Virtual Machine Dynamic Deployment Scheme Based on Double-Cursor Mechanism

SHUKUN LIU1, (Member, IEEE), CHAOLIANG LI2, ZHIMIN LIU3, AND QIANG ZHANG4

1College of Information Science and Engineering, Hunan Women’s University, Changsha 410004, China
2College of Computer and Information Engineering, Hunan University of Technology and Business, Changsha 410205, China
3College of Mathematics and Computational Science, Hunan First Normal University, Changsha 410205, China
4School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China

Corresponding author: Shukun Liu (liu_shukun@csu.edu.cn)

This work was supported in part by the Natural Science Foundation of Hunan Province under Grant 2018JJ2193 and Grant 2020JJ4250; in part by the Scientific Research Fund of Hunan Provincial Education Department under Grant 18A471, Grant 19A275, and Grant 19B208; in part by the Project of Hunan Education Science Achievement Appraisal Committee in 2020 under Grant XSB20BGD007; in part by the Project of Hunan Social Science Achievement Appraisal Committee in 2020 under Grant XSP20YBZ043; and in part by the Project of Ministry of Education Collaborative Education under Grant 201901050006.

ABSTRACT In accordance with the dynamically changing characteristics of user application load demand, the optimization of energy consumption in the process of virtual machine automatic deployment on the basis of the effective integration of user personalized resource demand in cloud platforms and the dynamic migration technology of virtual machines is studied in this work. Aiming at the problem of energy consumption optimization, an energy-efficient virtual machine deployment framework, which can be dynamically adjusted in accordance with the user’s actual resource needs, is designed. In the virtual machine deployment framework, this paper presents a virtual machine deployment selection algorithm based on a double-cursor control mechanism for the resource usage state of CPU and memory, which achieves a binocular optimization balance to a certain extent. Simulation experiment results show that when the virtual machine deployment framework proposed in this paper and the virtual machine deployment selection algorithm based on a double-cursor control mechanism are combined, the number of necessary active physical nodes in the cloud data center can be effectively controlled, and the frequency of virtual machine migration synchronization can be reduced. Consequently, the cloud data center can maintain low energy costs.

INDEX TERMS Double cursors, virtual machine, automatic deployment, virtual machine migration.

I. INTRODUCTION

To some extent, cloud computing [1], [2] is an effective extension of traditional parallel, distributed, grid, and edge computing. Present relevant research on cloud resource management mainly focuses on the following three aspects [3]: first, the primary goal is to reduce the energy consumption of cloud platforms [4], [5]; second, the main goal is to improve the utilization rate of system resources [6], [7]; third, resource management based on economics regards fault tolerance in virtual machine deployment as a key goal [8]. The virtual machine resource allocation model [2], which is based on user benefit maximization and the virtual machine deployment mechanism of cloud platforms, has a critical effect on effective resource utilization and energy consumption.

In the actual working environment, the establishment of a cloud data center has led to high energy consumption, and, to a certain extent, to the continuous increase in the hardware and software costs of operating service providers. With the rapid development of virtualization technology, effective correspondence between a single physical node and multiple virtual machine instances can be achieved [2], [3], and personalized configuration of virtual machines has become the main trend. A personalized virtual machine resource allocation method can control the overall energy consumption in the data center, thus improving the efficiency of resource use in each dimension [2], [3], [9].

A series of research has indicated that the user’s personalized resource needs have become a key factor for measuring the virtual machine deployment in a cloud platform. The number of necessary active physical nodes in the cloud platform can be adjusted through this factor; hence, the physical
nodes in the idle state can be switched to the energy-saving mode [2], [4], [5]. However, the correlation between user and load demands has caused great obstacles to virtual machine resource prediction. If resources cannot be accurately predicted, a series of problems, such as increased response time, interaction timeout, and even deployment failure, will often occur during virtual machine deployment [3], [5].

Therefore, properly handling the relationship between energy consumption and performance and providing a high and reliable quality-of-service (QoS) guarantee while solving energy consumption are key issues for resource suppliers [3]. On the basis of the dynamic coordination mechanism of double cursors, this paper proposes a double-cursor control method to solve the problem of virtual machine deployment. The double-cursor mechanism is a dynamic measurement and adjustment mechanism for specific resources (for example, CPU and memory). CPU and memory can be dynamically adjusted into a coordinated state of resource mixing through the cursor mechanism, which can effectively improve resource utilization. The double-cursor mechanism for the control of CPU and memory resources plays an important role in personalized resource integration and virtual machine deployment for virtual machine migration.

In this study, first, we formulate our optimization problem as a 2D optimization problem based on a double-cursor control mechanism for the resource usage state of CPU and memory. This method is different from the traditional way. Second, this study proposes a new resource deployment process of the double-cursor mechanism, in which the problem of efficient utilization of physical resources can be equivalent to the problem of mapping sets, that is, to make a reasonable and effective correspondence between physical resources and customer tasks on the basis of CPU and memory. Third, this study proposes a framework for the dynamic deployment of virtual machines and presents a virtual machine deployment selection algorithm based on the double-cursor control mechanism for the resource usage state of CPU and memory, which achieves a binocular optimization balance. Simulation experiment results show that the number of necessary active physical nodes in the cloud data center can be effectively controlled, and the frequency of virtual machine migration synchronization can be reduced. Accordingly, the cloud data center can maintain low energy costs.

The rest of this paper is organized as follows. Section 2 surveys related works. Section 3 states the problem that we will address and presents related definitions. Section 4 describes the framework for the dynamic deployment of virtual machines and the double-cursor virtual machine resource allocation (DCVMRA) algorithm. Section 5 presents the experimental evaluation and analysis. Lastly, Section 6 concludes this paper and suggests a future research direction.

II. RELATED WORK
In the virtual machine deployment research, Literature [3] realized multi-objective optimization in the virtual machine deployment process through NSGA-II, which reduced the amount of migration in the virtual machine deployment process and improved the utilization efficiency of resources to a certain extent. Literature [4]–[6] regarded the energy consumption of physical nodes in a cloud platform as the main optimization goal, optimized the process of dynamic migration of virtual machine deployment, and realized the reduction of virtual machine migration under a specific deployment environment. Literature [6] focused on service-level agreement (SLA) that users are concerned about and determined that the optimization in the deployment process is still dominated by energy saving. The essential idea is to maximize the reduction of the number of open physical machines to maximize the use limit of physical resources; a power management strategy has been proposed to improve the integration of hardware and virtual machines [3].

In Literature [7], the idle and execution power data based on physical nodes were compared to explain the energy consumption during virtual machine deployment [3], [4]. The dynamic deployment and migration of virtual machines were solved through optimization, with the optimization goal of minimizing the energy consumption of physical nodes. A random task model was established in accordance with the randomness characteristics of dynamic deployment requirements, and temporary intervention was conducted in the deployment process. The main problem is that in the stochastic model constraint, task loss rate is relatively high [27], [28]; consequently, the expected results of users are difficult to achieve. Literature [8] proposed a resource allocation method based on SLA awareness, which can reduce the energy consumption of physical nodes and SLA violations. The coordination between performance and physical energy consumption is reflected in the virtual machine deployment process [4]. Literature [3] proposed a management allocation strategy for virtual machine resources with the main goal of reducing minimum energy consumption and SLA violations. The advantage of this policy is that it addresses the SLA issues that users are most concerned about. Literature [10] did not explicitly express the migration overhead in the process of virtual machine deployment, but it restricted the migration time of virtual machines, as reflected in the form of threshold. The migration process control of virtual machines was realized with an improved bin packing algorithm.

The rapid development of cloud platform data centers has gradually exposed the problems of rapid increase in energy consumption, excessive migration of virtual machines, and reduced resource utilization [11]–[13]. The abovementioned problems have become the key problems in cloud platforms [14]–[17]. The optimization methods adopted in the deployment of virtual machines in current cloud platforms are mainly manifested as single- and multi-objective optimization [18], [19]. For example, optimization is conducted in terms of the number of physical nodes that need to be activated, the application QoS, data center energy consumption, and virtual machine migration [20], [21]. Nevertheless, in reality, a conflict often exists among the aforementioned factors [22]–[24]. For instance, reducing the
number of physical nodes that are required to be activated in a data center will result in multiple virtual machine migrations [25], [26], increasing the number of virtual machine migrations and the overall energy consumption [4]. In accordance with the personalized adjustment of user resources, this paper mainly considers the effects of CPU and memory on physical resource utilization, host energy consumption, virtual machine migration, and SLA violations and proposes a virtual machine deployment scheme based on a double-cursor control mechanism with balanced performance and energy consumption.

In short, a comparison of typical literature is shown in Table 1.

### III. PROBLEM DESCRIPTION

Among the important reasons for the widespread promotion of cloud platforms in recent years are that they can strengthen resource sharing and maximize the effective utilization of physical resources while reducing physical energy consumption. The problem of efficient utilization of physical resources can be equivalent to the problem of mapping sets, that is, to make a reasonable and effective correspondence between physical resources and customer tasks.

#### A. DOUBLE-CURSOR MECHANISM

The cursor proposed in this article is not a cursor in a database but refers to two measurement parameters in the resource allocation process. The value of parameters can be determined using the double-parameter cooperative control mechanism, and the best configuration effect of physical resources can be achieved properly. A physical vernier caliper is simulated, in which only two dimensions of numerical CARDS are tight, to measure the precise length of an object. In the measurement process, the left calipers and the vernier calipers can be in different positions to determine only the relative parameter values. The two sides can eventually be the most accurately measured parameter values in the process of resource allocation. In this paper, CPU and memory are defined as two cursor parameters or double cursors for short. The mechanism of their cooperation is defined as the double-cursor mechanism. A simple operation process is shown in Figure 1.

![FIGURE 1. CPU and memory dimension cursors. This figure represents the synergy between CPU and memory dimension cursors for virtual machine deployment.](image)

#### B. SERVICE PROCESS

Multiple physical devices constitute a set of physical resources, and the provider combines them into a cloud platform (including several types of physical resources) on the basis of existing physical devices. The basic construction process of a cloud platform and the principle of the physical resource service process are shown in Figure 2.

### TABLE 1. Comparison of typical literature.

| Typical literature | Characteristic | Disadvantage |
|--------------------|----------------|--------------|
| Li Shuangli, Li Zhuhua, Yu Xinrong [3] | On the basis of NSGA-II, a multi-objective optimization in the virtual machine deployment process is realized, and the migration amount in the process is reduced. | They ignore the SLA issues that users are most concerned about. |
| Zhang Xiaoqing, XU Lipan, Wang Fupang [5] | The frequency of virtual machine migration can be reduced only in certain deployment environments. | The requirements for deployment environments are high, including requiring users to have clear resource type requirements. |
| Luo Jie, Zhang Zhiming, Gao Zhiqiang Cheng Chun [6] | The deployment process focuses on SLA issues of user concern, improving the further integration of hardware and virtual machines. | Resource utilization is relatively low. |
| Hu Zhigang, Xiao Hui, Li Keqin [7] | On the basis of a comparison of idle and execution power data of physical nodes, the energy consumption during virtual machine deployment is explained, and a separate power management strategy is proposed. | Task loss rate is relatively high; consequently, the expected results of users are difficult to achieve. |
| Liu Jicai, Wang Shangguan, Zhou Ao, Yang Fangchun [8] | A resource allocation method based on SLA awareness is proposed, which can reduce the consumption of physical nodes and SLA violations. The coordination between performance and physical energy consumption is reflected in the virtual machine deployment process. | No standardization of the classification criteria for using user requirements exists. |
| Rui Li, Qinghua Zheng, Xinj Li, Zheng Yan [9] | A virtual machine resource management allocation strategy is proposed. The primary goal is to reduce SLA violations. | The deployment process requires dynamic statistics of resource types, which takes a long time. |
| Xiaolong Xu, Xuyun Zhang, Masbool Khan, Wanchun Dou, Shengjun Xue, Shuai Yu [10] | The time of virtual machine migration is constrained to realize the process control of virtual machine migration by using an improved boxing algorithm. | The migration overhead during virtual machine deployment is not explicitly stated. |
the cloud platform, with different time series, a large number of user demands for resources will be collected to form a demand queue buffer, and the demand waiting queue buffer will be formed on the basis of the priority allocation condition of resources in time order. Each running task is executed in parallel and recycles physical resources in the cloud platform after completion.

In this paper, a fixed time period is regarded as an example to illustrate the problem. The user’s resource sequence within time period T is defined as R$_i$, where $t_i$ is a time point that is greater than 0 and less than T ($T > t_i > 0$; the user can simultaneously issue task resource requests). R$_i$ represents the set of resource requirements of the I$_i$th user, which can be specifically expressed as {r$_1$, r$_2$, r$_3$, ...... r$_n$}. {r$_1$, r$_2$, r$_3$, ...... r$_n$} is encapsulated into several independent virtual machines to form the corresponding sequence of r$_k$ and v$_j$ (Jth virtual machine) and ensure that every resource requirement is met. Deploying resource requirements to as few virtual machines as possible must be considered given how resources correspond to virtual machines at a specified point in time. In this way, the number of physical machines that are turned on in the actual state is reduced. Another issue that should be addressed is that during the changing process of $t_i$ into $t_{i+1}$, the degree of complementary CPU and memory should be considered. Virtual machines that can be combined and migrated should be calculated in accordance with the actual needs of users. Effective closure of free physical nodes and saving of the resource and energy costs of a cloud platform will dynamically adjust the demand of virtual machine resources and the number of physical resources. In this study, automatic allocation of virtual machine resources and effective recovery of physical resources are accomplished through the automatic adjustment of the double cursors of CPU and memory.

C. CURSOR RESOURCE REPRESENTATION

From the preceding description, we discuss how to place a virtual machine reasonably in accordance with the statistical multidimensional resource occupancy of a physical node in the process of virtual machine deployment. This paper regards a technical report on Microsoft virtual machine management as the reference standard to achieve a uniform description of performance. The evaluation standard of physical server resources mainly adopts CPU and memory basic resources as the measurement indicator. The main problem is described as follows. The CPU and memory are equivalent to the two dimensions of a cursor in the following representation. The correspondence between the physical and virtual resources is accomplished using a fixed value of the bidirectional cursor.

1. At a certain time, n physical nodes exist ($M = \{m_1, m_2, m_3, \ldots, m_n\}$, where M is a physical machine set, $m_i$ represents a physical node, and n represents the number of physical nodes in the cluster). Their resources mainly include memory, CPU, disk, network bandwidth, and I/O resources.

2. In a certain period of time, m virtual machines need to be effectively placed ($V = \{v_1, v_2, v_3, \ldots, v_m\}$, where V is a virtual machine set, $v_i$ represents a virtual machine node, and m represents the number of virtual machines needed in the cluster). The resource requirements for these virtual machines are assumed to include memory, CPU, disk, network bandwidth, and I/O resources.

3. A reasonable mapping between virtual machines and physical nodes required by users is necessary to find, which cannot only meet the resource requirements of each virtual machine but also minimize the number of physical nodes used. In this mapping relationship, the total resources of each dimension required by a virtual machine should not exceed the total resources of each dimension on the physical node on which it is placed.

For the illustration of the abovementioned multidimensional resources, only CPU and memory are considered in this paper.

IV. FRAMEWORK FOR THE DYNAMIC DEPLOYMENT OF VIRTUAL MACHINES

A. SYMBOL DESCRIPTION

In this scheme, the physical resource constraint set is represented as PRCS(Physical Resource Constraint Set), the resource demand state of $t_i$ at a certain time point is represented as RDS($t_i$), and the rate of change in USER $U_i$ demand is represented as Ra($U_i$). A cloud platform is generally composed of several data centers. H data centers are represented as $H_{h=1}^{H} RD(h)$. More symbols are described in Table 2.
TABLE 2. Common symbols.

| Symbol | Symbolic meaning |
|--------|------------------|
| H      | Number of data centers |
| DC     | Data center |
| D(h)   | h^b data center |
| RD(h)  | h^a data center resource |
| S      | Available service vector |
| S(V)   | Virtual machine set |
| S(P)   | Physical machine set |
| CPU    | CPU resource dimension |
| Mem    | Memory resource dimension |

B. RESOURCE ALLOCATION MODEL FOR DATA CENTER

The physical configuration of a data center hosts is diversified. The virtual machines generated in accordance with user requirements should be deployed on different physical nodes on the basis of specific requirements. A data center is supposed to consist of physical nodes, then the data center is represented as DC={dc_1, dc_2, dc_3, ..., dc_m}. A host on which virtual machine v_j is deployed is represented as DC_i(V_j), where j is between 1 and n, and i is between 1 and m. The set of virtual machines running on host DC_i is represented as V_{DC_i}. The number of Cpus applied using virtual machine v_j is expressed as CPU_{VR_j}, and the resource requirements of the total CPU of host DC_i are expressed as P_{cpu}(DC_i). The amount of memory requested by v_j is Mem_{VR_j}. The resources allocated for the total memory of host DC_i are represented as P_{mem}(DC_i). Then, we provide the definition, which is expressed as Formula (1).

\[ P_{mem}(DC_i) = \sum_{i=1}^{m} (x_{ij} \cdot VR_i) \]

The CPU utilization is defined as the total amount of resources allocated divided by the total amount of physical resources in the data center. Similarly, CPU utilization is expressed as the total amount of CPU resources allocated divided by the total amount of physical node resources. The formulas are expressed as (2) and (3).

\[ CPU(unity) = P_{cpu}(DC_i) = \sum_{i=1}^{m} (x_{ij} \cdot VR_i) / \tau(CPU) \]  
\[ MEM(unity) = P_{mem}(DC_i) = \sum_{i=1}^{m} (x_{ij} \cdot VR_i) / \tau(mem) \]

The physical resources in the data center are represented as \( RT = (r_1, r_2, r_3, r_4, ..., r_t, ..., r_m) \). \( r_i = \{ r_{i1}, r_{i2}, r_{i3}, ..., r_{im} \} \), where resource \( r_i \) is composed of the characteristic attributes of \( w \) different dimensions. Within service cycle \( L \) (the temporary fixed period is \( T \), where \( T > L > 0 \)), the available service time of \( r_i \) resources satisfies the characteristics of the Poisson distribution [14]; each resource is independent of others arbitrarily, that is, it satisfies \( \forall r_i \in RT, \cap_{i=0}^{m} r_i = \phi \). The available service vector in RT is expressed as \( SV = \{ sv_1, sv_2, ..., sv_n \} \), where \( sv_i (0 \leq i \leq m) \) is the available service rate of resource \( r_i \).

This paper assumes that user requirements need to be met by deploying the virtual machine set \( V = \{ v_1, v_2, v_3, ..., v_n \} \). \( V \) is a virtual machine matrix composed of N virtual matrix to meet the needs of customers. If at any moment, only one task of \( v_i \) of the virtual machine completes the task of \( t_j \), then \( K^{th} \) task \( \tau_j = \{ t_{jk}, t_{jk}, t_{jk}, ..., t_{jk} \} \) \( 0 \leq j \leq n \) of \( v_i \) of the virtual machine contains \( w \) characteristic attributes of different dimensions. The task generation rate vector of VM of the virtual machine is \( Q = \{ q_1, q_2, q_3, q_4, ..., q_j, ..., q_n \} \), \( 0 < j < n \), where \( q_j \) represents the task generation rate of \( v_j \) of the \( j^{th} \) virtual machine. Each task is independent of others, namely, \( v_i \in V, \cap_{i=0}^{j} t_j = \phi, \forall r_i \in R, t_j \in V \). If resource \( r_i \) can meet the requirements of task \( t_j \), then \( t_j \) can be assigned to \( r_i \) to perform the corresponding operations. In general, multiple resources \( r_i \) can satisfy task \( t_j \). However, in reality, task \( t_j \) is assigned to resource \( r_i \) with certain probability \( P_{ij} \) to realize an effective balanced allocation of resources.

The memory resource probability allocation matrix \( P_{m \times m} = (P_{ij})_{n \times m} \) and \( P_{ij} \geq 0 \), \( \sum_{i=0}^{n} P_{ij} = 1 \). The cloud platform resource-scheduling matrix is represented as \( M_{n \times m} = (M_{ij})_{n \times m} \). If \( v_j \) is assigned to resource node \( r_i \), then \( M_{ij} = 1 \); otherwise, \( M_{ij} = 0 \). The following conditions in Formula (4) must be met simultaneously in the entire process.

\[ \sum_{i=0}^{n} P_{ij} \cdot M_{ij} = 1 \]
\[ \sum_{i=0}^{n} P_{ij} \cdot M_{ij} \cdot t_j \leq r_i \]
\[ \sum_{i=0}^{n} M_{ij} \cdot t_j \leq r_i \]

The resources represented in the resource matrix are allocated to the required clients in a full probability manner. During the process of resource allocation, the allocated resources of any dimension cannot exceed the sum of physical resources. The sum of resources reflected in all encapsulated virtual machines should not exceed the total number of physical resources, generally no more than 75% of the physical resources; otherwise, the system performance will be affected (in accordance with Microsoft Report). This paper mainly focuses on two key indicators, CPU and memory, due to hardware limitations.

C. MODEL REPRESENTATION OF ENERGY CONSUMPTION

The classical representation method for energy consumption estimation is described in Literature [14], and the correlation between the power of physical nodes and the load
is expressed in detail. Specifically, the linear relationship between the power of physical nodes and the key factor (CPU) is expounded. Therefore, in this paper, we assume that the load and power of physical nodes follow a linear relationship [18]. In real virtual machine deployment, the energy consumption of the host in the data center is related to the workload and depends on the utilization of resources, such as CPU, memory, disk, and network card. Most studies have shown that the energy consumption of the host mainly comes from CPU and memory, which is highly positively correlated with the utilization of CPU and memory [15]–[17]. On this basis, the single-point energy consumption and total energy consumption of \( M_i \) host in the data center can be approximately expressed as Formula (5).

\[
\begin{align*}
EN_i &= \sum_{j=1}^{2} \int_0^t [\alpha \cdot EN_i(M_i(cpu_j)) + \beta \cdot EN_i(M_i(mem_j))] dt \\
EC &= \sum_{i=1}^{m} EN_i 
\end{align*}
\]

(5)

In Formula (5), \( EN_i \) represents the energy consumption of host \( M_i \) due to the changes in CPU utilization and memory, and \( EC \) represents the total energy consumption of the data center. \( \alpha \) and \( \beta \) are the correlation coefficient between memory and CPU. The total energy consumption \( EC \) of the data center, that is, \( \text{Min} (EC) \), should be minimized to save energy.

D. DESCRIPTION OF THE FRAMEWORK OF RESOURCE SUPPLY SERVICE

Under the condition of satisfying the abovementioned resource constraint conditions, the platform must meet the basic needs of users to improve the efficiency of service resources effectively. The dynamic framework is constantly changing in the process of dynamic change. Resource change also leads to service resource change. User requirements are constantly changing. The first phase requires dynamic collection and the collection of currently owned resources. Considering the main resource dimensions, this work regards CPU and memory as 2D indicators to measure.

On the basis of the above two dimensions, three aspects should be considered to conduct physical resource collection reasonably and effectively. Given that the physical resources in the cloud platform are an effective collection of comprehensive resources, the resources should be integrated in accordance with their types. However, user needs differ in the actual situation, and the user’s demand preferences should be classified. From the effective classification of resources and user’s demand preferences, a reasonable corresponding relation of resource demands is formed. The process of establishing an effective resource collection mechanism for a physical resource-planning model is described as follows:

First, the physical resource-planning center should be established to classify resources reasonably.

The system analyzes a resource complex composed of several physical nodes, which can decompose the categories of CPU and memory resources of each physical node and the corresponding specific number of resources, thus forming a unified and independent multicategory resource platform. The physical limitations of physical nodes are rapidly broken.

Second, customer demands should be analyzed and classified.

Each user requirement (ultimately can be translated into resource requirements) reflects the granularity of each user’s resource requirement, which effectively corresponds to the physical resource that is determined to exist. Therefore, limited physical resources should be combined with specific granularity needs to form an efficient resource correspondence relationship.

Third, the physical resource mapping should be unified.

The specific integrated process for resource mapping is shown in Figure 3 and Figure 4.

![Figure 3. Matching process for user requirement. This figure describes the main processes of resource classification and matching of user requirement resources with physical node resources.](image)

E. DOUBLE-CURSOR VIRTUAL MACHINE RESOURCE ALLOCATION ALGORITHM (DCVMRA)

1) THEORETICAL BASIS

This paper considers the influences of the dimensions of memory and CPU as the principal factors. The specific resource requirement that the user is asking for is a combination of CPU and memory. Assuming that the two are loop curve integrals, then CPU and memory are 2D independent variables randomly combined, as shown in Figure 5.

This process is equivalent to loop integrals based on 2D variables. If the value of \( Z \) is defined as the value of the loop integral function, then the values of C and M are derived in reverse, which are the effective values of 2D resources in the virtual resource scope.

The constraint conditions should meet Formula (6) and Formula (7).

\[
\int f(z) dz = 0
\]

(6)

That is,

\[
\begin{align*}
\int f(z) dz &= \int (u + iv)(dx + idy) \\
&= \int (udx - vdy) + i \int (vdx + udy)
\end{align*}
\]

(7)
FIGURE 4. Mapping of virtual machines in a cloud platform. The cloud platform conducts a reasonable mapping between requirements and resources in accordance with users’ requirement type and specific physical resource type. In the cloud platform, virtual demand queues will be formed in accordance with users’ requirements, which correspond to physical resources dynamically. After the user needs are met, the physical resources will be released, and the status of the physical resources will be dynamically updated in the cloud platform.

FIGURE 5. Schematic of loop integration. This paper deals with the calculation of the relationship between CPU and memory resources on the basis of a loop integration mechanism. This figure represents the loop integral calculation process of a 2D resource-to-resource mapping.

Formula (6) is the basic expression of a loop. Formula (7) is its expanded form, representing the integral form of CPU and memory resources on their respective dimensions and the interaction between U and V.

When two line integrals are added, Formula (8) can be obtained.

\[
\oint (Pdx + Qdy) = \iint (\partial Q/\partial x - \partial P/\partial y) dxdy
\]

\[
\oint (u dx - v dy) + i \oint (v dx + u dy)
\]

\[=
\oint (-\partial V/\partial x - \partial Q/\partial y) dxdy
\]

\[+ i \oint (\partial U/\partial x - \partial V/\partial y) dxdy\] (8)

Formula (8) represents the evolution process of Formula (7), indicates that P corresponds to CPU and Q corresponds to memory, and performs loop integral calculation on x and y.

In accordance with Formula (6), Formula (7), and Formula (8), Formula (9) can be obtained.

\[\partial V/\partial x - \partial Q/\partial y = 0\]

\[\partial U/\partial x - \partial V/\partial y = 0\]

\[\oint f(z)dz = 0\] (9)

Formula (9) indicates that if CPU and memory are in a stable equilibrium state, then the final result of loop integral should be zero. In accordance with this formula, the specific corresponding relationship between CPU and memory can be obtained, which serves as the variable correspondence rule.

On the basis of Green’s formula, in the closed interval, the unique value of resource index can be obtained on the integral curve after 2D resources are determined. This condition is equivalent to two dimensions being freely transformable cursors. Then, this method can be called double-cursor algorithm. Given the specific information on the loop integral, the relative matrix of CPU and memory can be obtained. Thus, an ordered resource allocation relation constraint is formed.

2) RESOURCE SUBTOTAL ALGORITHM

From the description in IV.E.1, a series of operations, such as resource collection and classification processing, is required to complete the effective pairing of resources. A typical resource collection classification algorithm (Algorithm 1) is shown as follows:

Algorithm 1: Typical Resource Collection Classification Algorithm

Input: Physical node number; determine Cpus number, type, and memory storage capacity in accordance with node number sequence.

Output: CPU computing power and memory capacity by category

(1). Initialize (pool,c,m,s (c),s(m));
/* Initialize the resource pool, c for CPU, m for memory, S (c) for the total number of Cpus, s(m) for the total amount of memory.*/
(2). Traverse(pn);
/* Traverse the set of physical nodes owned in the current state (number each physical node). pn denotes the set of physical nodes. */
(3). Sum(c,m);
/* Accumulate the sum of CPU and memory resources in accordance with the numbering order of physical nodes. */
(4). pool(c,m) = pair(CPU and memory);
chain-pointer(c,m) = pool(c,m);
/* Press the resource sequence into the resource pool once, and establish the chain pointer of each resource (in the form of singly or doubly linked list). */
(5). output(resource sequence);
/* Output the resource sequence corresponding to each pointer. Compare the optimized CPU–memory correspondence matrix. Thus, an ordered resource allocation relation constraint is formed. */

3) RESOURCE-MATCHING ALGORITHM

A typical resource matching algorithm (Algorithm 2) is shown as follows:

Algorithm 2: Resource Matching Algorithm
(1). Initialize (empty(Q(requirements)), c, m);
/* Create an empty user demand queue. */
(2). Q = requirements(c,m,u);
/* Join the queue in accordance with the actual order of user requirements. */
(3). p(x) = RAND();
p(y) = RAND();
(4). for every i(i = 1 and i = t)
candm read(c(ti), m(ti));
O = selection(c(ti), m(ti));
/* Read the user demand of ti moment from the user demand queue, and select the resource pointer from the resource pool in accordance with the user demand (ti moment is any moment during the service period), where candm represents an ordered pair of CPU and memory, and O represents the optimal order pair of CPU and memory resource sequence selected by the user at ti time. */
(5). calculate (c,m);
/* Assuming that c is the CPU resource and m is the memory resource, calculate the ordered pair set of c and m in combination with Formula (9), which meets the requirement of Formula (10). */

\[ \int f(c, m)dz = 0 \]  \hspace{1cm} (10)
(6). S = results(best,c,m);
empty(Q(requirements));
/* Return the ordered pair sequence of Step 5. On the basis of this sequence, virtual machine resource allocation and encapsulation are completed. */
(7). POOL = Q(requirements);
(8). empty(Q(requirements));
(9). S = Sort(POOL);
(10). Q = POOL[0];
(11). RETURN S, Q;

V. EXPERIMENTS

A. EXPERIMENTAL ENVIRONMENT

Our test environment includes a control node, a user database, and a local area network with 10 computing nodes connected by the network. All nodes are configured with an 8-core i7-8550-u CPU, an 8 GB memory, and a 1000 Mbps Ethernet card. All virtual machines are configured with three different types, having 6-, 4-, and 2-core CPUs and 8, 4, and 2 GB memories. The specific simulation configuration environment is Java environment (JDK 1.8), Eclipse SDK 4.5.0. The cloud platform adopts the simulation toolkit CloudSim4.0 [9]. CloudSim is a cloud-computing simulation platform launched by the University of Melbourne, Australia, which can effectively simulate resource consumption and other computing resources. This project mainly utilizes the following three key components of the simulation platform, namely, request generator, request allocator, and service node pool, which correspond to the classes Request_Generator, Dispatcher, and Server_Pool in CloudSim, respectively. Request_Generator and Dispatcher are subclasses of SimEntity. Server_Pool corresponds to a subclass of Power_Datacenter. The sequence of major event relationships is described in Figure 6.

The primary role of the class Request_Generator is to generate a personalized sequence of user requests on the basis of specific requirements, such as the frequency of user requests and resource type. The role of the class Dispatcher is to update the state of each physical node in real time, such as the number of requests on the current physical node, the number of requests that need to be released, and a series of information. The role of the class Server_Pool is to maintain the state of each physical node in the cloud platform properly during a specific experiment. The concrete interaction workflow of core components is presented in Figure 6.

The CloudSim cloud-computing environment [24] is used for simulation experiments to evaluate the algorithm performance. CloudSim is a discrete event simulator that supports virtual environment modeling and resource management. In this system, each PM is modeled as a double-core CPU with the same performance as HP ProLiant G4. The experimental platform is composed of a large-scale data center.
containing N heterogeneous physical nodes (the number of data centers is tentatively set at H). The evaluation attributes of each physical node I include two major factors: CPU and memory. CPU performance is measured in millions of instructions per second, and memory performance is measured on the basis of the amount of data stored per millisecond. In this experiment, we mainly monitor CPU and memory performance. In the experimental platform, the physical host is divided into 1800 and 2600 MIPS on the basis of the PERFORMANCE index of CPU. All hosts are simulated and deployed as double-core CPUs with memory capacities of 8, 4, and 2 GB. The details of the host parameter and virtual machine configurations for the CloudSim simulation platform are shown in Table 3 and Table 4, respectively.

**TABLE 3. Host parameter configuration.**

| Host            | CPU type | Core number | Memory |
|-----------------|----------|-------------|--------|
| HP ProLiant G4  | 15       | 4           | 4G     |
| HP ProLiant G5  | 17       | 8           | 8G     |

**TABLE 4. Virtual machine configuration.**

| Virtual machine instance | CPU(Hz) | Memory capacity |
|--------------------------|---------|-----------------|
| Basic type               | 2500    | 0.75            |
| Large instance           | 2000    | 3.85            |
| Small instance           | 1000    | 1.80            |
| Micro instance           | 500     | 0.63            |

In this experiment, the method of viewing Java program’s resource consumption (that is, reflecting a user’s resource demand) is mainly used to view the peak memory consumption (including virtual machine) and CPU consumption (ms) of Java program running, which reflects the host’s ability to cope with different user’s virtual machine requirements. The specific methods are as follows:

1. Download Jconsole.exe;
2. Run Jconsole.exe from the command line;
3. Select the virtual machine to be monitored in accordance with the displayed process number described in Table 5 (to view memory, thread, and other information).

In Table 5, kW stands for kilowatt. The first column denotes the CPU utilization, the second column denotes the HP G4 consumed power, and the third column denotes the HP G5 consumed power.

**TABLE 5. Comparison of utilization and power consumption.**

| CPU utilization (%) | HP G4 (kW) | HP G5 (kW) |
|---------------------|------------|------------|
| 0                   | 86         | 93.7       |
| 5                   | 87.3       | 95.1       |
| 10                  | 89.4       | 97         |
| 15                  | 91         | 99         |
| 20                  | 92.6       | 101        |
| 25                  | 94         | 103.4      |
| 30                  | 96         | 105        |
| 35                  | 97.2       | 107        |
| 40                  | 99.5       | 110        |
| 45                  | 101        | 114        |
| 50                  | 102        | 116        |
| 55                  | 104.2      | 119        |
| 60                  | 106        | 121        |
| 65                  | 107        | 123.4      |
| 70                  | 108        | 125        |
| 75                  | 110        | 126.8      |
| 80                  | 112        | 129        |
| 85                  | 113        | 131        |
| 90                  | 114        | 133        |
| 95                  | 115.6      | 134        |
| 100                 | 117        | 135        |

In this experiment, the method of viewing Java program’s resource consumption (that is, reflecting a user’s resource demand) is mainly used to view the peak memory consumption (including virtual machine) and CPU consumption (ms) of Java program running, which reflects the host’s ability to cope with different user’s virtual machine requirements. The specific methods are as follows:

1. COMPARISON OF ENERGY CONSUMPTION

With the continuous upgrading of host hardware facilities within the data center, such as multicore processors, large-capacity memory, and hard disks, the traditional CPU utilization–energy consumption model can no longer accurately describe the energy consumption. The real data provided by Standard Performance Evaluation Corporation are used to acquire an accurate value of energy consumption [7]. The power consumption of HP ProLiant G4 and G5 mainframes at different load levels is shown in Table 7. The consumption of different algorithms under fixed and similar tasks (physical nodes are all initially 80) is shown in the table, where T1–T5 represent different task numbers.
2) COMPARISON OF SERVICE LEVEL AGREEMENT (SLA) DEFAULT INDICATORS

In DCVMRA, the 2D cursors of CPU and memory are used to control the migration of virtual machine from the host with a large amount of tasks and avoid SLA violation by ensuring that the CPU and memory utilization of the host is lower than the resource threshold. Therefore, the default index of DCVMRA is superior to those of other algorithms. Figure 8 reflects the SLA default indicators of five different algorithms. Compared with other algorithms, DCVMRA has the lowest SLA default rate under the actual workload.

3) COMPARISON OF COMPREHENSIVE RESOURCE UTILIZATION

A similar number of virtual machine demands are randomly selected in a small range of users to compare the resource utilization of the five algorithms. The initial resource utilization is obtained by combining the actual number of physical nodes occupied and the ratio of total resource quantity. Each physical machine node is randomly selected, and the total amount of resources per physical node is different. The specific data given in this random experiment are shown in Table 9.

An intuitive data graph is provided in Figure 9 to analyze the resource utilization of the algorithm in Table 9 deeply. In terms of enabling the number of physical nodes, the number of physical nodes occupied by DCVMRA is basically the same as that occupied by the MEMETIC algorithm and is less than those of the FFD, BFD, and PABFD algorithms. The smaller the number of physical nodes is, the lower the energy consumption and the higher the resource utilization will be. The preceding analysis implies that DCVMRA can greatly reduce the number of physical servers that must be enabled and effectively improve the server cluster resource utilization. Figure 10 depicts that with the same number of virtual machine task requirements, DCVMRA requires the
TABLE 9. Performance comparison of five algorithms.

| Algorithm | Number of virtual machines | Number of physical machines | Resource utilization |
|-----------|----------------------------|-----------------------------|----------------------|
| FFD       | 29                         | 60                          |
| BFD       | 13                         | 63                          |
| PABFD     | 40                         | 67                          |
| MEMETIC   | 11                         | 70                          |
| DCVMRA    | 9                          | 73                          |
| FFD       | 30                         | 63                          |
| BFD       | 19                         | 66                          |
| PABFD     | 15                         | 69                          |
| MEMETIC   | 14                         | 73                          |
| DCVMRA    | 12                         | 75                          |
| FFD       | 35                         | 65                          |
| BFD       | 30                         | 68                          |
| PABFD     | 80                         | 73                          |
| MEMETIC   | 24                         | 74                          |
| DCVMRA    | 18                         | 76                          |
| FFD       | 40                         | 66                          |
| BFD       | 32                         | 70                          |
| PABFD     | 100                        | 77                          |
| MEMETIC   | 22                         | 80                          |
| DCVMRA    | 20                         | 81                          |
| FFD       | 50                         | 69                          |
| BFD       | 40                         | 73                          |
| PABFD     | 120                        | 79                          |
| MEMETIC   | 30                         | 82                          |
| DCVMRA    | 26                         | 83                          |

FIGURE 9. Comparison of resource utilization. The resource utilization of five algorithms under five different virtual machine tasks is compared.

lowest number of physical nodes to be turned on to achieve the goal of saving physical resources.

4) COMPARISON AND ANALYSIS OF VIRTUAL MACHINE MOBILITY

When redeploying virtual machines, DCVMRA prioritizes the hosts with lower resource utilization rate in the normal load than others, which ensures QoS at host running time from another perspective. Figure 4 demonstrates that DCVMRA effectively reduces the frequency of virtual machine migrations, thus reducing the effect of migration on QoS. Therefore, compared with other algorithms, DCVMRA has lower virtual machine mobility and reduces the risk of host overload. This algorithm also reduces the number of low-load hosts. The experimental results show that DCVMRA can make more reasonable virtual machine migration decisions and has better performance in reducing

FIGURE 10. Comparison of the number of physical nodes. The numbers of physical nodes to be turned on by the five algorithms in the case of five different virtual machine tasks are compared.

FIGURE 11. Comparison of the frequency of virtual machine migrations. The virtual machine migration of five algorithms is compared under the condition of five different virtual machine tasks.

TABLE 10. Virtual machine migration situation with random selection of fixed tasks.

| Algorithm | T1:200 (number of migration, mobility ratio) | T2:400 (number of migration, mobility ratio) | T3:600 (number of migration, mobility ratio) | T4:800 (number of migration, mobility ratio) | T5:1000 (number of migration, mobility ratio) |
|-----------|---------------------------------------------|---------------------------------------------|---------------------------------------------|---------------------------------------------|---------------------------------------------|
| FFD       | (2,10%) (2,17%)                             | (14,23%) (19,0%)                            | (22,95%) (19,0%)                            | (7,24%) (21,21%)                            |
| BFD       | (2,10%) (8,20%)                             | (15,25%) (17,6%)                            | (22,95%) (19,0%)                            | (20,24%) (21,21%)                           |
| PABFD     | (3,15%) (9,22%)                             | (12,28%) (17,6%)                            | (21,21%) (17,6%)                            | (21,21%) (17,6%)                            |
| MEMETIC   | (2,10%) (6,5%)                              | (11,18%) (16,0)                             | (20,0%) (14,0)                              | (14,14%) (14,14%)                           |
| DCVMRA    | (1,10%) (5,10%)                             | (8,13%) (14,0)                              | (17,5%) (14,0)                              | (14,14%) (14,14%)                           |
This paper presents a selection algorithm for virtual machine deployment based on a double-cursor control mechanism for the 2D resources of CPU and memory usage state, which achieves a biobjective optimization balance to a certain extent. First, a two-cursor algorithm is introduced to solve the problem of virtual machine deployment. Then, the DCVMRA proposed in this paper is compared with other relevant strategies. The simulation experimental results show that the proposed strategy can reduce the number of active servers in the data center, the energy consumption of data centers, and the amount of virtual machine migration. It can also realize multi-objective optimization. It has good performance in dealing with the problem of virtual machine deployment. However, with an increase in the number of virtual machine requests, the physical energy consumption exhibits a minimal rise.

In the future work, we will conduct enhanced optimization to reduce virtual machine mobility and physical consumption further.

VI. CONCLUSION
Aiming at the dynamically changing characteristics of user demand in a cloud platform and the effective integration of user personalized resource demand, a new virtual machine deployment strategy, that is, the double-cursor algorithm, is proposed in this paper. The proposed strategy is used to improve the resource utilization rate of cloud data centers, reduce the migration rate of virtual machines, and improve the energy consumption of physical resources.

The power consumption in the process of virtual machine deployment is the core factor of users’ concern. The energy consumption of the final physical resources resulting from deployment algorithms in different deployment frameworks varies greatly. In this paper, a resource classification and collection algorithm is first proposed, then a virtual machine deployment algorithm (DCVMRA) based on the 2D cursor mechanism is proposed. The algorithm is based on the classification of resources and user’s personalized needs. On the basis of different virtual machine loads (T1–T5), FFD, BFD, PABFD, and MEMETIC algorithms are compared with the DCVMRA algorithm proposed in this paper in terms of error rate, energy consumption, virtual machine mobility, failure rate, and resource utilization ratio. The results show that the algorithm proposed in this paper has relatively good advantages under the same conditions. In the next step, we will compare different types of load separately to find a better virtual machine deployment scheme.

the energy consumption of the data center than others. The number and mobility of virtual machines are determined during the experiment, as shown in Table 10. Figure 11 and Figure 12 show the comparison of the number and mobility of the virtual machines of the five algorithms. DCVMRA has good performance in controlling the frequency of virtual machine migrations. The root cause is the coordinated control of CPU and memory resources.

REFERENCES
[1] A. Beloglazov, J. Abawajy, and R. Buyya, Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing. Amsterdam, The Netherlands: Elsevier, 2012, pp. 20–22.
[2] L. Shuangli, L. Zhihua, and Y. Xinrong, “Virtual machine placement method based on multi-objective optimization,” J. Chongqing Univ. Posts Telecommun., vol. 32, no. 3, pp. 356–367, 2020.
[3] Z. Xiaoqing, “Research on optimization method of resource provision based on cloud computing environment,” Ph.D. dissertation, Dept. Elect. Eng., Wuhan Univ. Technol., Wuhan, China, 2013.
[4] X. Lijun and W. Fuqiang, “Optimization algorithm for cloud computing virtual machine deployment with balanced energy consumption and performance,” Lab. Res. Explor., vol. 37, no. 271, pp. 120–125, 2018.
[5] Z. Xiaoqing, “Optimal deployment strategy for high-efficiency cloud computing virtual machines,” Comput. Eng. Appl., vol. 52, no. 15, pp. 1–7, 2016.
[6] L. Jie, Z. Zhiming, G. Zhiqiang, and C. Chuan, “Research on resource allocation of cloud virtual machine based on benefit maximization,” Comput. Appl. Res., vol. 33, no. 10, pp. 2963–2966, 2016.
[7] H. Zhigang, X. Hui, and L. Keqin, “Virtual machine integration algorithm based on multi-objective optimization in cloud computing,” J. Hunan Univ., vol. 47, no. 2, pp. 116–124, 2020.
[8] J. Lijun, L. Zhao, X. Ao, and Y. Fangchun, “A virtual machine placement policy via biogeography-based optimization in the cloud,” in Proc. 12th Int. Conf. Ubiquitous Inf. Manage. Commun., 2018, pp. 54–55.
[9] R. Li, Q. Zheng, X. Li, and Z. Yan, “Multi-objective optimization for rebalancing virtual machine placement,” Future Gener. Comput. Syst., vol. 105, pp. 824–842, Apr. 2020.
[10] X. Xu, X. Zhang, M. Khan, W. Dou, S. Xue, and S. Yu, “A balanced virtual machine scheduling method for energy-performance trade-offs in cyber-physical cloud systems,” Future Gener. Comput. Syst., vol. 105, pp. 789–799, Apr. 2020.
[11] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, “A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems,” World Wide Web, vol. 23, no. 2, pp. 1275–1297, Mar. 2020.
[12] M. Hosseini Shirvani, A. M. Rahmani, and A. Sahafi, “A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: Taxonomy and challenges,” J. King Saud Univ.-Comput. Inf. Sci., vol. 32, no. 3, pp. 267–286, Mar. 2020.
[13] C. Wei, Z.-H. Hu, and Y.-G. Wang, “Exact algorithms for energy-efficient virtual machine placement in data centers,” Future Gener. Comput. Syst., vol. 106, pp. 77–91, May 2020.
[14] J. H. Y. G. Xiaoqing, “An intelligent scheduling algorithm for resource management of cloud platform,” Multimedia Tools Appl., vol. 24, no. 12, pp. 12–14, 2018.
[15] H. M. Naene, E. Zeinali, and A. T. Haghighat, “Adaptive Markov-based approach for dynamic virtual machine consolidation in cloud data centers with quality-of-service constraints,” Softw. Pract. Exper., vol. 50, no. 2, pp. 161–183, Feb. 2020.

[16] M. Zhang, H. Ren, and C. Xia, “A dynamic placement policy of virtual machine based on MOGA in cloud environment,” in Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. IEEE Int. Conf. Ubiquitous Comput. Commun. (ISPA/IUCC), Dec. 2017, pp. 885–891.

[17] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, and J. Li, “Virtual machine consolidated placement based on multi-objective biogeography-based optimization,” Future Gener. Comput. Syst., vol. 54, pp. 23–29, 2016.

[18] X. Zheng and Y. Xia, “Exploring mixed integer programming formulations for virtual machine placement with disk anti-colocation constraints,” Perform. Eval., vol. 135, pp. 56–57, Nov. 2019.

[19] S. Jin, S. Hao, and X. Qie, “A virtual machine scheduling strategy with a speed switch and a multi-sleep mode in cloud data centers,” J. Syst. Sci. Syst. Eng., vol. 28, no. 9, pp. 1–17, 2019.

[20] C. Mengjuan, C. Xingshu, and J. Xin, “Paging-measurement method for virtual machine process code based on hardware virtualization,” J. Comput. Appl., vol. 38, no. 2, pp. 305–309, 2018.

[21] W. Xiaodong and H. Jianjun, “Threshold-based energy-efficient VM scheduling in cloud datacenters,” (In Chinese), J. Huazhong Univ. Sci. Technol., vol. 46, no. 9, pp. 30–34, 2018.

[22] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, “An energy efficient ant colony system for virtual machine placement in cloud computing,” IEEE Trans. Evol. Comput., vol. 22, no. 1, pp. 113–128, Feb. 2018.

[23] Y. Han, J. Chan, T. Alpcan, and C. Leckie, “Using virtual machine allocation policies to defend against co-resident attacks in cloud computing,” IEEE Trans. Dependable Secure Comput., vol. 14, no. 1, pp. 95–108, Feb. 2017.

[24] D. Hao and L. Ye, “Research on optimal deployment of virtual machine based on multi-population genetic algorithm,” Control Eng., vol. 27, no. 2, pp. 335–341, 2020.

[25] T. Zhao, D. Xiao-Mei, C. Hong-Jian, M. Jing, and Y. Feng, “Multi-objective task scheduling algorithm based on reinforcement learning in cloud environment,” Mini Com Micro Comput. Syst., vol. 41, no. 2, pp. 285–290, 2020.

SHUKUN LIU (Member, IEEE) was born in Cangzhou, Hebei, China. He received the Ph.D. degree in computer science and technology from Central South University, Changsha, China, in 2016. From 2013 to 2019, he was an Associate Professor. Since 2020, he has been a Professor with the Department of Computer Science and Technology, Hunan Women’s University. His research is supported by the Hunan Provincial Natural Science Foundation and the Hunan Provincial Education Department Scientific Research Foundation. His major research interests include cloud computing, computer networks, virtualization technology, database technology, data mining, and software engineering. He has published more than 30 technical articles and books in the above areas. He is a member of CCF.

ZHIMIN LIU was born in Chenzhou, China. He received the B.E. degree from Northeastern University, in 2006, the M.E. degree from South China Agricultural University, in 2010, and the Ph.D. degree from Central South University in 2017. He is currently a Lecturer with the School of Mathematics and Computational Science, Hunan First Normal University. His current interests include wireless camera sensor networks, sensor cloud, and crowd sensing.

QIANG ZHANG received the B.Sc. degree from Central South University, Changsha, China, in 2011, the M.Sc. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2014, and the Ph.D. degree in computer science from Central South University, in 2019. He was a Visiting Ph.D. Student with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is currently a Lecturer with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. His research interests include privacy preserving, cloud computing, and information retrieval.

**CHAO LIANG LI** was born in Xiangtan, Hunan, China in 1972. He received the B.Sc. degree in applied mathematics from Huaihe Normal University, Huaibei, China, in 1996, and the M.Sc. degree in computer application technology and the Ph.D. degree in computer software and theory from Central South University, Changsha, China, in 2004 and 2015, respectively. From 2004 to 2006, he was a Lecturer. Since 2007, he has been an Assistant Professor with the Department of Computer Science and Technology, Hunan University of Business and Technology. His research is supported by the Hunan Provincial Natural Science Foundation and the Hunan Provincial Education Department Scientific Research Foundation. He is the author of one book and more than 20 articles. His research interests include artificial intelligence, big data, the Internet of Things (IoT), block chain, and privacy and security issues in the IoT. He has published more than 20 technical articles and books in the above areas.