This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3228402

An Energy-Efficient Dynamic Scheduling Method of Deadline-Constrained Workflows in a Cloud Environment

Guisheng Fan, Xingpeng Chen, Zengpeng Li, Huiqun Yu, Yingxue Zhang

***Abstract*—With the rapid development of cloud applications, the computing requests of cloud data centers have increased significantly, consuming a lot of energy, making cloud data centers unsustainable, which is very unfavorable from both the cloud provider's point of view and the environmental point of view. Therefore, it is crucial to minimize energy consumption and improve resource utilization while ensuring user service quality constraints. In this paper, we propose a hybrid workflow scheduling algorithm (Online Hybrid Dynamic Scheduling, OHDS), which aims to minimize the energy consumption of tasks and maximize service resource utilization while satisfying the sub-deadline and data dependency constraints of workflow tasks. Firstly, the data dependencies between workflow tasks are considered for multi-task merging, and sub-deadline constraints are assigned to workflow tasks based on task priority. Secondly, based on the independent nature of the tasks of different workflows, a hybrid scheduling of multiple workflows is performed to reduce service idle time. Then, the workflow task scheduling priority and its sub-deadlines are dynamically adjusted, and the service status is sensed by the CPU utilization of the service, and the workload on the overloaded/underloaded service is balanced by dynamic migration of virtual machines. Finally, the OHDS method is compared with three existing scheduling methods to verify its better performance in terms of scheduling energy consumption, scheduling success rate and service resource utilization.**

***Index Terms*—Task scheduling, deadline, energy consumption, resource utilization, cloud**

I. Introduction

WITH the rapid growth of cloud computing, cloud service providers continue to increase the size and number of services in cloud data centers, which also leads to large clusters of services that inevitably consume large amounts of power. Increasing energy costs and high energy consumption are in stark contrast to low cloud resource utilization, which is forcing cloud providers to continuously improve energy efficiency. Statistically, the average cloud service resource utilization is only 15% [1], while idle service resources dissipate more than 60% of their peak capacity [2]. Therefore, there are three reasons for the low service resource utilization: firstly, a workflow task cannot fully use a service resource, resulting in an idle service resource. Secondly, the data dependency between workflow tasks in cloud data centers inevitably leads to a large number of idle gaps in service resources, further reducing the utilization of service resources. Finally, peak workloads for cloud services are often several times the normal load, so over-provisioning of resources during off-peak hours is also inevitable.

Many researches often ignore that a single workflow task cannot fully use the host resources, and directly assign a single task and occupy a host, resulting in a serious decline in the utilization of host resources, and can not make good use of idle time slots of host resources. When multiple users compete for cloud hosting resources at the same time, workflow requests randomly arrive at the cloud hosting center under unpredictable conditions, and most research work tends to focus on one workflow request without considering the fairness among users. With the application of virtualization technology in cloud data centers, virtualization technology supports dividing a cloud host resource into several independent units (virtual machines) to perform multiple tasks at the same time, and dynamically migrates virtual machines to close host with low resource utilization. Considering that there is no data dependency between tasks of different workflows, the cross-execution of multi-workflow tasks makes better use of idle time slots of host resources, so as to improve resource utilization and reduce host energy consumption. Therefore, based on the independent characteristics of multi-workflow tasks, this paper uses multiple workflow hybrid scheduling to achieve host resource utilization, dynamically adjusts the scheduling priority and sub-deadline of workflow tasks to better meet time constraints, and migrates virtual machines on underloaded hosts to reduce power consumption.

In this work, we propose a hybrid workflow scheduling algorithm (OHDS) composed of two phases of task preprocessing and task scheduling to minimize the energy consumption of tasks and maximize service resource utilization. The main work of this paper is as follows:

(1) In the task preprocessing phase, we firstly combine multiple associated workflow tasks into a single task to reduce the data transmission overhead caused by the execution of associated tasks on different services during the task scheduling process. Secondly, we prioritize tasks by their latest start time and assign the initial sub-deadlines to tasks. Finally, a scheduling queue is generated based on task priorities for scheduling.

(2) In the task scheduling phase, we use workflow hybrid scheduling to reduce the idle time of virtual machines. We dynamically adjusted the unassigned tasks' direct subtask priority and subdeadlines with current scheduling information.

Guisheng Fan, Xingpeng Chen, Zengpeng Li, Huiqun Yu, and Yingxue Zhang are with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China (e-mail: gsfan@ecust.edu.cn. chxp2020@163.com. lizengpeng@126.com. yhq@ecust.edu.cn. 15154057373@163.com).

(Corresponding author: Guisheng Fan, Zengpeng Li, Huiqun Yu)

Furthermore, we migrate VMs on underloaded services to reduce service energy loss.

(3) In the experiments, this paper compares the OHDS method with existing methods (ESFS, REC-MCDM, and REEWS) using three datasets. The results show that the OHDS method has better performance in reducing execution energy consumption and improving resource utilization.

The rest of this paper is organized as follows: Section II reviews related work on workflow scheduling. Section III presents a model for the energy consumption optimization problem of workflow scheduling that satisfies time constraints. Section IV presents the optimization method. Section V is the experimental results and comparison. Section VI is the work summary and future work directions.

II. Related Work

The huge energy consumption of cloud data centers not only has a negative impact on the environment, but also leads to increased operating costs, so it cannot be ignored. In the past few years, many researchers have done a lot of researches on the problem of energy consumption optimization in cloud environment, and proposed many scheduling algorithms.

Safari et al. [3] proposed an energy consumption-aware scheduling algorithm based on dynamic voltage frequency scaling (DVFS) technology, which reduces the frequency of executing tasks by adjusting the voltage of the host, aiming to minimize the execution energy consumption and improve resource utilization while ensuring the task sub-deadline constraints. Choudhary et al. [4] proposed a power-aware virtual machine placement algorithm, which uses task clustering technology to combine tasks with short execution time and long execution time to reduce the number of tasks, thereby making the system scheduling costs are minimized and incorporating DVFS technology to reduce energy consumption. Stavrinides et al. [5] proposed a task scheduling strategy that considers energy consumption, cost, and Quality of Service (QoS) awareness. Tasks are sorted based on the earliest deadline priority strategy to generate a scheduling queue, and through dynamic voltage and frequency scaling techniques, as well as approximate computing. In order to save energy consumption and fill the task scheduling gap. Bhuiyan et al. [6] proposed an energy-aware scheduling algorithm to schedule multiple directed acyclic graph (DAG) tasks with deadline constraints on a multi-core platform. In order to further reduce energy consumption, the processors between tasks can be shared, light Loaded processors are turned off, and processors that meet the conditions are merged. Li et al. [7] proposed a cost- and energy-consumption-aware scheduling algorithm CEAS, which aims to minimize execution cost and reduce energy consumption under the premise of satisfying workflow time constraints. The algorithm uses task merging to reduce execution cost. And reuses idle virtual machines based on idle time recycling strategy to further reduce work energy consumption. Yuan et al. [8] proposes a simulated-annealing-based bees algorithm to provide fine-grained resource allocation and scheduling for tasks of heterogeneous applications. The algorithm aims to minimize energy costs, but it does not consider the problem of dependencies between tasks under workflow applications. Pietri et al. [9] adjusted the frequency for a given task to reduce the overall energy consumption while satisfying the deadline of tasks. The authors did not consider migrating virtual machines to reduce the static energy consumption of the hosts. Bugingo et al. [10] selected an appropriate CPU frequency for each VM to reduce energy costs. However, the authors ignored the DVFS technology that allows dynamic adjust CPU frequency. Garg et al. [11] proposed a reliability and energy efficient workflow scheduling algorithm (REEWS) which considers energy consumption and the application's reliability.

Saraswathi et al. [13] proposed an energy-aware scheduling algorithm EAWSTM based on task migration. Workflow tasks are independent of each other, and the optimal virtual machine is allocated to tasks according to power efficiency. After the tasks are merged, the tasks will be migrated to other optimal virtual machines for execution to save energy. Geng et al. [14] proposed a scheduling algorithm based on task repetition and task grouping in order to minimize workflow execution time, reduce energy consumption, and reduce execution cost. To reduce the data transfer overhead between tasks, the workflow tasks are divided into multiple groups, and combine multiple task groups to take advantage of idle time between tasks in a single task group to improve processor resource utilization. Garg et al. [15] proposed a workflow scheduling algorithm (ERES) that minimizes energy consumption, maximizes resource utilization, and minimizes workflow makespan under task deadlines and dependency constraints. Scheduling workflow tasks to virtual machines and dynamically deploy/undeploy virtual machines based on the needs of workflow tasks. This algorithm is based on a dual-threshold strategy for sensing the status of servers and balancing workloads on overloaded/underloaded servers by live migrating VMs. Mohammadzadeh et al. [16] based on the Antlion Optimization (ALO) algorithm and the Grasshopper Optimization Algorithm (GOA), a hybrid algorithm is proposed for multi-objective solving scheduling problems. The proposed algorithm improves the search performance by making a greedy strategy, using random numbers in a green cloud environment according to chaos theory, which aims to minimize the makespan, cost of performing tasks, energy consumption and increase throughput. Ahmad et al. [17] proposed an energy-efficient workflow scheduling algorithm that reduces energy consumption under user-specified budget constraints, this algorithm introduces a flexible mechanism to save energy consumption that incorporates energy and cost factor factors to fairly distribute the available budget for workflow tasks.

Based on the intelligent droplet algorithm and genetic algorithm, Kalra et al. [18] proposed a hybrid method for energy-aware scheduling of deadline-constrained workflows, which provides users with a non-dominated solution. In particular, it focuses on multiple goals of reducing program length, execution costs, and energy usage within user-specified deadlines. Toussi et al. [19] presented the DQWS workflow scheduling algorithm, which uses the divide-and-conquer method to minimize costs while satisfying the deadline constraint. Medara et al. [20] proposed an energy-efficient and reliability-

aware workflow task scheduling (EERS) algorithm in cloud environments, which aims to maximize energy savings and improve system reliability. First, the algorithm preserves task dependencies through a task-level calculation algorithm. Second, the communication cost is reduced by the task clustering algorithm, thereby reducing the energy consumption. Then, each task is assigned a sub-deadline through a sub-goal time distribution algorithm. Finally, through the cluster-VM mapping algorithm, the energy consumption is minimized and the system reliability is improved. Zhang et al. [21] proposed a mission-critical remapping algorithm (RMREC), which aims to reduce energy consumption. The algorithm is divided into two stages: In the first stage, the adjustable cost budget and the spare cost is determined according to the cost budget, the critical task path and the adjustable budget factor, and all workflow tasks are allocated to the virtual machine with the lowest energy consumption. In the second stage, key tasks are remapped according to the spares obtained in the first stage to reduce the energy consumption caused by task migration. Garg et al. [22] proposed a new scheduling algorithm that optimizes the reliability and energy consumption of applications and guarantees user-specified QoS constraints. The proposed algorithm works in four stages: task priority calculation, task clustering, deadline assignment, and assignment of clusters to processing elements with appropriate frequencies. Antolak et al. [23] presented a balancing heuristic for static task scheduling in a multi-verified time-based system architecture, with the main objective of minimising the energy consumed by the system without missing deadlines. Walia et al. [24] proposed a new Hybrid Scheduling Algorithm (HS) which is based on Genetic Algorithm (GA) and Flower Pollination Based Algorithm (FPA) for cloud environment to reduce task execution time, improve resource utilization and minimize energy consumption. Li et al. [25] proposed energy-aware task scheduling with deep reinforcement learning (DRL), a high-precision energy consumption model based on the real dataset SPECpower designed to facilitate environmental simulation, a partitioning-based heterogeneous resource proximal policy optimisation task scheduling algorithm based on real production situations, and autoencoders for processing high-dimensional spaces to accelerate DRL convergence. Bi et al. [26] proposes a dynamic hybrid metaheuristic algorithm based on simulated annealing and particle swarm optimization, which can guarantee the quality of service while minimizing the energy cost. The work packs applications with complementary multi-resource allocation requirements to improve performance and maximize profit, but they ignore migrating virtual machines to underload hosts to improve resource utilization further.

Considering the heterogeneous nature of cloud data center resources, Chen et al. [27] proposed an energy-efficient job scheduling algorithm that considers task dependencies in a cloud environment. The algorithm models energy consumption based on the frequency and number of cores of a virtual machine's CPU, and its main task is to divide each job into multiple tasks in a reasonable manner and schedule the tasks to the appropriate virtual machine to reduce energy consumption. Considering system response time and reliability, Hu et al. [28] propose an efficient heuristic algorithm aimed at solving the energy consumption optimisation problem for parallel workflows. The workflow execution time is first minimized while satisfying reliability constraints, and energy consumption is reduced by means of DVFS techniques and shutting down inefficient processors. The scheduling order of different workflow tasks affects the scheduling performance differently. Wang et al. [29] combine particle swarm optimisation and idle time slot awareness rules to propose a cloud workflow scheduling method that aims to minimize costs and improve resource utilization under workflow deadline constraints. Considering the endpoint communication contention when executing workflows, Wu et al. [30] propose a new usage scheduling model that introduces an awareness of communication contention during workflow scheduling to minimize workflow makespan. Efficient methods for scheduling cloud workflows are much needed. Wu et al. [31] propose a multi-objective evolutionary list scheduling (MOELS) algorithm that embeds classical list scheduling into a powerful multi-objective evolutionary algorithm (MOEA) designed to optimise the time horizon and economic cost of workflow scheduling. Different users often have different quality of service requirements. Li et al. [32] propose a Multi-swarm Co-evolution-based Hybrid Intelligent Optimization (MCHO) algorithm that minimizes the total makespan and cost while meeting the deadline constraints for each workflow.

In this work, we consider workflow scheduling with deadline-constrained and aim to reduce energy consumption while ensuring service quality constraints. We sense the service state by the CPU utilization of the service and balance the workload of services by dynamic migration of virtual machines to improve resource utilization.

III. System Model and Problem Formulation

In this section, we first discuss the models related to cloud workflow scheduling, mainly including the system model, task model and energy consumption model, and the parameters used in the models and their descriptions are summarised in Table 1. And the task scheduling problem is then formulated based on the dependencies between tasks.

A. System Model

The cloud data centers are composed of a large number of heterogeneous computing resources, namely $H = \{h_1, h_2, ..., h_n\}$, where n represents the number of hosts. For the host h_k can be represented by a tuple, that is, $h_k = < m_k, sc_k, n_k, P_k^{max}, f_k^{max}, VM_k >$, where m_k represents the host memory size, sc_k represents the storage capacity of the host, n_k represents the network bandwidth, P_k^{max} represents the maximum power of the service, f_k^{max} represents the maximum available frequency of the service, the host h_k can deploy multiple virtual machines, that is, $VM_k = \{vm_{k,1}, vm_{k,2}, ..., vm_{k,m}\}$. For a virtual machine $vm_{k,l}$ can be represented by a tuple $vm_{k,l} = < fk_{k,l}, mk_{k,l}, sk_{k,l} >$, where $fk_{k,l}$ represents the CPU frequency of the virtual machine, $mk_{k,l}$ represents the memory size of the virtual machine, and $sk_{k,l}$ represents the storage capacity of the virtual machine. Based on virtualization technology, all resources on a host can be shared by multiple

TABLE I
LIST OF SYMBOL

Abbreviation	Definition
h_k	host resources
m_k	host memory size
sc_k	host storage size
n_k	host network bandwidth
P_k^{max}	maximum host power
f_k^{max}	maximum host frequency
$vm_{k,l}$	virtual machines deployed on the host
$fk_{k,l}$	virtual machine CPU frequency
$mk_{k,l}$	virtual machine memory size
$sk_{k,l}$	virtual machine storage capacity size
$RT(vm_{k,l})$	virtual machine ready time
t_j^i	workflow task
st_j^i	workflow task start time
ft_j^i	workflow task end time
$tt_{p,j}^i$	workflow task data transfer time
ST_{vm}	virtual machine start time
ST_h	host start time
tw_j^i	workflow task computation load
et_j^i	workflow task execution time
$tr_{p,s}^i$	transfer load between workflow tasks
P_k	host execution power
r_k	host static power
y_k^t	host enable status
$EC(vm_{k,l})$	virtual machine energy consumption size
$EC(s_k)$	host energy consumption size
EC	cloud data center power consumption

virtual machines, and these virtual machines can be migrated between different hosts. In this way, the virtual machine on the underloaded host is migrated and the host is shut down, so as to improve resource utilization and reduce idle resources.

B. Task Model

Considering the host and virtual machine startup time, the start time, execution time, data transfer time, and workflow execution time of task t_j^i in workflow w_i are defined as follows:

(1) Task start time. In different cases, the start time st_j^i of the task t_j^i executed in the virtual machine $vm_{k,l}$ is defined as follows:

If task t_j^i and its parent task t_p^i are scheduled to execute on a virtual machine on the same host,

$$st_j^i = RT(vm_{k,l}) \tag{1}$$

among them, $RT(vm_{k,l})$ represents the virtual machine $vm_{k,l}$ ready execution time, which is the execution completion time for a virtual machine to process an existing task.

If task t_j^i and its parent task t_p^i are scheduled to execute on virtual machines on different hosts,

$$st_j^i = max(ft_i^p + tt_{p,j}^p, RT(vm_{k,l})) \tag{2}$$

If a new virtual machine is deployed on the host where the parent task t_p^i is located to execute the task t_j^i,

$$st_j^i = max(ft_i^p) + ST_{vm} \tag{3}$$

among them, ST_{vm} indicates the startup time of the virtual machine.

If a new virtual machine is deployed on a different host than the parent task t_p^i to execute task t_j^i,

$$st_j^i = max(ft_i^p) + ST_{vm} + tt_{p,j}^p \tag{4}$$

If deploying new hosts and new virtual machines to perform task t_j^i,

$$st_j^i = max(ft_i^p) + ST_h + ST_{vm} + tt_{p,j}^p \tag{5}$$

where ST_h indicates the startup time of the host. Therefore, the end time of task t_j^i can be defined as $ft_j^i = st_j^i + et_j^i$.

(2) Task execution time. Different virtual machines have different processing capabilities and the execution time of the task mainly depends on the performance of the assigned virtual machine. The execution time of the task t_j^i scheduled on the virtual machine $vm_{k,l}$ in the host h_k is defined as follows:

$$et_j^i = \frac{tw_j^i}{fk_{k,l}} \tag{6}$$

where tw_j^i indicates the computation load of task t_j^i.

(3) Data transfer time. If the parent task t_p^i and the child task t_s^i are scheduled to be executed on virtual machines of different hosts, data transfer time overhead will occur, and the transfer time between different virtual machines on the same host will be ignored. The data transfer time is defined as follows:

$$tt_{p,s}^i = \begin{cases} \dfrac{tr_{p,s}^i}{n_k} & \text{If } t_p^i \text{ and } t_s^i \text{ on different hosts} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

among them, $tr_{p,s}^i$ represents the data transmission load between task t_p^i and task t_s^i, and n_k represents the network bandwidth of the host where task t_p^i is located.

(4) Workflow execution time. The execution time of workflow w_i depends on the maximum end time in its workflow tasks and the workflow arrival time, which is defined as follows:

$$ET(w_i) = max(ft_j^i) - subTime_i \tag{8}$$

among them, $subTime_i$ indicates the arrival time of workflow w_i.

C. Energy Model

The energy consumption of a host h_k in the time period T can be expressed as $EC_k = P_k * T$, where P_k represents the power of the host h_k. The following equations for mainframe power and energy consumption are referenced from [33].

Knowing that the maximum power of the host h_k is P_k^{max}, the static power ratio is r_k, and the maximum CPU frequency is f_k^{max}, then the power of the host h_k can be formalized as:

$$P_k = r_k * P_k^{max} * y_k^t + \frac{(1 - r_k) * P_k^{max}}{(f_k^{max})^3} * (f_k)^3 \tag{9}$$

among them, y_k^t indicates whether host h_k is enabled at time t, and f_k is the CPU frequency of host h_k at time t. Among them, y_k^t and f_k change with time, and the rest are constants. The power of the host h_k at time t can be converted into the power generated by the virtual machine $vm_{k,l}$ deployed on the host h_k with a frequency of f_k at time t.

The start time $ST(vm_{k,l})$ and end time $FT(vm_{k,l})$ of the virtual machine $vm_{k,l}$ can be defined as:

$$ST(vm_{k,l}) = min(st_j^i) - ST_{vm} \tag{10}$$

$$FT(vm_{k,l}) = max(ft_j^i) \tag{11}$$

Therefore, the energy consumption generated by the virtual machine $vm_{k,l}$ during the use phase can be defined as:

$$\begin{aligned} EC(vm_{k,l}) &= \frac{(1-r_k) * P_k^{max}}{(f_k^{max})^3} * (f_k)^3 \\ &* (FT(vm_{k,l}) - ST(vm_{k,l})) \end{aligned} \tag{12}$$

For the host h_k, the start time $ST(h_k)$, the end time $FT(h_k)$, and the usage time $UT(h_k)$ can be defined as:

$$ST(h_k) = min(ST(vm_{k,l})) - ST_h \tag{13}$$

$$FT(h_k) = max(FT(vm_{k,l})) \tag{14}$$

$$UT(h_k) = FT(h_k) - ST(h_k) \tag{15}$$

Therefore, the energy consumption generated by the host h_k during the usage phase can be defined as:

$$EC(h_k) = r_k * P_k^{max} * UT(h_k) + \sum_{l=1}^{|VM_k|} EC(vm_{k,l}) \tag{16}$$

To sum up, the total energy consumption of the hosts in the cloud can be defined as:

$$EC = \sum_{k=1}^{n} EC(h_k) \tag{17}$$

D. Problem Formulation

As we all know, task scheduling is an NP-Complete problem. This paper aims to minimize execution energy consumption and maximize resource utilization under the premise of satisfying workflow time constraints. Therefore, the energy consumption optimization problem can be formally described as the following formula:

$$Min \sum_{k=1}^{n} EC(h_k) \tag{18}$$

In addition to energy consumption, this paper aims to maximize the resource utilization of the host, so the utilization of frequency when executing tasks is also an important optimization indicator:

$$Max(\sum_{i=1}^{m} \sum_{j=1}^{|T_i|} cpu_j^i * et_j^i) / \sum_{k=1}^{n} f_k^{max} * wt_k \tag{19}$$

among them, m represents the number of workflows in W, $|T_i|$ represents the number of workflow tasks, cpu_j^i represents the execution frequency of workflow tasks, et_j^i represents the execution time of workflow tasks, and n represents the number of cloud hosts, wt_k indicates the running time of the host.

(1) Decision variables

$x_{j,(k,l)}^i$ represents the mapping relationship between task t_j^i and virtual machine $vm_{k,l}$. $x_{j,(k,l)}^i$ is 1 if task t_j^i is scheduled to execute to virtual machine $vm_{k,l}$ in host h_k, and it is 0 otherwise.

$$x_{j,(k,l)}^i = \begin{cases} 1 & \text{If } t_j^i \text{ executes on } vm_{k,l} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

(2) Task dependency constraints

The dependencies between workflow tasks are constrained as follows:

$$ReadyTime \leq st_{s,(q,h)}^i \tag{21}$$

The parent and child tasks can be divided into five cases: parent and child tasks on the same host, parent and child tasks not on the same host, parent and child tasks on the same host and need to deploy a new virtual machine, parent and child tasks not on the same host and need to deploy a new virtual machine, and need to deploy a new host and virtual machine. $ReadyTime$ is the readiness time of the task t_p^i in the above five cases, and $st_{s,(q,h)}^i$ the start time of t_p^i's child task t_s^i on virtual machine $vm_{q,h}$.

(3) Deadline constraints

When all tasks in the workflow w_i are scheduled to be completed, the completion time of the workflow w_i is the maximum end time of the workflow task, that is:

$$ft_i = \max_{t_j^i \in T_i} \left\{ ft_{j,(k,l)}^i \right\} \tag{22}$$

where $ft_{j,(k,l)}^i$ is the end time of the task t_j^i on virtual machine $vm_{k,l}$.

Let d_i be the deadline of the workflow w_i. The workflow deadline constraints are as follows:

$$ft_i \leq d_i, \forall w_i \in D \tag{23}$$

(4) Host resource constraints

Cloud hosts include resources such as CPU and memory. Based on virtualization technology, all resources of a host can be shared by several virtual machines. Therefore, the following constraints need to be met in the allocation of host resources:

$$f_k^{max} - \sum_{l=1}^{|VM_k|} fk_{k,l} \geq 0, \forall h_k \in H \tag{24}$$

$$m_k - \sum_{l=1}^{|VM_k|} mk_{k,l} \geq 0, \forall h_k \in H \tag{25}$$

IV. Hybrid Scheduling Method

The workflow scheduling approach proposed in this chapter is divided into two main phases.

(1) Task pre-processing phase. Firstly, multiple associated workflow tasks are combined into a single task to reduce the data transfer overhead incurred by associated tasks executing on different services during task scheduling. Secondly, each workflow task is assigned a scheduling priority by the earliest

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3228402

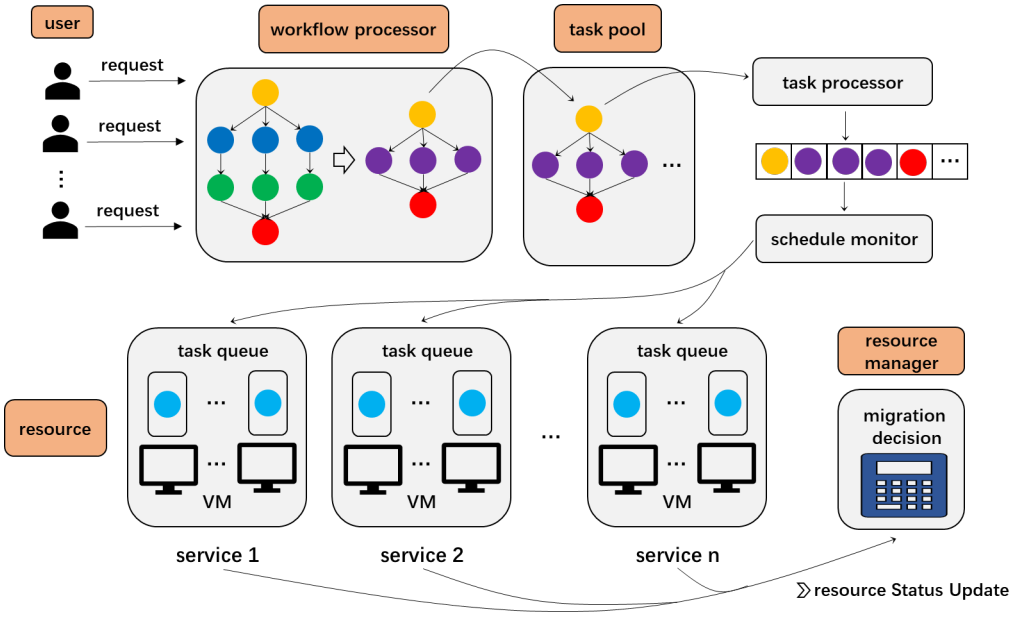

Fig. 1. The workflow scheduling architecture.

completion time of the task and an initial sub deadline is assigned to the workflow task based on the latest completion time of the task. Finally, a task queue is generated based on the scheduling priority to wait for resource allocation.

(2) Task scheduling phase. Workflow hybrid scheduling is used to reduce the idle time of VMs; for scheduled tasks, their direct sub-task priorities and the sub deadlines of subsequent unassigned tasks are dynamically adjusted by the current scheduling information; for each enabled service, the VMs on the underloaded service are dynamically migrated to reduce the service energy loss.

The proposed workflow scheduling architecture based on the above two phases is shown in Fig. 1. The scheduling architecture of the cloud data center is divided into three layers: the user layer, the scheduling layer and the resource layer. The user layer sends service requests to the application, and the scheduling layer processes the user requests based on the resource layer, i.e. the scheduling layer is the bridge between the user layer and the resource layer in the whole architecture. The resource layer in the cloud differs from traditional systems in that it consists of a service layer and a virtual machine layer. In addition, the available virtual machines in the virtual machine layer can be dynamically increased or decreased depending on the resources available for the service to which they belong.

The scheduling layer consists of five main components.

(1) A workflow processor processes workflows and merges directly and uniquely associated workflow parent and child tasks into a single task to reduce the data transfer overhead incurred when associated tasks are dispatched to different virtual machines during the scheduling process.

(2) The task pool stores all tasks awaiting scheduling that have been processed by the workflow processor.

(3) The task processor assigns sub deadlines to each workflow task and calculates the task priority to generate a task scheduling queue.

(4) The scheduling monitor generates workflow task VM-services mappings and dynamically adjusts workflow task priorities and subdeadlines based on scheduling information.

(5) The resource manager dynamically migrates VMs based on the resource utilization of the cloud service.

Algorithm 1 Task-merging (WF)

Input: Workflow w_i consisting of n tasks.
Output: Workflow $w_i^{'}$ after task merging is completed

```
Generate task queue Queue through entry task t^i_entry
for each task t^i_p in Queue do
    if t^i_p has only one child t^i_s then
        if t^i_s has only one parent t^i_p then
            Update the computation load tw^i_s of task t^i_s to tw^i_p + tw^i_s
            Update the immediate parent task set pred(t^i_s) of task t^i_s
            Update the child task set succ(t^i_p) of the immediate parent task of task t^i_p
        end if
    end if
    Remove t^i_p from Queue
end for
```

A. Task Merging

In order to reduce the data transmission overhead caused by the execution of associated tasks between different hosts during the workflow scheduling process, multiple associated workflow tasks are combined into a single task. This not only reduces the cost of data transmission between associated tasks, but also reduces the time overhead incurred when cyclically monitoring scheduling information. The workflow task merging conditions are as follows: task t_p^i is the direct

Algorithm 2 Priority-Adjustment(task)

Input: currently scheduled task t_j^i
Output: task scheduling sequence $Queue$

```
list_child = a collection of direct child tasks of task t_j^i;
queue_unsched = unscheduled task queue after task t_j^i
for each task t_u^i in queue_unsched do
    if list_child contains task t_u^i then
        Add the task t_u^i to the list_prior;
    end if
end for
collection_prior = list_prior the collection of scheduling sequences;
for each sequence q_prior in collection_prior do
    if the sequence q_prior satisfies the task's child deadlines and dependencies then
        Calculate the energy consumption of the current scheduling sequence q_prior;
    end if
    Choose a sequence q_prior with the least energy consumption;
end for
Use the sequence q_prior to update the task scheduling sequence Queue;
return Queue;
```

parent task of task t_s^i, when task t_p^i has the only direct child task t_s^i, and task t_s^i has the only direct parent task t_p^i, Task t_p^i and task t_s^i are combined into task t_{p+s}^i. When task t_p^i and task t_s^i are assigned to different hosts for execution, data transfer costs will be incurred, and task merging will avoid this cost. Algorithm 1 specifies the specific steps of task merging.

B. Task Priority Assignment

Task pre-processing stage. The earliest end time of a workflow task is ranked as the initial priority of the task to generate the scheduling queue, where the average processing power of the virtual machine is used to calculate the execution time of the workflow task, taking into account the data transfer between tasks. Thus the earliest end time of a task, $eft(t_j^i)$, is defined as follows.

$$eft(t_j^i) = \begin{cases} et_j^i, & \text{if } t_j^i \text{ is an entry task} \\ et_j^i + \max\left\{eft(t_p^i) + tt_{p,j}^i\right\}, & \text{otherwise} \end{cases} \tag{26}$$

among them, et_j^i represents the execution time of task t_j^i, $tt_{p,j}^i$ represents the data transmission time between task t_j^i and the parent task, $pred(t_j^i)$ is the immediate parent task set of task t_j^i. Therefore, the earliest start time of task t_j^i can be defined as $est(t_j^i) = eft(t_j^i) - et_j^i$.

Task scheduling phase. Dynamically adjusting the scheduling priority of workflow tasks. Based on the current scheduling task, the set of its direct subtasks is obtained, and the direct subtasks in the order of the current scheduling queue are selected for priority adjustment based on the priority order of the current scheduling task, and the dynamic priority adjustment pseudo-code is shown in Algorithm 2.

Algorithm 3 Sub-Deadline-Initialization(WF)

Input: workflow w_i consisting of n tasks.
Output: workflow $w_i^{'}$ after task sub-deadline initialization completes

```
Build a task list based on the workflow topology list_p;
for each task t_j^i in list_p do
    Use formula (4-5) to calculate the earliest end time eft(t_j^i) of task t_j^i;
end for
Build a task list based on the workflow inverse topology structure list_q;
for each task t_j^i in list_q do
    Use formula (4-6) to calculate the latest end time lft(t_j^i) of task t_j^i;
    Use formula (4-7) to calculate the sub-deadline dd_j^i of task t_j^i;
end for
return workflow w_i';
```

C. Task Deadline Distribution

As with task priority allocation, task deadlines are allocated in two stages.

Task preprocessing stage. This paper calculates the latest end time of a task, $lst(t_j^i)$, based on the earliest completion time of the workflow task and considers the data transfer delay between tasks, which is defined as follows:

$$lft(t_j^i) = \begin{cases} eft(et_{exit}^i), \text{if } t_j^i \text{ is an export task} \\ \max\left\{lft(t_s^i) + et_s^i - tt_{j,s}^i\right\}, \text{ otherwise} \end{cases} \tag{27}$$

among them, $succ(t_j^i)$ represents the direct subtask set of task t_j^i, and the latest start time of task t_j^i can be defined as $lst(t_j^i) = lft(t_j^i) - et_j^i$. Therefore, it is possible to calculate its child deadlines for workflow tasks in reverse topological order starting from the exit task, and the deadline dd_j^i for each task is defined as follows:

$$dd_j^i = \frac{lft(t_j^i)}{eft(et_{exit}^i)} * deadline^i \tag{28}$$

The relevant pseudocode is shown in Algorithm 3:

Task scheduling stage. After the current task has been scheduled, it is adjusted for unscheduled tasks based on their initialized subdeadlines. Unlike the initialized sub-deadlines in the task pre-processing phase, the execution time and data transfer time of scheduled tasks are calculated according to their actual allocated services and virtual machines.

D. Task Scheduling

The main process of workflow task scheduling is shown in Algorithm 4. First, based on the dependencies between workflow tasks, multiple tasks are merged into a single task, and task priorities and sub-deadlines are initialized; then based on task priorities, task queues are initialized. Finally, perform scheduling based on the task queue, perform task migration based on the resource utilization of the currently used host before task scheduling, and dynamically adjust the priority and

Algorithm 4 Task-Scheduling(WF)

Input: workflow w_i consisting of n tasks.
Output: task resource mapping $< Resource, Allocation >$

while new workflow w_i arrives **do**
  Task-Merging(w_i);
  for each task t_j^i in w_i **do**
    Use formula (26) to calculate the earliest end time $eft(t_j^i)$ of task t_j^i;
  end for
  Task sub-deadline initialization $\rightarrow$ Sub-Deadline-Initialization(w_i);
  for each task t_j^i in scheduling queue $queue_s$ **do**
    Task migration $\rightarrow$Task-Migration(Resource);
    $\{s_k, vm_{k,l}\}$ = Cheapest-Energy-Consumption(t_j^i);
    Update the start time $st^i_{j,(k,l)}$ of task t_j^i and end time $ft^i_{j,(k,l)}$;
    if collection $list_{alloc}$ is not empty **then**
      t_p^i = the latest scheduled task assigned by the virtual machine $vm_{k,l}$;
      for task t_c^j in candidate queue $queue_c$ **do**
        if task t_c^j satisfies dependency constraints and $max\{ft^i_{p,(k,l)}, st^j_{c,(k,l)}\} + et^j_{c,(k,l)} \leq st^i_{j,(k,l)}$ **then**
          Update the start time $st^j_{c,(k,l)}$ of task t_c^j and end time $ft^j_{c,(k,l)}$;
        end if
      end for
    end if
    Adjust the priority of direct subtasks $\rightarrow$ Priority-Adjustment(t_j^i);
    Adjust the sub-deadline of unscheduled tasks $\rightarrow$ Sub-Deadline-Initialization(w_i);
  end for
  Update scheduling queue $queue_s$ and candidate queue $queue_c$;
end while
return $< Resource, Allocation >$;

Algorithm 5 Cheapest-Energy-Consumption(task)

Input: currently scheduled task t_j^i
Output: hosts and virtual machines $\{s_k$,$vm_{k,l}\}$ assigned to task t_j^i

$list_{vm}$ = the set of virtual machines in the used resource pool;
for virtual machine $vm_{k,l}$ in $list_{vm}$ **do**
  Select $\{h_k, vm_{k,l}\}$ with the smallest energy consumption for task t_j^i;
end for
$list_s$ = the set of hosts in the used resource pool;
for virtual machine $vm_{k,l}$ in virtual machine type list **do**
  if $ft^i_{p,l} \leq dd^i_j$ **then**
    for each host h_k in $list_s$ **do**
      Select $\{h_k, vm_{k,l}\}$ with the least energy consumption for task t_j^i;
    end for
    for host h_k in host type list **do**
      Select $\{h_k, vm_{k,l}\}$ with the least energy consumption for task t_j^i;
    end for
  end if
end for
if $\{s_k, vm_{k,l}\}$ is empty **then**
  for virtual machine $vm_{k,l}$ in $list_{vm}$ **do**
    Select $\{h_k, vm_{k,l}\}$ with the smallest execution end time of task t_j^i;
  end for
  for virtual machine $vm_{k,l}$ in virtual machine type list **do**
    if task t_j^i executes in the virtual machine vm_l with the minimum end time **then**
      for each host h_k in $list_s$ **do**
        Select $\{h_k, vm_{k,l}\}$ with the least energy consumption of task t_j^i;
      end for
      for each host h_k in host type list **do**
        Select $\{h_k, vm_{k,l}\}$ with the least energy consumption of task t_j^i;
      end for
    end if
  end for
end if
return $\{s_k, vm_{k,l}\}$;

sub-deadline of unscheduled tasks according to the scheduling information after task scheduling.

It is worth noting that for a single workflow, due to the data dependencies between different tasks, during the execution of the workflow, the virtual machine has a large number of idle time periods, resulting in idle waste of host resources. However, there is no data dependency restriction between different workflow tasks, so different workflows are executed in the same virtual machine in an appropriate order, and the idle time period of the virtual machine will be reduced or even eliminated.

Assuming that there are tasks t_p^i and t_s^i in the workflow w_i that are executed on the virtual machine $vm_{k,l}$, the end time of task t_p^i is less than the start time of task t_s^i, that is, $ft^i_{p,(k,l)} < st^i_{s,(k,l)}$. There is a workflow w_j, and the start time of its task t_h^j executed on the virtual machine $vm_{k,l}$ is less than the start time of the task t_s^i, that is, $st^j_{h,(k,l)} < st^i_{s,(k,l)}$. If $max\{ft^i_{p,(k,l)}, st^j_{h,(k,l)}\} + et^j_{h,(k,l)} \leq st^i_{s,(k,l)}$, that is, the end time of task t_h^j after task t_p^i is less than or equal to the start time of task t_s^i, then task t_h^j is allocated to the virtual machine $vm_{k,l}$ and executed after task t_p^i is completed, which effectively reduces the idle time between task t_p^i and task t_s^i. Therefore, in the workflow scheduling process, this paper adopts the multi-workflow hybrid scheduling method, which aims to reduce the idle time of virtual machines, improve the utilization of host resources, and reduce the energy consumption of scheduling.

In the multi-workflow hybrid scheduling process, when a new workflow arrives, its tasks are preprocessed and stored

in the task pool. The task pool is divided into two parts: the scheduling queue $queue_s$ and the candidate queue $queue_c$. The unscheduled tasks of the currently scheduled workflow are stored in the scheduling queue, and the unscheduled tasks that arrive later in the workflow are stored in the candidate queue, that is, the scheduling queue task is the main scheduling object, and the candidate queue is mainly used to fill the idle time of the virtual machine. When scheduling a scheduling queue task, first update the used service resources and make a VM migration decision based on its resource utilization, see Algorithm 6 for the related process; then select a service and VM with the lowest energy consumption for the current task, see Algorithm 5 for the related process; finally update the service resources and task information. When the scheduling queue task scheduling is completed, first determine whether the idle time of the virtual machine allocated by the scheduling task satisfies the task filling constraint, and if it does, select a suitable task from the candidate queue for idle time filling, and update the task resource mapping result. After each round of task scheduling is completed, the direct sub-task priority of the current task and the sub-deadlines of all unscheduled tasks are adjusted based on the scheduling information. When all tasks in the scheduling queue have been scheduled, the first arriving workflow is selected from the candidate queue, its tasks are stored in the scheduling queue and the candidate queue is updated, and the above scheduling steps are repeated until all workflow tasks are scheduled.

Algorithm 5 aims to select the resource map with the least energy consumption to execute for the current task. First, by selecting the used host and virtual machine, selecting the used host and creating a new virtual machine, and creating a new host and virtual machine, select the host and virtual machine that satisfy the task sub-deadline constraints and execute the least energy consumption. Secondly, if a suitable host and virtual machine are not obtained, the host and virtual machine with the smallest task end time are selected according to the above three methods, and if the end time is the same, the selection is made according to the energy consumption of task execution. Finally, update the host and virtual machine status and return.

E. Task Migration

The integration of host resources in cloud data centers is an NP-hard problem. This paper dynamically migrates workflow tasks to target hosts and virtual machines based on virtualization technology to improve host CPU utilization, and shuts down idle hosts to save host resources and reduce energy consumption.

Taking the host CPU utilization as the main decision-making factor, calculate the currently used CPU utilization of each host and the overall average CPU utilization, and divide the currently used host resources into normal hosts and underloaded hosts. Under the premise of resource constraints, migrating workflow tasks assigned to underloaded hosts to normal hosts and virtual machines and shutting down underloaded hosts, so as to improve the CPU utilization of the normal host

Algorithm 6 Task-Migration(Resource)

Input: the currently used host resource $Resource$
Output: host resource after resource adjustment $Resource^{'}$

```
list_s = set of used hosts;
for each host h_k in list_s do
    Use formula (29) to calculate the CPU resource utilization of host h_k;
end for
Use formula (30) to calculate the overall average CPU resource utilization;
list_u = used underload host collection;
list_n = set of used normal hosts;
for each host h_u in list_u do
    list_vm = the set of virtual machines in host h_k;
    for each virtual machine vm_{k,l} in list_vm do
        for each host h_n in list_n do
            if the available frequency of host h_n is greater than or equal to the frequency of virtual machine vm_{k,l} then
                Migrate the virtual machine vm_{k,l} to the host h_n after the current task is executed;
            end if
        end for
    end for
    Shut down the host h_u;
end for
return Resource';
```

and reduce the idleness of the resources of the underloaded host. Host CPU utilization is defined as follows:

$$RU_k = \frac{\sum_{l=1}^{|VM_k|} fk_{k,l}}{f_k^{max}} \tag{29}$$

among them, $fk_{k,l}$ is the CPU frequency of the virtual machine $vm_{k,l}$, and f_k^{max} is the maximum CPU frequency of the host h_k, so the average CPU resource utilization is defined as follows:

$$avg_{RU} = \frac{\sum_{k=1}^{n} RU_k}{n} \tag{30}$$

It is worth noting that VM migration decisions are made before new tasks are scheduled. Traverse the virtual machine list of the underload host to make migration decisions. Before the virtual machine is migrated, it needs to complete its assigned tasks and then migrate to the new host. When the migration of all the virtual machines of the underload host is completed, it needs to be timely off to avoid its static energy consumption.The task migration decision is made based on the above CPU resource utilization, and the specific process is shown in Algorithm 6.

The algorithm time complexity analysis is based on a workflow consisting of n tasks. The OHDS algorithm whose maximum dependency of workflow tasks is $n(n-1)/2$, so the time complexity of the deadline assignment phase is $O(n^2)$, the time complexity of task sorting based on priority is $O(nlogn)$. And the upper limit of the number of candidate VMs in the task scheduling phase is $n+v$, where v is the

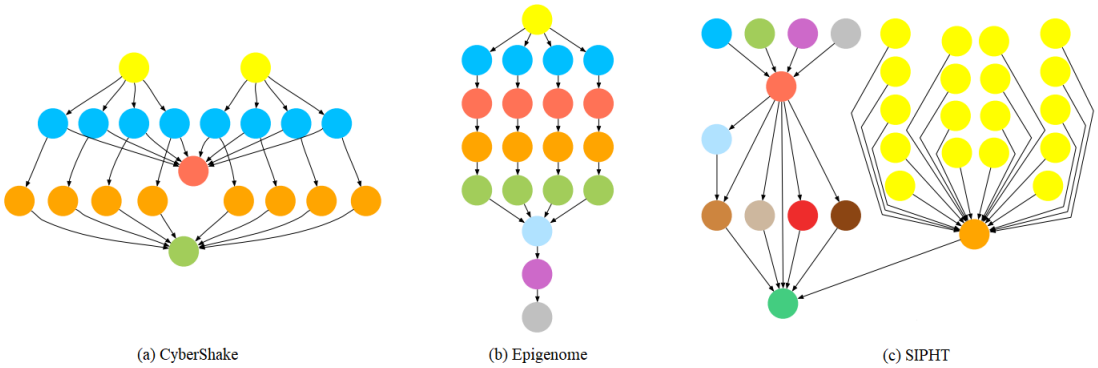

Fig. 2. The structure of scientific workflows.

set of VM type set, and the time complexity of traversing the candidate VMs during task round-robin scheduling is $O(n(n+v))$. The time complexity of the task scheduling phase is $O(n^3(n+s))$ considering task subdeadlines and scheduling priority adjustment, and the need to perform VM migration operations before new tasks are scheduled. Thus the complexity of the OHDS algorithm time is $O(n^3(n+s))$.

V. Evaluation

In this paper, the proposed OHDS method is compared experimentally with the ESFS algorithm, the REC-MCDM algorithm and the REEWS algorithm. The main principles and steps of the other three scheduling methods are shown below.

(1) Energy-aware Stepwise Frequency Scaling (ESFS). The ESFS algorithm [9] references the HEFT algorithm to construct task priorities based on the critical path length of each workflow task to the egress task. Running at the maximum host frequency to assign each task the host with the earliest completion time. The execution frequency of each task is then gradually adjusted, aiming to reduce the energy consumption of the hosts as much as possible while satisfying the workflow deadline constraint.

(2) Runtime-Energy-Cost Multi Criteria Decision Making (REC-MCDM). The REC-MCDM algorithm [10] constructs an initial task resource mapping based on the HEFT algorithm through the maximum VM frequency. The execution time and energy consumption of each VM with different processing power are then calculated and the highest performing VM frequency is returned for each task by weighted comparison.

(3) Reliability and Energy Efficient Workflow Scheduling (REEWS). The REEWS algorithm [11] is mainly applied to priority-constrained applications in the cloud and consists of four main phases. Firstly, task priority is calculated based on workflow topology ordering; secondly, workflow tasks are divided into different task clusters to minimize the communication cost between tasks; then QoS constraints are assigned based on user-defined workflow deadlines; and finally, virtual machines of appropriate frequency are assigned to task clusters to minimize energy consumption, etc.

Even though the algorithm in [19] is recently proposed, it tackles a different optimization problem, that is, minimizing the execution cost subject to the deadline constraint for a single workflow. By contrast, the identified problem of this paper aims to minimize energy consumption and maximize resource utilization while satisfying the deadline constraints for multiple workflows. Additionally, the algorithm in [19] always selects the cheapest computing resources while meeting the deadline, neglecting the optimization of energy consumption and resource utilization. Considering the root difference in problem definition between them, the algorithm in [19] cannot be applied directly for comparison.

A. Experimental Settings

Three widely used workflows (i.e. CyberShake, Epigenome and SIPHT), each with different data and computational characteristics, have been chosen for this paper. (a) The CyberShake workflow is used by the Southern California Earthquake Center to characterize the severity of earthquakes in an area by generating earthquake distribution maps. The workflow is characterized as data-intensive and has a high demand on service memory and CPU resources. (b) The Epigenome workflow is used to automate various genomic testing operations and its workflow is characterized as CPU-intensive. (c) The SIPHT workflow is used to automate searches of the National Center for Biotechnology Information database for sRNA-encoding genes. Its workflow is characterised as CPU-intensive. Detailed information on the above workflows can be found and studied in the literature [12]. These three workflows cover all the basic features (one-to-one, one-to-many, many-to-one and many-to-many), with 50, 200, 400 and 600 task sizes selected for each workflow respectively.

In addition, 10 types of real service hosts are selected to simulate service clusters in the cloud data center and an upper limit of 1000 resources is assumed for each service, where the main parameters of the 10 services are shown in Table 2. Five types of virtual machine templates are used in the experiments in this paper, with the CPU frequency requirement of the templates ranging from 200 to 1000 in steps of 200. The average network bandwidth between the different services is 1.0Gpbs and the energy consumption for transferring 1GB of data is 2.3W. The start-up time and shutdown time of the hosts and virtual machines are 96.9s and 30.0s respectively.

TABLE II
HOST CONFIGURATION PARAMETERS

Name	RAM	Max Fre	Static EC	Max EC
PowerEdge R630	64	2.3	51.2	287
Fujitsu TX2560	64	2.3	40.0	264
RH2288H V2	48	2.4	68.7	137
Altos R380 F2	24	2.2	71.5	316
PowerEdge R720	24	2.2	52.7	250
Gateway GT350	12	3.0	79.5	264
IBM x3650 M3	16	3.0	56.1	218
Acer R380 F1	16	2.4	88.1	197
Xserce3,1	18	2.9	173	334
Proliabr DL160	16	2.5	148	233

B. Metrics

Each scheduling algorithm has a scheduling solution that does not meet the workflow deadline, so the scheduling success rate is used to represent how well each scheduling algorithm meets the deadline constraint. Considering that longer idle time of a service will lead to an increase in overall static energy consumption, the energy loss of the service is further illustrated by comparing the deployment of virtual machines on enabled services.

In addition to energy consumption, resource utilization is also an important optimisation metric, so the concept of resource utilization is introduced, using the ratio of the frequency of VMs required to perform workflow tasks during workflow scheduling to the maximum frequency of enabled services to represent current resource utilization. In summary, this paper evaluates the effectiveness of each scheduling algorithm on the workflow scheduling problem by comparing the energy consumption, scheduling success rate and service resource utilization of each algorithm.

C. Results

The performance of the four algorithms in terms of execution energy and resource utilization is evaluated by increasing the time relaxation factor λ from 0.01 to 0.05 in steps of 0.01.

In this paper, we compare the workflow execution energy consumption of the four algorithms with different time relaxation factors, and the results are shown in Fig.3. It can be found that the workflow execution energy consumption of the four scheduling algorithms tends to be stable overall in the three different workflow datasets, regardless of the variation of the time relaxation factor. In addition, the OHDS method consumes less energy than the other three scheduling algorithms in all three workflow experiments.

In the workflow scheduling process, the OHDS algorithm performs multi-task merging based on the dependency relationship between workflow tasks, effectively reducing the energy consumption incurred when enabling services to wait for task execution due to data transfer between tasks. In addition, the hybrid scheduling of multiple mutually independent workflow tasks effectively reduces the idle time of the virtual machine, improves execution efficiency and reduces the overall energy consumption of the service.

The concept of scheduling success rate is introduced in this paper to provide a more comprehensive representation of the effectiveness of scheduling decisions. Fig.4 shows the success rates of the task scheduling decisions for the above four methods for three different types of workflow sets, as a more detailed representation of the specific situation of meeting workflow deadlines. It can be seen from Figure 4 that the OHDS method proposed in this paper maintains a high scheduling success rate regardless of the time relaxation factor, while the ESFS, REC-MCDM and REEWS algorithms gradually increase their scheduling success rates as the deadlines are relaxed.

In this paper, the earliest and latest completion times of workflow tasks are used as the basis for dividing the deadlines. In the task scheduling process, the resources that meet the task sub-deadlines are selected first, otherwise the service resource with the lowest task completion time is selected to meet the deadline constraint of the workflow. And when the task scheduling is completed, the priority and sub-deadlines of subsequent unscheduled tasks are dynamically adjusted according to the current task resource mapping information to assign the best scheduling order and constraints to subsequent tasks.

In this paper, the service resource utilization in the workflow scheduling phase is represented by the ratio of the sum of the total required frequency of the workflow tasks and the maximum frequency of the service activation phase during the workflow task execution phase, as shown in Fig.5. As the time slack factor becomes larger, the post-tasks on the same VM have more time to wait for the predecessor tasks to complete, eliminating the need to create more new service instances to execute workflow tasks, effectively reducing the idle time of the VM and improving resource utilization.

As can be seen from Fig.5, the service resource utilization of each algorithm tends to be smooth and upward overall among the three different types of workflows, and the OHDS method proposed in this paper has the highest service resource utilization regardless of the variation of the slack factor. Considering that different workflow tasks do not interfere with each other, the OHDS method effectively reduces the idle time slot of enabled VMs and improves resource utilization by mixing scheduling of multiple workflows in the scheduling process.

Fig.6 represents the CPU utilization of the enabled services, i.e. the deployment of VMs, during the task execution phase. With the gradual change of the slack factor, the overall CPU utilization of the services for the four algorithms remains stable. The OHDS method proposed in this paper is used to improve the CPU utilization of the enabled service by migrating the VMs on the underloaded service, and thus the service CPU utilization of the OHDS method outperforms the other scheduling algorithms in all three workflow collections of different sizes.

VI. CONCLUSION

Based on the independence between different workflow requests, this paper proposes an online hybrid dynamic scheduling algorithm - OHDS - to minimize workflow execution

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3228402

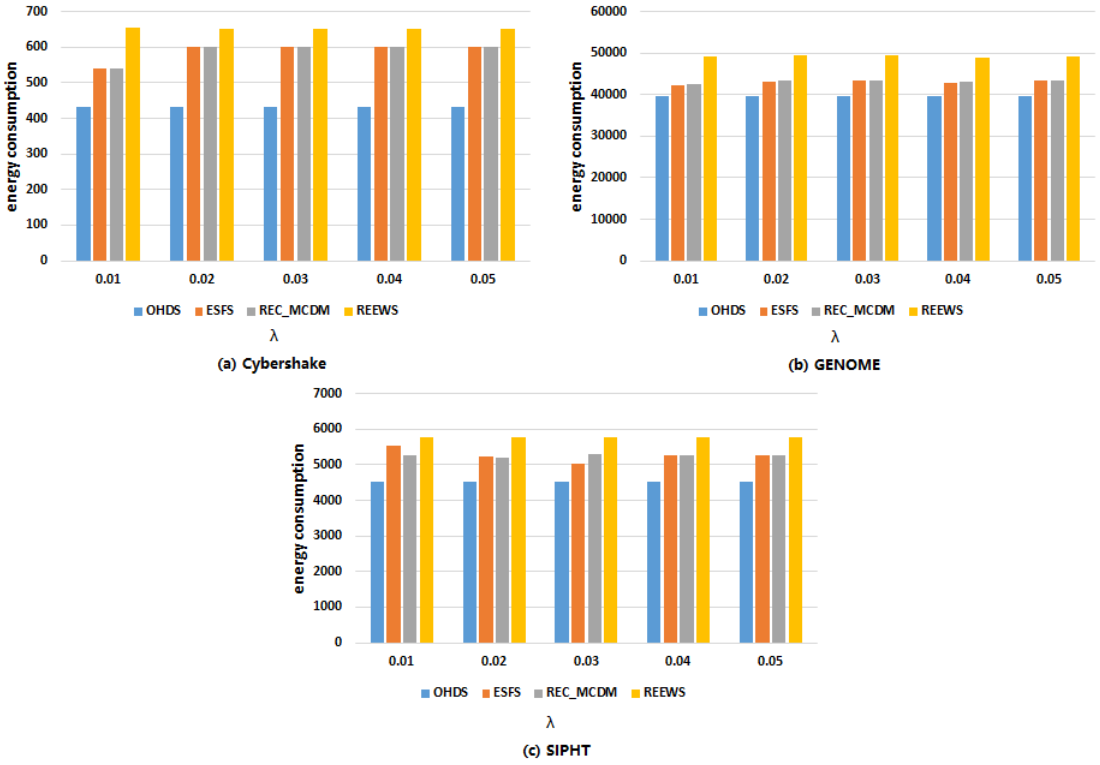

Fig. 3. The energy consumption of each algorithm when λ varies from 0.01 to 0.05.

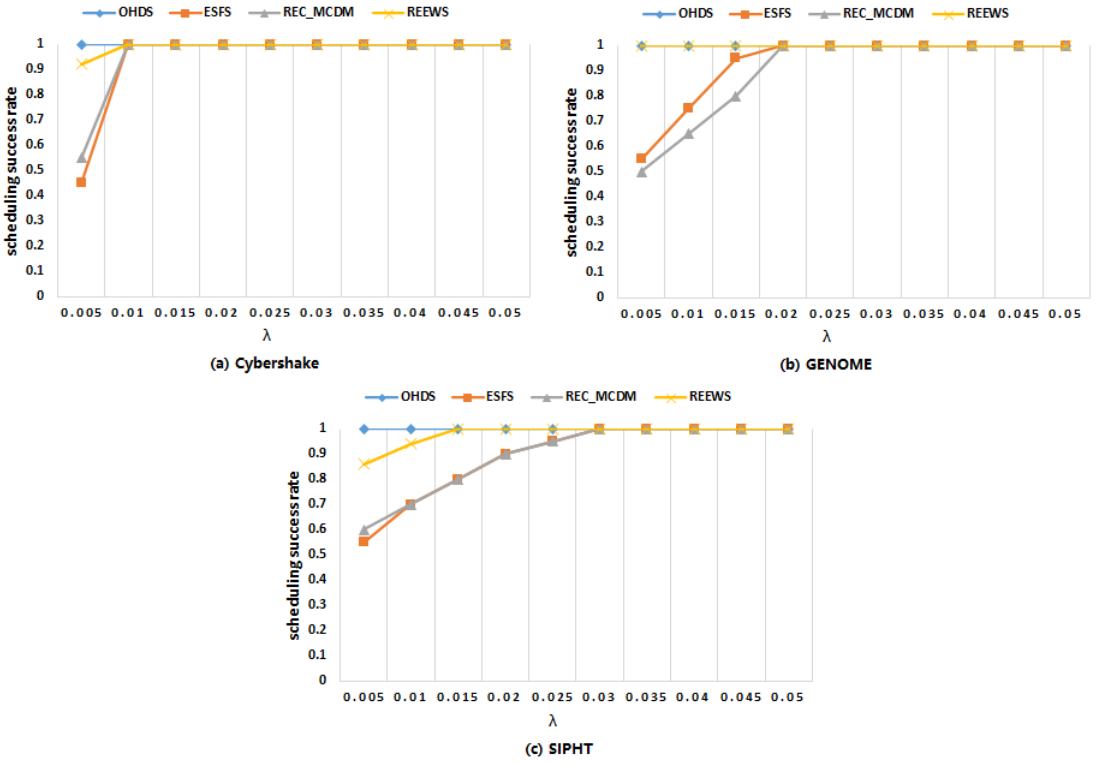

Fig. 4. The scheduling success rate for each algorithm with λ varies from 0.005 to 0.05.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3228402

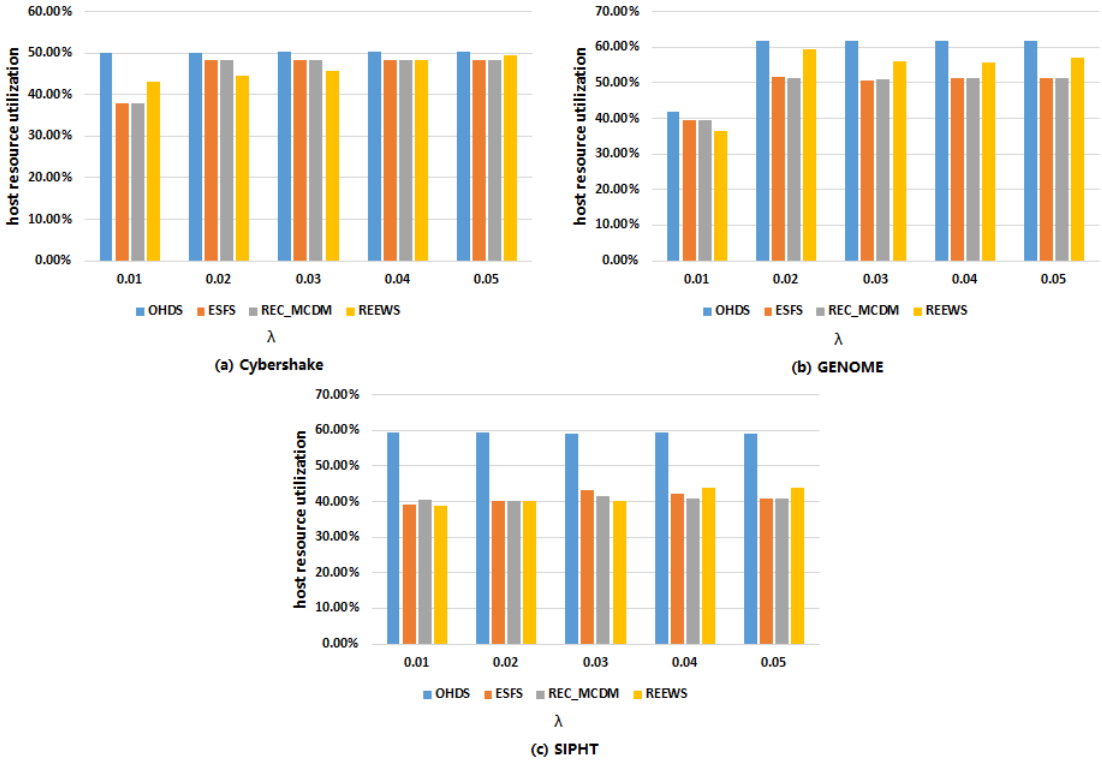

Fig. 5. The service resource utilization for each algorithm with λ varies from 0.01 to 0.05.

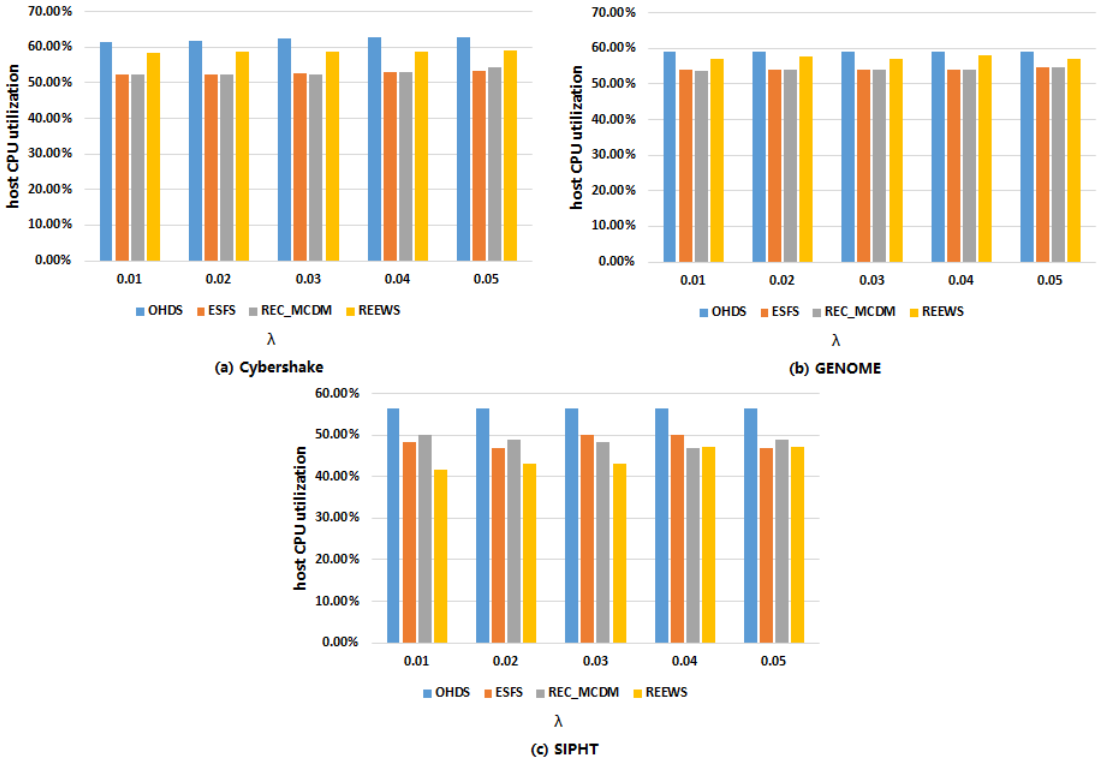

Fig. 6. The service CPU utilization for each algorithm with λ varies from 0.01 to 0.05.

energy consumption and improve service resource utilization while satisfying workflow time constraints and inter-task data dependency constraints. Firstly, multiple associated workflow tasks are combined into a single task to reduce the data transfer overhead incurred by associated tasks executing on different services during task scheduling. Secondly, each workflow task is assigned a scheduling priority by the earliest completion time of the task and an initial sub deadline is assigned to the workflow task based on the latest completion time of the task. Finally, workflow hybrid scheduling is used to reduce the idle time of VMs; for scheduled tasks, their direct sub-task priorities and the sub deadlines of subsequent unassigned tasks are dynamically adjusted by the current scheduling information; for each enabled service, the VMs on the underloaded service are dynamically migrated to reduce the service energy loss. Based on three different types of workflow collections, the proposed approach in this paper is experimentally compared with three currently available scheduling algorithms, and the OHDS approach performs better in terms of execution energy consumption, scheduling success rate and service resource utilization.

In subsequent research work, the task merging method in the pre-processing phase will first be improved to suit all types of workflows; reinforcement learning methods will be referred to for the workflow task scheduling decision problem; and more inter-constrained optimisation objectives will be introduced for trade-off and comparison under the constraint of satisfying the quality of service.

REFERENCES

[1] Li X, Garraghan P, Jiang X, et al. Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(6): 1317-1331.

[2] Yan H, Wang H, Li X, et al. Cost-efficient consolidating service for aliyun's cloud-scale computing[J]. IEEE Transactions on Services Computing, 2019, 12(1): 117-130.

[3] Safari M, Khorsand R. Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment[J]. Simulation Modelling Practice and Theory, 2018, 87: 311-326.

[4] Choudhary A, Govil M C, Singh G, et al. Task clustering-based energy-aware workflow scheduling in cloud environment[C]. //2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2018: 968-973.

[5] Stavrinides G L, Karatza H D. An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations[J]. Future Generation Computer Systems, 2019, 96: 216-226.

[6] Bhuiyan A, Guo Z, Saifullah A, et al. Energy-efficient real-time scheduling of DAG tasks[J]. ACM Transactions on Embedded Computing Systems, 2018, 17(5): 1-25.

[7] Li Z, Ge J, Hu H, et al. Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds[J]. Services Computing, IEEE Transactions on, 2018, 11(4): 713-726.

[8] Yuan H, Zhou M C, Liu Q, et al. Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed green clouds[J]. IEEE/CAA Journal of Automatica Sinica, 2020, 7(5): 1380-1393.

[9] Pietri I, Sakellariou R. Energy-aware workflow scheduling using frequency scaling[C]. //2014 43rd International Conference on Parallel Processing Workshops, 2014: 104-113.

[10] Bugingo E, Zhang D, Zheng W. Constrained energy-cost-aware workflow scheduling for cloud environment[C]. //2020 IEEE 13th International Conference on Cloud Computing (CLOUD), 2020: 40-42.

[11] Garg R, Mittal M, Son L H. Reliability and energy efficient workflow scheduling in cloud environment[J]. Cluster Computing, 2019, 22(4): 1283–1297.

[12] Juve G, Chervenak A, Deelman E, et al. Characterizing and profiling scientific workflows[J]. Future Generation Computer Systems, 2013, 29(3):682-692.

[13] Saraswathi S, Balamurugan S. Energy-aware workflow scheduling algorithm for the deployment of scientific workflows in cloud[J]. Smart Intelligent Computing and Applications, 2019, 104: 153-162.

[14] Geng X, Mao Y, Xiong M, et al. An improved task scheduling algorithm for scientific workflow in cloud computing environment[J]. Cluster Computing, 2019, 22(3): 7539-7548.

[15] Garg N, Singh D, Goraya M S. Energy and resource efficient workflow scheduling in a virtualized cloud environment[J]. Cluster Comput, 2021, 24(2):767–797.

[16] Mohammadzadeh A, Masdari M, Gharehchopogh F S. Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm[J]. Journal of Network and Systems Management, 2021, 29(3): 1-34.

[17] Ahmad W, Alam B, Atman A. An energy-efficient big data workflow scheduling algorithm under budget constraints for heterogeneous cloud environment[J]. The Journal of Supercomputing, 2021, 77(10):11946–11985.

[18] Kalra M, Singh S. Multi-objective energy aware scheduling of deadline constrained workflows in clouds using hybrid approach[J]. Wireless Personal Communications, 2021, 116(3):1743–1764.

[19] Khojasteh Toussi G, Naghibzadeh M. A divide and conquer approach to deadline constrained cost-optimization workflow scheduling for the cloud[J]. Cluster Computing, 2021, 24(3): 1711-1733.

[20] Medara R, Singh R S. Energy efficient and reliability aware workflow task scheduling in cloud environment[J]. Wireless Personal Communications, 2021, 119(2):1301–1320.

[21] Zhang L, Wang L, Wen Z, et al. Minimizing energy consumption scheduling algorithm of workflows with cost budget constraint on heterogeneous cloud computing systems[J]. IEEE Access, 2020, 8:205099-205110.

[22] Garg R, Mittal M, Son L H. Reliability and energy efficient workflow scheduling in cloud environment[J]. Cluster Comput, 2019, 22(4):1283–1297.

[23] Antolak E, Pułka A. Energy-efficient task scheduling in design of multithread time predictable real-time systems[J]. IEEE Access, 2021, 9:121111-121127.

[24] Walia N K, Kaur N, Alowaidi M, et al. An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments[J]. IEEE Access, 2021. 9:117325-117337.

[25] Li J, Zhang X, Wei Z, et al. Energy-aware task scheduling optimization with deep reinforcement learning for large-scale heterogeneous systems[J]. CCF Transactions on High Performance Computing ,2021, 3(4):383–392.

[26] Bi J, Yuan H, Tan W, et al. Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center[J]. IEEE Transactions on Automation Science and Engineering, 2015, 14(2): 1172-1184.

[27] Chen R, Chen X, Yang C. Using a task dependency job-scheduling method to make energy savings in a cloud computing environment[J]. The Journal of Supercomputing, 2202, 78(3): 4550–4573.

[28] Hu B, Cao Z, Zhou M. Energy-Minimized Scheduling of Real-Time Parallel Workflows on Heterogeneous Distributed Computing Systems[J]. IEEE Transactions on Services Computing, 2022, 15(5): 2766-2779.

[29] Wang Y, Zuo X. An Effective Cloud Workflow Scheduling Approach Combining PSO and Idle Time Slot-Aware Rules[J]. IEEE/CAA Journal of Automatica Sinica, 2021, 8(5):1079-1094.

[30] Wu Q, Zhou M, Wen J. Endpoint Communication Contention-Aware Cloud Workflow Scheduling[J]. IEEE Transactions on Automation Science and Engineering, 2022, 19(2):1137-1150.

[31] Wu Q, Zhou M, Zhu Q, et al. MOELS: Multiobjective Evolutionary List Scheduling for Cloud Workflows[J]. IEEE Transactions on Automation Science and Engineering, 2019, 17(1):166-176.

[32] Li H, Wang D, Zhou M, et al. Multi-Swarm Co-Evolution Based Hybrid Intelligent Optimization for Bi-Objective Multi-Workflow Scheduling in the Cloud[J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(9):2183-2197.

[33] Zhu X, Yang L T, Chen H, et al. Real-time tasks oriented energy-aware scheduling in virtualized clouds[J]. IEEE Transactions on Cloud Computing, 2014, 2(2):168-180.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3228402

Guisheng Fan received the B.S. degree in computer science from the Anhui University of Technology in 2003, and the M.S. and Ph.D. degrees in computer science from the East China University of Science and Technology in 2006 and 2009, respectively, where he is currently a Research Assistant with the Department of Computer Science and Engineering. His research interests include formal methods for complex software system, service oriented computing, and techniques for analysis of software architecture.

Xingpeng Chen is currently pursuing the M.D. degree in computer technology with the Department of Computer Science and Engineering, East China University of Science and Technology. He research interests include software engineering, cloud computing, cloud workflow scheduling.

Zengpeng Li received the B.E. degree from East China University of Science and Technology, Shanghai, China, in 2019, he is currently working toward the Ph.D. degree with the Department of Computer Science and Engineering. His current research interests include cloud computing, edge computing, and microservices.

Huiqun Yu (Senior Member, IEEE) received the B.S. degree in computer science from Nanjing University in 1989, the M.S. degree in computer science from the East China University of Science and Technology (ECUST) in 1992, and the Ph.D. degree in computer science form Shanghai Jiaotong University in 1995. He is currently a Professor of computer science with the Department of Computer Science and Engineering, ECUST. His research interests include software engineering, high confidence computing systems, cloud computing, and formal methods.

Yingxue Zhang is currently pursuing the M.D. degree in computer science and technology with the Department of Computer Science and Engineering, East China University of Science and Technology. He research interests include software engineering, cloud computing.