Energy-Efficient Resource Utilization for Heterogeneous Embedded Computing Systems

Jing Huang, Renfa Li, *Senior Member, IEEE*, Jiyao An, *Member, IEEE*, Derrick Ntalasha, Fan Yang, and Keqin Li, *Fellow, IEEE*

Abstract—In this paper, the joint optimization problem with energy efficiency and effective resource utilization is investigated for heterogeneous and distributed multi-core embedded systems. The system model is considered to be fully a heterogeneous model, that is, all nodes have different maximum speeds and power consumption levels from the perspective of hardware while they can employ different scheduling strategies from the perspective of applications. Since the concerned problem by nature is a multi-constrained and multi-variable optimization problem in which a closed-form solution cannot be obtained, our aim is to propose a power allocation and load balancing strategy based on Lagrange theory. Furthermore, when the problem cannot be fully solved by Lagrange approach, a data fitting method is employed to obtain core speed first, and then load balancing schedule is solved by Lagrange method. Several numerical examples are given to show the effectiveness of the proposed method and to demonstrate the impact of each factor to the present optimization system. Finally, simulation and practical evaluations show that the theoretical results are consistent with the practical results. To the best of our knowledge, this is the first work that combines load balancing, energy efficiency, hardware heterogeneity and application heterogeneity in heterogeneous and distributed embedded systems.

Index Terms—Embedded and distributed systems, energy efficiency, effective resource utilization, load distribution, power allocation, queueing model

1 Introduction

1.1 Motivation

A typical complex embedded system will have a heterogeneous distributed multi-core architecture that can respond to a variety of complicated computational requests at the application level. It is common for complex embedded systems, such as automotive electronics and avionics systems, to have over 60 Electronic Control Units (ECUs)[30], with each ECU dedicated to handling numerous tasks of different sizes and levels of urgency. As the complexity of embedded systems continues to increase to meet the demands of modern applications for increased computational power and performance, the need for energy efficiency and effective resource utilization will become increasingly significant. Current and future embedded systems must be able to assign general tasks to nodes in a manner that improves resource utilization without affecting dedicated tasks. Power must be allocated reasonably to each node in order to achieve minimum power usage by the system. Attaining optimal allocation of tasks and power in a distributed system is a well-known multi-variable optimization problem. In light of these issues, the development of heterogeneous distributed embedded systems is challenging.

In heterogeneous systems, the architecture of each node may differ, so the characteristics of nodes may vary. Each node might have different maximum and minimum core speed, or a different power consumption level [29]. The performance of the overall system can be influenced by any node. Therefore, to achieve energy efficiency in heterogeneous environments, the characteristics of each node must be considered carefully. From the point of view of distributed systems, each node is assigned preloaded dedicated tasks, and each task may have different task arrival rate and task size. To achieve effective utilization of resources, a distributed system requires an efficient load balancing algorithm that can assign tasks appropriately to each node. From the point of view of embedded systems, dedicated tasks executed on specified nodes are more important or urgent than general tasks. Moreover, each class of dedicated tasks has a different degree of urgency. To utilize all the available resources efficiently, each node should be set with an appropriate scheduling policy corresponding to the degree of urgency of dedicated tasks assigned to it. From the point of view of the overall system, computing performance is a vital metric when a system's Quality of Service (QoS) is being evaluated. Thus, the QoS still needs to be guaranteed.

Balancing all of these factors is a challenge for the development of heterogeneous distributed and embedded systems

- *J. Huang, R. Li, J. An, D. Ntalasha, and F. Yang are with the College of Computer Science and Electronic Engineering of Hunan University, National Supercomputing Center in Changsha, Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha 410082, China. E-mail: jingh@hnu.edu.cn, lirenfa@vip.sina.com, anbobcn@aliyun.com, dbntalasha@gmail.com, yangfanf117@126.com.*
- *K. Li is with the College of Computer Science and Electronic Engineering of Hunan University, National Supercomputing Center in Changsha, Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha 410082, China Department of Computer Science, State University of New York, New Paltz, NY 12561. E-mail: lik@newpaltz.edu.*

Manuscript received 6 Sept. 2016; revised 28 Mar. 2017; accepted 5 Apr. 2017. Date of publication 0 . 0000; date of current version 0 . 0000.
Recommended for acceptance by A. Yakovlev.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2017.2693186

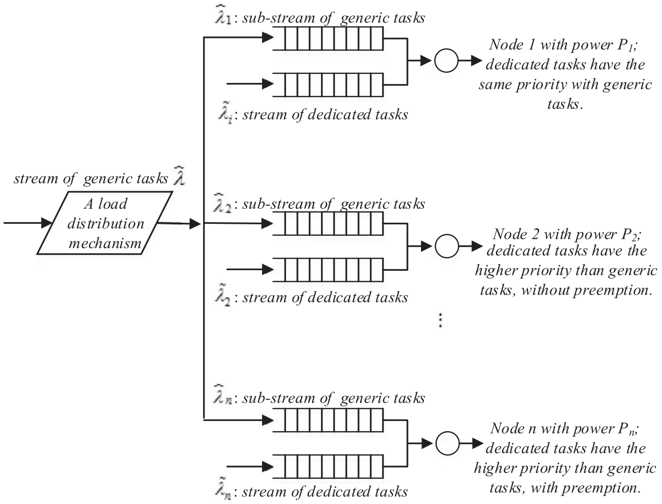

Fig. 1. System structure.

that are both energy efficient and making the best use of resources. Although there are many studies of the diverse aspects this problem, most of the existing research don't consider these factors jointly. Therefore, it is important to study how energy efficiency and high resource utilization can be achieved together on heterogeneous and distributed embedded systems.

1.2 Our Contributions

In this paper, we study the problem of assigning a set of general tasks to the computing nodes of a computational heterogeneous distributed embedded system, wherein each node is preloaded with a different number of dedicated tasks, equipped with a DVFS feature. The structure of the system is shown in Fig. 1. A node can be treated as a computational unit, which may include processor, memory etc.

Changing a node from its sleep state to a running state takes a long time [1]. In embedded environments, a node may be assigned important tasks that cannot be delayed. Consequently, we don't have the option to put an embedded node to sleep, even if its core is not working. In our investigations, to balance the power consumption and time delay, we assume that a core continues to run at a low frequency even when it is idle. Clearly, the power consumption differs when the core is working and when it is not working. Therefore, the cores can be considered to have two distinct modes [25]:

- *Core busy-power*: The power consumption of a core when there are tasks running on the core, is the major power consumption of a core.
- *Core idle-power*: The power consumption of a core when there is no task running.

We view each node as an M/M/1 queueing model with infinite waiting queue capacity [24], and define three queueing disciplines-Discipline 1, Discipline 2, and Discipline 3-each one of which could be employed by any node. The details of the disciplines are as follows:

- *Discipline 1*: All general tasks and dedicated tasks on this node are scheduled on a first-come, first-served basis, without priority. We identify this discipline as, "dedicated tasks without priority."
- *Discipline 2*: On this node, the queueing principle is that dedicated tasks are always scheduled before general tasks. All tasks are executed without interruption. We identify this discipline as, "prioritized dedicated tasks without preemption."
- *Discipline 3*: Dedicated tasks are always scheduled before general tasks on this node, with preemption. We term this discipline as, "prioritized dedicated tasks with preemption."

Our aim is to find the minimum overall power consumption of the system, along with the response time of general tasks, within an acceptable range. Our major contributions are as follows:

- To the best of our knowledge, this work is the first study of the minimum power consumption problem in heterogeneous distributed embedded systems that considers the load distribution in combination with the characteristics, queuing discipline, and idle speed of each node.
- We propose an algorithm for finding the optimal load distribution and power allocation scheme of the system, such that the overall power consumption of the system is minimized.
- We are the first to take the optimal solutions as training data to fit the relationship between the task size and core speed, and then use optimal load balancing to solve the problem when the problem cannot be solved by a Lagrangian system. Experimental results show this strategy to be efficient.
- Based on our algorithm, we show the influence of different parameters on the optimal power allocation and load distribution. These parameters include idle speed of core, as well as power consumption exponent α, preloaded tasks, queueing discipline, and number of nodes in the system. We provide numerical examples to demonstrate the effectiveness of our algorithm for each parameter. Furthermore, we give an example where all parameters are different. Simulation and practical evaluations show that the theoretical results are consistent with the practical results.

Our study focuses on a well-defined, multi-constrained, and multi-variable optimization problem. The investigation in this paper has made significant contribution to high-performance and energy-efficient computing in modern heterogeneous and distributed embedded systems.

2 Related Work

Because energy efficiency is a primary concern for embedded systems, especially for systems with limited power, this topic has been studied extensively, and a large body of literature exists [2], [3], [4], [5], [6]. In recent years, supercomputer operators also have paid considerable attention on energy efficiency because supercomputers have very large power requirements. While supercomputers are focused on performance as their most significant metric, the technique used by embedded systems to achieve energy efficiency is similar to that of supercomputers. Energy efficiency is about making power consumption proportional to system utilization [20] in a manner that decreases unnecessary energy loss. There are many approaches to achieving power reduction. Most commonly, dynamic voltage and frequency scaling (DVFS) [22], [23] is implemented at the operating system level to manage power and to regulate the frequency and voltage of CPUs. Generally speaking, two DVFS techniques exist for multi-core systems: One is global DVFS, which scales the frequency and voltage of all the cores simultaneously, and the other is local DVFS, which regulates the frequency and voltage on a per core basis [7]. Experiments indicated that local DVFS could achieve better performance than global DVFS [8], [9], but it is more complicated.

The energy efficiency of embedded systems has been studied by a number of researchers. Because the architectures and applications for embedded systems are quite diverse, researchers have needed to establish various theories to study the problem of energy efficiency in these different systems. In [26], the authors investigated the tradeoff between inter-application concurrency with performance and power consumption under various system configurations. They proposed a runtime optimization approach to achieve energy efficiency, implemented on a real platform called Odroid XU- 3. In [27], the minimum energy consumption was obtained based on a running model generated through regression-based learning of energy/performance trade-offs between different computing resources in the system. In [28], to support application quality of service and to save energy, an energy-efficient soft real-time CPU scheduler for mobile devices was proposed that primarily ran multimedia applications.

In addition to embedded computing, energy efficiency also plays an important role in cloud computing, which is marked by huge and increasing power consumption. The techniques for achieving energy efficiency used in multicore embedded systems and cloud computing systems are similar. Therefore, they could learn from each other. In [10], the author used DVFS and workload dependent dynamic power management to improve system performance and to reduce energy consumption. In [11], based on a cooperative game-theoretical approach and DVFS technology, the authors investigated the problem of allocating tasks onto a computational grid, with the aim of minimizing simultaneously the energy consumption and the makespan. In [12], the authors also employed a game-theoretic approach to study the problem of minimizing energy consumption in a distributed system.

An efficient load balancing strategy is a key component to building out any distributed architecture. The complexities are reflected in the extensive body of literature on the topic, as exemplified by the excellent reference collection given in [13]. The purpose of load balancing is to assign tasks appropriately to nodes in terms of the workload and computing power of each node. In [15], researchers proposed a fault tolerant, hybrid load balancing strategy for a heterogeneous grid computing environment. In [16], the authors addressed the problem of optimal load balancing of tasks when power is constrained.

The queueing discipline has also been studied widely. In [14], two types of cases were considered, namely, systems with and without special tasks. The authors addressed the problem of minimizing the average response time of generic tasks. Both [17] and [18] studied optimal load distribution in heterogeneous distributed computer systems with both generic and dedicated applications. In [17], each node was modeled as an M/G/1 non-preemptive queuing system, and was applied to several types of dedicated tasks, while in [18], each node was treated as an M/M/1 non-preemptive queuing system. The authors of [19] assumed that each node was preloaded with dedicated tasks, and three conditions were taken into account: Dedicated tasks without priority, and prioritized dedicated tasks with and without preemption. Each node was treated as an M/G/1 queueing system, and the authors focused on the problem of optimal load balancing of general tasks.

In distributed heterogeneous embedded systems, in order to achieve energy efficiency and effective utilization of resources, it is necessary to consider the combination of node heterogeneity, applications urgency (priority of tasks, which might be different for each node), energy efficiency, and the idle CPU state. To the best of our knowledge, present studies on load balancing and energy efficiency have not considered fully all of these factors together.

3 System Model and Problem Formulation

3.1 Power Model

The power dissipation of an embedded processor core mainly consists of three parts, namely, dynamic, static, and short-circuits consumption, among which dynamic power consumption is the dominant component. The dynamic power consumption can be expressed by $P = kCV^2f$ where k is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency. Given that $s \propto f$ and $f \propto V$, then $P_i \propto s_i^{\alpha_i}$, where α_i is around 3 [21]. For ease of discussion, we model the power allocated to processor core with speed s_i as ${s_i}^{\alpha_i}$.

The *core busy-power* is different from *core idle-power*. There are implied energy-frequency and frequency-performance relations. In this paper, the performance (speed) is defined as the number of instructions a core can perform per second (IPS). Therefor, the dynamic power is ${s_i}^{\alpha_i}$ when the core is working at frequency f_i and the corresponding speed is s_i. When a core is not working, because there are no instructions to perform, it is inappropriate to define the core speed directly. In that case, our research focuses on the power consumption rather than core speed. Therefore, when the core is idle, we assume the speed to be s_{Ii}, corresponding to a frequency f_i, such that ${s_{Ii}}^{\alpha_i}$ equals the actual power of the core, i.e., ${s_{Ii}}^{\alpha_i} = C{V_i}^2 f_i$. A processor core still consumes some amount of basic power P_i^* that includes static power dissipation, short circuit power dissipation, and other leakages and wasted power. Therefore, the power model can be formulated as

$$\begin{aligned} P_i &= (s_i^{\alpha_i} + P_i^*)\rho_i + (s_{Ii}^{\alpha_i} + P_i^*)(1-\rho_i) \\ &= \rho_i s_i^{\alpha_i} + (1-\rho_i) s_{Ii}^{\alpha_i} + P_i^* \\ &= \left(\widehat{\lambda}_i \widehat{r} + \widetilde{\lambda}_i \widetilde{r}_i\right) s_i^{\alpha_i - 1} + \left(1 - \frac{\widehat{\lambda}_i \widehat{r} + \widetilde{\lambda}_i \widetilde{r}_i}{s_i^{\alpha_i}}\right) s_{Ii}^{\alpha_i} + P_i^*. \end{aligned} \tag{1}$$

3.2 Queueing Model

The queueing model is used to formulate and study the problem of power allocation and load balancing in a heterogeneous distributed embedded environments. Taking n as the number of heterogeneous embedded computing nodes

TABLE 1
Mathematical Notations in This Paper

Symbol	Definition
s_i	The core speed of node v_i when it's core is busy
s_{Ii}	The core speed of node v_i when it's core is idle
α_j	power consumption exponent
$\widetilde{\lambda}_i$	Arrival rate of dedicated tasks to v_i
$\widehat{\lambda}_i$	Arrival rate of general tasks to v_i
λ_i	$= \widetilde{\lambda}_i + \widehat{\lambda}_i$
$\widehat{\lambda}$	$= \widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n$
$\widehat{r}$	Average task size of general tasks
$\widetilde{r}_i$	Average task size of dedicated tasks on v_i
$\widetilde{x}_i = \widetilde{r}_i/s_i$	Average execution time of dedicated tasks on v_i
$\widehat{x}_i = \widehat{r}/s_i$	Average execution time of general tasks on v_i
$\widehat{\rho}_i$	$\widehat{\lambda}_i \cdot \widehat{x}_i = \widehat{\lambda}_i\widehat{r} \big/ s_i$
$\widetilde{\rho}_i$	$\widetilde{\lambda}_i \cdot \widetilde{x}_i = \widetilde{\lambda}_i\widetilde{r}_i \big/ s_i$
$\rho_i = \widehat{\rho}_i + \widetilde{\rho}_i$	Average percentage of time that node v_i is busy
$\widehat{T}_i$	Average response time of general tasks on v_i
$\widehat{T}$	Acceptable time of general tasks on system

$v_1, v_2, \ldots, v_n$ (simply called as a node), each of which has its own dedicated set of jobs that follow a Poisson stream of tasks with arrival rate $\widetilde{\lambda}_i$ that can only be executed on it. There exists a general Poisson stream of tasks with arrival rate $\widehat{\lambda}$ that needs to be executed by being split into n substreams $\widehat{\lambda}_i$ assigned to each node. Thus, each node deals with a combined stream of dedicated and general tasks. The task size of dedicated and general tasks are exponential random variables rd_i and rg, respectively, with mean $\widetilde{r}_i$ and $\widehat{r}$, respectively. Thus, the two types of mean execution times on node v_i are $\widetilde{x}_i = \widetilde{r}_i/s_i, \widehat{x}_i = \widehat{r}/s_i$, respectively. Since the arrival rate and processing rate of tasks are subject to Poisson distribution, we can treat each node as an M/M/1 queueing system. Parameters used are shown in Table 1. To maintain the queue steady, we assume that $\rho_i < 1$, for all $1 \le i \le n$.

3.3 Problem Formulation

We specify our multi-variable optimization problem as follows: given n numbers of embedded nodes $v_1, v_2, \ldots, v_n$, the arrival rates $\widetilde{\lambda}_1, \widetilde{\lambda}_2, \ldots, \widetilde{\lambda}_n$ and average task size $\widetilde{r}_1, \widetilde{r}_2, \ldots, \widetilde{r}_n$ of dedicated tasks on each node, the total arrival rate $\widehat{\lambda}$ and average task size $\widehat{r}$ of general tasks, the idle-speed $s_{I1}, s_{I2}, \ldots, s_{In}$, base power supply $P_1^*, P_2^*, \ldots, P_n^*$, queueing discipline of each node, and the acceptable response time $\widehat{T}$ of generic tasks, find the task arrival rates $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ and core speeds $s_1, s_2, \ldots, s_n$ on each node such that the power consumption of the system $P = \sum_{i=1}^n P_i$ is minimized while satisfying the following constraints

$$\widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n = \widehat{\lambda}, \tag{2}$$

$$\frac{\widehat{\lambda}_1}{\lambda}\widehat{T}_1 + \frac{\widehat{\lambda}_2}{\lambda}\widehat{T}_2 + \cdots + \frac{\widehat{\lambda}_n}{\lambda}\widehat{T}_n \le \widehat{T}. \tag{3}$$

4 The Proposed Method

Each node is treated as an M/M/1 queuing system and has different queuing disciplines. Different queuing disciplines have different expressions of response time of general tasks. Thus, all nodes are divided into three groups according to the queuing discipline. We assume that group G_1 includes all those nodes whose queuing discipline is dedicated tasks without priority, group G_2 includes all those nodes whose queuing discipline is prioritized dedicated tasks without preemption, and group G_3 includes all those nodes whose queuing discipline is prioritized dedicated tasks with preemption.

Let $\widehat{T}_i$ denote the response time of generic tasks on node v_i. For node v_i belongs to group G_1 ($v_i \in G_1$), we have [24, p. 700]

$$\widehat{T}_i = \frac{\widehat{r}}{s_i} + \frac{\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^{\,2}}{s_i\left(s_i - \widehat{\lambda}_i\widehat{r} - \widetilde{\lambda}_i\widetilde{r}_i\right)}. \tag{4}$$

For node v_i belongs to group G_2 ($v_i \in G_2$), we have [24, p. 702]

$$\widehat{T}_i = \frac{\widehat{r}}{s_i} + \frac{\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^{\,2}}{\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i\right)\left(s_i - \widetilde{\lambda}_i\widetilde{r}_i - \widehat{\lambda}_i\widehat{r}\right)}. \tag{5}$$

For node v_i belongs to group G_3 ($v_i \in G_3$), we have [24, p. 704]

$$\widehat{T}_i = \frac{1}{s_i - \widetilde{\lambda}_i\widetilde{r}_i}\left(\widehat{r} + \frac{\widehat{\lambda}_i\widehat{r}^2 + \widetilde{\lambda}_i\widetilde{r}_i^{\,2}}{s_i - \widetilde{\lambda}_i\widetilde{r}_i - \widehat{\lambda}_i\widehat{r}}\right). \tag{6}$$

Our objective function is

$$\begin{aligned} &P\left(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n\right) \\ &= \sum_{i=1}^n \left(\left(\widehat{\lambda}_i\widehat{r} + \widetilde{\lambda}_i\widetilde{r}_i\right) s_i^{\alpha_i - 1} + \left(1 - \frac{\widehat{\lambda}_i\widehat{r} + \widetilde{\lambda}_i\widetilde{r}_i}{s_i}\right) s_{Ii}^{\alpha_i} + P_i^*\right), \end{aligned} \tag{7}$$

subject to

$$\sum_{v_i \in G_1} \frac{\widehat{\lambda}_i}{\lambda}\widehat{T}_i + \sum_{v_j \in G_2} \frac{\widehat{\lambda}_j}{\lambda}\widehat{T}_j + \sum_{v_k \in G_3} \frac{\widehat{\lambda}_k}{\lambda}\widehat{T}_k \le \widehat{T},$$

and

$$\widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n = \widehat{\lambda}.$$

Since the background of this problem is clear, we can use Lagrange system to solve our problem. We set

$$\begin{aligned} &\psi\left(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n\right) \\ &= \widehat{T} - \frac{1}{\widehat{\lambda}}\left(\sum_{v_i \in G_1} \widehat{\lambda}_i\widehat{T}_i + \sum_{v_i \in G_2} \widehat{\lambda}_i\widehat{T}_i + \sum_{v_i \in G_3} \widehat{\lambda}_i\widehat{T}_i\right), \end{aligned} \tag{8}$$

and

$$\varphi\left(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n\right) = \widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n - \widehat{\lambda}, \tag{9}$$

as two constraint functions. According to Lagrange system, we have

$$\begin{aligned} \nabla P = {} & \phi\nabla\varphi(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n) \\ & + \tau\nabla\psi(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n), \end{aligned}$$

that is,

$$\begin{aligned} \frac{\partial P}{\partial\widehat{\lambda}_i} = {} & \phi\frac{\partial\varphi\left(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n\right)}{\partial\widehat{\lambda}_i} \\ & + \tau\frac{\partial\psi\left(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n\right)}{\partial\widehat{\lambda}_i}, \end{aligned} \tag{10}$$

for all $1 \le i \le n$, where ϕ and τ are two Lagrange multipliers, and

$$\frac{\partial P}{\partial s_i} = \phi \frac{\partial \varphi\left(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n\right)}{\partial s_i} + \tau \frac{\partial \psi\left(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n, s_1, s_2, \ldots, s_n\right)}{\partial s_i}. \tag{11}$$

Based on Equation (10), we get

$$\widehat{r} {s_i}^{\alpha_i - 1} - \frac{\widehat{r} {s_{Ii}}^{\alpha_i}}{s_i} = \phi - \tau\left(\widehat{T}_i + \widehat{\lambda}_i \frac{\partial \widehat{T}_i}{\partial \widehat{\lambda}_i}\right), \tag{12}$$

where for all $v_i \in G_1$, we have

$$\frac{\partial \widehat{T}_i}{\partial \widehat{\lambda}_i} = \frac{\widehat{r}^2\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right) + \widetilde{\lambda}_i {\widetilde{r}_i}^2 \widehat{r}}{s_i\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i - \widehat{\lambda}_i \widehat{r}\right)^2};$$

for all $v_i \in G_2$, we have

$$\frac{\partial \widehat{T}_i}{\partial \widehat{\lambda}_i} = \frac{\widehat{r}}{s_i} + \frac{\left(2\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right) - {\widehat{\lambda}_i}^2 \widehat{r}^3}{\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right)\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i - \widehat{\lambda}_i \widehat{r}\right)^2};$$

for all $v_i \in G_3$, we have

$$\frac{\partial \widehat{T}_i}{\partial \widehat{\lambda}_i} = \frac{\widehat{r}^2\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right) + \widetilde{\lambda}_i \widehat{r} {\widetilde{r}_i}^2}{\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right)\left(s_i - \widehat{\lambda}_i \widehat{r} - \widetilde{\lambda}_i \widetilde{r}_i\right)^2}.$$

Based on Equation (12), for all $v_i \in G_1$, we can get

$${\widehat{\lambda}_i}^2 a_i + \widehat{\lambda}_i b_i + c_i = 0, \tag{13}$$

where

$$\begin{aligned}
a_i &= \widehat{r}^2 R_i, \\
b_i &= -2\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right)\widehat{r} R_i, \\
c_i &= -\left(\left(\widehat{r} - R_i\right)\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right) + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right), \\
R_i &= \frac{\phi s_i + \widehat{r} {s_{Ii}}^{\alpha_i} - \widehat{r} {s_i}^{\alpha_i}}{\tau s_i}.
\end{aligned}$$

Solving Equation (13), we can obtain

$$\widehat{\lambda}_i = \frac{t_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{t_i\left(\widehat{r} t_i + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\tau}{d_i}}, \tag{14}$$

for all $v_i \in G_1$; similarly, based on Equation (12), we can get

$$\widehat{\lambda}_i = \frac{t_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{t_i\left(\widehat{r} t_i + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)s_i}{d_i t_i/\tau + \widehat{r}\widetilde{\lambda}_i \widetilde{r}_i}}, \tag{15}$$

for all $v_i \in G_2$; and

$$\widehat{\lambda}_i = \frac{s_i - \widetilde{\lambda}_i \widetilde{r}_i}{\widehat{r}} - \frac{1}{\widehat{r}}\sqrt{\frac{\left(\widehat{r} t_i + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\tau s_i}{d_i}}, \tag{16}$$

for all $v_i \in G_3$, where

$$t_i = s_i - \widetilde{\lambda}_i \widetilde{r}_i, d_i = \widehat{r} {s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r} {s_i}^{\alpha_i}.$$

Based on Equation (11), we take the partial derivative with respect to s_i, that is,

$$-\tau \widehat{\lambda}_i \frac{\partial \widehat{T}_i}{\partial s_i} = (\alpha_i - 1)\left(\widehat{\lambda}_i \widehat{r} + \widetilde{\lambda}_i \widetilde{r}_i\right){s_i}^{\alpha_i - 2} + \frac{\left(\widehat{\lambda}_i \widehat{r} + \widetilde{\lambda}_i \widetilde{r}_i\right)}{{s_i}^2} {s_{Ii}}^{\alpha_i}, \tag{17}$$

where for all $v_i \in G_1$ we have

$$\frac{\partial \widehat{T}_i}{\partial s_i} = -\left(\frac{\widehat{r}}{{s_i}^2} + \frac{\left(\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\left(2s_i - \widehat{\lambda}_i \widehat{r} - \widetilde{\lambda}_i \widetilde{r}_i\right)}{{s_i}^2\left(s_i - \widehat{\lambda}_i \widehat{r} - \widetilde{\lambda}_i \widetilde{r}_i\right)^2}\right),$$

for all $v_i \in G_2$ we have

$$\frac{\partial T_i}{\partial s_i} = -\frac{\widehat{r}}{{s_i}^2} - \frac{\left(\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\left(2\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right) - \widehat{\lambda}_i \widehat{r}\right)}{\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right)^2\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i - \widehat{\lambda}_i \widehat{r}\right)^2},$$

and for all $v_i \in G_3$ we have

$$\frac{\partial \widehat{T}_i}{\partial s_i} = -\left(\frac{\left(\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\left(2\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right) - \widehat{\lambda}_i \widehat{r}\right)}{\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i - \widehat{\lambda}_i \widehat{r}\right)^2} + \widehat{r}\right) \times \frac{1}{\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right)^2}.$$

Through taking the derivative with respect to $\widehat{\lambda}_i$ and s_i respectively, we have obtained the Equations (14), (15), (16) and (17) for all $1 \le i \le n$. Basing on these Equations, our problem is modified to find the appropriate ϕ, τ and each node speed s_i to satisfy the conditions Equations (2) and (3).

This is a well-defined multi-variable optimization problem which is difficult to get a closed-form solution especially that different queueing disciplines have different expressions of $\widehat{\lambda}_i$ and $\partial \widehat{T}_i \big/ \partial s_i$. Thus, we have to devise the numerical solution. We consider

$$f_i(s_i, \phi, \tau) = \begin{cases} \dfrac{t_i}{\widehat{r}} - \dfrac{1}{\widehat{r}}\sqrt{\dfrac{t_i\left(\widehat{r} t_i + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\tau}{d_i}}, v_i \in G_1; \\ \dfrac{t_i}{\widehat{r}} - \dfrac{1}{\widehat{r}}\sqrt{\dfrac{t_i\left(\widehat{r} t_i + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)s_i}{d_i t_i/\tau + \widehat{r}\widetilde{\lambda}_i \widetilde{r}_i}}, v_i \in G_2; \\ \dfrac{t_i}{\widehat{r}} - \dfrac{1}{\widehat{r}}\sqrt{\dfrac{\left(\widehat{r} t_i + \widetilde{\lambda}_i {\widetilde{r}_i}^2\right)\tau s_i}{d_i}}, v_i \in G_3; \end{cases} \tag{18}$$

and

$$g_i(s_i, \widehat{\lambda}_i) = \frac{\widehat{\lambda}_i \widehat{r} + \widetilde{\lambda}_i \widetilde{r}_i}{-\widehat{\lambda}_i \partial \widehat{T}_i \big/ \partial s_i}\left((\alpha_i - 1){s_i}^{\alpha_i - 2} + \frac{{s_{Ii}}^{\alpha_i}}{{s_i}^2}\right), \tag{19}$$

where

$$t_i = s_i - \widetilde{\lambda}_i \widetilde{r}_i, d_i = \widehat{r} {s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r} {s_i}^{\alpha_i}.$$

Since $\widehat{\lambda}_i$ is viewed as a function of s_i, ϕ and τ, and τ is treated as a function of s_i and $\widehat{\lambda}_i$, it needs to find the domain definition of functions $f_i(s_i, \phi, \tau)$ and $g_i(s_i, \widehat{\lambda}_i)$. The tasks arrival rate $\widehat{\lambda}_i$ must be larger than zero, and $\rho_i < 1$. Hence, we have

$$\begin{cases} \widehat{\lambda}_i \geq 0; \\ s_i - \widetilde{\lambda}_i \widetilde{r}_i - \widehat{\lambda}_i \widehat{r} \geq 0; \\ \widehat{r} s_{Ii}^{\alpha_i} + \phi s_i - \widehat{r} s_i^{\alpha_i} > 0; \end{cases} \tag{20}$$

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

In real situations of distributed environments, α_i may be a decimal and not the same for different nodes, therefore, it is impossible to obtain a closed-form solution of Equation (20). We shall give the numerical solution in Section 5.

How to obtain the appropriate $\widehat{\lambda}_i$, s_i, ϕ and τ based on $f_i(s_i, \phi, \tau)$ and $g_i(s_i, \widehat{\lambda}_i)$ that can satisfy constraint conditions Equations (2) and (3) will be introduced in Section 5. We give the derivations with respect to s_i of function $f_i(s_i, \phi, \tau)$ and $g_i(s_i, \widehat{\lambda}_i)$, which will be used in Section 5

$$f_i'(s_i, \phi, \tau) = \frac{1}{\widehat{r}} - \frac{h_i'(s_i)}{\widehat{r} 2\sqrt{h_i(s_i)}},$$

where $h_i(s_i) = \dfrac{t_i H_i \tau}{d_i}$, and

$$h_i'(s_i) = \frac{\tau}{d_i}\left(\widehat{r} t_i + H_i\left(1 - \frac{t_i(\phi - \alpha_i \widehat{r} s_i^{\alpha_i - 1})}{d_i}\right)\right),$$

for all $v_i \in G_1$; $h_i(s_i) = \dfrac{t_i H_i s_i \tau}{d_i t_i + \tau \widehat{r} \widetilde{\lambda}_i \widetilde{r}_i}$, and

$$h_i'(s_i) = \tau\left(\frac{(H_i + t_i \widehat{r}) s_i + t_i H_i}{d_i t_i + \tau \widehat{r} \widetilde{\lambda}_i \widetilde{r}_i} - \frac{(d_i' t_i + d_i) t_i H_i s_i}{\left(d_i t_i + \tau \widehat{r} \widetilde{\lambda}_i \widetilde{r}_i\right)^2}\right),$$

for all $v_i \in G_2$; $h(s_i) = \dfrac{H_i \tau s_i}{d_i}$, and

$$h'(s_i) = \tau \frac{(\widehat{r} s_i + H_i) d_i - d_i' H_i s_i}{d_i^2},$$

for all $v_i \in G_3$, and

$$t_i = s_i - \widetilde{\lambda}_i \widetilde{r}_i, d_i = \widehat{r} s_{Ii}^{\alpha_i} + \phi s_i - \widehat{r} s_i^{\alpha_i},$$
$$H_i = \widehat{r}\left(s_i - \widetilde{\lambda}_i \widetilde{r}_i\right) + \widetilde{\lambda}_i \widetilde{r}_i^2, d_i' = \phi - \widehat{r} \alpha_i s_i^{\alpha_i - 1}.$$

We get

$$\begin{aligned} g_i'(s_i, \widehat{\lambda}_i) = & \left(-\left((\alpha_i - 1) s_i^{\alpha_i - 2} + \frac{s_{Ii}^{\alpha_i}}{s_i^2}\right)\right. \\ & \times \left(\frac{\widehat{\lambda}_i'}{\widehat{\lambda}_i} \widetilde{\lambda}_i \widetilde{r}_i + \frac{\Delta\left(\partial \widehat{T}_i / \partial s_i\right) / \Delta s_i}{\partial \widehat{T}_i / \partial s_i}\left(\widehat{\lambda}_i \widehat{r} + \widetilde{\lambda}_i \widetilde{r}_i\right)\right) \\ & + \left((\alpha_i - 1)(\alpha_i - 2) s_i^{\alpha_i - 3} - \frac{2 s_{Ii}^{\alpha_i}}{s_i^3}\right)\left(\widehat{\lambda}_i \widehat{r} + \widetilde{\lambda}_i \widetilde{r}_i\right)\Bigg) \\ & \times \frac{1}{\widehat{\lambda}_i \partial \widehat{T}_i / \partial s_i}, \end{aligned}$$

where

$$\begin{aligned} -\frac{\Delta\left(\partial \widehat{T}_i / \partial s_i\right)}{\Delta s_i} = & \frac{-2\widehat{r}}{t_i^3} + \frac{2t_i - \widehat{\lambda}_i \widehat{r}}{t_i^2 \left(t_i - \widehat{\lambda}_i \widehat{r}\right)^2} \\ & \times \left(\widehat{\lambda}_i' \widehat{r}^2 - \left(\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2\right)\right. \\ & \times \left.\left(\frac{1}{t_i} + \frac{1 - \widehat{\lambda}_i' \widehat{r}}{t_i - \widehat{\lambda}_i \widehat{r}}\right)\right) - \frac{1}{t_i} \frac{\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2}{\left(t_i - \widehat{\lambda}_i \widehat{r}\right)} \\ & \times \left(\frac{1}{t_i^2} + \frac{1 - \widehat{\lambda}_i' \widehat{r}}{\left(t_i - \widehat{\lambda}_i \widehat{r}\right)^2}\right), \end{aligned}$$

for all $v_i \in G_1$;

$$-\frac{\Delta\left(\partial \widehat{T}_i / \partial s_i\right)}{\Delta s_i} = \frac{\widehat{r}}{\lambda s_i^2}\left(f_i'(s_i) - \frac{2}{s_i} \widehat{\lambda}_i\right) + \frac{D_i}{\lambda t_i \left(t_i - \widehat{\lambda}_i \widehat{r}\right)^2},$$

for all $v_i \in G_2$, where

$$\begin{aligned} D_i = & f_i'(s_i)\left(2 - \frac{\widehat{\lambda}_i \widehat{r}}{t_i}\right) \times \left(2\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2\right) \\ & - \frac{\widehat{\lambda}_i\left(\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2\right)}{t_i\left(t_i - \widehat{\lambda}_i \widehat{r}\right)} \times \left(6t_i + \widehat{\lambda}_i \widehat{r}\left(\frac{2\widehat{\lambda}_i \widehat{r}}{t_i} - 6\right)\right. \\ & \left. + \widehat{r}^2 f_i'(s_i)\left(\widehat{\lambda}_i \widehat{r} - 3t_i\right)\right), \end{aligned}$$

$t_i = s_i - \widetilde{\lambda}_i \widetilde{r}_i$; and

$$\begin{aligned} -\frac{\Delta\left(\partial \widehat{T}_i / \partial s_i\right)}{\Delta s_i} = & \frac{-2\widehat{r}}{t_i^3} + \frac{2t_i - \widehat{\lambda}_i \widehat{r}}{t_i^2\left(t_i - \widehat{\lambda}_i \widehat{r}\right)^2} \\ & \times \left(f_i'(s_i) \widehat{r}^2 - \left(\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2\right)\left(\frac{1}{t_i} + \frac{1 - f_i'(s_i) \widehat{r}}{t_i - \widehat{\lambda}_i \widehat{r}}\right)\right) \\ & - \left(\widehat{\lambda}_i \widehat{r}^2 + \widetilde{\lambda}_i \widetilde{r}_i^2\right) \\ & \times \left(\frac{(1 - f_i'(s_i) \widehat{r}) t_i^2 + \left(t_i - \widehat{\lambda}_i \widehat{r}\right)^2}{t_i^3 \left(t_i - \widehat{\lambda}_i \widehat{r}\right)^3}\right), \end{aligned}$$

with $t_i = s_i - \widetilde{\lambda}_i \widetilde{r}_i$, for all $v_i \in G_3$.

Based on above equations, the next section will introduce how to employ algorithms to obtain the appropriate τ, ϕ and s_i of each node that satisfy Equations (2) and (3).

5 The Algorithm

We will implement the algorithm to solve the present multi-variable optimization problem. And, how to obtain the definition domain of s_i is described in Section 5.1, while Section 5.2 introduces how to find the appropriate Lagrange multipliers ϕ and τ based on $f_i(s_i)$ and $g_i(s_i, f_i(s_i))$ in Equations (18) and (19) under the constraint conditions in (2) and (3). Furthermore, since that the difference between the preloaded tasks and α of each node is large, and then it is difficult to accomplish load balancing only by the Lagrange theory, Section 5.3 will solve the multi-variable optimization problem by combining with the Lagrange method and data fitting technique.

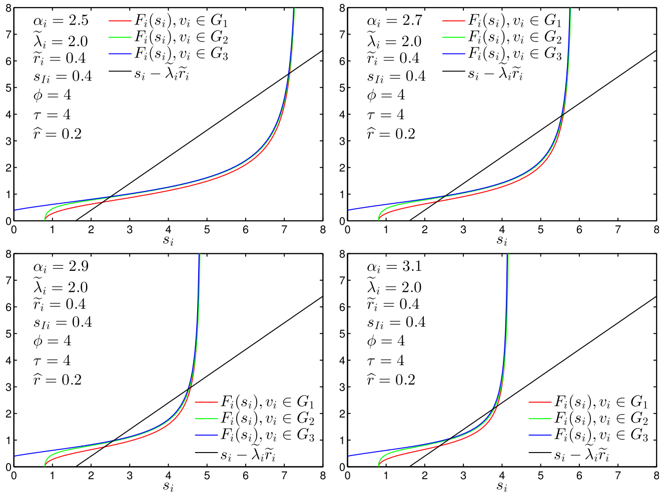

Fig. 2. Several examples of $F_i(s_i)$.

5.1 Defining the Search Space of s_i

As mentioned in Section 4, we view $f_i(s_i, \phi, \tau)$ and $g_i(s_i, \widehat{\lambda}_i)$ as functions of s_i, for all $1 \le i \le n$, the domain definition of which is defined by Equation (20). However, it is impossible to get a closed-form solution with regard to s_i for Equation (20). We have to devise a numeric solution. We consider

$$f_i(s_i, \phi, \tau) = t_i - F_i(s_i),$$

where

$$F_i(s_i) = \begin{cases} \sqrt{t_i\left(\widehat{r}t_i + \widetilde{\lambda}_i\widetilde{r}_i^{\,2}\right)\tau \big/ d_i}, v_i \in G_1; \\ \sqrt{\dfrac{t_i\left(\widehat{r}t_i + \widetilde{\lambda}_i\widetilde{r}_i^{\,2}\right)s_i\tau}{d_it_i + \tau\widehat{r}\widetilde{\lambda}_i\widetilde{r}_i}}, v_i \in G_2; \\ \sqrt{\left(\widehat{r}t_i + \widetilde{\lambda}_i\widetilde{r}_i^{\,2}\right)\tau \big/ d_i}, v_i \in G_3; \end{cases}$$

and

$$t_i = s_i - \widetilde{\lambda}_i\widetilde{r}_i, d_i = \widehat{r}{s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r}{s_i}^{\alpha_i}.$$

Given ϕ and τ, functions $F_i(s_i)$ for all $1 \le i \le n$ have the similar changing trends as core speed s_i changes. Fig. 2 shows an example of $F_i(s_i)$. Assume $F_i(s_i)$ and t_i intersect at two points $(s_{a_i}, F_i(s_{a_i}))$ and $(s_{b_i}, F_i(s_{b_i}))$. If $s_{a_i} \le s_i \le s_{b_i}$, then $f_i(s_i) \ge 0$; else $f_i(s_i) \le 0$. The two values s_{a_i} and s_{b_i} are respectively the lower bound and upper bound of domain definition of function $f_i(s_i)$. In order to search for the values of s_{a_i} and s_{b_i}, we need a point s_ξ which satisfies $f_i(s_\xi) > 0$. According to the rule that if $s_{a_i} \le s_i \le s_{b_i}$, then $f_i(s_i) \ge 0$; else $f_i(s_i) \le 0$, we can respectively employ binary search to find the value of s_{a_i} between $[\widetilde{\lambda}_i\widetilde{r}_i, s_\xi]$ and the value of s_{b_i} between $[s_\xi, s_{i\_\max}]$, where $s_{i\_\max}$ represents the solution of $\widehat{r}{s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r}{s_i}^{\alpha_i} = 0$. Basing on the Lagrange Mean Value Theorem, there must be a point s_ξ between s_{a_i} and s_{b_i} that makes

$$F_i'(s_\xi) = \frac{F_i(s_{b_i}) - F_i(s_{a_i})}{s_{b_i} - s_{a_i}} = \frac{t_i(s_{b_i}) - t_i(s_{a_i})}{s_{b_i} - s_{a_i}} = 1.$$

When $s_{a_i} \le s_i \le s_\xi$, we have $F_i'(s_\xi) \le 1$ and $f_i(s_i) \ge 0$; while $s_\xi \le s_i$, we have $F_i'(s_\xi) \ge 1$ and $f_i(s_i) \ge 0$. To take advantage of this feature, Squeeze theorem and binary search method are used to quickly find a point s_ξ satisfying $f_i(s_\xi) > 0$. Then, taking the value of s_ξ to find the low bound s_{a_i} and upper bound s_{b_i} (See Algorithm 2). The binary search will be mostly used in our algorithms. In order to avoid repeatedly using a list of search methods, we define it in Algorithm 1.

Algorithm 1. biSearch($var, lb, ub, criterion$)

```
Input: var, lb, ub, criterion
Output: var
while (ub − lb > ε) do
    var ← (ub + lb)/2;
    if (criterion) then
        ub ← var;
    else
        lb ← var;
    end if
end while
return var.
```

Algorithm 2. getDomainof $s_i(\widetilde{\lambda}_i, \widetilde{r}_i, \widehat{r}, \phi, \tau)$

```
Input: λ̃_i, r̃_i, r̂, φ, τ.
Output: lbs_i, ubs_i.
lb ← λ̃_i r̃_i; ub ← MaxS_i;
s_i ← biSearch(s_i, lb, ub, φs_i + r̂ s_Ii^α_i − r̂ s_i^α_i < 0);
s_max ← s_i; lb ← λ̃_i r̃_i;
while (f_i(s_i, φ, τ) < 0) do
    if (f'_i(s_i, φ, τ) < 0) then
        lb ← s_i;
    else
        ub ← s_i;
    end if
    s_i ← (ub + lb)/2;
end while
lb ← λ̃_i r̃_i; ub ← s_i; s_ξ ← s_i;
s_i ← biSearch(s_i, lb, ub, f_i(s_i, φ, τ) > 0));
lbsi ← s_i; lb ← s_ξ; ub ← s_max;
s_i ← biSearch(s_i, lb, ub, f_i(s_i, φ, τ) < 0));
ubsi ← s_i;
return ubsi, lbsi.
```

5.2 Searching for Lagrange Multipliers

Our target is to find the appropriate ϕ , τ and all of s_i $(1 \le i \le n)$, which can make conditions Equations (2) and (3) be satisfied basing on $f_i(s_i)$ and $g_i(s_i, f_i(s_i))$. Our strategy is that by fixing a Lagrange multiplier we try to search for an appropriate value of the other Lagrange multiplier which can make one constraint condition (Equations (2) or (3) be satisfied, then adjusting the value of first Lagrange multiplier, under the new value, we continue to search the corresponding value of the other Lagrange multiplier. The process will finish only if the two appropriate Lagrange multipliers are found, or if the loop conditions are violated.

5.2.1 *Searching Lagrange Multiplier* τ

Theorem 1. *If there is no dedicated task on a node* ($\widetilde{\lambda}_i = 0$), *then the speed of the node* s_i *is independent of* τ, *and has the following form:*

$${s_{Ii}}^{\alpha_i}(1 - \widehat{r}) = \left(\widehat{r}\alpha_i{s_i}^{\alpha_i - 1} - \phi\right)s_i.$$

Proof. Taking $\widetilde{\lambda}_i = 0$ into $f_i(s_i)$ and $\partial \widehat{T}_i \big/ \partial s_i$, we can get

$$\widehat{\lambda}_i = \frac{s_i}{\widehat{r}}\left(1 - \sqrt{\frac{\widehat{r}\tau}{{s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r}{s_i}^{\alpha_i}}}\right), \tag{21}$$

and

$$\begin{aligned} \frac{\partial \widehat{T}_i}{\partial s_i} &= -\frac{1}{{s_i}^2}\left(\widehat{r} + \frac{\widehat{\lambda}_i \widehat{r}^2\left(2s_i - \widehat{\lambda}_i\widehat{r}\right)}{\left(s_i - \widehat{\lambda}_i\widehat{r}\right)^2}\right) \\ &= -\frac{1}{{s_i}^2}\left(\widehat{r} + \frac{\frac{s_i}{\widehat{r}}(1-L_i)\widehat{r}^2\left(2s_i - \frac{s_i}{\widehat{r}}(1-L_i)\widehat{r}\right)}{\left(s_i - \frac{s_i}{\widehat{r}}(1-L_i)\widehat{r}\right)^2}\right) \\ &= -\frac{\widehat{r}}{{s_i}^2}\left(1 + \frac{{s_i}^2(1-L_i)(1+L_i)}{{s_i}^2{L_i}^2}\right) \\ &= -\frac{\widehat{r}}{{s_i}^2}\frac{1}{{L_i}^2} \\ &= -\frac{{s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r}{s_i}^{\alpha_i}}{\tau {s_i}^2}, \end{aligned} \tag{22}$$

where

$$L_i = \sqrt{\frac{\widehat{r}\tau}{{s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r}{s_i}^{\alpha_i}}}.$$

Substituting Equations (21) and (22) into Equation (17), we can get

$$\tau = \frac{\widehat{r}}{\frac{{s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r}{s_i}^{\alpha_i}}{\tau {s_i}^2}}\left((\alpha_i - 1){s_i}^{\alpha_i - 2} + \frac{{s_{Ii}}^{\alpha_i}}{{s_i}^2}\right),$$

that is,

$$\frac{{s_{Ii}}^{\alpha_i} + \phi s_i - \widehat{r}{s_i}^{\alpha_i}}{{s_i}^2} = \widehat{r}\left((\alpha_i - 1){s_i}^{\alpha_i - 2} + \frac{{s_{Ii}}^{\alpha_i}}{{s_i}^2}\right).$$

Basing on the above equation, we obtain

$${s_{Ii}}^{\alpha_i}(1 - \widehat{r}) = \left(\widehat{r}\alpha_i {s_i}^{\alpha_i - 1} - \phi\right)s_i,$$

and the theorem is proven.

If there are dedicated tasks on a node, it is very difficult to directly solve this problem by using mathematical derivation, and impossible to get a closed-form solution. Through observing the form of $g_i(s_i, f_i(s_i))$, we notice that if we set $\widetilde{\lambda}_i = 0$ (there are no dedicated jobs), then the form of $g_i(s_i, f_i(s_i))$ will be translated into

$$g_i(s_i, \widehat{\lambda}_i) = \frac{1}{-\partial \widehat{T}_i \big/ \partial s_i}\left((\alpha_i - 1){s_i}^{\alpha_i - 2} + \frac{{s_{Ii}}^{\alpha_i}}{{s_i}^2}\right).$$

The response time T_i could be treated as a convex function of s_i. Thus, $-\partial \widehat{T}_i \big/ \partial s_i$ will decrease as s_i increases. ${s_i}^{\alpha_i - 2}$ is an increasing function of s_i, and ${s_{Ii}}^{\alpha_i} / {s_i}^2$ is a decreasing function of s_i, which implies that the above equation should decrease as s_i increases and then increase as s_i continues to increase. The feature of function $g_i(s_i, f_i(s_i))$ will not be changed even if $\widetilde{\lambda}_i$ is not equal to zero. Virtually this feature can be observed from a great deal of data experiments. Fig. 3 shows an example of four nodes, and each node is preloaded with different amount of tasks.

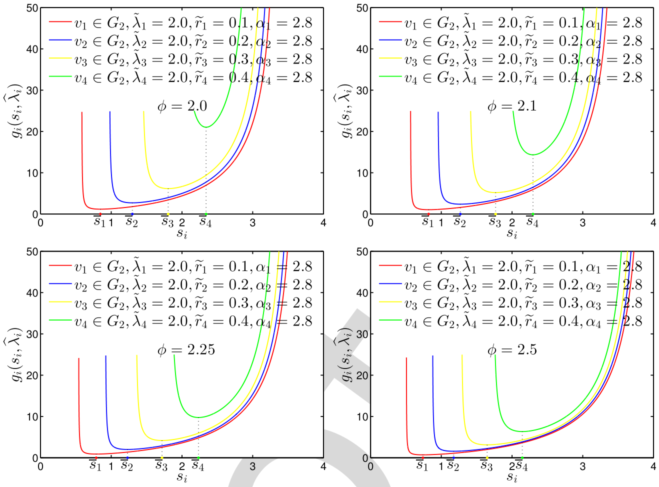

Fig. 3. Several examples of function $g_i(s_i, \widehat{\lambda}_i)$.

Algorithm 3. find turning Point $\overline{s_i}(\phi, \widetilde{\lambda}_i, \widetilde{r}_i)$

Input: $\phi, \widetilde{\lambda}_i, \widetilde{r}_i$.
Output: $\overline{s_i}$.
1: $lb, ub \leftarrow getDomainofs_i(\widetilde{\lambda}_i, \widetilde{r}_i, \widehat{r}, \phi, \tau)$;
2: $s_i \leftarrow biSearch(s_i, lb, ub, g_i'(f_i(s_i), s_i) \le 0)$;
3: **return** $\overline{s_i} \leftarrow s_i$.

Each $g_i(s_i, \widehat{\lambda}_i)$ has its minimum value, meaning that there is a point $\overline{s_i}$ of speed s_i that makes $g_i(s_i, \widehat{\lambda}_i)$ obtain the minimum value. The $\overline{s_i}$ can be obtained by using the derivative of $g_i(s_i, \widehat{\lambda}_i)$ with respect to s_i (See Algorithm 3). The Lagrange multiplier τ should have the same value for each $g_i(s_i, \widehat{\lambda}_i)$, where $1 \le i \le n$. Thus, the low bound of $g_i(s_i, \widehat{\lambda}_i)$ is the maximum value of all minimum values of $g_i(s_i, \widehat{\lambda}_i)$, namely, the maximum value of all $g_i(\overline{s_i}, f_i(\overline{s_i}))$. From Fig. 3, we can observe that for each $g_i(s_i, \widehat{\lambda}_i)$ there are two values of s_i that can be mapped onto the same value of τ, one located at the left side of $\overline{s_i}$ and the other one located at the right side of $\overline{s_i}$. Notice that for the same τ, the difference in speed s_i of each node with different preloaded tasks is large, and the change in value of function $g_i(s_i, \widehat{\lambda}_i)$ as s_i changes is drastic, if we take the left side value as the value of s_i, this implies that the left side value is not a suitable value of s_i. In fact, we have tried to take the left side value as the value of s_i, and the result is abnormal. Thus, given a value of τ, we adopt the right side value corresponding to the τ as the value of s_i. At the right side of $\overline{s_i}$, each $g_i(s_i, \widehat{\lambda}_i)$ is a monotone increasing function of s_i. Hence, given a value of $g_i(s_i, \widehat{\lambda}_i)$, we can immediately get the corresponding value of s_i (See Algorithm 4).

Algorithm 4. Calculate $s_i(\tau, \overline{s}_i)$

Input: $\tau, \overline{s}_i$.
Output: s_i.
1: $lb \leftarrow \overline{s}_i$; $ub \leftarrow ubsi$;
2: $s_i \leftarrow biSearch(s_i, lb, ub, g_i(s_i, f_i(s_i)) < \tau)$;
3: **return** s_i.

In order to satisfy the conditions in Equations (2) and (3), we need to adjust ϕ and τ. Each $g_i(s_i, \widehat{\lambda}_i)$ is treated as a function of s_i, and still represents the value of τ. Function $f_i(s_i)$

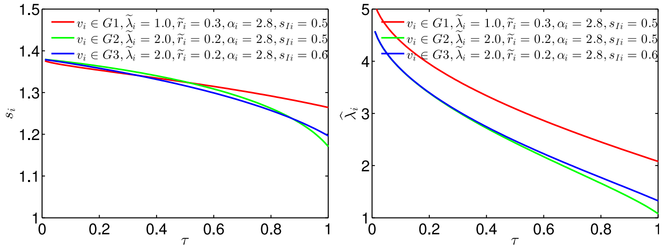

Fig. 4. The s_i and $\widehat{\lambda}_i$ changing tendency as τ changes.

representing $\widehat{\lambda}_i$ also contains τ. Thus, the value of $g_i(s_i, \widehat{\lambda}_i)$ should equal the value of τ included in $f_i(s_i)$. Let $g_i(s_i, \widehat{\lambda}_i) = \tau$, through Algorithm 4 we can get the value of s_i corresponding to the τ. Our target is to get an appropriate τ which can make Equation (3) is satisfied. Thus, we have to adjust the value of τ. By analyzing Equation (17), we observe that the s_i will decrease as τ increases. In Fig. 4, we respectively give a series of s_i and corresponding $\widehat{\lambda}_i$, which are the solutions of $g_i(s_i, \widehat{\lambda}_i) = \tau$ when τ is set to a different values. In distributed systems, if the core speed of a node is reduced, then the node will be assigned with lesser tasks, this leads to the average response time of general tasks on this node to reduce. Since $\widehat{T}_i \propto \widehat{\lambda}_i$, $\widehat{\lambda}_i \propto s_i$ and $s_i \propto 1/\tau$, then $\widehat{\lambda}_i \widehat{T}_i \propto 1/\tau$, this implies that given a appropriate value of ϕ, the binary search method could be employed to find the appropriate τ that can be used to get all speeds s_i making the constraint condition Equation (3) be satisfied (see Algorithm 5).

Algorithm 5. CalculateAll s_i

Input: $\phi, \widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_n, \widetilde{r}_1, \ldots, \widetilde{r}_n, \widehat{r}, s_{I1}, \ldots, s_{In}, \widehat{T}$.
Output: $s_1, s_2, \ldots, s_n$.
1: **for** $(1 \leftarrow i; i \leq n; i \leftarrow i+1)$ **do**
2: $\quad \overline{s}_i \leftarrow$ findTurningPoint $\overline{s}_i(\phi, \widehat{\lambda}_i, \widetilde{r}_i)$;
3: **end for**
4: $ub \leftarrow \zeta$; $lb \leftarrow 0$;
5: $\tau, s_i \leftarrow biSearch(\tau, lb, ub,$
$\quad \frac{1}{\lambda}\sum_{i=1}^{n} f_i(calculates_i(\tau, \overline{s_i}), \phi, \tau)\widehat{T}_i < \widehat{T})$;
6: **return** $\tau, s_1, s_2, \ldots, s_n$.

5.2.2 *Searching Lagrange Multiplier ϕ*

It can be observed from Equation (18) that for all $1 \leq i \leq n$, if we reduce the value of ϕ, then the value of $f_i(s_i)$ will decrease. Thus, actually, $f_i(s_i)$ could be viewed as increasing function of ϕ. Since given an appropriate ϕ, a τ that makes condition Equation (3) be satisfied could be obtained, the condition Equation (2) also can be met by adjusting ϕ. In fact, we have the following rule: For all s_i solved in Algorithm 5 that satisfy Equation (3), $\widehat{\lambda}_i$ will be increasing monotonically with ϕ, this indicates that $\sum_{i=1}^{n} \widehat{\lambda}_i$ will increase monotonically with ϕ. In terms of the rule, our solution to the problem of optimal power allocation and load distribution can be described as follows:

- Step 1: Given a ϕ, using Algorithm 5 to find the τ that equals to all of $g_i(s_i, \widehat{\lambda}_i)$ $(1 \leq i \leq n)$, as well as can make Equation (3) be satisfied.
- Step 2: Based on Step 1, adjust ϕ until the condition Equation (2) is satisfied.

Through the above Steps (1) and (2) we can find the optimal solution to our problem. However, Fig. 2 suggests that if the value of ϕ becomes smaller, the value of $g_i(s_i, \widehat{\lambda}_i)$ will become larger, this means that a small ϕ will be matched with a large τ. By observing Equation (18), we know that $f_i(s_i)$ may be less than zero when ϕ is excessively small and τ is too large. Under this situation, there might not exist such a common τ that makes all s_i satisfy Equation (3), this means that using Steps (1) and (2) cannot solve the current problem. According to a common τ that exists, we can find a threshold of ϕ called as ϕ_B, when $\phi \geq \phi_B$ there will exist a common τ that can make all s_i satisfy Equation (3) $(1 \leq i \leq n)$, while $\phi < \phi_B$ there will not exist a common τ. Since a ϕ is matched with a $\widehat{\lambda}$, there exist a $\widehat{\lambda}_B$ corresponding to the ϕ_B. When $\widehat{\lambda} \geq \widehat{\lambda}_B$, we can find the optimal solution to our problem basing on Lagrange system; when $\widehat{\lambda} < \widehat{\lambda}_B$, Lagrange system cannot be used to solve the problem because Lagrange multipliers cannot be found. We call the searching process for ϕ_B as $calbdof\phi$, due to limited space, it is moved to the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2017.2693186. Algorithm 6 can be employed to solve our problem under the situation that $\widehat{\lambda} > \widehat{\lambda}_B$. How to deal with the situation that $\widehat{\lambda} < \widehat{\lambda}_B$ is described in next section.

Algorithm 6. Caculate_P1

Input: $\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_n, \widetilde{r}_1, \ldots, \widetilde{r}_n, \widehat{r}, \widehat{T}$.
Output: $\widehat{\lambda}_1, \ldots, \widehat{\lambda}_n, s_1, \ldots, s_n, \phi, \tau$.
1: $\phi_B \leftarrow calbdof\phi$;
2: $lb \leftarrow \phi_B$;
3: **repeat**
4: $\quad \phi \leftarrow 2\phi$;
5: $\quad s_1, s_2, \ldots, s_n \leftarrow calculateAll\ s_i()$;
6: **until** $\widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n > \widehat{\lambda}$
7: $ub \leftarrow \phi$;
8: **while** $(ub - lb > \varepsilon)$ **do**
9: $\quad \phi \leftarrow (lb + ub)/2$;
10: $\quad s_1, s_2, \ldots, s_n \leftarrow calculateAlls_i()$;
11: $\quad$ **if** $(\widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n < \widehat{\lambda})$ **then**
12: $\quad\quad lb \leftarrow \phi$;
13: $\quad$ **else**
14: $\quad\quad ub \leftarrow \phi$;
15: $\quad$ **end if**
16: **end while**
17: **return** $\widehat{\lambda}_1, \ldots, \widehat{\lambda}_n, s_1, \ldots, s_n, \phi, \tau$.

5.3 Data Fitting

Notice that the key factor for solving our problem is how to determine the speed s_i for each node. In other words, load balancing depends on power allocation. Therefore, the first work to solve our problem should be how to determine the core speed for each node. We cannot adopt Lagrange system to obtain the optimal core speed of each node when $\widehat{\lambda} < \widehat{\lambda}_B$, while it is easy for us to get a lot of optimal allocation data when $\widehat{\lambda} \geq \widehat{\lambda}_B$. Since the background of our problem is clear, we insist that there exists a mapping relationship between arrive rate of general tasks $\widehat{\lambda}$ and each core speed s_i. Since a lot of optimal allocation data can be obtained by using Lagrange system when $\widehat{\lambda} \geq \widehat{\lambda}_B$, these data could be employed as training data to fit the relationship between arrive rate of general tasks $\widehat{\lambda}$ and each core speed s_i. The details are described as follows:

TABLE 2
Numerical Data in Section 6.1 When System Priority Strategy Is Dedicated Jobs without Priority

i	$\widehat{\lambda}_i$	s_i	ρ_i	P_i	T_i
1	2.2312497	1.8916260	0.6710496	3.8558076	0.4821204
2	2.2363917	1.8912320	0.6720050	3.8824050	0.4836255
3	2.2482624	1.8903044	0.6742187	3.9441767	0.4871508
4	2.2685086	1.8886612	0.6780213	4.0507656	0.4933329
5	2.2978527	1.8861329	0.6835975	4.2082049	0.5027002
6	2.3360110	1.8825563	0.6909771	4.4187123	0.5156827
7	2.3817199	1.8777717	0.7000403	4.6807678	0.5326177

TABLE 3
Numerical Data in Section 6.2 When System Priority Strategy Is Dedicated Jobs without Priority

i	$\widehat{\lambda}_i$	s_i	ρ_i	P_i	T_i
1	2.9169995	2.0225077	0.7293420	4.6642845	0.5480374
2	2.7258834	1.9465756	0.7283380	4.3662016	0.5673108
3	2.5540980	1.8778833	0.7275368	4.0986037	0.5863338
4	2.3990575	1.8154982	0.7269173	3.8573070	0.6051059
5	2.2586006	1.7586370	0.7264604	3.6388327	0.6236268
6	2.1309091	1.7066375	0.7261487	3.4402690	0.6418969
7	2.0144441	1.6589363	0.7259671	3.2591625	0.6599162

Assume that through Lagrange system, we have a group of optimal speed allocation data points $(\widehat{\lambda}, s_1, s_2, \ldots, s_n)_1$, $(\widehat{\lambda}, s_1, s_2, \ldots, s_n)_2$, ... ,$(\widehat{\lambda}, s_1, s_2, \ldots, s_n)_N$. N is the number of the data points, $\widehat{\lambda}$ is different for each data point, and $\widehat{\lambda} > \widehat{\lambda}_B$ for all the N data points. We want to estimate each core speed s_i $(1 \le i \le n)$ under the situation that arrive rate of general tasks is $\widehat{\lambda}$ and $\widehat{\lambda} < \widehat{\lambda}_B$. We shall fit the data using a polynomial function of the form

$$s_i = w_{i0} + w_{i1}\widehat{\lambda} + \cdots + w_{iM}\widehat{\lambda}^M = \sum_{k=0}^{M} w_k \widehat{\lambda}^k,$$

where M is the order of the polynomial, and $\widehat{\lambda}^k$ denotes $\widehat{\lambda}$ raised to the power of k. The polynomial coefficients w_{i0}, w_{i1}, w_{iM} require to be solved. We adopt root-mean-square to fit the data, which can immediately get the values of w_{i0}, w_{i1}, w_{iM}, since each of them has a closed form solution. Once we get the polynomial coefficients $w_{i0}, w_{i1}, \ldots, w_{iM}$, then we obtain the equivalent speed s_i of node v_i, the remaining works is to find the appropriate $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$, ϕ and τ subject to $\widehat{\lambda}_1 + \widehat{\lambda}_2 + \cdots + \widehat{\lambda}_n = \widehat{\lambda}$, and $\frac{1}{\widehat{\lambda}}(\widehat{\lambda}_1\widehat{T}_1 + \widehat{\lambda}_2\widehat{T}_2 + \cdots + \widehat{\lambda}_n\widehat{T}_n) \le \widehat{T}$. Since speeds for all nodes have been obtained, the appropriate $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$, ϕ and τ can be obtained by removing the steps for searching speeds in Algorithm 6. Due to space limitation, the algorithm regrading how to solve the problem under the situation that $\widehat{\lambda} < \widehat{\lambda}_B$ is moved into the supplementary material, available online.

For a system under analysis, we first calculate the threshold $\widehat{\lambda}_B$ and ϕ_B, and then determine which method could be adopted to solve our problem. It is worth noting that although the result is solved by searching method, all methods we adopted are binary search methods. Moreover, the search for s_i and $\widehat{\lambda}_i$ $(1 \le i \le n)$ of each node is independent except for the shared Lagrange multipliers ϕ and τ. This implies that our method can exploit distribution and parallelism to solve the problem when the system scalability factor (n) is large.

6 Numerical Examples

In this section, we demonstrate a number of numerical examples. All parameters in our examples are for illustration purposes only, and could be changed to any other real values. In heterogeneous distributed parallel computing environments, each parameter of a node can have an impact on power allocation and load distribution. We show these impacts, and sum up the objective laws observed from our experimental data in the latter part of each section.

6.1 The Impact of Idle Speed s_{Ii}

In this section, we consider the impact of idle speed s_{Ii} on power allocation and load distribution. We consider a group of $n = 7$ embedded nodes. We assume that $\widetilde{\lambda} = 2.0$ per second, $\widetilde{r}_i = 0.3$ (giga instructions), $\alpha_i = 2.7$, $P_i^* = 0.1$ Watts, for all $1 \le i \le n$. Further $s_{I1} = 0.2$, $s_{I2} = 0.4$, $s_{I3} = 0.6$, $s_{I4} = 0.8$, $s_{I5} = 1.0$, $s_{I6} = 1.2$, $s_{I7} = 1.4$ IPS, $\widehat{\lambda} = 16$ per second, $\widehat{r} = 0.3$ (giga instructions), and $\widehat{T} = 0.5$ seconds. We show the optimal load distribution $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_7$, the optimal node speeds $s_1, s_2, \ldots, s_7$, the node utilizations $\rho_1, \rho_2, \ldots, \rho_7$, the node power consumption $P_1, P_2, \ldots, P_n$ and the average general task response time $T_1, T_2, \ldots, T_7$. Results shown in Table 2 are for all nodes in the system employing the *Discipline 1*, "dedicated tasks without priority", and the system power consumption is 29.04084 Watts. The similar results can be obtained when system queueing disciplines are set to *Disciplines* 2 and 3. Due to space limitation, they are moved to the supplementary material, available online.

From this section, we can observe that the system will assign more tasks to a node with higher *core idle-power*, which has a physical meaning, since the node consumes more power when it is idle, trying to reduce its idle time can decrease power loss.

6.2 The Impact of Power Consumption Exponent α_i

In this section, we consider the impact of α_i on power allocation and load distribution. We also consider a group of $n = 7$ embedded nodes. We assume that $\widetilde{\lambda} = 2.0$ per second, $\widetilde{r}_i = 0.3$ (giga instructions), $s_{Ii} = 0.3$ IPS, $P_i^* = 0.1$ Watts for all $1 \le i \le n$; $\widehat{\lambda} = 17$ per second. Further $\widehat{r} = 0.3$ (giga instructions), $\widehat{T} = 0.6$ seconds, $\alpha_1 = 2.6$, $\alpha_2 = 2.65$, $\alpha_3 = 2.7$, $\alpha_4 = 2.75$, $\alpha_5 = 2.8$, $\alpha_6 = 2.85$, and $\alpha_7 = 2.9$. Results shown in Table 3 are for all nodes in the system employing the *Discipline 1*, and the system power consumption is $P = 27.324661$ Watts. The results for all nodes employing *Disciplines 2 and 3* are moved to supplementary material, available online.

From this section, we can observe that the system will assign more tasks to a node with a smaller value of α_i, which has a physical meaning, i.e., if a node is capable of performing at the same capacity of work as other nodes, but consumes less power, then assigning more tasks to the node is reasonable.

6.3 The Effect of Data Fitting

In this section, we consider the case that using Lagrange system cannot obtain the optimal power allocation and load

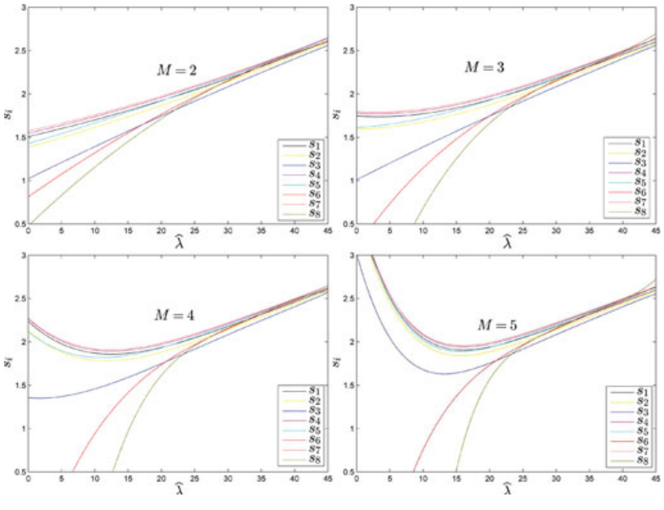

Fig. 5. Fitting results.

distribution strategy. We fit the relationships between the arrival rate of general tasks and core speeds for each node. The average task size and acceptable response time of generic tasks are $\widehat{r} = 0.25$ (giga instructions) and $\widehat{T} = 0.5$ seconds, respectively. The other parameters are $\widetilde{\lambda}_i = 2.0$, $\alpha_i = 2.6$, $s_{Ii} = 0.5$ for all $1 \le i \le 8$, $(\widetilde{r}_1, \widetilde{r}_2, \widetilde{r}_3, \widetilde{r}_4, \widetilde{r}_5, \widetilde{r}_6, \widetilde{r}_7, \widetilde{r}_8)$ is (0.2, 0.3, 0.4, 0.2, 0.3, 0.4, 0.2, 0.4), nodes 1, 2, 3 employe *Discipline 1*, nodes 4, 5, 6 employe *Discipline 2*, and nodes 7, 8 employe *Discipline 3*. For each node v_i, we select 170 data points ($\widehat{\lambda}$ and corresponding s_i) from region $23 \le \widehat{\lambda} \le 40$ as training data set, which have been solved with Lagrange system. Fig. 5 shows the fitting results when the order of polynomial M is $M = 2$, $M = 3$, $M = 4$ and $M = 5$. Once the fitting function is obtained, we could obtain the speed s_i immediately for each node corresponding to a given $\widehat{\lambda}$. Therefore, based on s_i, it is easy for us to obtain the task allocation $\widehat{\lambda}_i$ assigned to each node, as well as the power consumption for the overall system. Table 4 shows the power consumptions corresponding to different values of M and $\widehat{\lambda}$.

The solution reached by this fitting method might not be an optimal solution. As we all know, a genetic algorithm (GA) can solve non-linear problems, and achieve a global approximate optimal solution. In order to check the quality of solutions, we compare our results with the solutions produced by the genetic algorithm from the genetic algorithm toolbox GAOT in MATLAB. We use the same group size and input parameters as Section 6.6, and set the order of the polynomial as $M = 4$. The results of this comparison are shown in Fig. 6. Meanwhile, we compare the solutions provided by fitting method with optimal solutions. The results appear in Fig. 7.

From Fig. 6, we observe that our solutions, most of the time, are better than the solutions provided by the GA algorithm. This result implies that the quality of solutions obtained by our method is good. From Fig. 7, we observe that the difference between fitting solutions and optimal solutions are within 0.01 W. This finding implies that the fitting solutions could replace the optimal solutions to certain degree. The benefit of the fitting method is that it could reduce the search process for finding each core speed.

TABLE 4
Numerical Data in Section 6.3

M	$\widehat{\lambda} = 18$	$\widehat{\lambda} = 19$	$\widehat{\lambda} = 20$	$\widehat{\lambda} = 21$	$\widehat{\lambda} = 22$
2	25.1558 W	26.4602 W	27.8053 W	29.1919 W	30.6206 W
3	25.1039 W	26.4173 W	27.7718 W	29.1678 W	30.6055 W
4	25.0748 W	26.3923 W	27.7526 W	29.1553 W	30.5994 W
5	25.0687 W	26.3827 W	27.7427 W	27.7427 W	30.5968 W

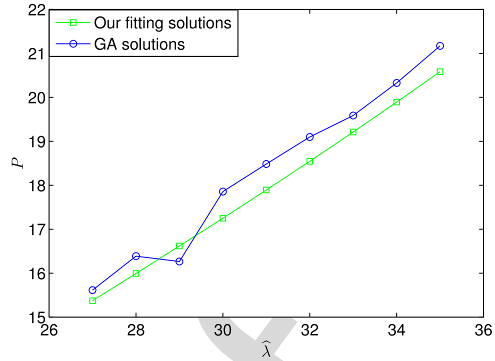

Fig. 6. Fitting versus GA.

These results demonstrate that a relationship exists between the total task arrival rate and the core speed s_i; the bigger the order of the polynomial M, the closer it is to the optimal power allocation. Moreover, this work provides an important insight. When the difference between the preloaded tasks and α of each node is large, the workload of the system result in an imbalance between each node prior to assigning general tasks to each node. It is difficult to accomplish load balancing when the task arrive rate $\widehat{\lambda}$ is small. Thus, under these circumstances, it is possible that using a Lagrange system cannot solve the problem of optimal power allocation and load balancing on the system.

6.4 The Impact of Preloaded Tasks

Due to space limitation, these derivations are moved to the supplementary material, available online.

6.5 The Impact of Queueing Discipline

Due to space limitation, these derivations are moved to the supplementary material, available online.

6.6 The Situation of a Fully Heterogeneous System

Due to space limitation, this section is moved to the supplementary material, available online.

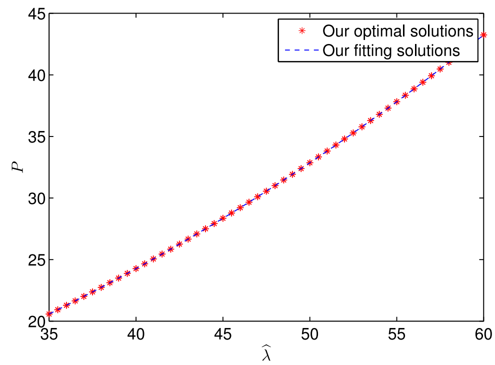

Fig. 7. Fitting versus optimal.

TABLE 5
Numerical Data in Section 7

i	$\widehat{\lambda}_i$	s_i	ρ_i	P_i	T_i
1	1.952	0.664	0.678	1.548	0.893
2	1.952	0.664	0.678	1.548	0.893
3	1.308	0.643	0.686	1.532	1.132
4	1.308	0.643	0.686	1.532	1.132
5	0.638	0.587	0.677	1.487	1.664
6	0.638	0.587	0.677	1.487	1.664

6.7 The Impact of System Scalability

Due to space limitation, this section is moved to the supplementary material, available online.

6.8 Performance Comparison

Due to space limitation, this section is moved to the supplementary material, available online.

7 Experiment Evaluation

The results shown in Section 6 are theoretical results. In this section, we provide our findings concerning the differences between the theoretical results and the results obtained from experimental evaluation.

In the evaluation experiments we consider the system consisting of six nodes, the average arrive rate and task size of general tasks are $\widehat{\lambda} = 7.8$ and $\widehat{r} = 0.2$ respectively, the average arrive rates and task size of dedicated tasks on each node are $\widetilde{\lambda}_i = 0.6, 1 \leq i \leq 6$, and $\widetilde{r}_1 = \widetilde{r}_2 = 0.1$, $\widetilde{r}_3 = \widetilde{r}_4 = 0.3$, $\widetilde{r}_5 = \widetilde{r}_6 = 0.45$, respectively, and the basic power and power consumption exponent is $P_i^* = 1.35W$ and $\alpha_i = 3.0$ $(1 \leq i \leq 6)$ respectively. The idle speed is $s_{Ii} = 0, 1 \leq i \leq 6$. For this investigation, all nodes employed *Discipline 1*. Based on these parameters, we obtained the optimal power and allocation of tasks shown in Table 5. Based on Table 5, the experimental evaluation is divided into two parts as follows.

7.1 Simulation Evaluation

The following discussion reviews the differences between the theoretical values and simulation values obtained from the execution of an established number of tasks.

The result listed in Table 5 is a theoretical value. To investigate the difference between the theoretical value and actual simulation value, we generated a number of general and dedicated tasks. The arrival interval times and task sizes for general tasks are exponential random variable $1/\widehat{\lambda}_i$ and $\widehat{r}$ respectively; for dedicated tasks the arrival interval times and task sizes are $1/\widetilde{\lambda}_i$ and $\widetilde{r}_i$ respectively.

TABLE 6
Simulation Results

i	TT	$RTGT$	GN	$watters$	P_i	T_i
1	19,434.67	34,822.44	38,228	31,378.00	1.554	0.912
2	19,370.9	33,757.77	38,298	31,276.79	1.551	0.884
3	36,713.94	53,570.74	48,159	58,555.09	1.539	1.114
4	36,545.64	55,196.02	48,212	58,320.22	1.535	1.143
5	65,144.49	68,610.00	40,653	100,650.40	1.480	1.685
6	65,573.72	68,663.13	40,930	101,284.43	1.484	1.674

TABLE 7
Platform Parameters

CPU	OS	Memory	DVFS tool
Cortex-A20	Debian 4.7.2-5	1 G	cpufreq
Frequency	1.01 0.960 0.912 0.864 0.816 0.768 0.744 0.720 0.696 0.672 0.648 0.600 0.528 0.480 0.408 0.384 0.360 0.336 GHz		
Max transition latency	2 ms		

Once these times are established, we schedule these tasks. The scheduling results are shown in Table 6, in which "TT" represents the total time, "$RTGT$" represents the response time for all of general tasks, "GN" represents the numbers of general tasks, "$Watters$" represents the total power cost, $P_i = watters/TT$ represents the average power cost per second, and $T_i = RTGT/GN$ represents the average response time of general tasks. By comparing Tables 5 and 6, we find that there is a good agreement between the theoretical and simulation results regarding the average response time of general tasks T_i and power cost per seconds P_i.

7.2 Practical Evaluation

Based on Table 5 and the tasks generated in Section 7.1, we will investigate the difference between theoretical results and practical results on a real platform consisting of the six nodes (embedded boards) corresponding respectively with nodes mentioned in Section 7, and its detailed parameters are listed in Table 7. The testing process is divided into three steps.

Step 1: We need to test the core speed (IPS) and power when core is operating at various frequencies. A program commonly consists of a number of assembly instructions, such as JUMP, MOV, CMP, ADD, and MUL. By recording the actual number of assembly instructions and corresponding execution times, we are able to obtain the IPS. Power is the product of current and voltage. The board voltage is kept at 5 V in our experiment. Notice that the tested power not only includes the processor's power, but also the power of other components. While the power of the processor is dynamic, the power of the other components is relatively stable. Thus, we are able to treat the core's static power and all of the other component's power as the basic power P_i^*, which could be obtained by setting the core's frequency to 0 GHz. When the core is idle, its *core idle-power* equals the node's (embedded board) power minus P_i^*. In this experiment, the frequency of the Cortex-A20 dual core CPU is set at 336 MHz when the status of its core is idle. In real environments, the power consumption exponent α_i is usually defined as 3.0. Thus, the idle speed can be calculated based on the *core idle-power* and α_i. The data obtained from tests are shown in Table 8, where PB represents the power of node when there are tasks running, and PI represents the power of node that there is no task running, and s_i is derived by $s_{Ii} = \sqrt[\alpha]{\frac{PI - P_i^*}{2}}$.

Step 2: Since that the optimal speed shown in Table 5 is computed theoretically, in the actual test platform, the real core frequency needs to be adjusted to map the corresponding core speed into theory value. Based on Tables 5 and 8, we adjust the core frequency in terms of the smallest gap between

TABLE 8
IPS and Power

freqency(GHz)	1.01	0.960	0.912	0.864	0.816	0.336	0
s_i (Giga IPS)	0.67	0.635	0.60	0.57	0.54	0.22	-
PB(W)	1.9	1.85	1.8	1.75	1.7	1.40	-
PI(W)	1.5	1.5	1.475	1.45	1.45	1.35	1.35
s_{Ii}	0.42	0.42	0.4	0.37	0.37	0	0

the optimal and practical speed. For examples, the closest frequency that we could get speed 0.667, 0.644, and 0.58 Giga IPS is 1.01, 0.96, and 0.864 GHz, respectively. Therefore, each node's frequency is adjusted to corresponding level.

Step 3: In the course of experiment, by recording the arrival, start and completion time of each task, we can calculate the total time (TT), execution time (ET) and response time of all tasks. Note that Cortex-A20 includes two cores. The power consumption per seconde for a node with one core can be calculated as

$$P_i = \frac{\left(\frac{P_B - P_i^*}{2} + P_i^*\right) \times ET + \left(\frac{P_I - P_i^*}{2} + P_i^*\right) \times (TT - ET)}{TT}.$$

The P_B and P_I are obtained from Table 8.

The TT, ET, $RTGT$ (response time of all general tasks), ρ_i, P_i, and T_i are shown in Table 9. From the Tables 5 and 9, we can find out that the errors of response time between optimal and practical result are less than 0.06 seconds (3.6 percent), and the errors of power are less than 0.04W (2.5 percent). We analyse that the errors between theoretical value and practical value are due to the following reasons. (1) The speeds are made a slight adjustment. (2) Some power may be ignored. (3) Speed or power test process may be uncertain. (4) Core environment exists noise etc. In summary, the experiments show that the present theoretical results are basically in line with the practical results.

8 Conclusion

In this paper, we have studied the joint optimization problem of load balancing and power allocation in heterogeneous distributed embedded systems. From the perspective of hardware, we specify that all nodes in the system are heterogeneous, with each node having a different maximum speed and power consumption. We also specify that the priority of each task is different on each node, and the speed of each core is different from the perspective of the application.

We propose an efficient algorithm to solve the joint optimization problem using a Lagrange method. When the problem could not be solved using the Lagrange method, we design an algorithm to determine the appropriate speed of each core by using a fitting data method to fit the relationship between task arrival rate and core speed. This approach solves the problem. Extensive numerical examples are given to demonstrate the impact of each factor on the system. Furthermore, we employe both simulation and practical evaluation to show that present theoretical results are consistent with the practical results. This research makes an original contribution to optimal load balancing and power allocation with performance constraint for multiple embedded computing nodes in heterogeneous and distributed embedded systems.

TABLE 9
Practical Results

i	TT	ET	$RTGT$	GN	ρ_i	P_i	T_i
1	19,434.7	13,187.9	34,257.2	38,228	0.678	1.536	0.892
2	19,370.7	13,181.3	33,757.8	38,298	0.680	1.537	0.873
3	38,006.9	26,296.4	57,747.2	49,890	0.691	1.505	1.154
4	37,874.5	26,353.6	59,572.7	49,903	0.695	1.506	1.193
5	65,329.0	44,860.5	70,754.8	40,725	0.686	1.487	1.734
6	65,573.8	44,870.4	71,793.8	40930	0.684	1.486	1.753

Acknowledgments

The work reported in this paper was supported by National Key Research and Development Plan of China Grant No. 2016YFB0200405, the Natural Science Foundation of China (Grant No. 61370097, 61370095, 61502162). The corresponding author is Renfa Li.

References

[1] J. Park, D. Shin, N. Chang, and M. Pedram, "Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors," in *Proc. ACM/IEEE Int. Symp. Low-Power Electron. Des.*, Aug. 2010, pp. 419–424.
[2] S. H. Kim, et al., "C-lock: Energy efficient synchronization for embedded multicore systems," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 1962–1974, Aug. 2014.
[3] E. Viegas, A. Santin, A. Fanca, R. Jasinski, V. A. Pedroni, and L. S. Oliveira, "Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 163–177, Jan. 2017, doi: 10.1109/TC.2016.2560839.
[4] A. Munir, S. Ranka, and A. Gordon-Ross, "High-performance energy-efficient multicore embedded computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 4, pp. 684–700, Apr. 2012.
[5] W. Dally, et al., "Efficient embedded computing," *IEEE Comput.*, vol. 41, no. 7, pp. 27–32, Jul. 2008.
[6] J. Balfour, "Efficient embedded computing," Ph.D. dissertation, EE Dept., Stanford Univ., Stanford, CA, USA, May 2010.
[7] G. Kornaros, *Multi-core embedded systems*. New York, NY, USA: Taylor and Francis Group, CRC Press, 2010.
[8] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. IEEE 33rd Int. Symp. Comput. Archit.*, Jun. 2006, pp. 78–88.
[9] R. Jayaseelan and T. Mitra, "A hybrid local-global approach for multi-core thermal management," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2009, pp. 314–320.
[10] 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.–Jun. 2016.
[11] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 3, pp. 346–360, Mar. 2009.
[12] B. Yang, Z. Li, S. Chen, T. Wang, and K. Li, "Stackelberg game approach for energy-aware resource allocation in data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3646–3658, Dec. 2016, doi: 10.1109/TPDS.2016.2537809.
[13] B. A. Shirazi, A. R. Hurson, and K. M. Kavi, Eds., *Scheduling and Load Balancing in Parallel and Distributed Systems*. Los Alamitos, CA, USA: IEEE Comput. Soc. Press, 1995.
[14] K. Li,"Optimal load distribution for multiple heterogeneous blade servers in a cloud computing environment," *J. Grid Comput.*, vol. 11, no. 1, pp. 948–957, 2012.
[15] J. Balasangameshwara and N. Raju, "A hybrid policy for fault tolerant load balancing in grid computing environments," *J. Netw. Comput. Appl.*, vol. 35, no. 1, pp. 412–422, 2012.
[16] 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.

[17] K. W. Ross and D. D. Yao, "Optimal load balancing and scheduling in a distributed computer system," *J. ACM*, vol. 38, no. 3, pp. 676–690, 1991.

[18] F. Bonomi and A. Kumar, "Adaptive optimal load balancing in a nonhomogeneous multiserver system with a central job scheduler," *IEEE Trans. Comput.*, vol. 39, no. 10, pp. 1232–1250, Oct. 1990.

[19] K. Li, "Optimal load distribution in nondedicated heterogeneous cluster and grid computing environments," *J. Syst. Archit.*, vol. 54, no. 1/2, pp. 111–123, 2008.

[20] L. Barroso and U. Holzle, "The case for energy-proportional computing," *IEEE Comput.*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[21] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proc. 41st Annu. Des. Autom. Conf.*, 2004, pp. 868–873.

[22] Q. Qiu and M. Pedram, "Dynamic power management continuous-time Markov decision processes," in *Proc. 36th Des. Autom. Conf.*, 1999, pp. 555–561.

[23] M. Weiser, B. Welch, A. J. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proc. Int. Symp. Operating Syst. Des. Implementation*, 1994, pp. 13–23.

[24] A. O. Allen, *Probability, Statistics, and Queueing Theory with Computer Science Applications*, 2nd ed. Boston, MA, USA: Academic, 1990.

[25] M. J. Walker, et al., "Accurate and stable run-time power modeling for mobile and embedded CPUs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 1, pp. 106–119, Jan. 2017.

[26] A. Aalsaud, R. Shafik, A. Rafiev, F. Xia, S. Yang, and A. Yakovlev, "Power-aware performance adaptation of concurrent applications in heterogeneous many-core systems," in *Proc. ACM Int. Symp. Low Power Electron. Des.*, 2016, pp. 368–373.

[27] S. Yang, et al., "Adaptive energy minimization of embedded heterogeneous systems using regression-based learning," in *Proc. 25th Int. Workshop Power Timing Modeling Optimization Simul.*, 2015, pp. 103–110.

[28] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," in *Proc. ACM Symp. Operating Syst. Principles*, 2003, pp. 149–163.

[29] M. Goraczko, J. Liu, D. Lymberopoulos, S. Matic, B. Priyantha, and F. Zhao, "Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems," in *Proc. Design Autom. Conf.*, 2008, pp. 191–196.

[30] G. Xie, G. Zeng, L. Liu, R. Li, and K. Li, "High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems," *J. Syst. Archit.*, vol. 70, pp. 3–14, 2016.

Jing Huang received the MSc degree in computer science and technology from Hunan University, Changsha, China, in 2013. He is currently working toward the PhD degree at Hunan University. His research interests include parallel computing, high-performance computing, distributed computing, energy-efficient computing, heterogeneous computing, cloud computing, and machine learning.

Renfa Li is a full professor of computer science and electronic engineering, and the dean of College of Computer Science and Electronic Engineering, Hunan University, China. He is the director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. He is also an expert committee member of National Supercomputing Center in Changsha, China. His major research includes embedded systems, distributed systems, and cyber-physical systems. He is a senior member of the IEEE, and the ACM.

Jiyao An received the MSc degree in mathematics from Xiangtan University, China, in 1998 and the PhD degree in mechanical engineering from Hunan University, China, 2012. He was a visiting scholar in the Department of Applied Mathematics, University of Waterloo, Ontario, Canada, from 2013 to 2014. Since 2000, he joined the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, where he is currently a professor. His research interests include cyber-physical systems (CPS), Takagi-Sugeno fuzzy systems, parallel and distributed computing, and computing intelligence. He has publish more than 50 papers in international and domestic journals and refereed conference papers. He is an active reviewer of international journals. He is a member of the IEEE and the ACM, and a senior member of the CCF.

Derrick Ntalasha received the BSc degree from Copperbelt University (CBU), Zambia, in 2002 and the MSc degree in information systems engineering from the University of Manchester, United Kingdom, in 2005. He is currently working toward the PhD degree in the College of Information Science and Engineering, Hunan University, China. His research interests include context awareness computing, middleware, parallel and distributed systems, and Internet of Things (IoT)

Fan Yang received the MSc degree in computer science and technology from Hunan University, Changsha, China, in 2011. He was a visit scholar with Michigan State University, from 2014-2015. He is currently working toward the PhD degree with Hunan University. His research interests include cyber physical systems, embedded systems, and computer architecture. He is a member of China Computer Federation.

Keqin Li is a SUNY distinguished professor of computer science. His current research interests include parallel computing and highperformance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things, and cyber-physical systems. He has published more than 470 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Services Computing*, the *IEEE Transactions on Sustainable Computing*. He is a fellow of the IEEE.