma^k - \pi^k h^{k+1}$.
3: **if** $||h^{k+1}|| \le \eta^k$ **then**
4: **if** $||T'_{\max}(\lambda^{k+1}) - T'_{\max}(\lambda^k)|| \le \partial$ **then**
5: Terminate the iteration and get $\lambda^* = \lambda^{k+1}$.
6: **else**
7: Let $\pi^{k+1} = \pi^k, \eta^{k+1} = \eta^k/(\pi^k)^{0.9}, \partial^{k+1} = \partial^k/\pi^k$.
8: **else**
9: Let $\pi^{k+1} = 1.5\pi^k, \eta^{k+1} = 1/(\pi^k)^{0.1}, \partial^{k+1} = 1/\pi^k$.
10: **end if**
11: Let $k = k + 1$ and return to step 1.

The minimization problem subject to bounded constraints at step k in (26) is addressed by utilizing the SQP-BHom method, with an initial solution denoted as $x^0 = \lambda^k$. By employing a second-order expansion of (25), the aforementioned problem is transformed into a quadratic programming subproblem, taking on the following form:

$$\begin{aligned}\min\ q(d) &= \nabla T'_{\max}(x^j)^T d - (\gamma^k)^T A(x^j) d\\&\quad + \frac{1}{2} d^T H(x^j) d + \frac{\pi^k}{2}||h(x^j) + A(x^j)d||^2\\&\quad s.t.\ -x^j < d < I - x^j, \qquad (27)\end{aligned}$$

where

$$A(x^j)^T = [\nabla h_1(x^j), \nabla h_2(x^j), \nabla h_3(x^j)],$$

$h_i(x^j)$ is the i-th component of $h(x^j)$ and $H(x^j)$ is the approximate Hesse matrix obtained by using the Symmetric Rank-One (SR1) formula for the Augmented Lagrange function. We ensure that the matrix $H(x^j)$ is positive definite by adding a sufficiently large multiple of the unit matrix. Moreover, to ensure the global convergence of the sequential quadratic programming (SQP) method [38] and overcome the Maratos effect, we adopt the augmented Lagrange function presented in (25) as the value function, and utilize the Armijo line search criterion to determine the search step size $l^j = \vartheta^{(m^j)}$ for obtaining the next iteration point $x^{j+1} = x^j + l^j d^j$, where $\vartheta \in (0, 1)$, and m^j is the smallest integer satisfying the following equation:

$$\mathcal{L}_A(x^j + \vartheta^{(m^j)} d^j, \gamma^k; \pi^k) - \mathcal{L}_A(x^j, \gamma^k; \pi^k) < \varpi\, \vartheta^{(m^j)} \nabla_x \mathcal{L}_A^T d^j,$$

where $\varpi \in (0, 0.5)$.

Multivariate nonlinear optimization problems often require significant amounts of memory and computing time to

form and solve SQP methods, making them unsuitable for resource-limited heterogeneous multiprocessor systems. Therefore, in this paper, we employ the following Stochastic Block Coordinate Descent (SBCD) method [39], which involves splitting the multivariate problem into a series of small-scale subproblems. In each iteration, the approximate Karush-Kuhn-Tucker (KKT) condition of the original problem is used as a criterion for the working set B and the non-working set N.

$$|\mathcal{G}(x^j, \gamma^k)_i| \geq \xi, \quad 0 < x_i^j < I_i$$

where $1 \leq i \leq n$, $\mathcal{G}(x^j, \gamma^k) = \nabla T'_{\max}(x^j) - A(x^j)^T \gamma^k$. In particular, the indicator i that violates the approximate KKT condition is randomly added to the working set B, with the constraint $|\mathrm{B}| \leq q$ satisfied. Furthermore, q is a predetermined upper bound on the dimensionality of the subproblem, and the subproblem (27) can be decomposed into the following form:

$$\begin{aligned} q_k(d) = & \frac{1}{2}[d_B^T \ d_N^T] \begin{bmatrix} H_{BB}^j & H_{BN}^j \\ H_{NB}^j & H_{NN}^j \end{bmatrix} \begin{bmatrix} d_B \\ d_N \end{bmatrix} \\ & - (\gamma^k)^T [A_B^j A_N^j] \begin{bmatrix} d_B \\ d_N \end{bmatrix} \\ & + [(\nabla T'^{(j)}_{\max(B)})^T \ (\nabla T'^{(j)}_{\max(N)})^T] \begin{bmatrix} d_B \\ d_N \end{bmatrix} \\ & + \frac{\pi^k}{2} ||(h^j + [A_B^j \ A_N^j] \begin{bmatrix} d_B \\ d_N \end{bmatrix})||^2 \\ = & \frac{1}{2} d_B^T (H_{BB}^j + \pi^k (A_B^j)^T A_B^j) d_B \\ & + ((\nabla T'^{(j)}_{\max(N)})^T - (\gamma^k)^T A_N^j) d_N \\ & + \frac{1}{2} d_N^T H_{NN}^j d_N \\ & + (d_N^T H_{BN}^j + (\nabla T'^{(j)}_{\max(B)})^T - (\gamma^k)^T A_B^j \\ & + \pi^k (h^j + A_N^j d_N^j)^T A_B^j) d_B \end{aligned}$$

We set the value of component d_N^j to a fixed value. Then the subproblem (27) can be expressed as a QP block subproblem with the independent variable d_B^j:

$$\begin{aligned} \min \ & \frac{1}{2} d_B^T (H_{BB}^j + \pi^k (A_B^j)^T A_B^j) d_B \\ & + (d_N^T H_{BN}^j + (\nabla T'^{(j)}_{\max(B)})^T - (\gamma^k)^T A_B^j \\ & + \pi^k (h^j + A_N^j d_N^j)^T A_B^j) d_B \\ s.t. \ & - x_B^j < d_B < I_B - x_B^j. \end{aligned} \tag{28}$$

The Accelerated Proximal Gradient Homotopy (APG-Hom) method is employed to solve the subproblem (28), with the method steps and details described in the APP-Hom method developed by Wang et al. [34]. The APG method is used to predict a good initial point, which can reduce the number of iterations required for the homotopy tracking process. Meanwhile, the Hom method is utilized to solve the subproblem (28) efficiently. During the chunked subproblem-solving process, the non-working set N contains zero components, i.e., $d_N^j = 0$ and $d^j = (d_B^j; d_N^j)$. The next iteration point of the SQP is $x^{j+1} = x^j + l^j d^j$.

The framework of the Sequential Quadratic Programming Block Homotopy (SQP-BHom) method is shown in Algorithm 4. It is worth noting that $P_{[0,I]}(x_B^j - \nabla \mathcal{L}_A(x_B^j, \gamma^k; \pi^k))$ in line 3 of Algorithm 4 denotes the projection of the variable x_B^j on the working set B within the bound constraint $[0, I]$.

Algorithm 4 Sequential Quadratic Programming Block Homotopy Method (SQP-BHom)

Input: $\varpi > 0, \vartheta > 0, \gamma^k \in \mathcal{R}^3, \pi^k > 0, \partial^k > 0, h^0 \in \mathcal{R}^3, \nabla T'^{(0)}_{\max} \in \mathcal{R}^n, A^0 \in \mathcal{R}^{m \times n}, H^0 \in \mathcal{R}^{n \times n}, x^0 \in \mathcal{R}^n, d^0 \in \mathcal{R}^n, j = 0$.
Output: λ^{k+1}.
1: Update the working set B^j.
2: Calculate h^j, $\nabla T'^{(j)}_{\max(B)}$, A_B^j, H_{BB}^j.
3: **if** $||x_B^j - P_{[0,I]}(x_B^j - \nabla \mathcal{L}_A(x_B^j, \gamma^k; \pi^k))|| \leq \partial^k$ **then**
4: Terminate the iteration and return $\lambda^{k+1} = x^j$.
5: **APG-Hom** method for solving subproblem (28), let $d_N^j = 0$, $d^j = (d_B^j; d_N^j)$.
6. **Armijo line search** to determine the search step m^j, $l^j = \vartheta^{(m^j)}$.
7: Let $x^{j+1} = x^j + l^j d^j$.
8: The **SR1** method to update the Hesse matrix and add a sufficiently large unit matrix to make it positive definite.
9: Let $j = j + 1$ and return to step 1.

In summary, the scheduling optimization algorithm named BCAL-SQP-BHom is presented to enable the load distribution $\lambda_1', \lambda_2', \ldots, \lambda_n'$ for real-time tasks. By incorporating the distribution result as a known variable into the scheduling algorithm for non-real-time tasks, we can achieve federated scheduling for both real-time and non-real-time tasks while adhering to power constraints, thereby optimizing the overall system performance.

C. COMPLEXITY ANALYSIS

In this section, the time complexity of the typed task federated scheduling scheme is analyzed. Since the scheme combines non-real-time task scheduling (Algorithm 1 and Algorithm 2) and real-time task scheduling (Algorithm 3 and Algorithm 4), it is analyzed step-by-step according to these two parts. The details of the time complexity analysis are given below.

In non-real-time task scheduling, Algorithm 1 contains a While loop with $\log_2((ub - lb)/\varepsilon)$ iterations, where ub and lb represent the search upper and lower bounds of λ_i'', respectively, and ε is the accuracy which is the dominant component. For the sake of uniform analysis, I is used to denote the maximum length of all initial search intervals in this paper. Therefore, the time complexity of Algorithm 1 is less than $O(\log_2(I/\varepsilon))$. In Algorithm 2, lines 3 to 9 contain a While loop, a For loop, and a call to Algorithm 1. Assuming that the maximum number of iterations for the outermost

While loop is W, the time cost is bounded by $O(W \times n \times \log_2(I/\varepsilon))$. Lines 11 to 22 contain a While loop, a For loop, and a call to Algorithm 1, resulting in a time cost of at most $O(n \times (\log_2(I/\varepsilon))^2)$. Similarly, lines 24 to 26 have a time cost not exceeding $O(n \times \log_2(I/\varepsilon))$, and the time cost of lines 28 to 30 is about $O(n)$. Therefore, the time complexity of Algorithm 2 is less than $O((W + \log_2(I/\varepsilon) + 1) \times n \times \log_2(I/\varepsilon) + n)$.

In real-time task scheduling, Algorithm 4 serves as the inner iterative framework, where line 3 represents the termination condition, with a maximum number of iterations denoted as J. The APG-Hom method is employed for Algorithm 4 to solve subproblem (28), with the search step determined using the Armijo line search. Let A and S represent the maximum number of iterations that satisfy the termination conditions of the APG-Hom method and Armijo line search, respectively, ensuring that the time complexity of Algorithm 4 remains within $O(J \times (A + S))$. Algorithm 3, on the other hand, is an outer iterative framework, where line 4 denotes the termination condition for finding the optimal real-time task distribution, and the maximum iteration count is denoted as K. Therefore, the time complexity of Algorithm 3 is less than $O(K \times J \times (A+S))$. It should be noted that to validate the real-time performance of the proposed BCAL-SQP-BHom algorithm, its algorithmic time overhead is compared with other real-time scheduling algorithms in the DEADLINE section of Section V-C.

In real-world applications, the whole scheduling also encompasses additional factors including latency, communication, and context switching times. These factors depend on the particular environment and platform and are not discussed in this paper.

V. SIMULATION RESULTS AND PERFORMANCE ANALYSIS

In this section, we simulate and analyze the proposed federated scheduling optimization scheme for typed tasks with power constraints to demonstrate specific numerical scheduling results. We analyze the impact of system capacity, available power, and real-time task deadlines on system performance, using the maximum response time for real-time tasks, the average response time for non-real-time tasks, and load balancing as evaluation metrics. Meanwhile, we compare the proposed federated scheduling optimization scheme with other scheduling schemes. All numerical experiments are implemented using the MATLAB (R2019b) platform on a machine with Windows 11 operating system, Intel(R) Core (TM) i5-11320H @ 3.20 GHz processor, and 16 GB RAM.

A. SYSTEM PARAMETERS

In this subsection, we consider a set of $n = 7$ heterogeneous multiprocessor systems $S_1, S_2, \ldots, S_n$, each with $c_i = 2i$ processors and the system capacity $m_i = 4i$, where $1 \leq i \leq n$. The static power consumption of each local scheduling system is set to $P_i^* = 0.2$ Watts, where $1 \leq i \leq n$. We assume that the total task arrival rate in the global scheduling system is $\lambda = 60$ and divide real-time and non-real-time tasks in a ratio of 1:3. The specific simulation parameters are detailed in TABLE 3.

TABLE 3. System simulation parameters.

Simulation Parameter	Value
Number of local scheduling systems n	7
Number of processors c_i	[2,4,6,8,10,12,14]
System capacity m_i	[4,8,12,16,20,24,28]
Static power consumption P_i^*	0.2 Watts
Total task arrival rate λ	60
Real-time task arrival rate λ'	15
Non-real-time task arrival rate λ''	45
Average task execution size $\overline{r}$	0.5
Available power $\tilde{P}$	30 Watts
Real-time task deadline T_{dead}	1 second
Power factor α	3
Load balancing factor ε	0.01
Lagrange multiplier γ	[0,0,0]
Slack variable y	[0.5,0.5]
Penalty factor π	100

B. NUMERICAL ANALYSIS

In TABLE 4, we illustrate the results of federated scheduling for real-time and non-real-time tasks. As the number of processors $c_1, c_2, \ldots, c_7$ in the heterogeneous local scheduling systems $S_1, S_2, \ldots, S_7$ increases linearly, we observe that the load allocation $\lambda_1', \lambda_2', \ldots, \lambda_7'$ for real-time tasks, the load allocation $\lambda_1'', \lambda_2'', \ldots, \lambda_7''$ for non-real-time tasks, and the power consumption $P_1, P_2, \ldots, P_7$ of each system all increase. This is due to the fact that the ability to perform tasks is increased as the number of processors becomes larger, and more tasks are assigned to the corresponding core processors. Moreover, the utilization of processor cores $\rho_1'', \rho_2'', \ldots, \rho_7''$ remains relatively consistent, which indicates that a balanced load distribution is achieved. The average response times $T_1', T_2', \ldots, T_7'$ of real-time tasks in each system remain approximately consistent and meet the deadline constraints due to their strong preemption property. The non-real-time tasks with lower priority are affected by the preemption of real-time tasks, and the effect is more pronounced when the number of processor cores is smaller, leading to longer average response times. Consequently, the average response times $T_1'', T_2'', \ldots, T_7''$ of non-real-time tasks in the system $S_1, S_2, \ldots, S_7$ exhibit a downward trend, and the total average response time of $T'' = 0.692079$ seconds is obtained.

C. PERFORMANCE ANALYSIS

1) SYSTEM CAPACITY

Figure 3 presents the effects of the system queueing capacity, denoted as m, on the maximum response time $T'_{\max}$ for

TABLE 4. Load distribution under power constraints.

i	c_i	m_i	s_i	λ_i'	λ_i''	P_i	ρ_i''	T_i'	T_i''
1	2	4	0.783823	0.000035	3.073781	1.363129	0.523261	0.650656	0.704951
2	4	8	0.783823	0.101555	6.046078	2.726258	0.552476	0.650656	0.701015
3	6	12	0.783823	0.627564	4.827164	2.909148	0.591526	0.650657	0.695986
4	8	16	0.783823	1.552216	6.154106	4.014647	0.626771	0.650657	0.691256
5	10	20	0.783823	2.764222	7.309856	5.156544	0.655476	0.650658	0.688956
6	12	24	0.783823	4.186300	8.333664	6.322922	0.678850	0.650658	0.687675
7	14	28	0.783823	5.768112	9.255348	7.507350	0.698222	0.650659	0.686912

real-time tasks and the average response time T'' for non-real-time tasks, for varying numbers of processors c, respectively. It is worth noting that this paper is dedicated to the overall performance of the global system, so $m = \sum m_i$ and $c = \sum c_i$, where $1 \leq i \leq n$. It can be observed that with increasing system capacity, the maximum response time $T'_{\max}$ initially decreases sharply and then stabilizes, while the average response time T'' exhibits a trend of rapid decrease followed by gradual increase. This means that if the system capacity is insufficient, all arriving tasks cannot be processed in time, which leads to an oversupply of queue capacity and eventually seriously affects the system's performance. Therefore, both $T'_{\max}$ and T'' exhibit a significant decrease during the initial phase of queueing capacity increase as queueing pressure eases.

As can be seen in Figure 3, when the queueing capacity m increases to some extent, the maximum response time $T'_{\max}$ tends to stabilize. This is because real-time tasks have preemption priority and can deprive non-real-time tasks of their execution rights or delay their queue waiting time to meet their own real-time execution requirements. Therefore, the maximum response time $T'_{\max}$ almost keep constant if the system queueing capacity m is further increased. Moreover, $T'_{\max}$ keeps increasing as the number of processors increases. The reason is that the available power allocated to each processor decreases as the number of processors c increases if the same system power constraint is maintained, which results in a decrease in the task execution rate and an increase in $T'_{\max}$. Thus, a reasonable allocation of the number of processors, considering the system power constraint, can effectively fulfill the scheduling requirements of real-time tasks.

From Figure 3, it can also be observed that the average response time T'' maintains an increasing trend as the queueing capacity m increases in the later stages. The increase in capacity m will result in longer task queues and waiting times, leading to an increase in task response time. Moreover, if m increases indefinitely, the system will waste more space and time resources, which is undesirable in heterogeneous multiprocessor systems with limited hardware resources. Thus, an appropriate arrangement on the system capacity can effectively avoid additional delays and resource wastage

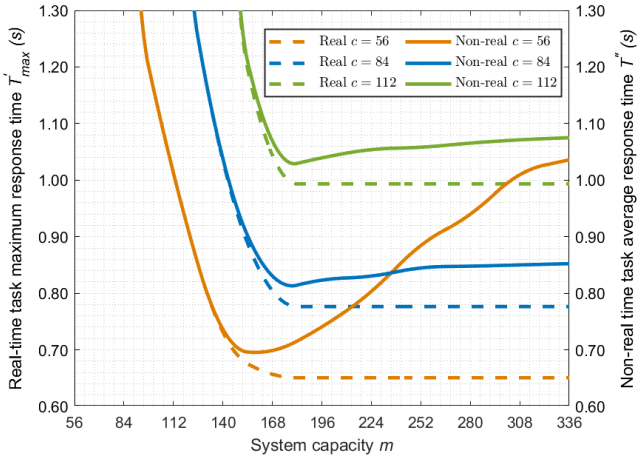

FIGURE 3. System capacity versus maximum response time for real-time tasks and average response time for non-real-time tasks.

resulting from queueing, thereby enhancing system performance. Furthermore, if the number of processors c is smaller, the available power and task execution rate per processor are higher, and the response time required for tasks should be shorter. However, due to the lower priority of non-real-time tasks, they may be preempted by real-time tasks and lead to re-queueing for execution, particularly when the number of processors is small ($c = 56$), the form of preemption is more intense, which leads to a sudden increase in T''. Therefore, it is crucial for performance improvement to reasonably control the number of processors and system queueing capacity within the fixed system power constraints.

2) AVAILABLE POWER

Figure 4 illustrates the impact of system available power P on the maximum response time $T'_{\max}$ and processor rate s for real-time tasks, respectively. It can be seen that the maximum response time $T'_{\max}$ for various arrival rates decreases as the available power keeps increasing. The processor speed s increases with the available power, indicating that less time would be taken to execute the real-time tasks at the same arrival rate, and the waiting time is correspondingly reduced. Meanwhile, if the arrival rate of tasks increases, the task

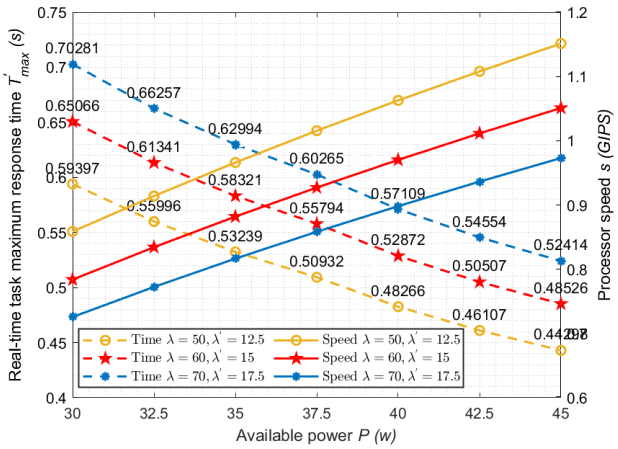

FIGURE 4. Available power versus maximum response time for real-time tasks and processor speed curve.

load assigned to each processor also increases, demonstrating that more time is consumed to respond to tasks within the same available power constraints. Similarly, with increasing arrival rates, more system power needs to be consumed to meet the task's real-time requirements while maintaining the same maximum task response time. As depicted in Figure 4, the maximum response times of the real-time tasks with varying arrival rates do not exceed their deadlines, ensuring the schedulability of the real-time tasks.

Figure 5 presents the relationship between the available power P and the average response time T'' of the non-real-time tasks as well as the processor rate s, for varying arrival rates, respectively. Similar to real-time tasks, the average response time of non-real-time tasks decreases with increasing available power and increases with higher task arrival rates. Moreover, the processor speed s increases as the available power keeps getting larger. Comparing Figure 4 and Figure 5, it can be revealed that the average response time T'' of non-real-time tasks exceeds the maximum response time

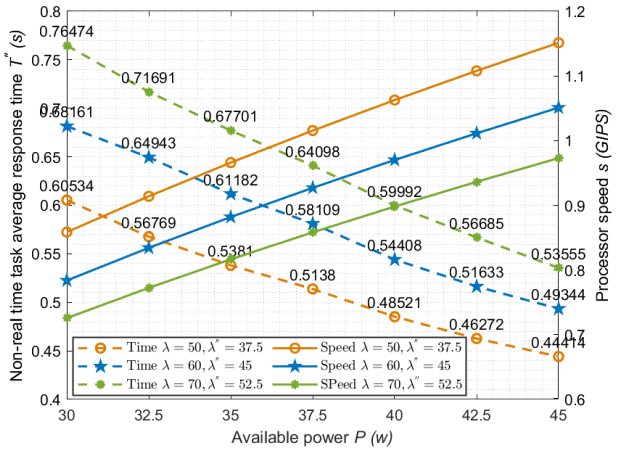

FIGURE 5. Available power versus average response time for non-real-time tasks and processor rate curves.

$T'_{\max}$ of real-time tasks with the same available power. The difference between T'' and $T'_{\max}$ will become larger as the arrival rate increases. This is due to the fact that with higher task arrival rates, the preemption probability of real-time tasks will increase, which may lead to a more pronounced trend of preemption and a wider response time gap between the two task types.

3) DEADLINE

In this subsection, we compare our proposed BCAL-SQP-BHom real-time scheduling algorithm with three established solvers for nonlinear minmax optimization problems, namely fmincon, OPTI (version 2.29), and KNITRO (version 13.02) to analyze the time overhead of these four algorithms at varying task arrival rates. Furthermore, we utilize the BCAL-SQP-BHom algorithm to determine the maximum response time of real-time tasks under various power constraints and analyze the schedulability of these tasks with different arrival rates. It is worth noting that the time overhead of the algorithm is not considered in the determination of the maximum response time for real-time tasks in this paper, as it is significantly affected by the hardware environment.

In Figure 6, the time overhead and the optimal maximum response time $T'_{\max}$ for real-time tasks are displayed for four real-time scheduling algorithms (fmincon, OPTI, KNITRO and BCAL-SQP-BHom), for different arrival rates of real-time tasks. The four scheduling algorithms exhibit an overall increasing trend in time overhead as the arrival rate of real-time tasks increases. This trend occurs because the complexity of achieving optimal real-time scheduling of task loads increases with higher real-time task arrival rates under the same power constraint, resulting in a larger time overhead.

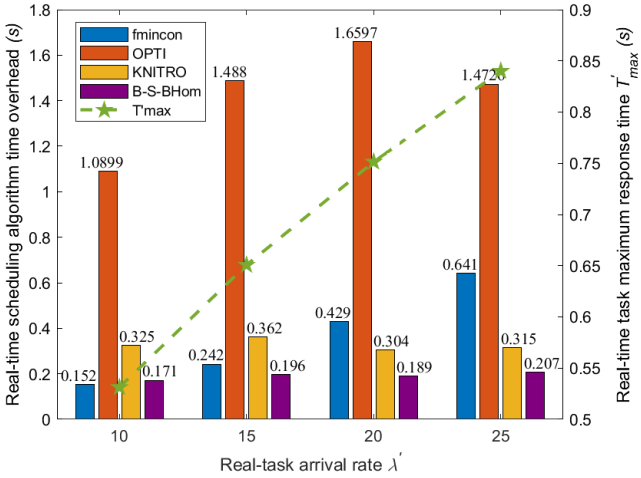

FIGURE 6. Comparison of the time overhead of different real-time scheduling algorithms.

It can also be observed from Figure 6 that as the arrival rate of real-time tasks increases, the time overhead of the OPTI algorithm consistently remains the highest among the four algorithms in the process of solving the optimal maximum response time $T'_{\max}$. On the other hand, the fmincon algorithm

exhibits the most variability in time overhead, whereas the KNITRO and BCAL-SQP-BHom algorithms demonstrate relatively stable changes in time overhead. Specifically, the time overhead of the KNITRO algorithm is around 0.3s, while that of the BCAL-SQP-BHom algorithm is around 0.18s, which indicates that the BCAL-SQP-BHom algorithm is more efficient in real-time scheduling. This can be attributed to the utilization of the SBCD method, which transforms the multivariate problem into a smaller-scale subproblem, thereby reducing the computational dimension and enhancing the overall efficiency of the scheduling algorithm.

Figure 7 illustrates the relationship between the real-time task arrival rate λ' and the maximum response time $T'_{\max}$ obtained from the BCAL-SQP-BHom algorithm with varying power constraints. It is evident that the maximum response time exhibits an increasing trend as the task arrival rate increases for specific available power. The reason is that the limited number of processors and available power make it challenging to process all tasks in parallel as the number of tasks increases. The coming real-time tasks are queued and additional waiting times are added, which leads to an increase in the maximum response time of real-time tasks. Furthermore, under the deadline constraint $T_{dead} = 1$ second, it can be seen that the schedulable demand for the arrival rate λ' within the range of $[15, 30]$ can be fully satisfied only when the system power P is set to $[35, 40]$. This can be attributed to the fact that higher system power results in faster task execution rates for individual processors, enabling the system to meet the response requirements of real-time tasks with shorter execution times, thus ensuring their real-time characteristics.

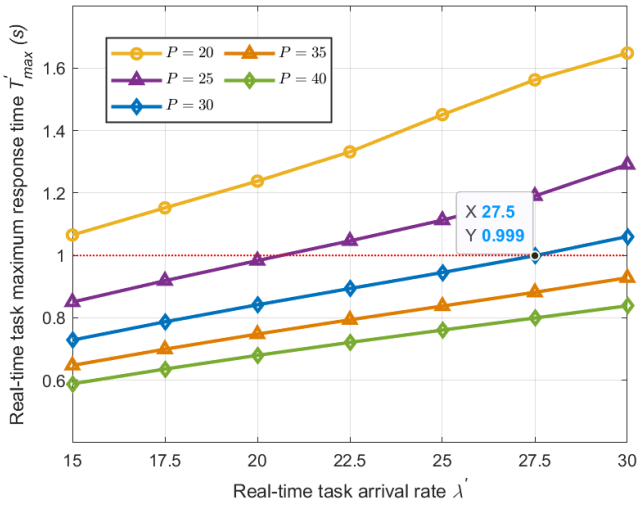

FIGURE 7. Real-time task arrival rate versus maximum response time for real-time tasks.

4) LOAD BALANCE

Figure 8 presents a comparison of the load balancing effectiveness σ of the BCAL-SQP-BHom algorithm proposed in this study with three other algorithms (fmincon, OPTI, and KNITRO). The comparison is conducted under varying real-time task arrival rates to validate the balancing performance of the proposed algorithms in real-time task scheduling scenarios. It can be seen that the load balancing effectiveness σ of all four scheduling algorithms demonstrates an increasing trend with an increase in the real-time task arrival rate. When the arrival rate is small, the load balancing effectiveness σ of the four algorithms remains relatively stable, with changes within the order of 10^{-4}. However, when the arrival rate increases to 25, the load balancing effectiveness σ of all four algorithms exhibits a noticeable upward trend. As the arrival rate increases, the number of tasks to be processed becomes larger, most of them require queuing and the additional waiting delay becomes larger, which results in an increase in the variance of the load metrics. Therefore, a decrease in the load balancing performance occurs as the arrival rate increases to some extent.

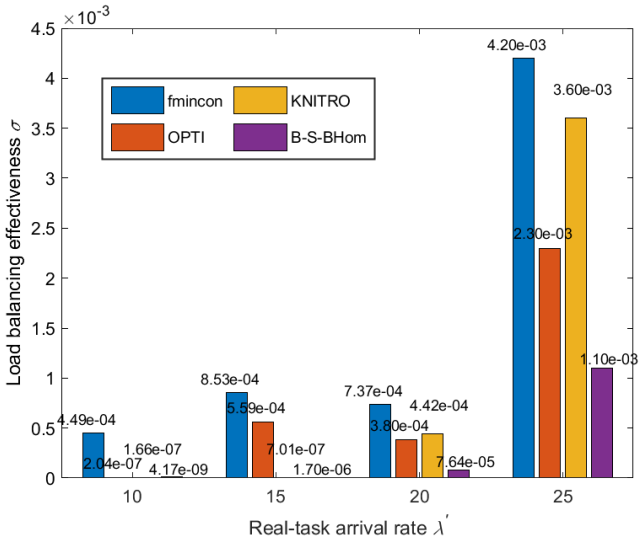

FIGURE 8. Comparison of load balancing effectiveness of different real-time scheduling algorithms.

Among the four classes of algorithms, the fmincon algorithm exhibits poor load balancing performance and the KNITRO algorithm lacks load balancing stability, whereas the OPTI algorithm and the BCAL-SQP-BHom algorithm demonstrate comparatively higher load balancing stability. Moreover, the BCAL-SQP-BHom algorithm outperforms the OPTI in terms of load balancing effectiveness. This is because the BCAL-SQP-BHom algorithm utilizes the Hom algorithm as a higher-order method to solve the blocking subproblem, resulting in a faster and more optimal load balancing solution. Therefore, the proposed BCAL-SQP-BHom algorithm can effectively ensure the balanced load distribution of real-time tasks in multiprocessor systems.

The load profiles of real-time and non-real-time tasks in the local systems for varying total task arrival rates are shown in Figure 9. Where the X-axis represents the total arrival rate of real-time and non-real-time tasks, the Y-axis denotes the local systems, and the Z-axis indicates the load distribution

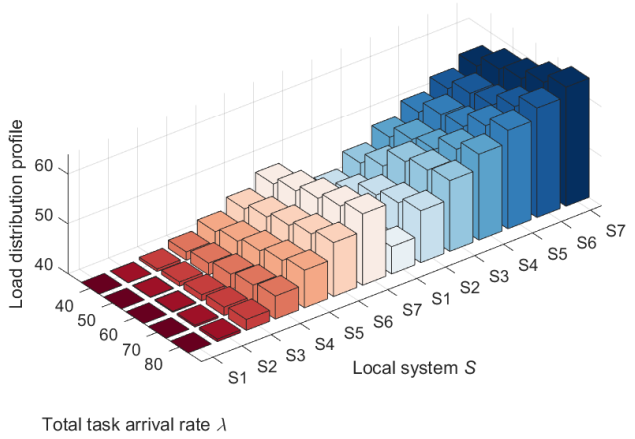

FIGURE 9. Load distribution of real-time and non-real-time tasks with different total task arrival rates.

of real-time (red part) and non-real-time tasks (blue part). It can be seen that both real-time and non-real-time task loads distributed in local systems $S_1 \sim S_7$ maintain an increasing trend as the total arrival rate increases. Additionally, for the same arrival rate, both real-time and non-real-time loads in local systems $S_1 \sim S_7$ demonstrate an increasing trend, with non-real-time task loads being larger than real-time task loads in the same local system. This corresponds to the settings in TABLE 3 of this paper, where the number of local system processors $c = [2, 4, 6, 8, 10, 12, 14]$, and the ratio of real-time task arrival rate to non-real-time task arrival rate is 1:3. As the total task arrival rate increases, the task load assigned to each processor core also increases. Similarly, processors with higher number of cores will be allocated larger task loads, which helps to obtain a well-balanced load distribution in multiprocessor systems.

5) SCHEME COMPARISON

In this subsection, a comparison is made between the proposed federated scheduling optimization scheme for typed tasks under power constraints and several scheduling optimization schemes:

(i) Optimal Load Balancing algorithm **(OLB)** [11], the system queueing model is $M/M/1$ with infinite queueing capacity, the typed tasks are divided into dedicated and generic tasks, and the dedicated tasks are preloaded without considering the time-bound characteristics, and the load distribution of the generic tasks is achieved using an optimal load balancing policy.

(ii) Bi-Directional Timing-Power Optimization algorithm **(BDTPO)** [17], the system queueing model is $M/M/1$ with infinite queueing capacity, the typed tasks are divided into dedicated and generic tasks, and the dedicated tasks are preloaded with deadlines, without considering the load distribution of dedicated tasks, followed by achieving the load distribution of the generic tasks using a bidirectional timing power optimization strategy.

(iii) The proposed federated scheduling scheme **(Proposed),** the system queueing model is $M/M/c/m$ with queueing capacity m, and the typed tasks are divided into real-time and non-real-time tasks, where real-time tasks have strong preemption priority and deadlines. The federated scheduling and load balancing for both real-time and non-real-time tasks are achieved by exploiting a federated scheduling strategy for typed tasks.

We set the static power consumption of each local scheduling system to be $P_i^* = 0.2$ Watts, where $1 \le i \le n$. The average task execution size is set to $\bar{r} = 0.25$ (giga instructions), the power available to the system is $\tilde{P} = 30$ Watts, the deadline for real-time tasks is $T_{dead} = 1$ second, and the total task arrival rate in the system is $\lambda = 25 \sim 50$. We divide real-time tasks and non-real-time tasks in a ratio of 2:3, i.e., the real-time task arrival rate is $\lambda' = 10 \sim 20$, and the arrival rate of non-real-time tasks is $\lambda'' = 15 \sim 30$. Furthermore, to ensure fairness in comparison, we model the system in this paper as a $M/M/1/m$ queueing system, as both OLB and BDTPO model heterogeneous multicore systems as $M/M/1$ queueing systems. Moreover, to ensure the generalizability of the experimental results, random distributions are used to preload dedicated tasks.

Figure 10 compares the impact of the arrival rate of real-time tasks on the maximum response time of real-time tasks under power constraints using different scheduling schemes. The results indicate that the maximum response time increases with the increase of the arrival rate for all three schemes. The maximum response time is the largest when the OLB scheme is selected, and when the arrival rate exceeds 20, the maximum response time exceeds the real-time task deadline $T_{dead} = 1$ second set by the system. This is because the OLB scheme does not take into account the time-limited constraints of real-time tasks. Furthermore, a comparison between the BDTPO scheme and the proposed scheme reveals that the introduction of limited queueing capacity and efficient allocation of real-time

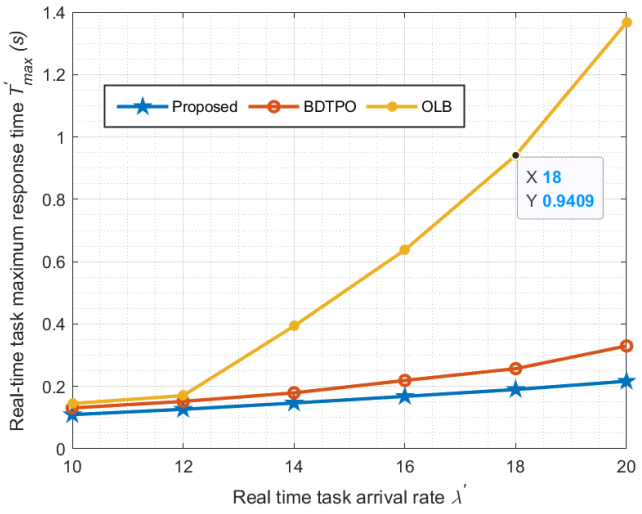

FIGURE 10. The relationship between real-time task arrival rate and maximum response time of real-time tasks with different scheduling schemes.

tasks improves the system performance. Therefore, the proposed scheme outperforms in real-time task scheduling and provides an efficient optimization method for system scheduling.

In Figure 11, the impact of the arrival rate of non-real-time tasks on the average response time of non-real-time tasks under power constraints is compared for different scheduling schemes. The average task response time shows an increasing trend for all schemes as the task arrival rate increases. It is evident that when the arrival rate exceeds 21, the average response time of the proposed scheme is significantly smaller than that of OLB and BDTPO. Due to the preload operations for real-time tasks in the OLB and BDTPO, processors are unable to perform effective load distribution based on the system power state, resulting in some processors becoming overloaded and eventually causing low-priority non-real-time tasks to be blocked in the waiting queue. This additional waiting latency leads to an increase in their average response time. It can also be observed that the system performance of the OLB scheme is slightly superior to that of the BDTPO scheme, and this performance gap becomes more pronounced with increasing arrival rate, confirming a trend reported in the literature [17]. Moreover, the limited system capacity design of the proposed scheme effectively mitigates additional delays caused by queueing and waiting. Thus, the proposed scheme has a superior performance compared to other schemes.

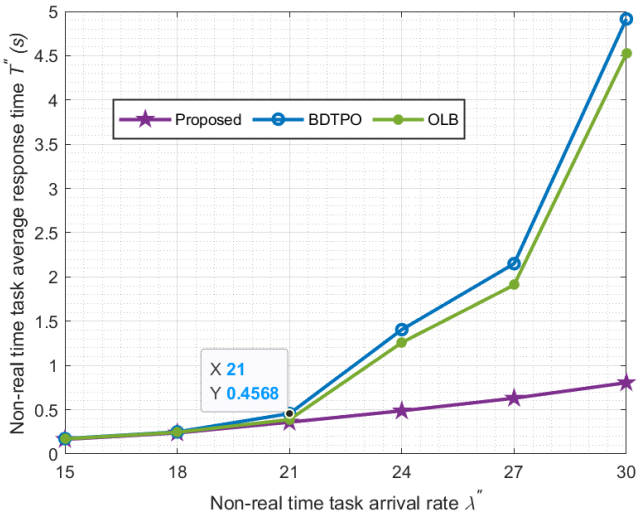

FIGURE 11. The relationship between the arrival rate of non-real-time tasks and the average response time of non-real-time tasks with different scheduling schemes.

In addition, it is worth noting that the multiprocessor parallelism feature of local systems is ignored in the literature [11], [17], and each local system is considered as a single processor core only. Nevertheless, the scalability and parallelism of multicore processors make them more desirable in practical application environments. Thus, taking a comprehensive view and comparing with the literature [11], [17], this paper possesses advantages in terms of model rationality, system scalability, and performance efficiency.

VI. CONCLUSION

This paper investigates the typed task federated scheduling problem for multiple heterogeneous multiprocessors with power constraints. A federated scheduling optimization scheme for real-time and non-real-time tasks is proposed under the given system power and real-time task deadlines, which can satisfy the schedulability requirements of real-time tasks and minimize the average response time of non-real-time tasks. In the simulation experiments, the results of federated scheduling are analyzed numerically in a concrete way, the performance of the system is evaluated with respect to the real-time characteristics of real-time tasks and the preempted characteristics of non-real-time tasks, and the proposed federated scheduling optimization scheme is compared with different scheduling schemes. Simulation results demonstrate that factors such as processor size, system capacity, available power, and task arrival rate have significant effects on system performance. Furthermore, the proposed federated scheduling optimization scheme and algorithms have high effectiveness and superior scheduling performance in power-limited systems. The models, algorithms, and results presented in this paper are also applicable to other multiprocessor systems and computing environments.

APPENDIX
PROOF OF THEOREM

First, we can minimize T'' by using the method of Lagrange multiplier, namely,

$$\begin{aligned}\nabla T''(\lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n) \\ = \phi \nabla F(\lambda_1'', \lambda_2'', \ldots, \lambda_n'') + \psi \nabla G(s_1, s_2, \ldots, s_n) \\ + \tau \nabla P(\lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n),\end{aligned} \tag{A1}$$

where ϕ, ψ, τ are the three Lagrange multipliers. Next, the partial derivative of the variable λ_i'' according to (A1) is obtained,

$$\frac{\partial T''}{\partial \lambda_i''} = \phi \frac{\partial F}{\partial \lambda_i''} + \tau \frac{\partial P}{\partial \lambda_i''} = \phi + \tau \bar{r} s_i^{\alpha_i - 1}, \tag{A2}$$

for all $1 \le i \le n$.

The partial derivative of the variable s_i is obtained as follows:

$$\begin{aligned}\frac{\partial T''}{\partial s_i} &= \psi \frac{\partial G}{\partial s_i} + \tau \frac{\partial P}{\partial s_i} \\ &= \psi \frac{\partial G}{\partial s_i} + \tau \bar{r} (\lambda_i' + \lambda_i'')(\alpha_i - 1) s_i^{\alpha_i - 2},\end{aligned} \tag{A3}$$

where

$$\begin{aligned}\frac{\partial G}{\partial s_i} &= \frac{\partial T_i'}{\partial s_i} = \frac{\partial T_i'}{\partial \rho_i'} \cdot \frac{\partial \rho_i'}{\partial s_i} \\ &= -\frac{\bar{r}}{c_i s_i^2} \cdot \frac{\partial L_i'}{\partial \rho_i'}\end{aligned}$$

for all $1 \leq i \leq n$. Next, we find the partial derivative of the variable λ_i'' in T'',

$$\begin{aligned}\frac{\partial T''}{\partial \lambda_i''} &= \frac{1}{\lambda''} \cdot \frac{\partial L_i}{\partial \lambda_i''} = \frac{1}{\lambda''} \cdot \frac{\partial L_i}{\partial \rho_i''} \cdot \frac{\partial \rho_i''}{\partial \lambda_i''} \\ &= \frac{\bar{r}}{\lambda'' c_i s_i} \cdot \frac{\partial L_i}{\partial \rho_i''}, \end{aligned} \tag{A4}$$

for all $1 \leq i \leq n$.

Also, find the partial derivative of the variable s_i in T'',

$$\begin{aligned}\frac{\partial T''}{\partial s_i} &= \frac{1}{\lambda''}\left(\frac{\partial L_i}{\partial s_i} - \frac{\partial L_i'}{\partial s_i}\right) \\ &= \frac{1}{\lambda''}\left(\frac{\partial L_i}{\partial \rho_i''} \cdot \frac{\partial \rho_i''}{\partial s_i} - \frac{\partial L_i'}{\partial \rho_i'} \cdot \frac{\partial \rho_i'}{\partial s_i}\right) \\ &= -\frac{\bar{r}(\lambda_i' + \lambda_i'')}{\lambda'' c_i s_i^2} \cdot \frac{\partial L_i}{\partial \rho_i''} + \frac{\bar{r}\lambda_i'}{\lambda'' c_i s_i^2} \cdot \frac{\partial L_i'}{\partial \rho_i'}, \end{aligned} \tag{A5}$$

for all $1 \leq i \leq n$.

According to (A2) and (A4), we have

$$\frac{\partial L_i}{\partial \rho_i''} = \frac{\lambda'' c_i s_i}{\bar{r}} \cdot (\phi + \tau \bar{r} s_i^{\alpha_i - 1}). \tag{A6}$$

Then, according to (A3), (A5), and (A6), it can be concluded that $\psi = -\lambda_i'/\lambda''$, and $\tau \bar{r} \alpha_i (\lambda_i' + \lambda_i'') s_i^{\alpha_i - 2} = -\phi(\lambda_i' + \lambda_i'')/s_i$, so we can get

$$s_i = \left(-\frac{\phi}{\tau} \cdot \frac{1}{\alpha_i \bar{r}}\right)^{1/(\alpha_i - 1)}, \tag{A7}$$

for all $1 \leq i \leq n$. The above result shows that n systems $S_1, S_2, \ldots, S_n$ have the same speed s, i.e., $s_1 = s_2 = \cdots = s_n$. Thus, the constraint $C9$ can be converted to

$$\begin{aligned} &P(\lambda_1'', \lambda_2'', \ldots, \lambda_n'', s_1, s_2, \ldots, s_n) \\ &= \sum_{i=1}^{n} (\lambda_i' + \lambda_i'') \bar{r} s_i^{\alpha_i - 1} + \sum_{i=1}^{n} c_i P_i^* \\ &= (\lambda' + \lambda'') \bar{r} s^{\alpha_i - 1} + \sum_{i=1}^{n} c_i P_i^* = \tilde{P}. \end{aligned} \tag{A8}$$

Finally, according to (A8), we can obtain the processor speed of the system as

$$s = \left(\frac{1}{(\lambda' + \lambda'')\bar{r}} (\tilde{P} - \sum_{i=1}^{n} c_i P_i^*)\right)^{1/(\alpha_i - 1)}. \tag{A9}$$

REFERENCES

[1] Y.-W. Zhang and R.-K. Chen, "A survey of energy-aware scheduling in mixed-criticality systems," *J. Syst. Archit.*, vol. 127, Jun. 2022, Art. no. 102524.

[2] A. A. Khan and M. Zakarya, "Energy, performance and cost efficient cloud datacentres: A survey," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100390.

[3] L.-C. Canon, L. Marchal, B. Simon, and F. Vivien, "Online scheduling of task graphs on heterogeneous platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 721–732, Mar. 2020.

[4] R. Pathan, P. Voudouris, and P. Stenström, "Scheduling parallel real-time recurrent tasks on multicore platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 4, pp. 915–928, Apr. 2018.

[5] K. M. Attia, M. A. El-Hosseini, and H. A. Ali, "Dynamic power management techniques in multi-core architectures: A survey study," *Ain Shams Eng. J.*, vol. 8, no. 3, pp. 445–456, Sep. 2017.

[6] J. Chen, Y. He, Y. Zhang, P. Han, and C. Du, "Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems," *J. Syst. Archit.*, vol. 129, Aug. 2022, Art. no. 102598.

[7] B. Pérez, E. Stafford, J. L. Bosque, and R. Beivide, "Sigmoid: An auto-tuned load balancing algorithm for heterogeneous systems," *J. Parallel Distrib. Comput.*, vol. 157, pp. 30–42, Nov. 2021.

[8] S. Liu, L. Zhang, W. Zhang, and W. Shen, "Game theory based multi-task scheduling of decentralized 3D printing services in cloud manufacturing," *Neurocomputing*, vol. 446, pp. 74–85, Jul. 2021.

[9] J. Yang, B. Jiang, Z. Lv, and K.-K.-R. Choo, "A task scheduling algorithm considering game theory designed for energy management in cloud computing," *Future Gener. Comput. Syst.*, vol. 105, pp. 985–992, Apr. 2020.

[10] Z. Ma, S. Guo, and R. Wang, "The virtual machines scheduling strategy based on M/M/c queueing model with vacation," *Future Gener. Comput. Syst.*, vol. 138, pp. 43–51, Jan. 2023.

[11] J. Huang, Y. Liu, R. Li, K. Li, J. An, Y. Bai, F. Yang, and G. Xie, "Optimal power allocation and load balancing for non-dedicated heterogeneous distributed embedded computing systems," *J. Parallel Distrib. Comput.*, vol. 130, pp. 24–36, Aug. 2019.

[12] W. Bai, J. Zhu, S. Huang, and H. Zhang, "A queue waiting cost-aware control model for large scale heterogeneous cloud datacenter," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 849–862, Apr. 2022.

[13] K. Li, "Optimal speed setting for cloud servers with mixed applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 1947–1955, Apr. 2019.

[14] K. Li, "Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 122–137, Apr. 2016.

[15] T. Atmaca, T. Begin, A. Brandwajn, and H. Castel-Taleb, "Performance evaluation of cloud computing centers with general arrivals and service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2341–2348, Aug. 2016.

[16] K. Li, "Optimal power and performance management for heterogeneous and arbitrary cloud servers," *IEEE Access*, vol. 7, pp. 5071–5084, 2019.

[17] J. Huang, R. Li, Y. Wei, J. An, and W. Chang, "Bi-directional timing-power optimisation on heterogeneous multi-core architectures," *IEEE Trans. Sustain. Comput.*, vol. 6, no. 4, pp. 572–585, Oct. 2021.

[18] K. Li, "Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing," *IEEE Trans. Sustain. Comput.*, Mar. 12, 2019, doi: 10.1109/TSUSC.2019.2904680.

[19] J. Li, J. J. Chen, K. Agrawal, C. Lu, C. Gill, and A. Saifullah, "Analysis of federated and global scheduling for parallel real-time tasks," in *Proc. 26th Euromicro Conf. Real-Time Syst.*, Jul. 2014, pp. 85–96.

[20] M. Han, T. Zhang, Y. Lin, and Q. Deng, "Federated scheduling for typed DAG tasks scheduling analysis on heterogeneous multi-cores," *J. Syst. Archit.*, vol. 112, Jan. 2021, Art. no. 101870.

[21] Z. Dong and C. Liu, "Schedulability analysis for coscheduling real-time tasks on multiprocessors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4721–4732, Nov. 2022.

[22] J. Zhou, M. Zhang, J. Sun, T. Wang, X. Zhou, and S. Hu, "DRHEFT: Deadline-constrained reliability-aware HEFT algorithm for real-time heterogeneous MPSoC systems," *IEEE Trans. Rel.*, vol. 71, no. 1, pp. 178–189, Mar. 2022.

[23] S. Moulik, R. Devaraj, and A. Sarkar, "HEALERS: A heterogeneous energy-aware low-overhead real-time scheduler," *J. Netw. Comput. Appl.*, vol. 208, Oct. 2022, Art. no. 103519.

[24] S. Moulik, Z. Das, R. Devaraj, and S. Chakraborty, "SEAMERS: A semi-partitioned energy-aware scheduler for heterogeneous MulticorE real-time systems," *J. Syst. Archit.*, vol. 114, Mar. 2021, Art. no. 101953.

[25] D. Ramegowda and M. Lin, "Energy efficient mixed task handling on real-time embedded systems using FreeRTOS," *J. Syst. Archit.*, vol. 131, Oct. 2022, Art. no. 102708.

[26] A. Goubaa, M. Khalgui, Z. Li, G. Frey, and M. Zhou, "Scheduling periodic and aperiodic tasks with time, energy harvesting and precedence constraints on multi-core systems," *Inf. Sci.*, vol. 520, pp. 86–104, May 2020.

[27] S. Chang, X. Zhao, Z. Liu, and Q. Deng, "Real-time scheduling and analysis of parallel tasks on heterogeneous multi-cores," *J. Syst. Archit.*, vol. 105, May 2020, Art. no. 101704.

[28] S. Chang, J. Sun, Z. Hao, Q. Deng, and N. Guan, "Computing exact WCRT for typed DAG tasks on heterogeneous multi-core processors," *J. Syst. Archit.*, vol. 124, Mar. 2022, Art. no. 102385.

[29] Z. He, K. Li, and K. Li, "Cost-efficient server configuration and placement for mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 9, pp. 2198–2212, Sep. 2022.
[30] K. Li, "Optimal load distribution for multiple classes of applications on heterogeneous servers with variable speeds," *Softw., Pract. Exper.*, vol. 48, no. 10, pp. 1805–1819, Oct. 2018.
[31] J. Huang, R. Li, J. An, D. Ntalasha, F. Yang, and K. Li, "Energy-efficient resource utilization for heterogeneous embedded computing systems," *IEEE Trans. Comput.*, vol. 66, no. 9, pp. 1518–1531, Sep. 2017.
[32] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.
[33] J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 45–58, Jan. 2014.
[34] G. Wang, B. Yu, and Z. Chen, "APP-Hom method for box constrained quadratic programming," 2020, *arXiv:1703.05001*.
[35] D. Pu and P. Yang, "A class of new Lagrangian multiplier methods," in *Proc. 6th Int. Conf. Bus. Intell. Financial Eng.*, Nov. 2013, pp. 647–651.
[36] I. Notarnicola, M. Franceschelli, and G. Notarstefano, "A duality-based approach for distributed min-max optimization," *IEEE Trans. Autom. Control*, vol. 64, no. 6, pp. 2559–2566, Jun. 2019.
[37] M. Kotti and K. I. Diamantaras, "Efficient binary classification through energy minimisation of slack variables," *Neurocomputing*, vol. 148, pp. 498–511, Jan. 2015.
[38] L. Fang, S. Vandewalle, and J. Meyers, "An SQP-based multiple shooting algorithm for large-scale PDE-constrained optimal control problems," *J. Comput. Phys.*, vol. 477, Mar. 2023, Art. no. 111927.
[39] K. Nakamura, S. Soatto, and B.-W. Hong, "Block-cyclic stochastic coordinate descent for deep neural networks," *Neural Netw.*, vol. 139, pp. 348–357, Jul. 2021.

XIAOHONG WEN received the B.S. degree in communication engineering from South-Central University for Nationalities, Wuhan, China, in 2021. She is currently pursuing the master's degree with the School of Microelectronics and Communication Engineering, Chongqing University, Chongqing, China. Her research interests include heterogeneous computing systems, distributed computing, parallel computing, and high-performance computing.

GUOJIN LIU received the Ph.D. degree in communication and information systems from Chongqing University, in 2009. He is currently an Associate Professor with the School of Microelectronics and Communication Engineering, Chongqing University. Until now, he has submitted more than 30 technical papers on the relevant fields. His research interests include digital image processing and communication signal processing.

DEJIAN LI (Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1999 and 2002, respectively, where he is currently pursuing the D.Eng. degree in electronic information.

He is currently the General Manager of the Digital IC Design Center, Beijing Smartchip Microelectronics Technology Company Ltd., China. His research interests include reliable architectures for industrial control chips, energy harvesting technologies for industrial sensor chips, and software-defined radios for industrial field communications.

YANTAO YU (Member, IEEE) received the B.S. (Hons.) and Ph.D. degrees in electrical engineering from the National University of Singapore, Singapore, in 2004 and 2009, respectively.

From 2008 to 2009, he was with Motorola Electronics Pte. Ltd., Singapore, as an RF Engineer. From 2009 to 2010, he was a Research Fellow with the National University of Singapore. He is currently an Associate Professor with the School of Microelectronics and Communication Engineering, Chongqing University, China. His research interests include microwave and RF communications and wireless communications.

HAISEN ZHAO received the B.S. degree in computer application engineering from the Taiyuan University of Technology, in 1992. He is currently a Senior Engineer and the Manager of Information Communication Company of Jinzhong Power Supply Company of State Grid Shanxi Province Electric Power Company. His research interests include power information and communication technology, and power security control technology.

TIANCONG HUANG received the B.S. and M.S. degrees in welding technology and automation and the Ph.D. degree in circuits and systems from Chongqing University, Chongqing, China, in 1993, 1996, and 2010, respectively. His research interests include new generation broadband mobile communication, power system communication and informatization, special communication, and emergency communication.