IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 13, 2020, accepted March 19, 2020, date of publication March 24, 2020, date of current version April 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982956

CPU–GPU Utilization Aware Energy-Efficient Scheduling Algorithm on Heterogeneous Computing Systems

XIAOYONG TANG AND ZHUOJUN FU

College of Information and Intelligence, Hunan Agricultural University, Changsha 410128, China

Corresponding author: Zhuojun Fu (fzj@hunau.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFD0301506, in part by the National Natural Science Foundation of China under Grant 61972146 and Grant 61672219, in part by the Hunan Provincial Key Research and Development Program under Grant 2018GK2055, and in part by the Double First-Class Construction Project of Hunan Agricultural University under Grant SYL201802029.

ABSTRACT Nowadays, heterogeneous computing systems have proven to be a good solution for processing computation intensive high-performance applications. The main challenges for such large-scale systems are energy consumption, computing node CPU-GPU utilization dynamic variability, and so on. In response to these challenges, this study first provides heterogeneous computing systems architecture and parallel application job model. Then, we build system computing node CPU-GPU utilization model and analyze job execution energy consumption. We also deduce the optimal CPU-GPU utilization and job deadline scheduling constraint. Third, we propose a systems CPU-GPU utilization aware and energy-efficient heuristic greedy strategy (UEJS) to solve this job scheduling problem. To improve the algorithm global optimization ability, we design a hybrid particle swarm optimization algorithm (H-PSO), which incorporates the heuristic greedy strategy into the bio-inspired search optimization technique. The rigorous experimental results clearly demonstrate that our proposed H-PSO outperforms heuristic greedy strategy, Max-EAMin, and Genetic Algorithm in terms of the average energy consumption of jobs and system job rejection rate. In particular, H-PSO is better than UEJS by 36.5%, Max-EAMin by 36.3%, and GA by 46.7% in term of the job average energy consumption for heterogeneous system with high workload.

INDEX TERMS Heterogeneous computing systems, CPU-GPU utilization, energy consumption, job scheduling, particle swarm optimization.

I. INTRODUCTION

In the past few years, many computation intensive high-performance applications, such as image processing, natural language processing, have been deployed on large-scale heterogeneous computing systems based on CPU-GPU [1]. This is mainly due to the low cost, good expansibility characteristics of that systems [1], [2]. One of the main challenges for such infrastructure is energy consumption, which generates high operational costs and produces a dramatic increase in carbon. For example, the first ranked on the top-500 list Supercomputer Summit has an operating power of 10.096 MW [3], which its energy consumption is about 2.8 times of the total electricity consumption of universities in Guangzhou University Town. This Supercomputer is a classical CPU-GPU heterogeneous system, which consists of 4608 computing nodes [3].

Due to the heterogeneity of system and the complexity of parallel applications, the resource (or CPU-GPU) utilization of heterogeneous computing nodes usually changes dynamically [4], [5]. The dynamic variability of CPU-GPU utilization also results in the changing of their power consumption. In general, the power consumption increases nonlinearly with the increase of computing nodes CPU-GPU utilization [5]–[7]. Therefore, for large-scale heterogeneous computing systems, the energy consumption of computing nodes with different CPU-GPU utilizations is different. Moreover, the total energy consumption of heterogeneous computing systems will also greatly fluctuate depending on the jobs and computing nodes dispatching schemes. This problem can be summed up as a combinatorial optimization problem with the goals of satisfying the application

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

performance requirements and minimizing system energy consumption, which is widely known as a NP-complete in the general case [8].

Recently, techniques such as dynamic power management, dynamic voltage-frequency scaling (DVS), slack reclamation, resource hibernation, and so on, have been successfully applied in reducing the energy consumption of single computing resource [9]–[11]. These approaches are incorporated into job scheduling strategy to solve heterogeneous computing systems energy consumption optimization problem [12], [13]. However, most of them are not effectively deal with the relationship between resource (CPU-GPU) utilization dynamic variability and energy consumption.

In recognition of this, motivated by challenges of heterogeneous computing systems energy consumption, computing node CPU-GPU utilization, and job optimization scheduling, we propose two CPU-GPU utilization aware and energy-efficient job scheduling algorithms. Our contributions are multifold and can be summarized as follows:

- First, we build heterogeneous computing systems architecture, which consists of computing nodes with local queue. And, the parallel application job model based on CPU and GPU execution size is proposed.
- Secondly, we define system computing node CPU-GPU utilization model and analyze the relationship between computing node power consumption and CPU-GPU utilization. We also deduce job execution energy consumption and the optimal CPU-GPU utilization.
- Thirdly, we provide a systems computing node CPU-GPU utilization aware and energy-efficient heuristic greedy job scheduling algorithm (UEJS). To improve the algorithm global optimization ability, we combine the heuristic greedy strategy and bio-inspired search optimization method into a hybrid particle swarm optimization algorithm (H-PSO) to schedule jobs.
- Finally, the performance evaluation shows that our proposed hybrid particle swarm optimization algorithm (H-PSO) outperforms heuristic greedy strategy (UEJS), Max-EAMin, and GA in terms of the job average energy consumption and system job rejection rate.

The rest of this paper is as follows: The related work is summarized in Section 2. Section 3 provides heterogeneous computing systems architecture and parallel applications model. In Section 4, we describe computing node CPU-GPU utilization and job execution energy consumption model. The CPU-GPU utilization aware energy-efficient job scheduling algorithms are proposed in Section 5. In Section 6, we evaluate our proposed algorithms and analyze the experimental results. Finally, we present the conclusions of this paper and propose future studies in Section 7.

II. RELATED WORK

Many heuristic job scheduling algorithms have shown good performance in classical job scheduling problem, such as MET, Min-Min, Max-Min, and XSufferage [14]. The widespread used heuristic Min-Min algorithm first searches task's minimum completion time on each processor as task-processor set U. Then, the task-processor pair with overall minimum completion time from U is selected. The heuristic Max-Min is similar to Min-Min, and the difference is the second phase that Max-Min selects the task-processor pair with overall maximum completion time from U. The Min-Min, Max-Min are consider as the metric of scheduling algorithm for time optimization problem.

In recent years, a large number of scheduling algorithms that combine time and other system qualities of service (QoS), such as reliability, budget, deadline, energy consumption, have been emerged in the field of heterogeneous computing systems [16]–[19], [34], [39], [40]. Dogan and Özgüner [40] analyzed the task execution reliability on heterogeneous computing systems, and combined this into applications dynamic level to implemented the scheduling algorithm. Roy *et al.* studied the real-time precedence-constrained task scheduling problem on heterogeneous systems with multiple service levels (such as performance, reliability, cost) [34]–[36]. Zhang *et al.* first built the heterogeneous systems communication contention and task execution reliability model. Then, they proposed an efficient reliability management solution for task scheduling [16]. Reference [17] proposed an efficient task scheduling algorithm to satisfy the budget constraint and minimize the schedule length. Zong *et al.* proposed two energy aware job scheduling algorithms (EAD and PEBD) based on job duplication strategy for homogeneous clusters [18]. Chen *et al.* established a hyper-heuristic framework and proposed a quantum-inspired high-level learning strategy to solve performance-constrained energy optimization problem [19]

On the other hand, there are many studies that focus on optimizing the energy consumption and the above other objectives [12], [15], [20]. Huang *et al.* focus on co-management of system reliability and energy consumption, and propose an energy-efficient fault-tolerant scheduling algorithm [20]. Tarplee *et al.* adopted linear programming methods to obtain good tradeoff between application makespan and heterogeneous computing systems energy consumption [37]. To guarantee task deadline and reduce energy consumption, Qin *et al.* use integer linear programming to assign individual basic blocks by a proper operational frequencies [12]. These research works demonstrate that the optimal scheduling strategy can effectively reduce system energy consumption.

Many bio-inspired meta-heuristic evolutionary algorithms are also applied in dealing with such multi-objective scheduling problem [21]–[24], [27]. For example, Deng *et al.* proposed an improved Ant Colony Optimization (ACO) algorithm to effectively solve the traveling salesmen problem and the actual gate assignment problem [21]. Xiong *et al.* proposed a Johnson's-rule-based Genetic Algorithm (GA) to solve task scheduling problem of cloud data-centers [22]. In our previous work [23], we proposed a task scheduling strategy based on Genetic Algorithm with budget

constraint for heterogeneous cloud systems. Reference [24] used the strength pareto evolutionary algorithm (SPEA) and non-dominated sorting genetic algorithm (NSGA) to optimize the performance of energy and temperature. In recent years, some improved and hybrid particle swarm optimization (PSO) algorithms have been applied in computing systems job scheduling problem and have shown good performance [25], [26]. Kumar and Sharma first defined the fitness function based on execution time, makespan, cost, energy consumption, task rejection ratio, and throughput. Then, they developed a PSO-COGENT algorithm to schedule jobs in cloud environment while considering deadline as constraint [27]. However, the efficiency of its multi-objective fitness function still needs further study.

A few research works also involved system resource utilization. In [28], they presented a two-layer coordinated CPU utilization control architecture, which the frequency scaling for the CPU utilization of each processor and the rate adaptation can control the utilizations of all the processors. Sandokji *et al.* surveyed the underutilization GPGPU cores and proposed a warp scheduler to improve GPU utilization [38]. Stavrinides and Karatza investigated the impact of resource utilization and the energy consumption for distributed system. Their research results reveal that system resource utilization variability can affect its energy consumption [7]. However, this work assumes a linear relationship between energy consumption and workload, which is inconsistent with the nonlinearity of the actual system. Moreover, the Max-EAMin scheduling algorithm did not consider jobs' execution size between CPU and GPU. In this paper, we focus on the optimizing energy consumption by considering the system computing node CPU-GPU utilization, which implemented by a heuristic greedy job scheduling algorithm and a hybrid particle swarm optimization technique.

III. HETEROGENEOUS SYSTEMS MODELS

This section describes the heterogeneous systems architecture and parallel applications model used in our study. For future reference, we summary the notations introduced in this section in Table 1.

A. SYSTEMS ARCHITECTURE

In this paper, the heterogeneous systems are considered as a finite set of m computing nodes, which are supported by many racks. Generally, these computing nodes based on DVFS enabled CPU-GPU processors are connected by high-speed interconnection networks, such as Infiniband and Myrinet. For example, computing node of Summit consist of two 22-core IBM Power9 CPUs and six NVIDIA Volta GV100 GPUs, and these nodes are linked by a dual-rail Mellanox EDR InfiniBand [3]. For heterogeneous systems, the number and type of computing nodes' CPU, GPU are different. Therefore, each node has its own multi-core CPU and manycore GPU computation capacity (the amount of computation that can be performed in a unit of time, such as TFLOPS). Figure 1 depicts a large-scale heterogeneous computing systems architecture.

TABLE 1. Definitions of the heterogeneous systems notations.

Notation	Definition
Θ	A set of m heterogeneous computing nodes
cn_j	The jth computing node
$cp(cn_j)$	CPU computation capacity
$cm(cn_j)$	Number of CPUs
$gp(cn_j)$	GPU computation capacity
$gm(cn_j)$	Number of GPUs
$cu(cn_j)$	CPU utilization
$gu(cn_j)$	GPU utilization
$nd(cn_j)$	Computing node latest execution finish time
Λ	A set of n jobs
$CE(J_i)$	Job CPU execution size
$GE(J_i)$	Job GPU execution size
$RD(J_i)$	Job deadline
$CP(J_i)$	Job critical path execution size

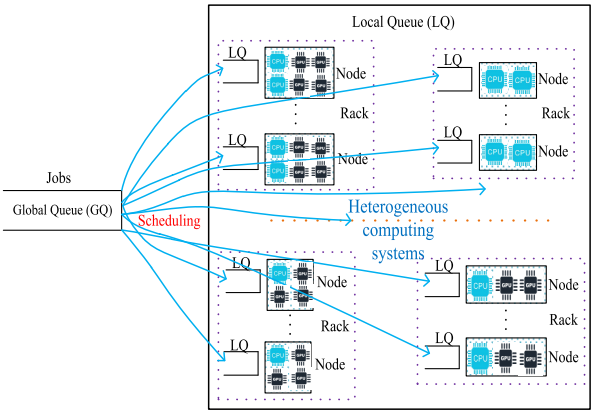

FIGURE 1. Heterogeneous computing systems.

In this study, the computing node set of heterogeneous systems is expressed as $\Theta = \{cn_1, cn_2, \cdots, cn_m\}$. It is assumed that each computing node cn_j consists of the same type of CPU and the same type of GPU. We let $cp(cn_j)$ denotes CPU computation capacity, $cm(cn_j)$ is the number of CPUs, and $gp(cn_j)$ denotes GPU computation capacity, $gm(cn_j)$ is the number of GPUs. The system computing node CPU-GPU utilization is defined as the ratio of the local queue actual computational requirement (such as the total execution size) to the aggregate computation capacity of such computing resources. Here, we let $cu(cn_j)$ denotes computing node cn_j CPU utilization, and $gu(cn_j)$ represents its GPU utilization. The heterogeneous systems also maintain a global parallel

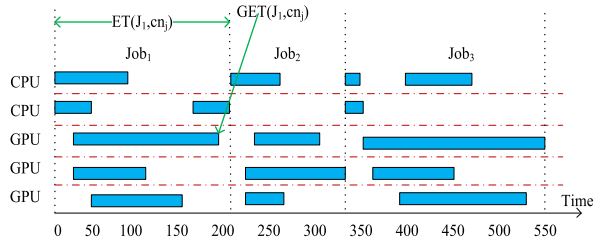

FIGURE 2. An example of computing node job execution.

applications (or jobs) queue (GQ) and computing nodes' local queue (LQ). The symbol $nd(cn_j)$ represents the latest execution finish time for all jobs in the local queue of the computing node cn_j.

B. PARALLEL APPLICATIONS

User parallel applications (or jobs) are first submitted to heterogeneous systems and entered the global queue. Then, these applications are partitioned into n bag-of-tasks (BoT) parallel jobs $\Lambda = J_1, J_2, \cdots, J_n$, which can be dispatched to computing node's local queue according to the scheduling policy. Typically, for heterogeneous systems based on CPU-GPU, the parallel portions of jobs execute on the GPU, and the serial portions will be processed by CPU [29]. Therefore, each job J_i is associated with a three tuple representation $< CE, GE, RD >$, where $CE(J_i)$ denotes job J_i CPU execution size (such as Tera Operations), and $GE(J_i)$ is GPU execution size, which can be estimated by code profiling, historic table, or statistical prediction techniques [30]. We use $RD(J_i)$ to represent job J_i deadline. We also assume that all jobs are ready at the scheduling point, and regardless of their arrival time in this study.

Generally speaking, due to the different of job parallel portions execution size and precedence constraints, the execution time of job parallel portions execution on GPUs are also different [31]. Therefore, the largest parallel portions execution time is considered as the application GPU execution time. In such case, some GPUs will be idle in some time intervals. Similarly, the computing node CPU will also be idle waiting for the result of the parallel portions to continue the serial portions. Figure 2 shows an example of jobs execution on computing node CPU and GPU. From these phenomena and analysis, we can conclude that the system computing node cannot work at full load in most time.

Here, we let $CP(J_i)$ as the execution size of the critical path of parallel job. Therefore, the job J_i GPU execution time $GET(J_i, nc_j)$ on computing node nc_j can be defined as

$$GET(J_i, nc_j) = Max\left\{\frac{CP(J_i)}{gp(cn_j)}, \frac{GE(J_i)}{gp(cn_j) \times gm(cn_j)}\right\}. \quad (1)$$

And, the job J_i execution time $ET(J_i, nc_j)$ on node nc_j is the maximum of its GPU execution time and CPU execution time, which is expressed as

$$ET(J_i, nc_j) = Max\left\{GET(J_i, nc_j), \frac{CE(J_i)}{cp(cn_j) \times cm(cn_j)}\right\}. \quad (2)$$

IV. CPU-GPU UTILIZATION AND ENERGY MODEL

A. CPU-GPU UTILIZATION

We define the heterogeneous systems computing node resource (CPU-GPU) utilization $cu(J_i, cn_j), gu(J_i, cn_j)$ of parallel job as

$$\begin{cases} cu(J_i, cn_j) = \dfrac{CE(J_i)}{cp(cn_j) \times cm(cn_j) \times ET(J_i, nc_j)}, \\ gu(J_i, cn_j) = \dfrac{GE(J_i)}{gp(cn_j) \times gm(cn_j) \times ET(J_i, nc_j)}. \end{cases} \quad (3)$$

The system computing node CPU, GPU utilization and its latest execution finish time $nd(cn_j)$ are depended on the jobs assigned. Here, we assume that this jobs set as Φ_j, and the computing node execution finish time $nd(cn_j)$ is computed by

$$nd(cn_j) = \sum_{J_i \in \Phi_j} ET(J_i, nc_j). \quad (4)$$

Moreover, the computing node CPU and GPU utilization are

$$\begin{cases} cu(cn_j) = \dfrac{\sum_{J_i \in \Phi_j} CE(J_i)}{cp(cn_j) \times cm(cn_j) \times nd(cn_j)}, \\ gu(cn_j) = \dfrac{\sum_{J_i \in \Phi_j} CE(J_i)}{gp(cn_j) \times gm(cn_j) \times nd(cn_j)}. \end{cases} \quad (5)$$

B. SCHEDULING CONSTRAINT

For each job, its execution finish time must before its deadline. Therefore, the following condition must be met for all jobs that are scheduled on corresponding computing nodes.

Condition 1 (Deadline Constraint): For any job $J_i \in \Lambda$ and computing node $cn_j \in \Theta$

$$nd(cn_j) + ET(J_i, nc_j) \leq RD(J_i). \quad (6)$$

As the job and corresponding computing node pair satisfies the above condition, this pair may be a feasible schedule solution, and the computing node latest execution finish time $nd(cn_j)$ may be changed as the following

$$nd(cn_j) = nd(cn_j) + ET(J_i, nc_j). \quad (7)$$

C. ENERGY CONSUMPTION MODEL

For heterogeneous systems computing node, we focus on the energy consumption of CPU and GPU. This is due to the fact that they are typically consume most energy of computing node compare to other components [29]. The computing node power consumption of CPU/GPU based on complementary metal oxide semiconductor (CMOS) logic circuits nonlinearly increases as the resource utilization increases [6]. Figure 3 shows the relationship between resource utilization and power consumption of computing node, which the data are measured from a computing node with 2 Xeon E7-8870 CPUs and 2 Tesla V100 GPUs. From these data, we can conclude that the power consumption and resource utilization of the computing node approximately follow the following rules

$$y = \alpha + \beta \times log_2(1 + x). \quad (8)$$

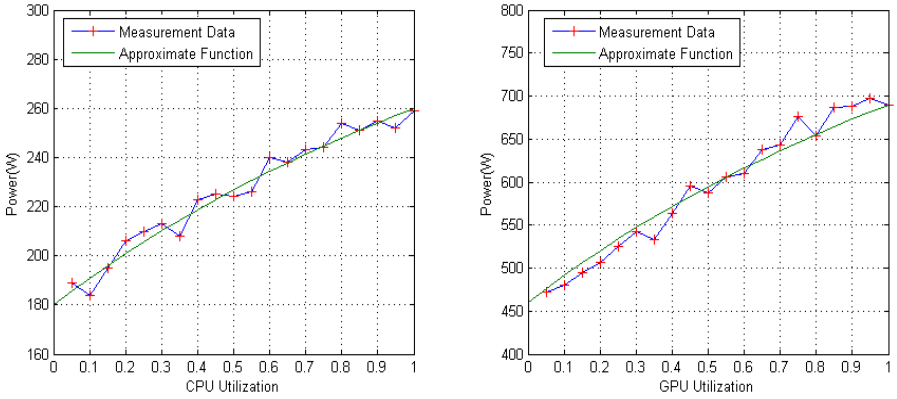

FIGURE 3. The relationship between resource utilization and power consumption of computing node.

where α, β are constant, $x \in [0, 1]$ is CPU-GPU utilization, and y is power consumption.

In this study, the α is expressed as symbol $P_{idle}(nc_j)$, which is the power consumed by computing node cn_j resource when idle. We also assume that symbol $P_{max}(nc_j)$ denotes computing node cn_j power consumption when the resource at 100% utilization. Thus, the constant β is defined as

$$\beta = P_{max}(cn_j) - P_{idle}(cn_j). \tag{9}$$

Consequently, the computing node cn_j power consumption can be defined by

$$\begin{cases} P^c(cn_j) = P^c_{idle}(nc_j) + [P^c_{max}(cn_j) - P^c_{idle}(cn_j)] \\ \qquad \times log_2(1 + cu(cn_j)), \\ P^g(cn_j) = P^g_{idle}(nc_j) + [P^g_{max}(cn_j) - P^g_{idle}(cn_j)] \\ \qquad \times log_2(1 + gu(cn_j)). \end{cases} \tag{10}$$

where $P^c(cn_j)$ and $P^g(cn_j)$ are computing node cn_j CPU and GPU power consumption, respectively. Furthermore, the power consumption of job J_i that is assigned to computing node cn_j is

$$\begin{cases} P^c(J_i, cn_j) = P^c_{idle}(nc_j) + [P^c_{max}(cn_j) - P^c_{idle}(cn_j)] \\ \qquad \times log_2(1 + cu(J_i, cn_j)), \\ P^g(J_i, cn_j) = P^g_{idle}(nc_j) + [P^g_{max}(cn_j) - P^g_{idle}(cn_j)] \\ \qquad \times log_2(1 + gu(J_i, cn_j)). \end{cases} \tag{11}$$

The energy consumption $E(J_i, cn_j)$ of job J_i on computing node cn_j can be defined as an integral of that node's power consumption and job execution time, which is

$$\begin{aligned} E(J_i, cn_j) &= \int_{ST(J_i,cn_j)}^{FT(J_i,cn_j)} P(J_i, cn_j)dt \\ &= P(J_i, cn_j) \times [FT(J_i, cn_j) - ST(J_i, cn_j)] \\ &= P(J_i, cn_j) \times ET(J_i, cn_j). \end{aligned} \tag{12}$$

where, the $ST(J_i, cn_j)$, $FT(J_i, cn_j)$ are the job's start time and finish time on computing node. Therefore, the job's CPU and GPU energy consumption are

$$\begin{cases} E^c(J_i, cn_j) = P^c(J_i, cn_j) \times ET(J_i, cn_j), \\ E^g(J_i, cn_j) = P^g(J_i, cn_j) \times ET(J_i, cn_j). \end{cases} \tag{13}$$

Therefore, the job J_i on computing node cn_j energy consumption can be expressed as

$$E(J_i, cn_j) = E^c(J_i, cn_j) + E^g(J_i, cn_j) + E_{other}. \tag{14}$$

where the E_{other} is the energy consumption of other components of computing node. Then, the computing node cn_j energy consumption is

$$E(cn_j) = \sum_{J_i \in \Phi_j} E(J_i, cn_j). \tag{15}$$

The heterogeneous systems average energy consumption of jobs is defined as

$$EV = \frac{\sum_{j=1}^{m} E(cn_j)}{SJNum}. \tag{16}$$

where $SJNum$ is the number of jobs that are successfully scheduled to heterogeneous systems computing nodes. This paper tries to get the optimal job average energy consumption with **C**ondition 1 constraint by scheduling jobs.

D. THE OPTIMAL CPU-GPU UTILIZATION

The above energy consumption is not only affected by the power consumption, but also by the execution time of the job. They are all related to resource utilization. In fact, the Eq. (3) can also be expressed as

$$\begin{cases} ET(J_i, nc_j) = \dfrac{CE(J_i)}{cp(cn_j) \times cm(cn_j) \times cu(J_i, cn_j)}, \\ ET(J_i, nc_j) = \dfrac{GE(J_i)}{gp(cn_j) \times gm(cn_j) \times gu(J_i, cn_j)}. \end{cases} \tag{17}$$

Here, we combined Eq. (13) and Eq. (17) as

$$\begin{cases} E^c(J_i, cn_j) = P^c(J_i, cn_j) \times \dfrac{CE(J_i)}{cp(cn_j) \times cm(cn_j) \times cu(J_i, cn_j)}, \\ E^g(J_i, cn_j) = P^g(J_i, cn_j) \times \dfrac{GE(J_i)}{gp(cn_j) \times gm(cn_j) \times gu(J_i, cn_j)}. \end{cases} \tag{18}$$

The partial derivative of Eq. (18) for CPU and GPU utilization are

$$\begin{cases} \dfrac{\partial E^c(J_i, cn_j)}{\partial cu(J_i, cn_j)} \\ = \partial P^c(J_i, cn_j) \times \dfrac{CE(J_i)}{cp(cn_j) \times cm(cn_j) \times cu(J_i, cn_j)} \\ \quad - P^c(J_i, cn_j) \times \dfrac{CE(J_i)}{cp(cn_j) \times cm(cn_j) \times cu^2(J_i, cn_j)}, \\ \dfrac{\partial E^g(J_i, cn_j)}{\partial gu(J_i, cn_j)} \\ = \partial P^g(J_i, cn_j) \times \dfrac{GE(J_i)}{gp(cn_j) \times gm(cn_j) \times gu(J_i, cn_j)} \\ \quad - P^g(J_i, cn_j) \times \dfrac{GE(J_i)}{gp(cn_j) \times gm(cn_j) \times gu^2(J_i, cn_j)}. \end{cases} \tag{19}$$

This equations can be approximately solved by assign them as zero

$$\begin{cases} \dfrac{\partial E^c(J_i, cn_j)}{\partial cu(J_i, cn_j)} = 0, \\ \dfrac{\partial E^g(J_i, cn_j)}{\partial gu(J_i, cn_j)} = 0. \end{cases} \tag{20}$$

Thus, we can get the computing node optimal CPU utilization $cu'(J_i, cn_j)$ and GPU utilization $gu'(J_i, cn_j)$ for system energy consumption saving.

V. THE PROPOSED JOB SCHEDULING ALGORITHM

A. A HEURISTIC GREEDY STRATEGY

In this study, we proposed a heuristic greedy strategy to solve this heterogeneous systems CPU-GPU utilization aware and energy-efficient job scheduling problem, which is formalized in **A**lgorithm 1 and we name it as UEJS. To improve scheduling performance, we use the following CPU-GPU utilization condition

Condition 2 (Utilization Constraint): For any job $J_i \in \Lambda$ and computing node $cn_j \in \Theta$

$$\begin{cases} |cu(J_i, cn_j) - cu'(J_i, cn_j)| \leq \varepsilon, \\ |gu(J_i, cn_j) - gu'(J_i, cn_j)| \leq \varepsilon. \end{cases} \tag{21}$$

where the weight ε is used to ensure that the CPU-GPU utilization of job J_i on computing node cn_j as close as possible to its optimal utilization.

At the beginning, we first initialize the system computing nodes parameters, such as the computing node latest execution finish time $nd(cn_j)$. Then, for each unscheduled job J_i, we try to find a computing node cn_j that can satisfy the **C**ondition 1, 2 and with the minimum job execution energy consumption (Steps 3-12). At last, for all these job and computing node $< J_i, cn_j >$ pairs, the algorithm will find the optimal pair and assign this job to the corresponding heterogeneous computing node (Steps 13). The algorithm removes job J_i from unscheduled job set and updates systems computing node cn_j parameters in Steps 14-16.

The time complexity of this heuristic greedy scheduling strategy is usually expressed in terms of the number of jobs n, and the system computing nodes m. The time complexity of our proposed UEJS is analyzed as follows: Initialize system computing nodes parameters in Step 1 and compute job average energy consumption in Step 18 can be done in $O(m)$ time. The time complexity of optimal job and computing node $< J_i, cn_j >$ pairs searching by Steps 3-12 are $O(nm)$ time. In this UEJS algorithm, there has n jobs need to repeat Step 3-17, and the overall time complexity is $O(n^2m)$.

Algorithm 1 The CPU-GPU Utilization Aware and Energy-Efficient Heuristic Greedy Algorithm (UEJS)

Input: BoT applications set Λ and system computing nodes set Θ.
Output: Jobs and computing nodes assignment $< J_i, cn_j >$.

1 Initialize system computing nodes parameters;
2 **while** *there are unscheduled jobs* **do**
3 **for** *each unscheduled job* J_i **do**
4 **for** *each computing node* nc_j **do**
5 Use Eq. (2) to compute job execution time $ET(J_i, nc_j)$;
6 Use Eq. (3) to compute job CPU-GPU utilization;
7 Use Eq. (20) to compute optimal CPU-GPU utilization;
8 **if** *job* J_i *and computing node* nc_j *satisfy* ***C****ondition 1,2* **then**
9 Use Eq.(10)-(14) to compute energy consumption $E(J_i, cn_j)$.
10 **end**
11 **end**
12 **end**
13 Find a job and computing node $< J_i, cn_j >$ with the minimum energy consumption $E(J_i, cn_j)$;
14 Assign job J_i to computing node cn_j;
15 Remove job J_i from unscheduled job set;
16 Update computing node cn_j latest execution finish time $nd(cn_j)$.
17 **end**
18 Use Eq.(15)-(16) to compute job average energy consumption.

B. A HYBRID PARTICLE SWARM OPTIMIZATION ALGORITHM

The above heuristic greedy scheduling algorithm (UEJS) has good local optimization performance, but the global optimization ability needs to be improved. Therefore, we proposed a hybrid system CPU-GPU utilization aware and energy-efficient job scheduling algorithm (H-PSO), which incorporates the heuristic greedy strategy and bio-inspired search technique. Where, the chosen evolutionary technique is particle swarm optimization (PSO) that is derived from the study of simulating social behaviors of flock birds [26], [32]. The PSO model consists of individuals,

referred to as particles, which each individual represents a solution of jobs and computing nodes assignment $< J_i, cn_j >$ pairs. In this study, we assume that $X_{i,j} = 1$ if job J_i is scheduled on computing node cn_j, and $X_{i,j} = 0$ otherwise. Thus, the individual can be expressed as the following matrix:

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,m} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,m} \\ \vdots & \vdots & \cdots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,m} \end{bmatrix} \tag{22}$$

Accordingly, the PSO individual of such matrix X must satisfy the following constraints:

$$\sum_{i=1}^{n} X_{i,j} = 1, \quad \forall j. \tag{23}$$

In our proposed hybrid PSO algorithm, we use a population with K individuals to share experience and discovery knowledge. This algorithm has two phases, the first phase is tried to find the start position and step of jobs by PSO. Then, the heuristic greedy strategy is used to find an optimal system computing node for chosen jobs.

For the first phase, the jobs are stored in the cycle scheduling list, and we try to select jobs by its start position and step. Here, the start position and step (x_{id}^k, s_{id}^k) are the PSO kth particle current position, and (vx_{id}^k, vs_{id}^k) are the k particle current velocity. The k particle own best experience position is denoted as $p_{k,x}^{best}$, $p_{k,s}^{best}$, and the PSO global optimal position of the whole population is represented as g_x^{best}, g_s^{best}. During each PSO iteration i, the following equations are used to update the velocity and position of particle, which keeps on finding the optimal solution.

$$\begin{cases} vx_{id+1}^k = \omega_1 \times vx_{id}^k + c_1 \times Rand() \times (p_{k,x}^{best} - x_{id}^k) \\ \qquad\qquad + c_2 \times Rand() \times (g_x^{best} - x_{id}^k), \\ x_{id+1}^k = x_{id}^k + vx_{id+1}^k. \end{cases} \tag{24}$$

$$\begin{cases} vs_{id+1}^k = \omega_2 \times vs_{id}^k + c_1 \times Rand() \times (p_{k,s}^{best} - s_{id}^k) \\ \qquad\qquad + c_2 \times Rand() \times (g_s^{best} - s_{id}^k), \\ s_{id+1}^k = s_{id}^k + vs_{id+1}^k. \end{cases} \tag{25}$$

where variable ω is called the inertia weight, and we set $\omega_1 = 0.4, \omega_2 = 0.3$. c_1 is the cognitive coefficient of particle self-recognition component, and c_2 is the social coefficient based on the whole population experience. Here, we set $c_1 = 2, c_2 = 2$. $Rand()$ is a function that can generate random numbers from the range of [0, 1]. The fitness function used in this proposed hybrid PSO algorithm is the job average energy consumption and is defined in Eq. (16).

At the second phase, the heuristic greedy strategy is used to find the optimal computing node for each job. The job J_i is chosen from job list with start position x_{id}^k and step s_{id}^k, the heuristic greedy strategy will find a computing node cn_j with the minimum job execution energy consumption. At last, this algorithm assigns job J_i to the corresponding computing node cn_j and update computing node cn_j parameters. The hybrid PSO algorithm is described as **A**lgorithm 2.

Algorithm 2 The Hybrid PSO Algorithm (H-PSO)

Input: BoT applications set Λ and system computing nodes set Θ.
Output: Jobs and computing nodes assignment $< J_i, cn_j >$.

1 Initialize system computing nodes parameters;
2 Initialize a population with K particles by random positions, velocities;
3 **while** *the PSO stop condition is not met* **do**
4 #pragma omp parallel for;
5 **for** *each particle k* **do**
6 **for** *each job J_i chosen from job list with start position x_{id}^k and step s_{id}^k* **do**
7 **for** *each computing node nc_j* **do**
8 Use Eq. (2) to compute job execution time $ET(J_i, nc_j)$;
9 Use Eq. (3) to compute job CPU-GPU utilization;
10 Use Eq. (20) to compute optimal CPU-GPU utilization;
11 **if** *job J_i and computing node nc_j satisfy **C**ondition 1,2* **then**
12 Use Eq.(10)-(14) to compute energy consumption $E(J_i, cn_j)$.
13 **end**
14 **end**
15 Find a computing node cn_j with the minimum energy consumption $E(J_i, cn_j)$;
16 Assume job J_i to computing node cn_j;
17 Update computing node cn_j parameters.
18 **end**
19 Use Eq. (15)-(16) to evaluate the particle fitness;
20 Update the particle best position $p_{k,x}^{best}$, $p_{k,s}^{best}$ and global best position g_x^{best}, g_s^{best};
21 Update start position of particle by Eq. (24);
22 Update step of particle by Eq. (25).
23 **end**
24 **end**
25 Find the optimal particle by global best position g_x^{best}, g_s^{best};
26 Assign job J_i to corresponding computing node cn_j;
27 Use Eq.(15)-(16) to compute job average energy consumption.

The time complexity of Steps 6-18 in our proposed H-PSO algorithm is $O(nm)$ time. We assume that the PSO iteration is $\Re$ and the overall time complexity of H-PSO is $O(\Re Knm)$. Compared with our proposed heuristic greedy scheduling algorithm, the time complexity of H-PSO algorithm increases greatly as the PSO iteration $\Re$ is much larger than job number. However, we observe from **A**lgorithm 2 that the particles (or feasible scheduling solutions) can be computed in parallel. In this study, we adopt OpenMP parallel method to

TABLE 2. The heterogeneous systems computing nodes parameters.

Node Type	CPU capacity (TFLOPS)	CPUs	GPU capacity (TFLOPS)	GPUs	P^c_{idle}	P^c_{max}	P^g_{idle}	P^g_{max}	P_{other}
1	2.8	2	10.6	4	200W	280W	980W	1300W	860W
2	3.2	1	4.3	6	90W	145W	1650W	2180W	1070W
3	1.8	4	5.6	2	320W	580W	460W	540W	720W
4	3.2	2	20.2	4	210W	330W	1180W	1360W	1080W
5	2.2	1	12.4	2	120W	160W	520W	750W	920W

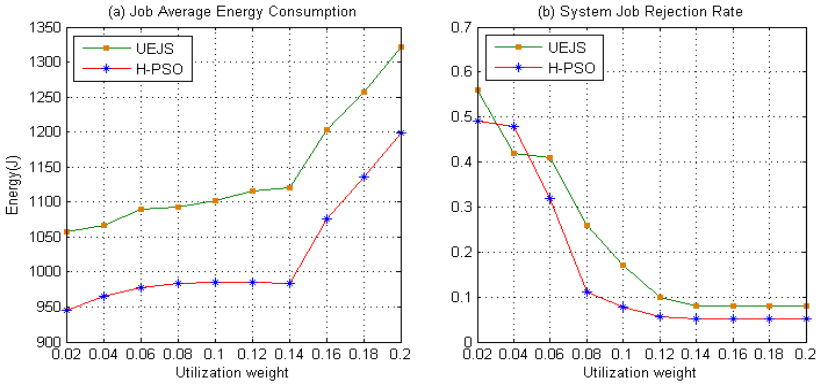

FIGURE 4. Various CPU-GPU utilization condition weight ε. (a) job average energy consumption; (b) system job rejection rate.

accelerate our proposed H-PSO (See line 4). We also conduct experiments with 400 jobs for this H-PSO algorithm on a computer with Intel i7 8700 CPU. The parallel H-PSO algorithm overhead is 1.4s and that of serial algorithm is 9.492. Thus, the speedup of parallel H-PSO can achieve 6.78.

VI. EXPERIMENTAL EVALUATION

In this section, the proposed heuristic greedy strategy UEJS and hybrid particle swarm optimization algorithm (H-PSO) are compared with the Max-EAMin [7] and the traditionally Genetic Algorithm (GA) [22], [23] by CloudSim. The heterogeneous computing systems simulator includes 5 type of computing nodes, and their parameters are listed in Table 2. These experiments are conducted on a computer with Intel i7 8700 CPU. The comparison metrics chosen in this paper are the job average energy consumption (Eq. (16)), system job rejection rate, and the average CPU, GPU utilization of computing nodes, which are defined as

$$\begin{cases} sjRR = \dfrac{n - SJNum}{n}, \\ avCU = \dfrac{\sum_{j=1}^{m} cu(cn_j)}{m}, \\ avGU = \dfrac{\sum_{j=1}^{m} gu(cn_j)}{m}. \end{cases} \tag{26}$$

where $sjRR$, $avCU$, $avGU$ are the system job rejection rate, average CPU utilization, and GPU utilization, respectively.

A. PARALLEL APPLICATIONS

In order to achieve practical insights into the effectiveness of our proposed CPU-GPU utilization aware and energy-efficient job scheduling algorithms, we simulate parallel applications. The applications characteristics, such as CPU execution size $CE(J_i)$, GPU execution size $GE(J_i)$, and critical path execution size $CP(J_i)$. Some of them like the Parallel Workloads Archive HPC2N trace [33], and some are come from the actual application execution log file of Shanghai Supercomputing Center and Changsha National Supercomputing Center. The others are randomly generated applications by the uniform distribution within the range of [500, 3500] tera operations for CPU, and [2000, 210000] tera operations for GPU, and critical path execution size $CP(J_i)$ is the [0.2,0.5] times of job GPU execution size. The jobs relative deadline $RD(J_i)$ is set as the 3 times of job average execution time.

B. VARIOUS UTILIZATION CONSTRAINT WEIGHT

In the first experiments, we evaluate the performance of CPU-GPU utilization condition weight ε to our proposed heuristic greedy strategy UEJS and hybrid PSO (H-PSO) job scheduling algorithm. Figure 4 plots the experimental results of job average energy consumption and system job rejection rate by varying weight ε from 0.02 to 2 with steps of 0.02. These experiments are conducted on 100 computing

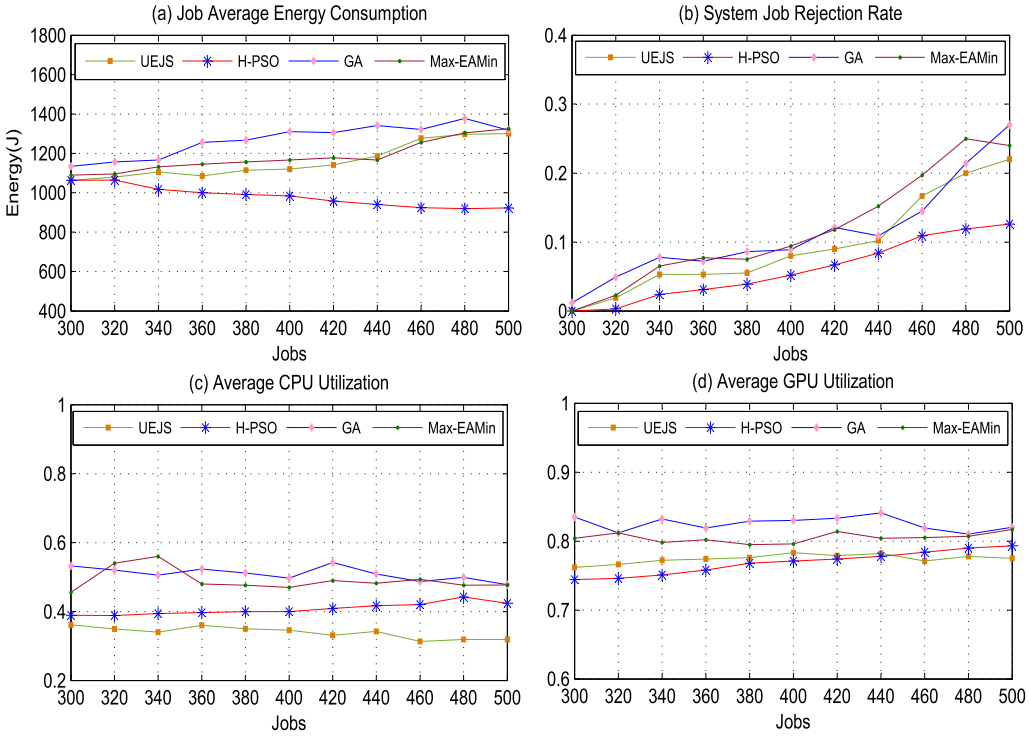

FIGURE 5. Various jobs on heterogeneous systems with 100 computing nodes. (a) energy consumption; (b) rejection rate; (c) CPU utilization; (d) GPU utilization.

nodes with 400 jobs. From Figure 4, we can conclude that the optimal utilization condition weight ε is 0.14. This is mainly because small ε can effectively save the job average energy consumption, such as job average energy consumption $EV = 965$ for H-PSO and $EV = 1067$ for UEJS as $\varepsilon = 0.04$. However, when the weight ε is greater than 0.14, the job average energy consumption of the two algorithms increases rapidly. On the other hand, when weight ε is lower than 0.12, the system job rejection rate changes greatly and is high. And, when ε is greater than or equal to 0.14, the system job rejection rate is basically stable and will not decrease with the growth of weight ε. In the following experiments, the UEJS and hybrid PSO algorithms set the weight ε of **C**ondition 2 to 0.14.

C. COMPARISON RESULTS

In the second set of experiments, we evaluate the performance of H-PSO, UEJS, and GA on heterogeneous systems with 100 computing nodes. The number of jobs is varied from 300 to 500 with steps of 20. Figure 5 shows the experimental results of this group of evaluation. From Figure 5(a), we can conclude that our proposed H-PSO, UEJS are better than Max-EAMin by 20.6%, 1.9%, and GA by 29.4%, 9.2% in term of the job average energy consumption, respectively. This performance improvement is due to the fact that GA is a job processing time optimization algorithm and not consider the job execution energy consumption. The power consumption of Max-EAMmin is very simple, and the relationship between the maximum power consumption and resource utilization is linear. Moreover, Max-EAMmin derived from Max-Min mainly focuses on the time optimization and not the energy consumption.

The job average energy consumption of UEJS, Max-EAMin, and GA increases as the number of jobs increases. However, for H-PSO algorithm, the job average energy consumption slowly decreases as the the number of jobs increases. In fact, for a large number of jobs, the job average energy consumption of H-PSO algorithm significantly outperforms that of UEJS, Max-EAMin, and GA. The example of 480 jobs is that the job average energy consumption of UEJS is 1297J, Max-EAMin is 1304J, GA is 1378J, and that of H-PSO is 919J. We observe from Figure 5(a) that H-PSO is over UEJS by 36.5%, Max-EAMin by 36.3%, and GA by 46.7% in term of the job average energy consumption with number of jobs from 440 to 500. In this case, our proposed hybrid PSO algorithm is much better than UEJS. Actually, H-PSO performs better than UEJS by 18.4% in term of the job average energy consumption. These experimental results also reveal that the H-PSO algorithm is very suitable for system with high workload.

The experimental results of system job rejection rate are shown in Figure 5(b), which shows that the system job rejection rate increases as the number of jobs increases. However, the system job rejection rate growth trend is relatively slow for H-PSO algorithm. In fact, the H-PSO is better than UEJS by 37%, Max-EAMin by 49.3%, and GA by 47.5% in term of system job rejection rate, respectively. This experimental phenomenon also shows that H-PSO can effectively schedule more jobs. The resource CPU-GPU utilization experimental results of heterogeneous systems computing nodes are shown

TABLE 3. The time overheads associated with our proposed algorithm UEJS and H-PSO.

Algorithm/Overhead(s)	300	320	340	360	380	400	420	440	460	480	500
UEJS	0.5	0.7	0.8	0.8	0.9	1	1.1	1.2	1.3	1.5	1.5
H-PSO	1.15	1.3	1.35	1.35	1.35	1.4	1.4	1.45	1.5	1.55	1.6

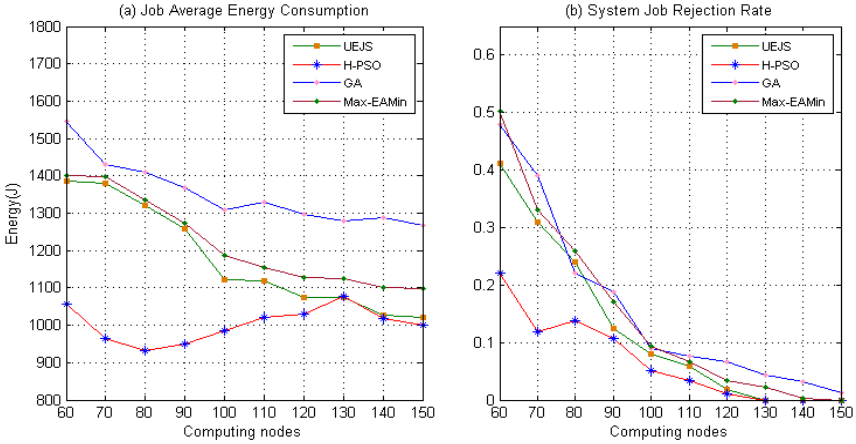

FIGURE 6. Various computing nodes with 400 jobs.(a) job average energy consumption; (b) system job rejection rate.

in Figure 5(c), Figure 5(d), where the GA is higher than H-PSO by 20%, UEJS by 33.4%, Max-EAMin by 3.7% for average CPU utilization, and H-PSO by 6.9%, UEJS by 6.2%, Max-EAMin by 2.5% for average GPU utilization, respectively. This experimental phenomena also demonstrate that GA can make better use of resources but consume more energy.

Table 3 lists the time overheads of our proposed heuristic greedy strategy UEJS and hybrid particle swarm optimization algorithm (H-PSO). From this table, we can conclude that the greedy algorithm UEJS is much faster than H-PSO for systems with a small number of jobs. However, as the job number increases, their time overheads get closer and closer.

The third group of experiments is conducted for 400 jobs and shown in Figure 6. Here, we change the number of computing nodes from 60 to 150, at the steps of 10. We observe from Figure 6(a) that our proposed H-PSO algorithm also outperforms UEJS by 17.4%, Max-EAMin by 21.6%, and GA by 37.8% in term of the job average energy consumption. Especially for the system with few nodes and high workload, such as 80 computing nodes, H-PSO algorithm is better than UEJS by 41.8%, Max-EAMin by 43.5%, and GA by 51.4% for job average energy consumption. The system job rejection rate experimental results of Figure 6(b) also demonstrate that our proposed H-PSO algorithm has high successful job schedule rate. Therefore, we can conclude that the H-PSO is the best system CPU-GPU utilization aware and energy-efficient job scheduling algorithm in this study.

VII. CONCLUSION AND FUTURE WORK

Massive energy consumption has become a major challenge for modern heterogeneous computing systems based on CPU-GPU. One of good solutions to overcome this problem is energy-efficient scheduling technique. However, the dynamic variability of resource utilization in the system computing node aggravates the difficulty of solving this problem. In this paper, we define system computing node CPU-GPU utilization model, and analyze the relationship between computing node power consumption and CPU-GPU utilization. Then, we propose a heuristic greedy job scheduling strategy (UEJS) and a hybrid particle swarm optimization algorithm (H-PSO) to reduce energy consumption. The experimental results show that our proposed H-PSO is better than UEJS, Max-EAMin, and GA in terms of the average job energy consumption and system job rejection rate.

We would like to mention two directions in this domain for further research. First, we shall extend the relationship between computing node power consumption and CPU-GPU utilization to improve the accuracy of energy consumption calculation. Second, we plan to study more complex version of hybrid particle swarm optimization algorithm, in which the reliability, cost issues are taken into account.

REFERENCES

[1] E. Yang, S.-H. Kim, T.-W. Kim, M. Jeon, S. Park, and C.-H. Youn, "An adaptive batch-orchestration algorithm for the heterogeneous GPU cluster environment in distributed deep learning system," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2018, pp. 725–728.

[2] C. Chen, K. Li, A. Ouyang, Z. Zeng, and K. Li, "GFlink: An in-memory computing architecture on heterogeneous CPU-GPU clusters for big data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1275–1288, Jun. 2018.

[3] *TOP 500 List*. Accessed: Feb. 15, 2020. [Online]. Available: https://www.top500.org/lists/2019/11/

[4] S. Alsubaihi and J.-L. Gaudiot, "A runtime workload distribution with resource allocation for CPU-GPU heterogeneous systems," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2017, pp. 994–1003.

[5] D. Cheng, J. Rao, C. Jiang, and X. Zhou, "Elastic power-aware resource provisioning of heterogeneous workloads in self-sustainable datacenters," *IEEE Trans. Comput.*, vol. 65, no. 2, pp. 508–521, Feb. 2016.
[6] A. M. Haywood, J. Sherbeck, P. Phelan, G. Varsamopoulos, and S. K. S. Gupta, "The relationship among CPU utilization, temperature, and thermal power for waste heat utilization," *Energy Convers. Manage.*, vol. 95, pp. 297–303, May 2015.
[7] G. L. Stavrinides and H. D. Karatza, "The impact of workload variability on the energy efficiency of large-scale heterogeneous distributed systems," *Simul. Model. Pract. Theory*, vol. 89, pp. 135–143, Dec. 2018.
[8] M. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979.
[9] Z. Lu and Y. Yao, "Marginal performance: Formalizing and quantifying power Over/Under provisioning in NoC DVFS," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1903–1917, Nov. 2017.
[10] K. Ma, X. Wang, and Y. Wang, "DPPC: Dynamic power partitioning and control for improved chip multiprocessor performance," *IEEE Trans. Comput.*, vol. 63, no. 7, pp. 1736–1750, Jul. 2014.
[11] A. Suyyagh and Z. Zilic, "Energy and task-aware partitioning on single-ISA clustered heterogeneous processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 2, pp. 306–317, Feb. 2020.
[12] Y. Qin, G. Zeng, R. Kurachi, Y. Li, Y. Matsubara, and H. Takada, "Energy-efficient intra-task DVFS scheduling using linear programming formulation," *IEEE Access*, vol. 7, pp. 30536–30547, Mar. 2019.
[13] D. Cheng, X. Zhou, P. Lama, M. Ji, and C. Jiang, "Energy efficiency aware task assignment with DVFS in heterogeneous Hadoop clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 70–82, Jan. 2018.
[14] F. Dong and S. Akl, "Scheduling algorithms for grid computing: State of the art and open problems," School Comput., Queens Univ., Kingston, Kingston, ON, Canada, Tech. Rep. 2006-504, 2006.
[15] X. Tang, X. Liao, J. Zheng, and X. Yang, "Energy efficient job scheduling with workload prediction on cloud data center," *Cluster Comput.*, vol. 21, no. 3, pp. 1581–1593, Sep. 2018.
[16] L. Zhang, K. Li, W. Zheng, and K. Li, "Contention-aware reliability efficient scheduling on heterogeneous computing systems," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 3, pp. 182–194, Jul./Sep. 2018.
[17] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, and K. Li, "Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems," *Future Gener. Comput. Syst.*, vol. 74, pp. 1–11, Sep. 2017.
[18] Z. Zong, A. Manzanares, X. Ruan, and X. Qin, "EAD and PEBD: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters," *IEEE Trans. Comput.*, vol. 60, no. 3, pp. 360–374, Mar. 2011.
[19] S. Chen, Z. Li, B. Yang, and G. Rudolph, "Quantum-inspired hyper-heuristics for energy-aware scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 6, pp. 1796–1810, Jun. 2016.
[20] K. Huang, X. Jiang, X. Zhang, R. Yan, K. Wang, D. Xiong, and X. Yan, "Energy-efficient fault-tolerant mapping and scheduling on heterogeneous multiprocessor real-time systems," *IEEE Access*, vol. 6, pp. 57614–57630, Oct. 2018.
[21] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.
[22] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, "A Johnson's-Rule-Based genetic algorithm for Two-Stage-Task scheduling problem in data-centers of cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 597–610, Jul./Sep. 2019.
[23] X. Tang, X. Li, and Z. Fu, "Budget-constraint stochastic task scheduling on heterogeneous cloud systems," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 19, p. e4210, Oct. 2017.
[24] H. F. Sheikh, I. Ahmad, and S. A. Arshad, "Performance, energy, and temperature enabled task scheduling using evolutionary techniques," *Sustain. Comput., Inform. Syst.*, vol. 22, pp. 272–286, Jun. 2019.
[25] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Comput.*, vol. 62, pp. 1–19, Feb. 2017.
[26] L.-D. Chou, H.-F. Chen, F.-H. Tseng, H.-C. Chao, and Y.-J. Chang, "DPRA: Dynamic power-saving resource allocation for cloud data center using particle swarm optimization," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1554–1565, Jun. 2018.
[27] M. Kumar and S. C. Sharma, "PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint," *Sustain. Comput., Inform. Syst.*, vol. 19, pp. 147–164, Sep. 2018.
[28] X. Wang, X. Fu, X. Liu, and Z. Gu, "Power-aware CPU utilization control for distributed real-time systems," in *Proc. 15th IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2009, pp. 233–242.
[29] J. Y. Jang, H. Wang, E. Kwon, J. W. Lee, and N. S. Kim, "Workload-aware optimal power allocation on single-chip heterogeneous processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 6, pp. 1838–1851, Jun. 2016.
[30] X. Tang, K. Li, G. Liao, K. Fang, and F. Wu, "A stochastic scheduling algorithm for precedence constrained tasks on grid," *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1083–1091, Oct. 2011.
[31] G. Xie, G. Zeng, L. Liu, R. Li, and K. Li, "Mixed real-time scheduling of multiple DAGs-based applications on heterogeneous multicore processors," *Microprocessors Microsyst.*, vol. 47, no. A, pp. 93–103, 2016.
[32] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment," *Future Gener. Comput. Syst.*, vol. 97, pp. 361–378, Aug. 2019.
[33] *Parallel Workloads Archive*. Accessed: Feb. 10, 2020. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/
[34] S. K. Roy, R. Devaraj, A. Sarkar, K. Maji, and S. Sinha, "Contention-aware optimal scheduling of real-time precedence-constrained task graphs on heterogeneous distributed systems," *J. Syst. Archit.*, vol. 105, May 2020, Art. no. 101706.
[35] S. K. Roy, R. Devaraj, and A. Sarkar, "Optimal scheduling of PTGs with multiple service levels on heterogeneous distributed systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 157–162.
[36] R. Devaraj, "A solution to drawbacks in capturing execution requirements on heterogeneous platforms," *J. Supercomput.*, early access, Jan. 8, 2020, doi: 10.1007/s11227-020-03145-w.
[37] K. M. Tarplee, R. Friese, A. A. Maciejewski, H. J. Siegel, and E. K. P. Chong, "Energy and makespan tradeoffs in heterogeneous computing systems using efficient linear programming techniques," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 6, pp. 1633–1646, Jun. 2016.
[38] S. Sandokji, F. Essa, and M. Fadel, "A survey of techniques for warp scheduling in GPUs," in *Proc. IEEE 7th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2015, pp. 600–606.
[39] X. Tang and X. Liao, "Application-aware deadline constraint job scheduling mechanism on large-scale computational grid," *PLoS ONE*, vol. 13, no. 11, 2018, Art. no. e0207596.
[40] A. Dogan and F. Özgüner, "Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 308–323, Mar. 2002.

XIAOYONG TANG received the M.S. and Ph.D. degrees from Hunan University, China, in 2007 and 2013, respectively. He is currently a Professor with the College of Information and Intelligence, Hunan Agricultural University. He has published over 50 research articles in refereed journals and conference proceedings, such as the IEEE TRANSACTIONS ON COMPUTERS (TC), the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS (TII), IEEE ACCESS, JPDC, FGCS, and ICPP. His research interests include distributed computing systems scheduling, parallel computing, cloud computing, and parallel algorithms. He is a Reviewer of TC, TPDS, TSC, TII, JPDC, FGCS, CPE, IEEE ACCESS, and so on.

ZHUOJUN FU received the M.S. degree from Hunan Agricultural University, China, in 2010. He is currently an Associate Professor with the College of Information and Intelligence, Hunan Agricultural University. His research interests include parallel algorithms, intelligence decision systems, and cloud computing.