Design and Implementation of Virtualization Cloud Computing System Intelligent Terminal Application Layer

Hongtao Ni and Lixia Yan*

Computing Science & Artificial Intelligence College, Suzhou City University, Suzhou 215104, China
E-mail: hongtao.ni@szcu.edu.cn; yanlixia99@163.com
**Corresponding Author*

Received 13 July 2024; Accepted 01 September 2024

Abstract

Cloud task scheduling has become a trend, and the shortcomings of traditional scheduling algorithms can be optimized through mathematical models of other cloud task scheduling scenarios. In order to improve the virtualization data processing effect of intelligent terminal application layer, this paper proposes an improved krill swarm optimization algorithm based on adaptive weight. The optimization of cluster load balancing and task average response time ratio are used to improve the convergence and accuracy of task scheduling algorithm. Moreover, this paper uses CloudSim simulation tool to conduct experiments to verify the effectiveness of the proposed model. In addition, this paper proposes an application-based virtualization method, which virtualizes the application programs inside the host machine into the virtualization software inside the virtual machine, so that the virtual machine can access it. Finally, this paper verifies the reliability of the proposed method with experiments, thus providing a theoretical reference for the subsequent design of intelligent terminal application layer virtualization cloud computing system. Compared with the traditional way of using physical hardware, using virtual machine hardware is more flexible, efficient and safe, which brings great convenience to the development and deployment of applications.

Keywords: Intelligent terminal, application layer, virtualization, cloud Computing.

1 Introduction

Once the concept of cloud computing was put forward, it has attracted the common attention of domestic and foreign research institutions for its characteristics of virtualization, high scalability and on-demand service. At the same time, the popularity of mobile terminals such as mobile phones and tablet computers and the development of mobile communication technology have also pushed cloud computing services to the high point of the times. The Internet has poured into a huge number of users, data and demand, and the demand for high-performance cloud computing capabilities and convenient and reliable cloud service models is imminent. First of all, the mainstream application model of the Internet has gradually developed from the Client/Server model to the Browser/Server model, which greatly reduces the development and operation costs, improves the distribution, and can provide fast and convenient services for cross-platform and cross-terminal users at the same time. With the gradual commercialization of the concept of cloud computing, cloud service providers integrate their resources to help users obtain cloud computing services such as application landing, service landing and security guarantee in the form of cloud computing platform [1].

With the emergence of cloud computing platform services, ordinary people can enjoy hundreds of millions of super computing power per second, which has greatly changed the lifestyle of the public, and subverted the traditional technology and business model, making the scale of the Internet expand rapidly and the amount of data explode. In order to cope with such huge computing demand and the growth of various new resource demand, cloud service providers are constantly increasing the number of infrastructure resources and expanding the scale of physical resource pools. However, the traditional resource management methods have gradually fallen behind, and the resource utilization rate of data centers has gradually decreased [2]. Under such an urgent background, how to reasonably and effectively allocate network resources has become a hot spot of common concern in academia and industry, and resource management has gradually become one of the core issues of cloud computing.

Different from central cloud computing, the terminal devices of edge cloud computing are usually embedded devices with limited hardware resources, such as sensors, cameras, vehicle electronic devices, etc. Moreover, its computing power is low, the physical environment in which the edge cloud is located is complex and diverse, and in many cases, the space, temperature, and power system cannot guarantee the normal state. However, at the same time, the edge side requires extremely high real-time and computational performance. The contradiction between the two makes it difficult for the traditional CPU architecture to meet the needs of edge cloud [3].

To improve the virtualization data processing efficiency of intelligent terminal application layer, this paper proposes an improved krill swarm optimization algorithm based on adaptive weights. Optimize the convergence and accuracy of task scheduling algorithms using two objective formulas: cluster load balancing and task average instruction response time ratio A virtualization software that virtualizes the internal applications of the host into virtual machines is proposed, which enables virtual machines to access the application based virtualization method. The reliability of this method is verified through experiments, providing theoretical reference for the design of future intelligent terminal application layer virtualization cloud computing systems. Compared with traditional physical hardware usage, using hardware in the form of virtual machines is more flexible, efficient, and secure, bringing great convenience to the development and deployment of applications.

The innovation of this article lies in the background and significance of cloud computing, as well as the application of deep learning algorithms. It proposes an adaptive inertia weight krill swarm algorithm and an improved particle swarm and krill swarm hybrid algorithm suitable for cloud computing task scheduling. Using hardware in the form of virtual machines is more flexible, efficient, and secure, bringing great convenience to the development and deployment of applications.

2 Related Work

At present, there are numerous virtualization solutions suitable for different scenarios in the industry. Commercial solutions include PikeOS from SYSGO, LynxSecure from LynuxWorks, and VxWorksMILSPlatform from WindRiver. The above scheme is based on the idea of separating the kernel to achieve virtualization and has a similar architecture. The idea of separating the kernel divides processor hardware resources into multiple tamper proof and non bypassable high guarantee partitions, hosting various GuestOS, runtime environment (RTE), and APIs, supporting applications of different security levels, critical levels, real-time or non real-time [4]; And set up strict control of information flow between partitions and peripheral devices to isolate the partitions; The virtual machine monitor (Hypervisor) does not include user models, shell access, dynamic memory, and device drivers, and these auxiliary tasks are pushed into the virtual machine. The separation of kernel architecture minimizes the code base of the Hypervisor, makes it lighter, and high assurance and tamper proof partitions improve system security. Although commercial solutions have an irreplaceable position in terms of performance, security, and lightweight, these solutions are not open-source, have limited relevant information, and are expensive, which brings inconvenience to users [5].

In a cloud environment, a physical storage device is virtualized into multiple logical devices and provided for use by multiple virtual machines. The increase in the number of logical devices will inevitably result in the storage I/O performance of each virtual machine being lower than the I/O performance when monopolizing physical devices in general operating systems. Therefore, optimizing storage performance has become one of the key issues in improving cloud platform performance [6]. In the current field of virtualization, CPU virtualization technology and memory virtualization technology have become quite mature, with relatively ideal performance. However, as one of the core virtualization technologies, I/O virtualization has relatively lagged behind in development, and virtualization I/O performance has also become a key factor affecting the performance of virtual machines in cloud platforms [7]. The experiment in reference [8] shows that the storage virtualization I/0 mechanism has become a performance bottleneck for cloud platform I/0 virtualization. In I/O virtualization based on the LiInlX kernel driver, most of the storage software overhead is caused by context switching, data replication, interrupts, resource synchronization, and other factors leading to the kernel I/O stack [9].

The emergence of cloud computing and other service methods has led to many projects being applied in the industry. However, in order to broaden its application value, task scheduling optimization algorithms should directly affect customers' feelings in cloud computing. The marketing and promotion of cloud computing is particularly important in a large number of fields, and it is the most crucial part of cloud computing. Many scholars have applied deep learning methods to cloud task scheduling. The exploration of cloud task scheduling has always been a research direction and network hotspot in both academia and industry. Task scheduling plays a crucial role in improving resource utilization in cloud computing big data centers, as it is a key technology for efficient operation [10]. Single objective and relational task scheduling are two parts of cloud computing task scheduling, and they usually choose different scheduling optimization algorithms due to the differences in overall scheduling objectives and the differences in dependencies between tasks [11]. The common algorithms in cloud computing scheduling are generally: (1) Traditional scheduling algorithms: Although these algorithms are relatively easy to learn, they may have disadvantages such as weaker characteristics. Traditional scheduling algorithms include min min algorithm, FIFO algorithm, max min algorithm, and rotation scheduling algorithm [12].

Heuristic scheduling algorithms: Genetic algorithms, simulated annealing algorithms, particle swarm optimization algorithms, and their improvements are the main types of heuristic scheduling algorithms. Because when planning a single task, it is necessary to consider different types of overall planning goals and then choose a more suitable planning strategy to achieve the best results.

A scheduling algorithm for load balancing: Reference [13] studied and analyzed the traditional common min min algorithm, which is an improved version of the min min algorithm. Reference [14] proposes to improve the min min algorithm, which effectively balances the load by adjusting the task according to the min min, and sequentially allocates a portion of the solution resources with the highest load to other solution resources. Reference [15] compared the min min algorithm with the max min algorithm and chose the min max algorithm. Reference [16] proposes to improve the updating of attractants using resource functions to achieve load balancing.

The production scheduling algorithm with time as the core: Reference [17] clearly proposes the dual adaptability gene genetic quenching algorithm. Reference [18] clearly proposes a production scheduling algorithm based on particle swarm optimization algorithm. Reference [19] improved the ant colony algorithm. Reference [20] proposed a new ant colony algorithm that employs diversity and enhancement strategies to prevent inappropriate acquisition of long path pheromones by ants.

Cost centered scheduling algorithm: Reference [21] selected the PSO algorithm to search for the optimal scheduling plan that minimizes costs, with the main goal of minimizing costs. Reference [22] developed an intelligent scheduling experiment for cloud computing network resources based on SLA, which reduces scheduling costs by applying spider algorithms to the cloud computing scheduling process.

A multi-objective based scheduling algorithm: Reference [23] proposed an improved ant colony task scheduling algorithm, which defines a new constraint function formula and improves the initial attractant upgrade method to obtain the objective constraint function formula to obtain the global optimal solution. Reference [24] proposed a constrained task scheduling algorithm that takes into account multiple dimensions such as cost, CPU, and network bandwidth. Reference [25] proposes a fast convergence scheduling algorithm that optimizes customer metrics while achieving system metric constraints, while also ensuring the efficiency and fairness of cloud resources.

Although traditional scheduling algorithms are relatively easy to learn, they may have drawbacks such as weak characteristics. This article aims to explore cloud computing systems and cluster algorithms, fully considering the performance, cost, and service level of system software, in order to generate a relatively complete scheduling entity model. We need to find a balance that balances user experience while also implementing a scheduling strategy for cloud computing system performance.

3 Cloud Computing Task Scheduling Algorithm

Virtualization technology is the foundation of cloud computing, which can abstract and unify the computer software environment (such as applications, file systems, operating systems) and various computer hardware resources (such as memory, networks, storage, CPU). Abstraction of computer resources can break down geographical barriers and achieve unified management and use of computer resources in different regions. Virtualization technology enables multiple virtual machines with different operating systems to run simultaneously on a physical computer, and each virtual machine's operating environment is isolated and independent.

As a kind of flexible method with strong global optimization ability, natural heuristic algorithm shows great potential in solving complex optimization problems. Krill swarm algorithm is a natural heuristic optimization algorithm based on the foraging behavior of krill colonies, which has been successfully applied in many fields. However, there are still some shortcomings in local search ability and parameter setting of krill swarm algorithm, which brings challenges to the application of the algorithm in cloud computing task scheduling problems. To overcome these challenges, this paper proposes an adaptive inertial weight-based krill swarm algorithm (AIWKH). If the fitness value of the particle decreases after an iteration, its inertial force weight will be reset to zero at the next iteration. Moreover, the convergence of task scheduling algorithm is improved from two perspectives: task average command response speed and cluster web service accuracy.

3.1 Improved Strategy of Krill Swarm Algorithm

The krill swarm algorithm is a natural heuristic optimization algorithm based on the foraging behavior of krill populations, which has good global optimization ability and convergence speed. The algorithm principles mainly include the foraging behavior of krill populations, information transmission mechanisms, and dynamic update strategies.

Natural heuristic algorithms, as a flexible and globally optimized method, have shown great potential in solving complex optimization problems. Particle swarm optimization algorithm and krill swarm optimization algorithm are two natural heuristic optimization algorithms based on group behavior, which have achieved significant results in their respective fields. However, these two algorithms still have certain shortcomings in terms of local search ability, global optimization ability, and parameter settings. In order to fully leverage the advantages of these two algorithms, this paper proposes a cloud computing task scheduling method based on an improved particle swarm and krill swarm hybrid algorithm.

The main parameters of krill swarm algorithm play an important role in the performance of the algorithm. After dealing with the optimization problem, it is important to select reasonable basic parameters to improve the performance of the algorithm. Among them, the linear descent strategy is used to select the step size scaling factor Ct in the algorithm, and the larger Ct in early iteration stage makes the algorithm explore more feasible areas.

$$Ct = Ct_{\max} - \left(1 - \frac{t}{t_{\max}}\right) Ct_{\min} \tag{1}$$

Among them, $Ct_{\max}$ and $Ct_{\min}$ are the largest and smaller step scaling factors, respectively.

If the fitness value of the i-th particle drops after an iteration, then its inertia weight will be reset to zero at the next iteration, that is, $w_n = 0, w_f = 0$. If the fitness value of the i-th particle gets better after the iteration, then its

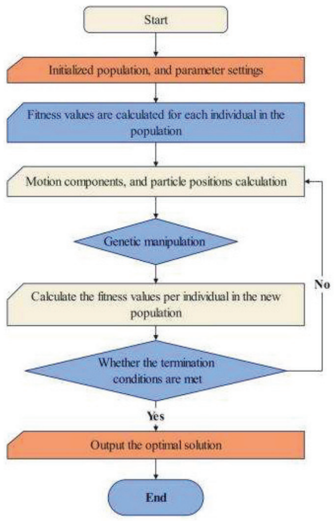

Start
Initialized population, and parameter settings
Fitness values are calculated for each individual in the population
Motion components, and particle positions calculation
Genetic manipulation
Calculate the fitness values per individual in the new population
Whether the termination conditions are met
No
Yes
Output the optimal solution
End

Figure 1 Flowchart of improved krill swarm algorithm.

inertia weight will not change at the next iteration.

$$w_{n,i}^{t} = \begin{cases} 0 & f(x_k^t) < f(x_k^{t-1}) \\ w_n & else \end{cases} \tag{2}$$

$$w_{f,i}^{t} = \begin{cases} 0 & f(x_k^t) < f(x_k^{t-1}) \\ w_f & else \end{cases} \tag{3}$$

Among them, f is the value of the fitness function, and the step size scaling factor Ct calculated by formula (4) is linearly reduced compared with that calculated by formula (1). Therefore, discrete system calculation can alleviate the downward trend of Ct and promote the convergence speed of particles in the subsequent.

$$Ct = Ct_{\max} - \left(1 - \frac{t}{t_{\max}}\right)^3 Ct_{\min} \tag{4}$$

Figure 1 is a flow chart of an improved krill swarm algorithm:

3.2 Scene Modeling and Optimization Objectives

In the development process of virtualization, performance issues are a concern that needs to be addressed in all application scenarios. Merely making

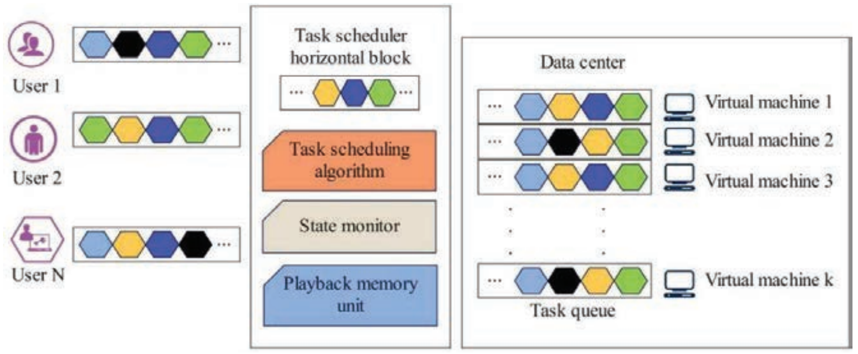


Figure 2 Schematic diagram of single data cloud task scheduler.

changes to virtual machine hardware cannot avoid the inherent performance loss of virtual machines in virtualization. The virtualization method using virtual machines usually requires a separate operating system, and this layer of operating system interface brings additional abstraction, which can result in multiple copies of the same data on the entire physical machine. This type of multiple copying is particularly significant in scenarios with a host operating system. Excluding multiple copies, the internal operating system and even the virtual machine hardware itself can cause the communication path between the programs inside the virtual machine and the outside world to become longer. Scene design and simulation can effectively analyze the actual effects of algorithms in various scenarios, which plays an important role in the subsequent algorithm time.

On the basis of cloud task scheduling environment, the performance of cloud computing scheduling is improved, and playback memory unit, state monitor and task scheduling algorithm are added to the task scheduler module. The planning process is shown in Figure 2. The information data between the task scheduler modules can be shared with each other, and the playback memory unit can store the historical information of task execution. Status monitor is the role of system monitoring, which is used to observe the surrounding environment, including task execution, CPU resource usage, and scheduling queue information.

(1) Task

The running instance of a program is a process, which is the basic entity of program execution. The appearance of a process improves the concurrency progress of the program and becomes the basic running unit of the program. Users submitting tasks to cloud service programs for processing is a process of enjoying cloud computing services. Each task can be expressed by the amount of data to be transmitted by the task, the instruction length of the program, CPU utilization and RAM utilization. During running, the frequency with which a task uses memory refers to the utilization rate of memory, and the CPU utilization rate reflected in running a task represents the computing power of the CPU.

(2) Virtual machine
Virtual machine (VM) refers to the virtual of computer software, which provides similar physics computer functions according to the simulation of software. Generally, virtual machines are divided into two types: one is system software virtual machine, and the other is program virtual machine. The running network resources required by the whole computer operating system are provided by the system software virtual machine. Program virtual machine is used to implement electronic computer programs in a mixed development environment. The high efficiency of virtual machine is measured by memory space, command running speed and its transmission speed.

The convergence and accuracy of task scheduling algorithm are the core indexes to judge the advantages and disadvantages of task scheduling algorithm. This chapter comprehensively analyzes and considers two optimization objectives as follows:

(1) Task average instruction response time ratio
The response time of the task job is defined as shown in formula (5):

$$reT_{ij} = taT_{ij} + waT_{ij} + exT_{ij} \tag{5}$$

In the above formula, taT_{ij} represents the data transmission time of job i from the task scheduler to the virtual machine in the data center, exT_{ij} represents the execution time of job i in the virtual machine j, and waT_{ij} represents the waiting time of job i in the virtual machine j.

The time taT_{ij} of data transmission is divided by the amount $data_i$ of data transmitted by the task by the network transmission speed from the scheduler to the virtual machine, and is expressed by $speed_j$, as shown in formula (6).

$$taT_{ij} = \frac{data_i}{speed_j} \tag{6}$$

The task sequence service scheduling algorithm, as the name implies, is executed in sequence according to the time sequence of task submission, as shown in formula (7):

$$waT_{ij} = \begin{cases} 0 & if\ taT_{ij} > \sum_{K=1}^{C} exT_{kj} \\ \sum_{K=1}^{C} exT_{kj} - taT_{ij} & if\ taT_{ij} \leq \sum_{K=1}^{C} exT_{kj} \end{cases} \tag{7}$$

Because there are only small differences in transfer rates between different memories, the execution time of a task depends on the instruction execution speed MIPS (million instructions per second), CPU utilization $CPU \times U_i$, and instruction length MI (million instructions):

$$exT_{ij} = \frac{mi_j}{CPU \times U_i \times mips_j} \tag{8}$$

$$mrR_{ij} = \frac{mi_j}{reT_j} \tag{9}$$

Combining formulas (5) to (9), the average instruction response time ratio of all tasks is:

$$avgMRR_{i,j} = E\left[\frac{mi_i}{\max\left(\frac{data_i}{speed_j}, \sum_{K=1}^{C} exT_{kj}\right) + \frac{mi_i}{CPI \times U_i \times mips_j}}\right] \tag{10}$$

$\forall i = 1, \ldots, I; \forall j = 1, \ldots, J$, I refers to the number of tasks and J refers to the number of virtual machines.

(2) Cluster load balancing

The standard deviation of CPU utilization is used to measure the degree of load balancing of the data center cluster. t is the execution time. The average CPU utilization of J virtual machines at time t is:

$$avgU_J^t = \frac{\sum_{j=1}^{J} CPU \times U_j^t}{J} \tag{11}$$

$CPU \times U_j^t$ refers to the CPU utilization of the j-th virtual machine at time t, and its standard deviation is:

$$stdU_J^t =^2 \sqrt{\frac{1}{J}\sum_{j=1}^{J}(CPU \times U_j^t - avgU_J^t)^2} \tag{12}$$

Combining formulas (11) and (12), it can be deduced that the average value of the standard deviation of CPU utilization is:

$$\begin{aligned} avgstdU_J^t &= E[stdU_J^t] \\ &= E\left[\sqrt[2]{\frac{1}{J}\sum_{j=1}^{J}(CPU \times U_j^t - avgU_J^t)^2}\right], T = 1, 2, \ldots, T \end{aligned} \tag{13}$$

(3) Total optimization objective
In order to better solve the multi-objective optimization problem, weight parameters are added, α is the weight of the average instruction response time ratio of the task, β is the weight of load balancing, CPU is the number of CPU cores participating in the operation, E is the expectation, π is the number convergence range, and the total optimization goal is:

$$\begin{aligned} \min_{\pi} \Bigg(\alpha E &\left[\frac{mi_i}{\max\left(\frac{data_i}{speed}, \sum_{K=1}^{C} exT_{kj}\right) + \frac{mi_i}{CPU \times U_i \times mips_j}} \right] \\ +\beta E &\left[\sqrt[2]{\frac{1}{J}\sum_{j=1}^{J}\left(CPU \times U_j^t - avgU_J^t\right)^2} \right] \Bigg) \end{aligned} \tag{14}$$

3.3 Application Virtualization Method

The performance issues of virtualization have now become the focus of virtualization research. Due to the increasing use of virtualization media such as virtual machines and containers in various environments, the performance overhead brought by virtualization has become an issue that cannot be ignored. And these costs are difficult to eliminate directly, so it is necessary to re evaluate the application program processes executed within the virtual machine, while ensuring the isolation of virtual machine virtualization, reducing redundant parts in program execution and communication, and improving the execution efficiency of the virtual machine.

Compared with type-1 virtualization, in type-2, the virtual machine manager does not run directly on the physical hardware, but runs inside an operating system running on the physical hardware. This system is called the host operating system (also known as the host machine). Programs running inside the host can be regarded as programs running on physical hardware.

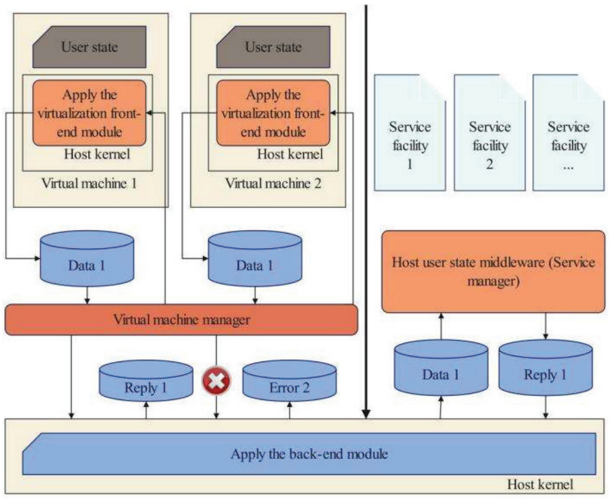


Figure 3 The overall architecture of application virtualization in type-2.

Therefore, the focus of application virtualization is to transfer the internal programs of virtual machines to the user space of the host machine, and at the same time, it is necessary to consider supporting common virtual machine management operations such as migration. This work is mainly based on Qemu Virtual Machine Manager, assisted by kernel bottom KVM module and paravirtualization framework vhost and virtio, and with the help of the above technologies, the internal programs of virtual machine are transferred to the host, and the virtual machine management operations such as migration and dynamic management are implemented.

The key point of applying virtualization framework is to provide a transparent, isolated from other programs and application-oriented data channel between virtual machine and host machine. The basic architecture of this work is shown in Figure 3. The framework is divided into four parts according to the affected areas: kernel component, host user mode component, virtual machine manager component and virtual machine user mode component.

The migration project using virtualization framework uses the same process for migration within this host and migration between different hosts: pre-migration process, migration process and post-migration recovery. Among them, the migration process is the same as that of the normal virtual machine to transfer data. After pre-migration, the process operation is the same, and the order is reversed. Figure 4 shows the detailed steps required for pre-migration. The pre-migration of application virtualization begins with

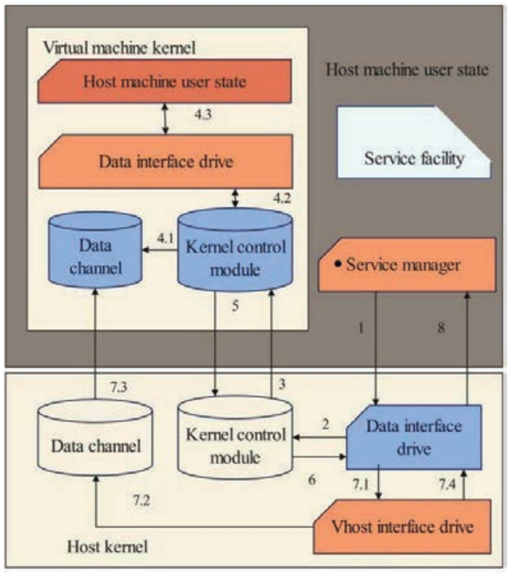


Figure 4 Pre-migration/post-migration process of application virtualization in type-2 virtualization.

the service manager, and the service manager will receive the migration signal from the virtual machine manager before the pre-migration.

Transport entities are organized in the framework into an index tree called an entity storage tree. Different from type-1, the entity cache tree does not exist globally in the host machine, but is stored in the virtual machine description structure. The entity cache tree also provides an optional lock for low-wait race control. Furthermore, each virtual machine descriptor contains two entity cache trees, and one is used to store data entities from the transmission cache and the other is used to store data entities from the receive cache. The virtual machine descriptor is similar in structure to the virtual machine description in type.1 and stores information related to virtual machines. In type.2, due to the necessity of fast indexing, virtual machine descriptors are stored using open-chain hash tables. This nested structure is shown in Figure 5. In the operation of application virtualization framework, the operation of transmission entities can be divided into three parts: insertion, deletion and query. All operations are performed inside the kernel where virtualization is applied.

There are two related operations of data packets: splitting and assembling. These operations are done in the kernel module where virtualization

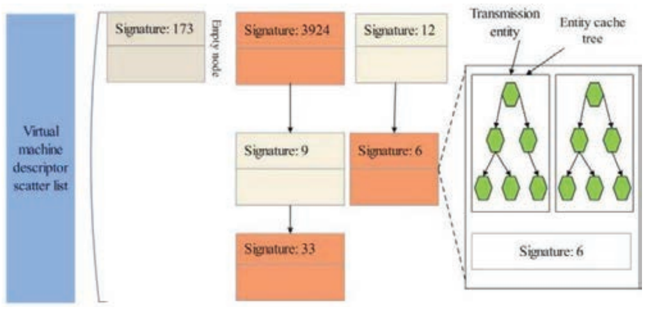


Figure 5 Organization of transport entities and virtual machine descriptors.

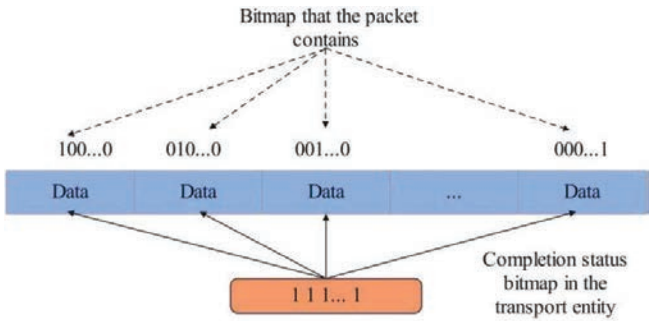

Bitmap that the packet contains
100...0
010...0
001...0
000...1
Data
Data
Data
...
Data
1 1 1... 1
Completion status bitmap in the transport entity

Figure 6 Relationship between completion bitmap in transmission entity and bitmap in data packet.

is applied. The payload data in the transmitted data packet originates from the transmitting entity, and finally needs to be restored to the payload of the transmitting entity. Therefore, for the processing operation of the data packet, it is necessary to provide an indication to provide the connection between the data packet and the related transmitting entity. This indication field is implemented using a bitmap, which can provide a computable and comparable packet identification function for different packets from the same transmitting entity. The bitmap in the transmission entity is referred to as the completion bitmap and is a subfield of the status field. This completion bitmap is a fixed-length bitmap containing the same number of valid bits as the packets split by the entity (Figure 6).

The service manager is the core part of the user mode component of the host. Unlike the service manager in type-1 virtualization, the service manager implemented in type-2 virtualization contains more functions. Figure 7 shows the main detailed design and operation flow of the service manager. The

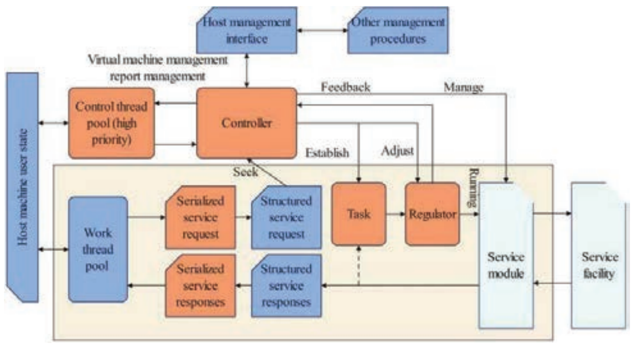

Host management interface
Other management procedures
Virtual machine management report management
Feedback
Manage
Host machine user state
Control thread pool (high priority)
Controller
Establish
Adjust
Seek
Running
Work thread pool
Serialized service request
Structured service request
Task
Regulator
Serialized service responses
Structured service responses
Service module
Service facility

Figure 7 Detailed design of service manager.

service manager consists of three parts: a thread pool, a controller, and a scheduler. The thread pool is responsible for allocating compute power while the service manager is running. There are two kinds of thread pools: a worker thread pool and a management thread pool. At the same time, each thread has different information processing capabilities. The dashed box in Figure 7 shows the flow of the service manager processing the virtual call request, in which the worker thread is responsible for unpacking the initial call data, and then calls the controller to find the corresponding service module and build the service task. Finally, the service task is handed over to the scheduler. At the same time, the worker thread is also responsible for wrapping the collected reply data after the service task is processed, and transfers it to the kernel.

4 Simulation Experiment

4.1 Experimental Environment

In the running environment of cloud computing, simulation tools can be used to create a controllable and repeatable computer environment, so that cloud computing users can conduct efficient tests in such an environment. CloudSim tool provides a simulation framework for computer systems without considering the details of related infrastructure and services. With the help of simulation tools, performance bottlenecks can be discovered in advance, and targeted adjustments can be made with the help of analysis of problems in advance. As shown in Figure 8, the SimJave layer can be used as the engine of discrete event simulation calculation, and is responsible for several core functions of the entire software, including queue creation and scheduling,

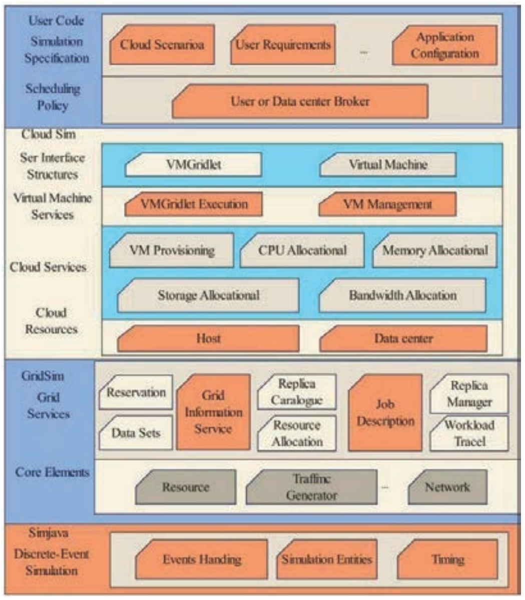


Figure 8 Cloud architecture.

communication between components, and creation of cloud system entities. The CloudSim simulation layer supports the simulation of cloud-based data centers, and cloud providers mainly design and run their scheduling strategies at this level to study the operational efficiency of different scheduling strategies. In the user code layer, users can set corresponding parameters according to the design needs.

CloudSim is the carrier of mapping between physical hosts and virtual machines, and supports the function of two-layer mapping. The working mode of CloudSim software is shown in Figure 9.

Among them, the main functions of CIS and DataCenter are to realize the interactive sharing of information, and VMScheduler class is mainly used to control virtual machines. The methods in DatacenterBroker class can be redesigned, that is, overloaded, and then the advantages and disadvantages of various scheduling strategies can be verified by simulation experiments.

The experimental environment configuration is shown in Table 1:

In order to avoid the situation that excessive search leads to difficult convergence, the number of iterations is limited in the experiment, which conforms to the finity of the algorithm. In this experiment, 150 is the maximum number of iterations of the algorithm [20].

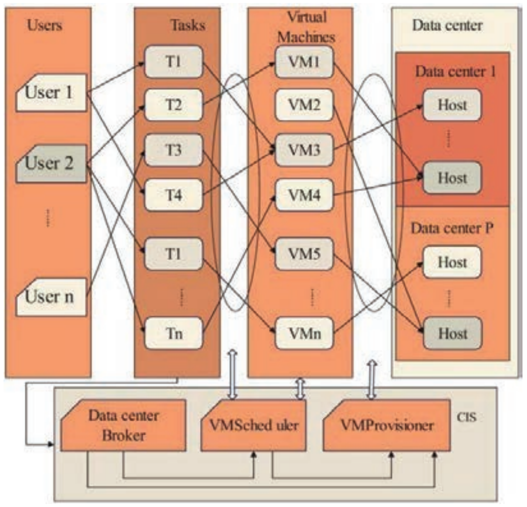


Figure 9 Working mode of CloudSim.

Table 1 Experimental environment

Name	Description
Operating system	windows11 ultimate 64-bit
Virtual machine	JDK1.9
Processor	Core i9-13900K
memory	32G
hard disk	1T
Development tools	Eclipse

4.2 Results

By using literature research methods, the commonly used cloud computing task scheduling algorithms in recent years are summarized. The more common algorithms are Ant Colony Optimization (ACO), Min Min algorithm, and QoSMin Min algorithm.

Ant colony algorithm (ACO), Min-Min algorithm and QoSMin-Min algorithm are used in the comparison experiment. Because these three algorithms are also commonly used algorithms for cloud computing task scheduling, these three algorithms have comparative reference value in performance.

After that, this paper compares the task execution time span. With the help of AIWKH algorithm and the running process of cloud computing scheduling, 8 computing nodes and 20 to 100 tasks are set in the experiment. In order to make the experimental results comparable, the differences including

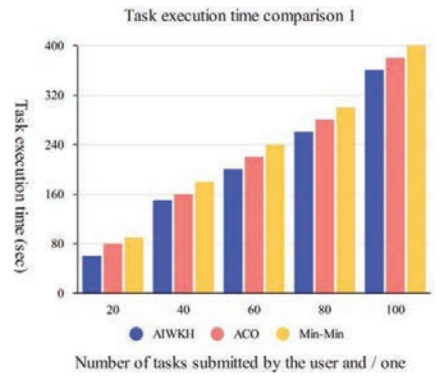


Figure 10 Comparison of execution time span.

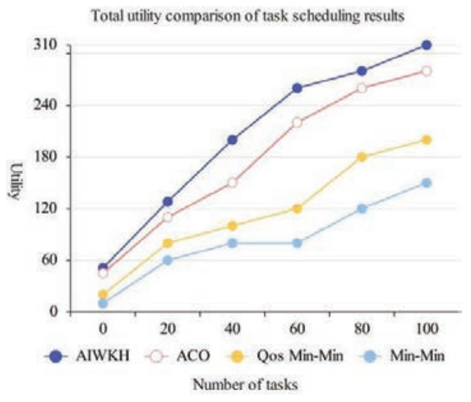


Figure 11 Comparison of total Qos utility values of users.

memory, CPU, unit time and network bandwidth are deliberately increased. With the help of three algorithms, the mean value is obtained after each run.

(1) Setting of Min-Min algorithm parameters: $\omega_1 = 0.6$, $\omega_2 = 0.2$, $\omega_3 = 0.2$;

(2) Setting of ant colony algorithm (ACO) parameters: $\alpha = 2.7$, $\beta = 1.0$, $\rho = 0.6$;

The task execution time span (unit seconds) results are shown in Figure 10.

In this paper, the QoSMin-Min algorithm is newly introduced as a reference in the comparative experiment. Setting of QoS utility evaluation parameters: $\omega_1 = 0.6, \omega_2 = 0.3, \omega_3 = 0.3$. The comparison of the scheduling results of the four algorithms is shown in Figure 11.

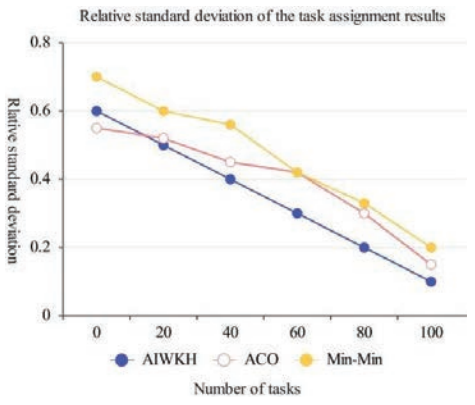


Figure 12 Relative standard deviation of task assignment results.

Next, this paper counts the task allocation on each virtual machine node. By calculating the relative standard deviation value, the advantages and disadvantages of the three algorithms in load balancing performance are visually shown, as shown in Figure 12.

In order to test the network throughput of the system, three experimental environments are set up in this section, which are non-virtualized environment, single virtual machine environment and dual virtual machine environment. This experiment tests the network throughput of the system when the host program sends data blocks of different sizes to OOS in three different environments. Among them, in the dual virtual machine environment, only the data of virtual machine 1 is observed to evaluate the performance of a single operating system in different experimental environments. In the experiment, multiple tests are carried out for data blocks of different sizes, and the average throughput is recorded. The test data are shown in Table 2.

Request response time is another important test index of network performance test, which reflects the system's ability to process network data. Test multiple times against blocks of different sizes, and record the average request response time. The test data are shown in Table 3.

4.3 Analysis and Discussion

It can be seen from Figure 10 that the task execution time span of AIWKH algorithm is the shortest, followed by ant colony algorithm, and the time span of Min-Min algorithm is the longest, and the difference in time span of the three algorithms increases as the number of tasks increases.

Table 2 Test results of throughput

Block Size (Bytes)	Non-virtualized Environment (Mb/s)	Single Virtual Machine Environment (Mb/s)	Dual Virtual Machine Environment (Mb/s)
64	30.31	7.39	1.91
128	34.66	7.77	2.07
256	36.30	8.18	2.71
512	37.04	8.84	4.00
1024	40.91	9.21	4.80
2048	48.90	10.07	6.15
4096	55.04	10.86	6.95
8192	58.37	11.99	9.23
16384	60.87	15.27	9.86

Table 3 Test results of request response time

Block Size (Bytes)	Non-virtualized Environment (us)	Single Virtual Machine Environment (us)	Dual Virtual Machine Environment (Mb/s)
64	95.88	181.15	245.59
128	100.36	192.50	266.60
256	102.76	203.14	282.58
512	111.67	209.75	295.69
1024	114.91	234.59	307.90
2048	118.75	605.52	694.03
4096	123.54	752.84	820.90
8192	128.69	847.40	997.89
16384	131.65	1,083.34	1,220.75

It can be seen from Figure 11 that the total utility of task scheduling of AIWKH algorithm is optimal under different number of tasks, followed by ACO algorithm, QoSMin-Min algorithm, and Min-Min algorithm is the worst, and the gap between the total utility of these four algorithms is increasing with the increase of the number of tasks.

It can be seen from Figure 12 that the standard deviation of the Min-Min algorithm is relatively large, while the ACO algorithm has uniform task distribution, better global search ability, and smaller standard deviation data. The AIWKH algorithm has smaller relative standard deviation and better global search ability under different number of tasks. To sum up, the cloud computing scheduling strategy of AIWKH algorithm has the ability to improve the resource load balancing of cloud system.

It can be seen from the test results in Table 2 that no matter how big or small the data block is, the throughput of the system in virtualized environment is lower than that in non-virtualized environment, and with the increase of the number of virtual machines, the throughput is even lower. The results show that the system performance is affected after virtualization, and the impact is further increased when the number of virtual machines increases.

It can be seen from the test results in Table 3 that the request response time in non-virtualized environment is shorter than that in virtualized environment, and as the number of data blocks increases, the request response time does not change much. However, in the virtualization environment, when the data block is larger than 1k, the request response time increases sharply, and compared with the single virtual machine environment, the request delay is more serious in the dual virtual machine environment.

In this paper, a parameter adjustment strategy and an adaptive inertial weight algorithm for krill swarm are proposed. Through these improved strategies, the local search ability and optimization efficiency of the algorithm are improved, and it is more suitable for solving cloud computing task scheduling problems. The scenario modeling of cloud task scheduling environment is constructed, and the optimization objectives are formulated. By establishing reasonable models and optimization objectives, the performance of the algorithm in practical applications can be better evaluated.

The network device full virtualization model implements virtualization of network devices based on instruction simulation, and every I/O operation instruction of the guest operating system gets trapped in program processing, which causes the system to frequently switch between virtual machines and programs and affects system performance. With the increase of the number of virtual machines, the switching overhead further increases, and the performance is further affected. The results show that the virtual system satisfies the spatiotemporal isolation, but its performance is affected. At the same time, when the number of virtual machines increases, the impact will increase, and the optimization should be improved in future work.

The design of a fully virtualized model for network devices and the implementation of virtual network card devices are correct. The system can run normally, and each virtual machine interacts with the real environment through the virtual machine to complete the operation of receiving and sending data. When there is more than one virtual machine in the system, they can reuse network devices in a time-sharing manner to meet the virtualization requirements of network devices.

5 Conclusion

Cloud task scheduling has become a trend, and the shortcomings of traditional scheduling algorithms can be optimized by other mathematical models of cloud task scheduling scenarios. Although krill swarm optimization is only suitable for solving some simple practical optimization problems, it can still be improved and applied. This study has a deeper understanding of the model structure of krill swarm algorithm, and also has some improvements to the algorithm. Moreover, this paper proposes application virtualization, and it is an application-based virtualization method that virtualizes the application programs inside the host machine into the virtual machine and the virtualization software makes the virtual machine accessible. In addition, this paper constructs the scenario modeling of cloud task scheduling environment with experiments, and formulates the optimization objectives. By establishing reasonable models and optimization objectives, the performance of the algorithm in practical applications can be better evaluated.

This article uses virtual machine online migration technology to achieve memory balancing between multiple hosts, but network interruptions or hard disk damage between hosts may cause virtual machine online migration to fail. Therefore, the next step will be to study the establishment of a host memory recycling technology similar to multi virtual machine balloon technology among multiple hosts. By uniformly allocating and using the idle memory recycling of the hosts, a wider range of memory resource scheduling will be achieved.

References

[1] Shukur, H., Zeebaree, S., Zebari, R., Zeebaree, D., Ahmed, O., and Salih, A. (2020). Cloud computing virtualization of resources allocation for distributed systems. *Journal of Applied Science and Technology Trends*, *1*(2), 98–105.

[2] Bhardwaj, A., and Krishna, C. R. (2021). Virtualization in cloud computing: Moving from hypervisor to containerization – a survey. *Arabian Journal for Science and Engineering*, *46*(9), 8585–8601.

[3] Katal, A., Dahiya, S., and Choudhury, T. (2023). Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing*, *26*(3), 1845–1875.

[4] El Kafhali, S., El Mir, I., and Hanini, M. (2022). Security threats, defense mechanisms, challenges, and future directions in cloud

computing. *Archives of Computational Methods in Engineering*, *29*(1), 223–246.

[5] Pallathadka, H., Sajja, G. S., Phasinam, K., Ritonga, M., Naved, M., Bansal, R., and Quiñonez-Choquecota, J. (2022). An investigation of various applications and related challenges in cloud computing. *Materials Today: Proceedings*, *51(3)*, 2245–2248.

[6] Vinoth, S., Vemula, H. L., Haralayya, B., Mamgain, P., Hasan, M. F., and Naved, M. (2022). Application of cloud computing in banking and e-commerce and related security threats. *Materials Today: Proceedings*, *51(2)*, 2172–2175.

[7] Zhang, P., Zhou, M., and Wang, X. (2020). An intelligent optimization method for optimal virtual machine allocation in cloud data centers. *IEEE Transactions on Automation Science and Engineering*, *17*(4), 1725–1735.

[8] Mughal, A. A. (2021). Cybersecurity Architecture for the Cloud: Protecting Network in a Virtual Environment. *International Journal of Intelligent Automation and Computing*, *4*(1), 35–48.

[9] Masdari, M., Gharehpasha, S., Ghobaei-Arani, M., and Ghasemi, V. (2020). Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. *Cluster Computing*, *23*(4), 2533–2563.

[10] Mishra, S. K., Sahoo, B., and Parida, P. P. (2020). Load balancing in cloud computing: a big picture. *Journal of King Saud University-Computer and Information Sciences*, *32*(2), 149–158.

[11] Bharany, S., Sharma, S., Khalaf, O. I., Abdulsahib, G. M., Al Humaimeedy, A. S., Aldhyani, T. H., ... and Alkahtani, H. (2022). A systematic survey on energy-efficient techniques in sustainable cloud computing. *Sustainability*, *14*(10), 6256–6264.

[12] Alam, T. (2020). Cloud Computing and its role in the Information Technology. *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, *1*(2), 108–115.

[13] Tang, X. (2021). Reliability-aware cost-efficient scientific workflows scheduling strategy on multi-cloud systems. *IEEE Transactions on Cloud Computing*, *10*(4), 2909–2919.

[14] Li, W., Wu, J., Cao, J., Chen, N., Zhang, Q., and Buyya, R. (2021). Blockchain-based trust management in cloud computing systems: a taxonomy, review and future directions. *Journal of Cloud Computing*, *10*(1), 35–44.

[15] Alsaidy, S. A., Abbood, A. D., and Sahib, M. A. (2022). Heuristic initialization of PSO task scheduling algorithm in cloud computing.

Journal of King Saud University-Computer and Information Sciences, *34*(6), 2370–2382.

[16] Ibrahim, I. M. (2021). Task scheduling algorithms in cloud computing: A review. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(4), 1041–1053.

[17] Rjoub, G., Bentahar, J., Abdel Wahab, O., and Saleh Bataineh, A. (2021). Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurrency and Computation: Practice and Experience*, *33*(23), e5919–e5930.

[18] Zheng, W., Muthu, B., and Kadry, S. N. (2021). Research on the design of analytical communication and information model for teaching resources with cloud-sharing platform. *Computer Applications in Engineering Education*, *29*(2), 359–369.

[19] Wei, W., Yang, R., Gu, H., Zhao, W., Chen, C., and Wan, S. (2021). Multi-objective optimization for resource allocation in vehicular cloud computing networks. *IEEE Transactions on Intelligent Transportation Systems*, *23*(12), 25536–25545.

[20] Shu, W., Cai, K., and Xiong, N. N. (2021). Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing. *Future Generation Computer Systems*, *124(1)*, 12–20.

[21] Ahmad, W., Rasool, A., Javed, A. R., Baker, T., and Jalil, Z. (2021). Cyber security in iot-based cloud computing: A comprehensive survey. *Electronics*, *11*(1), 16–25.

[22] Sriram, G. S. (2022). Edge computing vs. Cloud computing: an overview of big data challenges and opportunities for large enterprises. *International Research Journal of Modernization in Engineering Technology and Science*, *4*(1), 1331–1337.

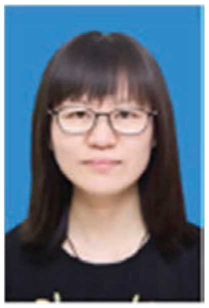
[23] Murad, S. A., Muzahid, A. J. M., Azmi, Z. R. M., Hoque, M. I., and Kowsher, M. (2022). A review on job scheduling technique in cloud computing and priority rule based intelligent framework. *Journal of King Saud University-Computer and Information Sciences*, *34*(6), 2309–2331.

[24] Masdari, M., and Zangakani, M. (2020). Green cloud computing using proactive virtual machine placement: challenges and issues. *Journal of Grid Computing*, *18*(4), 727–759.

[25] Hsieh, S. Y., Liu, C. S., Buyya, R., and Zomaya, A. Y. (2020). Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. *Journal of Parallel and Distributed Computing*, *139(1)*, 99–109.

Biographies

Hongtao Ni was born in Hebei, China, in 1982. From 2001 to 2005, he studied in North China Electric Power University and received his bachelor's degree in 2005. From 2006 to 2009, he studied in University of Science and Technology of China and received his Master's degree in 2009. Currently, he works in Suzhou City University. He has published three papers. His research interests are included Artificial intelligence, cloud computing.



Lixia Yan was born in Hebei, China,in 1980. From 2005 to 2007, she studied in Hebei Normal University and received her bachelor's degree in 2007. From 2010 to 2013, she studied in Soochow University and received her Master's degree in 2013. She has published a total of 7 papers. Her research interests are included Her research interests are included Computer applications, data mining.