JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 38, 17-39 (2022)
DOI: 10.6688/JISE.202201_38(1).0002

Multi-Objective Scheduling of Cloud Data Centers Prone to Failures

QING-HUA ZHU, JIA-JIE HUANG AND YAN HOU
School of Computer Science and Technology
Guangdong University of Technology
Guangzhou, 510006 P.R. China
E-mail: zhuqh@gdut.edu.cn; huangjj@mail2.gdut.edu.cn; houyan@gdut.edu.cn

Distributed data centers (DDCs) consume power energy increasingly to provide different types of heterogeneous services to global consumers. Consumers bring revenue to DDC providers according to actual quality of service (QoS) of their requests. High energy consumption caused by a DDC is paramount for its providers to solve. During the maintenance due to failures, network service providers have to guarantee continuously reliable services to their consumers to ensure their revenue. Therefore, it is highly challenging to schedule tasks among DDCs in a low-energy and high-QoS way. In this paper, we propose a novel hierarchical framework for solving the task scheduling and power management problem in DDCs. The proposed hierarchical framework comprises: (1) a tier for global task scheduling to the DDCs and (2) a local tier for distributed power management of local servers. The dataset transmission energy between DDCs is considered. Meanwhile, this approach optimizes three conflicting objectives: total cost, energy consumption during computations and transmissions, and application rejections or violations due to failures. The proposed method can also improve resource utilization. The experimental simulations on large scale parallel working datasets show that this method can save energy significantly and obtain high quality of service. Meanwhile, it can achieve a good trade-off between QoS and energy consumption in DDCs.

Keywords: cloud computing, energy-saving, random failures, dynamic voltage/frequency scaling

1. INTRODUCTION

Virtualization technology can greatly support cloud computing environments for efficient execution of applications on appropriate hosts [1]. With the growth of popularity of cloud computing, cloud infrastructure is built on a large number of servers, among which high performance computing (HPC) and large storage devices result in a large amount of energy consumption. Most of the energy consumption of a typical cloud data center is consumed by server operation, storage, and cooling. Energy consumption has become the most important issue on cloud computing. Many studies aim to improve the utilization of cloud resources [2-4].

In cloud computing, efficient resource utilization mainly relies on virtualization technology and fine-grained energy-saving technologies. To reduce the energy consumption of an active server, the frequency of its CPU and chipset can be scaled in terms of the workload of its task request by using dynamic voltage and frequency scaling (DVFS) technology. Dynamic power management (DPM) can be used to make physical machines sleep within a specified period of time to effectively save energy consumption. Compared with

Received August 23, 2020; revised November 4, 2020; accepted November 9, 2020.
Communicated by Changqiao Xu.

DVFS, the latter can reduce the number of active machines as much as possible, avoid unnecessary active machines with standby consumption, and meet the performance requirements in the current time period, which is achievable within a specified period of time.

In a large-scale cloud data center, the number of servers is so large that their component failures become routines [5, 6]. Despite of the failures affecting system performance, their impact on energy saving and operation costs have become an increasingly important concern to system designers and administrators [5, 6].

In this study, for HPC task scheduling of a distributed cloud system, in the global aspect, a fine-grained scheduling decision method is proposed in terms of resource utilization and energy saving in a reservation mode. To solve the problem of energy-aware task scheduling in a distributed cloud environment, the energy consumption of the dataset transmission is smoothed, and a trade-off is made between energy consumption and resource utilization in global scheduling. Based on dynamic cluster server configuration (DCSC) and dataset transmission cost, a cost-aware global scheduling algorithm (CGS) is proposed to allocate cloud resources for geologically distributed cloud systems subject to random faults.

The main contributions of this work are twofold. First, we propose a cost-sensitive scheduling algorithm based on dynamic cluster server on/off in the case of failures with considerations of dataset transmission cost. The proposed algorithms map each task to the required resources. Second, according to the workload or resource utilization of DC, the setting of the reservation ratio of servers and the operation of cluster server can be adjusted flexibly. The proposed algorithm balances the cost, energy consumption, and the number of application rejections with good performance.

2. RELATED WORK

The task scheduling problems aware of energy-efficiency have been extensively investigated [1, 7, 8] for distributed green data centers [1, 7, 8]. Apart from considering transmission energy, many approaches have been suggested to address the objective of minimizing energy consumption from different perspectives such as data center management architecture [9-11], scheduling workflows [12-14], electricity prices [15, 16], and resource scheduling strategy [2, 17, 18]. The studies in [19-24] focus on the energy consumption problems for scheduling HPC applications in cloud computing systems. The study in [25] considers workload scheduling in a heterogeneous cloud environment. Their algorithm based on Min-Min and Max-Max is better than other algorithms in the sense of makespan and average cloud utilization. A heuristic algorithm based on ant colony optimization (ACO) is proposed to minimize the execution cost of workflow in cloud computing under time constraints. In the study of [3], a cloud computing task scheduling method based on VM matching is proposed, which can effectively improve the task scheduling performance of the cloud and realize load balancing among various VMs.

Yacine *et al*. [26] propose two algorithms, namely, direct mobility heuristic (DMH) and iterative direct mobility heuristic (IDMH), to make up for the space limitation of A∗ algorithm. Jose *et al*. [27] define a comprehensive cost model which includes some utilities that customers provide for a certain degree of degradation when VMs are allocated to an overused environment. Both can bring benefits in terms of revenue and resource utilization.

Thus, it can be expanded flexibly according to the size of the data center. Workload execution time is also improved by incorporating the reallocation of service level agreement (SLA)-based computing power of virtual machines (VMs). Wang *et al*. [28] propose a method to calculate the optimal migration sequence and the network bandwidth used for each migration. For its feasibility, an approximation scheme combining linear approximation with a full polynomial time approximation is proposed, and the theoretical performance boundary and computational complexity have been obtained.

Juarez *et al*. [29] propose a dual-objective function optimization scheduling method for energy consumption or maximum completion time in heterogeneous cloud systems. They use an integrated cost function weighted by factor Alpha to represent user preferences for energy savings or execution time. Their heuristic algorithm sorts a given directed acyclic graph (DAG) task set by estimating the required energy, which identifies a separate subset of tasks as a preparatory step before allocating resources.

In addition to considering energy consumed by dataset transferring, there are many ways to minimize energy consumption from different perspectives. For example, green cloud computing [4], cost perception [18, 30], or transmission scheduling between computing nodes of the application [31]. Zhu *et al*. [32] formulate the workflow scheduling problem of simultaneous optimization of maximum completion time and cost as a multi-objective optimization problem in a cloud environment. They propose an algorithm based on evolutionary multi-objective optimization (EMO) to solve a workflow scheduling problem on infrastructure as a service (IaaS) platform.

Dabiah *et al*. [33] propose a heuristic scheduling algorithm for heterogeneous computing environments, which minimizes the total execution cost of the tasks while ensuring that the total completion time of the tasks does not exceed their deadlines. Aeshah *et al*. [34] propose a high performance computing application scheduling method that considers the transfer energy of datasets.

In addition, Gu *et al*. [35] have proved that DPM based on DVFS can be used to perform physical hibernation of the physical host/processor within a given period of time, which can effectively reduce the total energy consumption.

Alam *et al*. [36] provide a reliable resource allocation approach for cloud computing while minimizing the cost. A heuristic approach to cloud resource allocation is proposed, which focuses on the reliability problems caused by failures in cloud environments. Yuan *et al*. [37] present a cost-aware workload scheduling method to jointly optimize the number of active servers in each data center, and the selection of Internet service providers for the data centers. Yuan *et al*. [38] design a profit sensitive spatial scheduling approach to maximize the total profit of a distributed green data center by smartly scheduling all tasks of multiple applications to meet their response time constraints. Their approach can well utilize such spatial diversity of the above factors. Fu *et al*. propose a distributed VM construction strategy to enhance resource resilience by tracing failures [39]. Task rejections and deadline violations caused by failures of cloud resources have aroused much concern [39]. Extensive attention is paid to the reliability of cloud resource utilization.

The differences between this study and aforementioned ones are summarized as follows.

(1) Different from [36-39], to minimize total cost, energy consumption, and application deadline violations, we consider these factors jointly: random failures, the cost of tran-

smitting datasets across geographically distributed clouds, and cloud resource occupancy.

(2) Compared to [25, 35], we also take the resource footprint, energy for execution, and task execution costs in the scenario of random failures into account.

(3) Different from [29, 33-35], to further reduce energy consumption, our energy model uses DPM and DCSC, in which the task execution time is determined by the CPU frequency of a server. At the same time, we set flexible reservation ratio of servers and adaptive DPM decision according to workload fluctuation and the resource utilization of DCs.

The rest of this article is structured as follows. In Section 3, we formulate the addressed problem and system model. Then, Section 4 presents cost-sensitive scheduling algorithms which can effectively minimize total cost, energy consumption, and the number of rejected applications. Section 5 gives experimental evaluation. Section 6 summarizes this work.

3. SYSTEM MODEL AND PROBLEM FORMULATION

We consider q distributed data centers owned by different vendors, which forms a multi-cloud system, denoted by $C = \{C_1, C_2, C_3, \ldots, C_q\}$, where C_i is a data center. C_i has its own management server MS_j that relies on three components: a global scheduler, a local scheduler, and a resource controller. Fig. 1 provides an overview of the system model, illustrating the role of global and local schedulers when submitted applications are received.

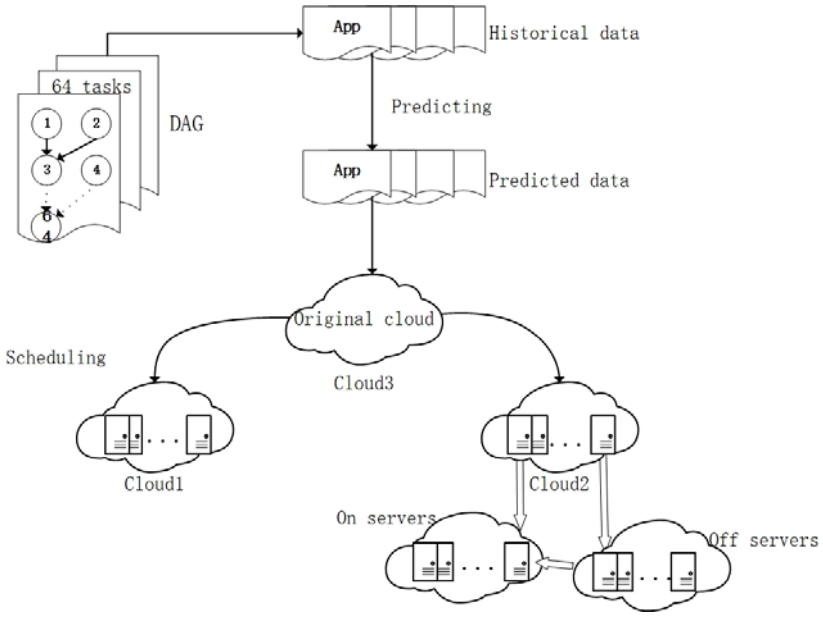

Fig. 1. System model.

This is very detrimental to application due to the non-periodicity of application arrivals. We use an LSTM-based (Long Short-Term Memory) workload prediction method for better scheduling. With little prior knowledge, we can extract links between different features from historical data. Here we make predictions about CPU utilization.

Table 1. Symbol list.

Symbol	Definition
M	Total number of servers
M_x^t	Number of servers in state x during timeslot t
$M_{x\to x'}^o$	Statistics number of servers in a transition from state x to state x'
$M_{reserve}^i$	Number of reserved servers in the cloud during timeslot t
M_{vio}	Number of application violations
μ_i	Process rate of active state i
L	Number of frequency levels.
P_x	Server power in server's state x
$P_{x\to x'}$	Power consumption in the server transition from state x to state x'
$P_f(l)$	Dynamic server power at frequency level $l \in \{1, 2, \ldots, L\}$
$D_{x\to x'}$	Latency of the server transition from state x to state x'
T	Period between the arrival time of x first task and the end of processing the last task
$T_{on\|\|off}$	Fixed period for the operation decision of switching on and off of a serve
$T_{s\|\|w}$	Fixed period for the operation decision of switching wakeup and sleep of a server
K	Length of a segment
k	Length of a timeslot
T_x^i	Duration of a server's state x during segment i
U	Number of timeslots to form a segment of time
E_C	Energy consumption of clouds
E	Total energy consumption
E_{on}^i	Energy consumption of booting up servers in a segment
E_{switch}^i	Energy consumption by a server's switching in segment i
T_fault_t	Duration of a failure
E_t	Dataset transmission energy consumption
ρ	Power usage efficiency (PUE)
A	Number of applications
$DataSize_i$	Data size of application i
$Infor\_App$	Relevant information about an application
ψ	Proportion of application violations
λ	Reserved process rate in a cloud
ε	Electricity price for data centers
η	Penalties for each application violation
ζ	Dataset transmission price

The local resource checker for each data center provides a provisional reservation (PR), which takes into account whether a data center is actually available or not, including estimated energy, and resource utilization. After an application is assigned to a corresponding cloud, its local scheduler applies scheduling algorithm cost-aware local scheduling (CLS) to map an application to a machine. The scheduling framework, energy model, and problem description are discussed next.

3.1 Scheduling Framework

Pre-scheduling primarily depends on the available resources and server status for a given period of time. In addition to uncertain resource failures, this scheduling model should avoid violating a deadline.

Energy optimization techniques are applied to schedule the submitted applications to ensure the performance requirements. It is required that there be at least one cloud that can meet an application deadline. Normally, every submitted application can be scheduled to execute, but it is possible that no cloud can meet the resources required by an application at its planned execution time, then it is marked in a rejected state. The framework supports combination rate strategy (CRS), improved cuckoo search (MCS) algorithm, hybrid chaotic particle search (HCPS) algorithm, and improved artificial bee colony algorithm (MABC) to select the most suitable cloud.

To ensure that the clouds should meet the application requirements and provide PR, we consider the following factors.

a) Estimating the energy required for each application,
b) Server status in each cloud, and
c) The unoccupied resources in each cloud.

The cloud selection strategy must adapt to any failures at any time in DDCs. A set of PRs are selected to perform the current task. The policy will check each PR to satisfy preferences for energy consumption, occupancy rate, performance, and their combinations. Afore mentioned factors b) and c) should be considered to choose the best cloud to fit a task.

When an application is rejected due to insufficiency of current computing resources, it will wait in a queue by our given strategy.

3.2 Energy Model

Advanced configuration and power interface (ACPI) standard is adopted in this study. Pursuant to ACPI, modern server power management hardware modules typically support multiple sleep states. In order to respond to more server types, it is assumed that a server has m sleep states and n active states, and a server in different states owns different processing capacities and capabilities. Let $N_g = \{1, 2, 3, \ldots, g\}$ denote a positive natural number set. Let $S\_sleep = \{sleep[1], sleep[2], \ldots, sleep[m]\}$ and $S\_active = \{active[1], active[2], \ldots, active[n]\}$. Normal transitions occur only between the sleep and idle states, and there is no switching between any two sleep states. Notice that the latency and power consumption of switching between different sleep modes and idle one are different. Server states fall into five levels: off, sleep, idle, fault, active, and reserve, which are denoted by *off*, *S_sleep*, *idle*, *fault*, *S_active*, and *reserve*, respectively. Let $\Im = \{off\} \cup S\_sleep \cup \{idle\} \cup S\_active \cup \{fault\} \cup \{reserve\}$. Each state of a server has a different frequency corresponding to a different processing rate and power. Let μ_i denote the process rate of the ith active state of a server ($i \in N_n$). P_x denotes a server's power in state $x \in \Im$. We define μ_{max} as the maximum processing rate of an active server. The relationships of servers' processing rate and power among different states are shown below.

$$\mu_n > \mu_{n\text{-}1} \ldots > \mu_2 > \mu_1 \tag{1}$$

$$P_{S\_active[n]} > P_{S\_active[n\text{-}1]} \ldots > P_{S\_active[2]} > P_{S\_active[1]} > P_{idle} > P_{S\_sleep[1]} > P_{S\_sleep[1]} \ldots > P_{S\_sleep[m]} \ (i \in N_n, j \in N_m) \tag{2}$$

The configuration of a sleep server determines the power and latency required to perform a wake-up operation. The greater power a sleep server has, the more energy is required to perform its wake-up operation and the shorter time to wake up. Similarly, idle or active servers need to perform sleep operations. The more energy it takes to get into a sleep state, the less time it takes to get sleep. Here, the power off state of a server can be viewed as a deep sleep one. The shutdown or bootup of a server requires very little power, but the corresponding delay is much longer than that of a normal wake-up operation. For a server, let $D_{x\to x'}$ denote the latency of its transition from state x to state x', where $x \in \mathfrak{I}$ and $x' \in \mathfrak{I}$. And $P_{x\to x'}$ denotes the power consumption of a server in a transition from state x to state x', where $x \in \mathfrak{I}$ and $x' \in \mathfrak{I}$. Their relationships are as follows.

$$P_{idle\to S\_sleep[1]} > P_{idle\to S\_sleep[2]} > \ldots > P_{idle\to S\_sleep[m]} > P_{idle\to off} \tag{3}$$

$$D_{idle\to S\_sleep[1]} < D_{idle\to S\_sleep[2]} < \ldots < D_{idle\to S\_sleep[m]} < D_{idle\to off} \tag{4}$$

$$P_{S\_sleep[1]\to idle} > P_{S\_sleep[2]\to idle} > \ldots > P_{S\_sleep[m]\to idle} > P_{off\to idle} \tag{5}$$

$$D_{S\_sleep[1]\to idle} < D_{S\_sleep[2]\to idle} < \ldots < D_{S\_sleep[m]\to idle} < D_{off\to idle} \tag{6}$$

Assume that there are M servers in each data center. Let T denote the period between the arrival time of the first task and the completion time of processing the last task, which is divided into $T_{on||off}$ segments. Each segment is set to be a fixed period. During segment t, let M_x^t denote the number of servers in state x. Let $M_{x\to x'}^t$ denote the statistics number of servers in a transition from state x to state x'. Let $S\_on = \mathfrak{I} \backslash \{off, reserve\}$. Let M_x^h denote the number of servers in state x during segment h. Let M_s^h denote the number of servers in state set S, *i.e.*, we have

$$M_s^h = \sum_{x \in S} M_s^h, \ h \in N_{T_{on||off}}. \tag{7}$$

In Eq. (7), S is one of the following state sets: S_active, S_on, and S_sleep.

The number of servers in a data center is calculated as follows,

$$M = M_x^h + M_{off}^h + M_{y\to off}^h + M_{off\to y}^h + M_{reserve}^h, \ h \in N_{T_{on||off}}, \ y \in S\_on, \tag{8}$$

$$M_{S\_on}^h = M_{S\_active}^h + M_{S\_sleep}^h + M_{\{idle\}}^h + M_{\{idle\}\to S\_sleep}^h + M_{S\_sleep\to\{idle\}}^h + M_{fault}^h, \ h \in N_{T_{on||off}}. \tag{9}$$

Each application consumes energy by its execution, failure handling, and dataset transfer, which are discussed below.

(A) Energy Formula for Executing Applications

The amount of energy to execute an application largely depends on the size of the application and the state of a server. When an application is scheduled to execute in a cloud, a server uses DPM to select its state. Then, it continues to adjust the processor frequency by using DVFS based on the application profile. Suppose that U timeslots form a segment

of time. Let K and k denote the length of a segment and a timeslot, respectively. K and k are set to be 120 minutes and 2 minutes, respectively. In other words, we divide T into $T_{s||w}$ timeslots.

$$T_{on||off} = T/K \tag{10}$$

$$T_{s||w} = T/K \tag{11}$$

$$U = K/k \tag{12}$$

The decision on turn-on/off is made at the beginning of each segment. Turn-on operations are performed at the end of current segment. Turn-off (shutdown) operations are performed at the beginning of current segment. Sleep operations are performed at the beginning of current timeslot. Wake-up operations are performed at the end of current timeslot. For the transition of a server, aforementioned sleep and wakeup operations ensure that enough servers are turned on or awakened to respond to requests for the next timeslot.

Let λ_i^K denote the incoming task requests during segment $i \in N_{T_{on||off}}$. Let λ_j^k denote the incoming task requests during timeslot $j \in N_{T_{s||w}}$. The maximum number of servers in the next timeslot is used as a decision variable for the current timeslot to ensure that the servers can satisfy all requests in the next timeslot. When the servers are making switch operations, the arrival of the requests should be considered to prevent the switch operations from affecting the processing of the current requests.

For a server, let D_{wakeup}, D_{sleep}, D_{off}, and D_{on} denote the maximum latency of its wakeup, sleep, off, and on operations, respectively.

Therefore, under the premise of ensuring a server's performance to process the request, transitions including $S\_sleep \rightarrow \{idle\}$, $\{idle\} \rightarrow S\_sleep$, $S\_on \rightarrow \{off\}$, and $\{off\} \rightarrow S\_on$ may keep a small number of servers in active- or on- state for the performance requirements in a timeslot, which may result in a portion of the energy loss, but can reduce the energy consumption through dynamic server on/off operations and a multi-sleep mode.

Let E_C denote the energy consumption of q data centers. During segment $i \in N_{T_{on||off}}$, let E_{switch}^i denote the energy consumption of servers in state x. Let E_{on}^i, $E_{reserve}^i$ and E_{switch}^i denote the energy consumption of booting up servers, the energy consumption of reserved servers and the energy consumption by servers' switching in segment $i \in N_{T_{on||off}}$, respectively. During segment i, let T_x^i stand for the duration of server state x in segment $i \in N_{T_{on||off}}$, ρ for the power usage efficiency (PUE), E_i for the energy consumption of data center C_i ($i \in N_q$), and $E_{x \rightarrow x'}$ for the energy consumption in the transition from state x to state x'. Therefore, the estimated energy consumption of clouds is calculated as follows:

$$E_x^i = M_x^i \times P_x \times T_x^i,\ i \in N_{T_{on||off}}, \tag{13}$$

$$E_{on}^i = E_{S\_active}^i + E_{idle}^i + E_{S\_sleep}^i + E_{idle \rightarrow S\_sleep}^i + E_{S\_sleep \rightarrow idle}^i + E_{fault}^i,\ i \in N_{T_{on||off}}, \tag{14}$$

$$E_{switch}^i = E_{on \rightarrow off}^i + E_{off \rightarrow on}^i,\ i \in N_{T_{on||off}}, \tag{15}$$

$$E_i = (1+\rho) \times \sum_{j=1}^{T_{on||off}} (E_{on}^j + E_{switch}^j + E_{reserve}^j), \tag{16}$$

$$E\_C = \sum_{i=1}^{q} E_i, i \in N_q. \tag{17}$$

(B) Energy formula for random failure occurrence

During the application execution, different components of a server may fail at any time, which breaks up the normal execution. In the event of a failure, we assume that the application being executed by a failed server will be moved by mirror replication to an idle server to continue its execution. During this migration, a failed server will continue to consume energy during the failure restore. Let T_{fault} denote the duration of a failure. We assume that the power of a server in the failed status remains as same as that prior to the failure, which is denoted by P_{fault}. Therefore, the extra energy consumption caused by a fault process is calculated below.

$$E_{fault} = M^{h}_{fault} \times P_{fault} \times T_{fault},\ t \in N_{T_{s||w}} \tag{18}$$

(C) Dataset transmission

Let A denote the number of applications. The set of applications is denoted by $App = \{app[1], app[2], \ldots, app[A]\}$. Let M_{vio} denote the number of application violations and ψ denote the proportion of application violations. Therefore, the proportion of application violations is calculated as follows:

$$\psi = M_{vio}/A. \tag{19}$$

The energy consumed by the dataset of application $app[i]$ is mainly related to its size and travel distance, which are denoted by $DataSize_i$ and $Distance_i$, respectively. A given original cloud is set up to accept all incoming tasks. After the allocation phase, a task may be performed in the original cloud or be sent to a target cloud data center to perform. Both its transmission from the source to target cloud and its execution on the target consume energy. After an application's execution, its output results are sent back to the original cloud, which also consumes energy. Following [35], the dataset transmission energy consumption and total energy consumption is:

$$E\_T = \sum\nolimits_{i \in N_A} (DataSize_i \times \frac{Distance_i}{500km}), \tag{20}$$

$$E = E\_C + E\_T. \tag{21}$$

3.3 Cost Model

The total cost is made of the cost caused by task executions and the fines caused by application violations. Let $Cost$, $Cost_{energy}$, and $Cost_{fault}$ denote the total cost, the cost caused by task execution, and the fines caused by application violations, respectively. Let ε, η and ζ denote the electricity price for data centers, the penalties for each application violation, and the dataset transmission price. The total cost is calculated by

$$Cost = Cost_{energy} + Cost_{fault}, \tag{22}$$

$$Cost_{energy} = E_C \times \varepsilon + E_t \times \zeta, \tag{23}$$

$$Cost_{fault} = M_{vio} \times \eta. \tag{24}$$

3.4 Problem Formulation

We assume that each application is committed to the original cloud at a specific time in DDCs. If a submitted task is known and the execution conditions are met for it, then a corresponding cloud can accept it, otherwise it will be rejected. If the application is accepted in a schedule, it may be successfully executed or interrupted in case that its server may fail during its execution. Our goal is to optimize the total energy consumption of all accepted applications across DDCs. At the same time, the reliability of system operations is guaranteed. Let R denote the tolerable response time. Efforts are made to minimize the total expenditure: (1) the total energy consumption; (2) the total cost; and (3) the ratio of application violations. Therefore, the objective function is to

Minimize: E, $Cost$, ψ

Subject to: Eq. (7) to Eqs. (18), (20), (23) to (24), and

$$\mu_{\max} \times M_{S\_active}^{h} > \lambda^{h},\ h \in N_{T_{on\|off}}, \tag{25}$$

$$\frac{1}{\mu_{\max} \times M_{S\_active}^{h} - \lambda^{h}} + \frac{1}{\mu_{\max}} \le R,\ h \in N_{T_{on\|off}}. \tag{26}$$

Once a task is rejected or interrupted, the penalty for a rejection or violation is usually high, which should be avoided in practice.

4. ALGORITHMS DESIGN

4.1 Mapping Applications to Clouds

After an application reaches the original cloud, the middleware assigns a token to the cloud that has the ability to complete before the deadline. When multiple clouds have the ability to complete at the same time, a strategy is adopted to obtain the desired result. We adopt CRS to trade off such multiple objectives by efficiently executing all applications while meeting their delay constraints, as far as possible to minimize the estimated energy consumption, the cost, the utilization and maximize the performance. In order to optimize multiple objectives and balance them, we use the coefficient of variation method to analyze each objective, and then calculate the weights of different objectives. The relative standard deviation (RSD) represents the dispersion among the objectives and plays a different role in making decisions.

As aforementioned, the local resource checker in each data center maintains a PR. All PRs in DDCs are stored in a list, which is denoted by PRs. Let *PRs.size* denote how many *PRs* (data centers) are available for a submitted application. Let e_i denote the energy consumption of C_i. Let $\bar{e}$ denote the average of energy consumption. In the aspect of energy, *RSD*(*Energy*) is defined below to evaluate the relevance of different objectives to the application.

$$RSD(Energy) = \sqrt[2]{\frac{\sum_{i=1}^{PRs.size} |\overline{E} - e_i|^2}{PRs.size}} / \overline{e} \tag{27}$$

Let Ut_i stand for the utilization rate of C_i, $\overline{Ut}$ for the average of all Ut_i's, μ_max_i for

the maximum processing rate of C_i, and $\overline{\mu\_\max}$ for the average of the maximum processing in all data centers. Similar to the definition of *RSD*(*Energy*), we have the evaluations for the utilization and performance.

$$RSD(Utilization) = \sqrt[2]{\frac{\sum_{i=1}^{PRs.size} |\overline{Ut} - Ut_i|^2}{PRs.size}} / \overline{Ut} \tag{28}$$

$$RSD(Performance) = \sqrt[2]{\frac{\sum_{i=1}^{PRs.size} |\overline{\mu\_\max} - \mu\_\max_i|^2}{PRs.size}} / \overline{\mu\_\max} \tag{29}$$

$$Total\_RSD = RSD(Energy) + RSD(Utilization) + RSD(Performance) \tag{30}$$

Then we use the coefficient of variation to calculate the weight of each objective as shown below.

$$W(Energy) = RSD(Energy)/Total\_RSD \tag{31}$$

$$W(Performance) = RSD(Performance)/Total\_RSD \tag{32}$$

$$W(Utilization) = RSD(Utilization)/Total\_RSD \tag{33}$$

Low dispersion means that the objective has little influence on decision-making. On the contrary, high dispersion means that the objective has a great impact on decision-making. During the decision evaluation, we give the objective higher weight when its index value becomes larger which means it has a greater influence on decision-making. Consequently, we can get better decision-making under the premise of trading off different objectives.

The execution list and waiting list for each application will be created in the original cloud during the execution. When an application arrives, the current state of each cloud is known, and the No. of a cloud capable of executing the application is recorded in the current executable list. After all of the clouds are evaluated, the executable list is queried. When there is no cloud in the executable list, an application is unable to execute at that time, then it is placed in a waiting list. If an application fails to start executing within the given time, it is considered to be rejected. A cloud is chosen as a host cloud if it is the only one can execute the current application.

To map each application to a desirable cloud, Algorithm 1 is proposed to implement aforementioned CRS scheme, where *Info_App* denotes the profile of an application.

Algorithm 1: Mapping applications to clouds

Input: *Info_App*, *C*, waiting list
Output: C_z

1. Initialization: *PRs*←∅, *token*←0
2. **For** each cloud management server of $C_i \in C$ do
3. **If** $\mu_{\max} \times (M - M^h_{S\_active}) > \lambda^{app}$ and *starttime* + *runtime* < *Deadline*
4. *token*←1
5. **Endif**
6. **EndFor**
7. **if** *PRs*←∅ **then**

8. add the application to the waiting list
9. **Else**
10. **If** *PRs.size* = 1 **then**
11. choose the cloud whose *token* = 1
12. **Else**
13. Calculate the *RSD* of Energy, Utilization and Performance by Eqs. (27)-(29)
14. Normalized the Energy, Utilization and Performance
15. Set weight of the Energy, Utilization and Performance by Eqs. (31)-(33)
16. Choose the best cloud $C_z \in PRS$ for the application
17. **Endif**
18. **Endif**
19. Return C_z.

Under the coefficient of variation method, the weight of each objective will be dynamically adjusted according to the requirements of each application and the cloud resources that meet the conditions. When an executable application is bound to a corresponding cloud, the executable list of the current application is published. The results are sent to PR when an appropriate cloud is chosen.

4.2 Cost-Sensitive Server Scheduling

After applications are dispatched to a cloud, they should be scheduled locally to corresponding servers to achieve a minimal energy consumption. Algorithm 2 is proposed to do so.

The input *Paras* of Algorithm 2 represents server configuration parameters. Its outputs *X*, *Y*, *F*, and *E* represent the decision variables for switching between turn-on and turn-off, sleep-wake switching, frequency scaling, and minimal total power consumption, respectively. Let *x*, *y*, and *f* denote the number of servers required for the current segment, the number of servers in the current timeslot, and the number of servers at different frequencies, respectively.

Algorithm 2: Cost-ware Server Scheduling

Input: $\bigcup_{i=1}^{T_{on\|off}} \{\lambda_i^K\}$, $\bigcup_{i=1}^{T_{s\|w}} \{\lambda_i^K\}$, K, k, U, *Paras*

Output: X, Y, F, E, λ.

1. Initialization: $X \leftarrow \varnothing$, $Y \leftarrow \varnothing$, $F \leftarrow \varnothing$, $E \leftarrow 0$, $\lambda \leftarrow 0$, $x \leftarrow 0$, $y \leftarrow 0$, $f \leftarrow 0$;
2. **For** $i \leftarrow 1$ to $T_{on\|off}$ do
3. **If** $\lambda_i^k < (\lambda_{\max} \times M \times 25\%)$
4. $\lambda \leftarrow 0.3\%$
5. **Else**
6. **If** $(\lambda_{\max} \times M \times 25\%) < \lambda_i^k < (\lambda_{\max} \times M \times 75\%)$
7. $\lambda \leftarrow 0.5\%$
8. **Else**
9. $\lambda \leftarrow 0.8\%$
10. **Endif**
11. **Endif**

12. **If** $\lambda_i^k < \lambda_{i+1}^k$ or $\lambda_i^k > (\lambda_{i+1}^k \times 120\%)$
13. $X_{i\text{-}1}$, $\{\lambda_i^k\}$ and *Paras* are constructed into matrix Ξ_1.
14. $x \leftarrow$ Call CPLEX by inputting Ξ_1
15. $X_i \leftarrow$ *Servers on/off* (x, i)
16. $X \leftarrow X \cup \{X_i\}$;
17. **For** $j \leftarrow 1$ to U do
18. **if** $\lambda_j^k > \lambda_{j+1}^k$ or $\lambda_j^k > (\lambda_{j+1}^k \times 120\%)$
19. $Y_{j\text{-}1}$, $\{\lambda_j^k\}$ and *Paras* are constructed into matrix Ξ_2.
20. $y, f \leftarrow$ Call CPLEX by inputting Ξ_2
21. $Y_j \leftarrow$ Servers Wakeup/Sleep(y, j);
22. $F_j \leftarrow$ Frequency adjustment (f, j);
23. **Endif**
24. $Y \leftarrow Y \cup \{Y_j\}$
25. $F \leftarrow F \cup \{F_j\}$
26. **EndFor**
27. **Endif**
28. **EndFor**
29. $E \leftarrow$ Energy consumption calculation (X, Y, F)
30. Return X, Y, F, E.

In lines 3-11 of Algorithm 2, the reserved process rate in a cloud is flexibly adjusted according to the resource utilization. After the workload for each segment is obtained, we adjust the reserved process rate according to the data center resource utilization in the current segment. Set a low threshold of reserved process rate when the workload is low (less than 25% of the data center's processing capacity). On the premise of dealing with failures, the energy consumption cost of the reserved server is reduced. Accordingly, the reserved process rate is increased when the workload is high. In lines 12-26 of Algorithm 2, the workload during the current segment is compared with that during the next segment. When the former workload is less than the latter workload, or 120% more than the latter workload, DPM operations are performed on servers' turn-on/off, sleep/wakeup, and DVFS are performed according to the workload in the specified timeslot, which can adapt to the workload fluctuations.

Now we look into the matrix construction procedure in lines 13 and 19. In order to execute a DPM operation, relevant parameters are constructed by matrices based on equations, inequalities, current workload, and server state parameters. Matrix construction is similar for sleep/wakeup, turn-on/off decision making, as exemplified here by turn-on/off decision making.

For segment 1, matrices can be constructed by the following steps. It is assumed that all servers are in an idle state before segment 1.

Step 1: For each $i \in \{1, 2, \ldots, T_{on\|off}\}$, initialize $M_{\{idle\}}^i = M \times (1 - \lambda)$, $M_{off}^k = 0$, $M_{S\_on}^i = M_{\{idle\}}^i$, $M_{on \to off}^i = 0$, $M_{off \to on}^i = 0$, $M_{reserve}^i = M \times \lambda$, $M_{fault}^i = 0$.
Step 2: *Current_next* $\leftarrow$ the number of failed servers that cannot recover by the end of current segment; $M_{fault}^{t+1} \leftarrow$ *Current_next*.
Step 3: Create a coefficient matrix for Eqs. (7)-(9) and (25)-(26) according to Steps 1-2.

For segment $i > 2$, the steps to construct a matrix are as follows:

Step 1: Count the numbers of servers currently in the reserved state, servers in a failure state, and servers that need to turn-on/off machine operations, respectively.
Step 2: *Current_next* ← the number of failed servers that cannot recover by the end of current segment; if the duration of a server failure is less than the delay of server turn-on operation, *Current_next* will also be considered as part of the reserved server in the next segment, and otherwise, some idle servers will become the reserved server.
Step 3: Create a coefficient matrix for Eqs. (7)-(9) and (25)-(26) according to Steps 1-2.

Therefore, redundant operation can be effectively reduced and the reliability of a server can be guaranteed under the condition that the energy consumption cost is roughly unchanged.

The decision variables for each segment determine the number of active servers at the current segment. On this basis, the decision for sleep mode switching is executed in each of timeslot of a segment. A matrix is obtained by constructing the relevant parameters in each timeslot through IBM ILOG CPLEX Optimization Studio. According to the obtained decision variables, a server is scheduled to sleep or wake up within a specified timeslot to ensure that energy consumption is minimized during the timeslot of execution. Adjusting the frequency of a server within a timeslot can further reduce its energy consumption. The results of each segment and timeslot are marked and all decision variables for the next task processing phase are updated in lines 15 and 21-22 of Algorithm 2.

4.3 Response to Failures

During the execution of a task, we assume that (1) a fault is likely to occur in every second; (2) the number of faults is stochastic; and (3) the fault duration satisfies the Poisson distribution. To cope with the situation in possible failures, a given number of servers are reserved to replace the failed ones. If the reserved servers cannot satisfy the replacement requirement, available servers in the resource pool can be scheduled to do so. Firstly, the numbers of both server failures per second and task requests executing on the failed server are obtained. Then, check the current cloud resource surplus and servers' status to see if the reserved servers can handle the current failure. If they can, they are enabled to perform tasks interrupted by failures; otherwise, the reserved and other available servers are enabled to make such a replacement. Finally, the actual execution time and status of the application are recalculated and updated which are outputted along with the energy and cost of handling a failure.

The above procedure is implemented by Algorithm 3, where we let λ stand for the reserved processing rate in the clouds, $M^t_{reserve}$ for the number of reserved servers in the clouds in timeslot t, and M_{vio} for the number of application violations. Notice that we do not consider to turn on the poweroff servers in the event of a server failure because the reliability of a server and the long latency are required to boot up.

Algorithm 3: Response to Failures

Input: M^t_{fault}, T_{fault}, *Info_App*, λ^t, *Paras*
Output: *Info_App*, E_{fault}, $Cost_{fault}$

1. $Cost_{fault} \leftarrow M_{vio} \times \eta$;
2. $\lambda \leftarrow M^t_{fault} \times T_{fault} \times \mu_{fault}$;
3. $\lambda_{reserve} \leftarrow M^t_{reserve} \times \mu_{max}$;
4. If $\lambda < \lambda_{reserve}$ then
5. $M^t_{reserve} \leftarrow M^t_{reserve} - M_{fault}$;
6. Update *Info_App*, E_{fault}, $Cost_{fault}$, $M^t_{reserve}$;
7. Else
8. $M_{fault} \leftarrow M_{fault} - M^t_{reserve}$;
9. Update λ with the new M_{fault};
10. Calculate the remaining required requests and the time to complete the application;
11. Update *Info_App*, E_{fault}, $Cost_{fault}$;
12. EndIf
13. Return *Info_App*, E_{fault}, $Cost_{fault}$.

5. EXPERIMENTAL EVALUTION

By simulations on CloudSim [40], the proposed algorithms based on CRS under different polices are compared with three benchmark algorithms to evaluate the total energy consumption for executing applications, the ratio of application rejections and violations under different loads, and the total cost for executing applications. Different workloads are given to show that the quantity of reservation servers impacts on the application executions.

Assume that there are five geographically dispersed data centers to form a multi-cloud system. Each cloud with 32000 VMs is set to be twice the VM capacity of its physical processors. They have three sleep states (*S_sleep*[i], $i \in N_3$), an idle state (*idle*), a poweroff state (*off*) and five active states (*S_sleep*[i], $i \in N_5$), as shown in Table 2. The relevant parameters for parallel workloads are shown in Table 3. We use the CPU utilization of all VMs as the input data of the prediction model. Table 4 shows the parameter settings in the simulation experiment. Suppose that the penalties for each application violation η is 2.5 times the maximum cost of executing applications. The parameter settings of the adopted benchmark algorithms are shown in Table 5. The parameters in our proposed algorithms have been presented afore. Because of the randomness of evolutionary algorithms, each benchmark algorithm is executed independently for 30 times to obtain the best solution as the result.

Table 2. Servers on different states.

State	Power(W)	State	Power(W)	Delay(s)
off	0, 0, 0, 0, 0	*off* → *on*	380	80
S_sleep [1]	36, 39, 44, 49, 56	*on* → *off*	280	60
S_sleep [2]	51, 46, 73, 97, 104	*S_sleep* [1] → *idle*	143	18
S_sleep [3]	69, 53, 89, 120, 127	*S_sleep* [2] → *idle*	135	15
idle	97, 87, 122, 150, 153	*S_sleep* [3] → *idle*	128	10
S_active [1]	110, 104, 140, 153, 159	*idle* → *S_sleep*[1]	100	4
S_active [2]	115, 120, 151, 165, 170	*idle* → *S_sleep*[2]	108	8
S_active[3]	136, 141, 173, 183, 189	*idle* → *S_sleep*[3]	114	10
S_active[4]	167, 172, 195, 220, 227	*idle* → *S_active* ($i \in N_5$)	0	10
S_active[5]	175, 187, 210, 243, 256			

Table 3. Parallel workloads.

Category	Max.n_{t_j}	Applications	Tasks in each app
Low-load	2556	200	64
Mid-load	8192	200	64
High-load	9120	200	64

Table 4. Simulation parameters.

Parameter	Definition	Value
q	Number of clouds.	5
N_{vm}	Number of VMs per cloud.	32000
K	Length of a segment.	120 minutes
k	Length of a timeslot.	2 minutes
ε	Electricity price for data centers.	1.2 $/kwh
ζ	Dataset transmission price.	0.2 kwh/GB
$P_{failure}$	Probability of random failures in data centers.	2%

Table 5. Benchmark algorithm parameters.

Algorithm	Parameter	Definition	Value
MCS	CN	Number of cuckoo nests.	20
	P_a	Possibility of finding the cuckoo's eggs.	0.50
	$Iter$	Maximum number of iterations.	800
	Q	Positive constant.	0.39
	b	Positive constant.	0.40
	$\alpha(i)$	Step size factor α of Levy flight: $\alpha(i) = b - Q \times \frac{\exp\left(\frac{10(i-1)}{Iter-1}\right)-1}{\exp(10)-1}$ $(i = 1, 2, \ldots, Iter)$	Not applicable to a function.
HCPS	PN	Number of particle.	20
	cf_1	Acceleration factor	1.5
	cf_2		1.5
	v	Maximum flying speed of the particles.	[−5, 5]
	ω_{max}	Upper bound of the inertia weight.	1.2
	ω_{min}	Lower bound of the inertia weight.	0.9
	ω^i	Inertia weight: $\omega^i = \omega^i - \frac{\omega_{\max}-\omega_{\min}}{i}$	N/A
	T_{start}	$T_{start} = \frac{Fitness(X_{gBest})}{\lg(5)}$, where $Fitness(X_{gBest})$ is the fitness function value of X_{gBest}. T_{start} denotes the initial temperature of SA.	N/A
	$F_{cooling}$	Temperature cooling rate.	0.95
	$Iter$	Maximum number of iterations.	1500
	p and v	Positions and velocities of particles are updated using Metropolis acceptance criterion.	N/A
MABC	CZ	Number of colonies.	30
	FS	Number of food source.	20
	PL	Number of leader bees.	20
	PF	Number of follower bees.	20
	PS	Number of scout bees.	2
	$Iter$	Maximum number of iterations.	800

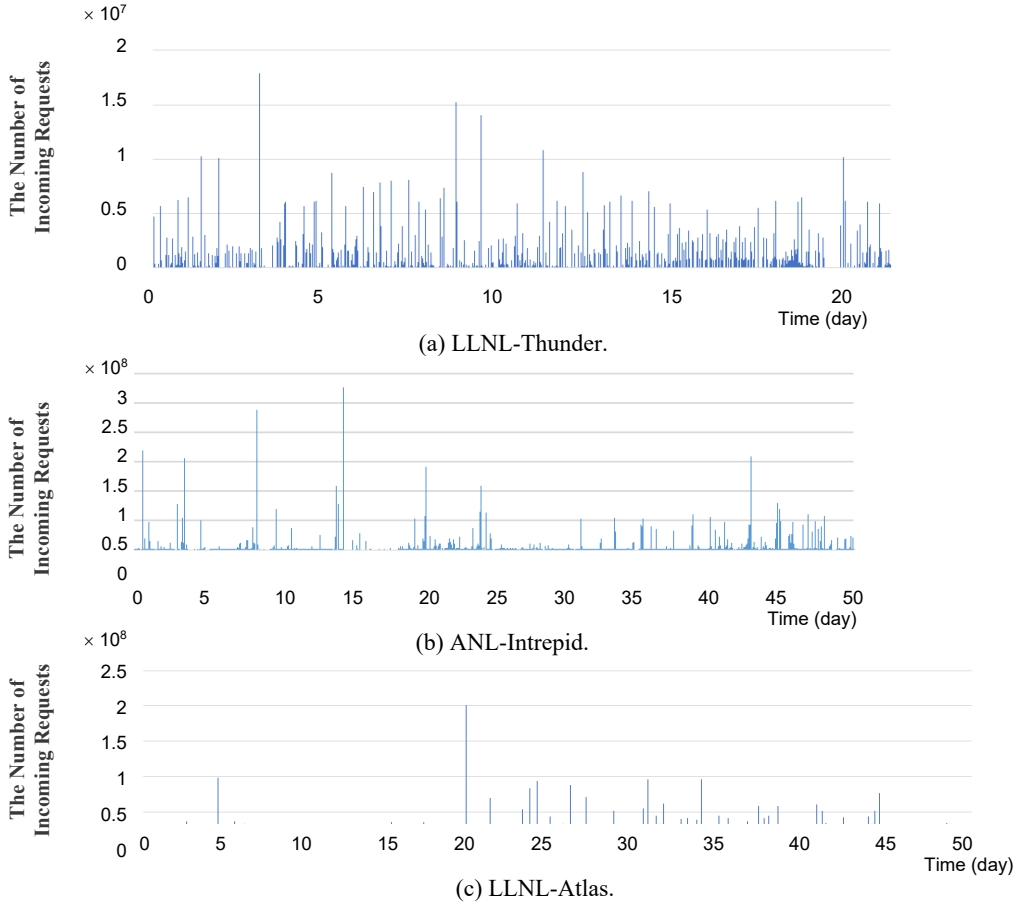

Fig. 2. Different logs of the traces of real events.

We assume that a random failure is given by a probability of 2%. In the simulations, a stochastic fault generation model based on Poisson distribution is implemented and server reservation settings are specified. Table 3 and Fig. 2 show three logs of the real event tracks [41] for the experiments.

As shown in Fig. 3, in the sense of energy optimization, the adopted DPM-DVFS technology has the best performance. Compared with conventional DVFS, the combination of dynamic server on/off scheduling and multi-sleep operations can greatly reduce the total energy consumption.

Suppose the deadline for executing a submitted application is 120% of its estimated execution time. All tasks in applications are CPU-bound.

Fig. 4 shows the number of application rejections for MCS, HCPS, MABC, CRS (using a fixed reservation ratio of servers), and CRS (using a flexible reservation ratio of servers) under various workloads, normal scenario, and failure scenario, respectively. Fig. 5 shows the total cost for MCS, HCPS, MABC, CRS (using a fixed reservation ratio of servers), and CRS (using a flexible reservation ratio of servers) under various workloads, normal scenario, and failure scenario, respectively.

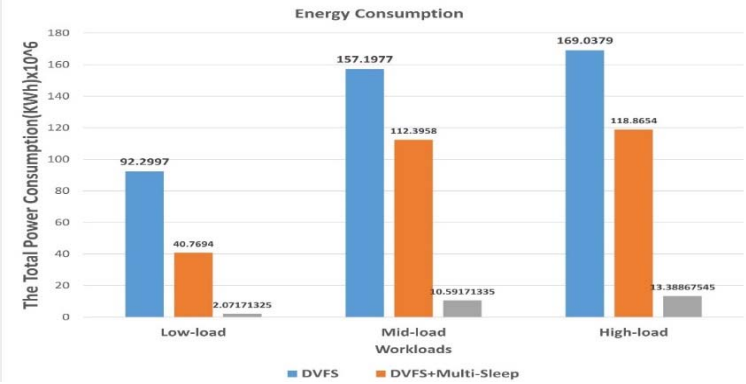

Fig. 3. Energy consumption.

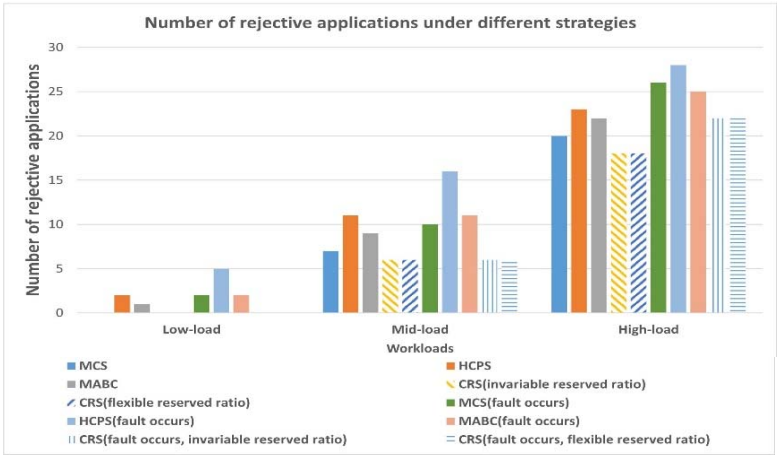

Fig. 4. Number of application rejections under different strategies.

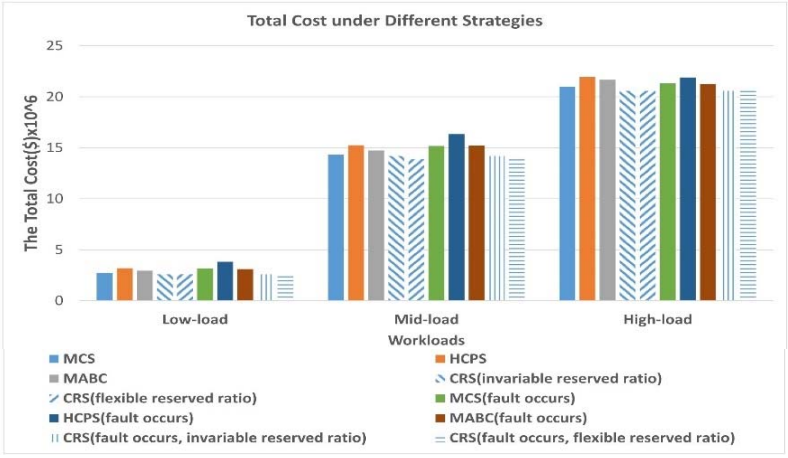

Fig. 5. Total cost under different strategies.

Regardless of failures, both cases CRS perform better than the other three benchmark algorithms under different workloads. CRS with a flexible reservation ratio of servers, compared with CRS with a fixed reservation ratio of servers, further reduces the cost of application execution while reducing the number of application rejections.

As can be seen from the experimental results, compared with other benchmark algorithms, CRS reduces the cost by 18% on average, and the number of application rejections also decreases about 21%. MCS performs better in reducing the cost and the number of application rejections than the other two benchmark algorithms do. The simulation for each benchmark algorithms has been performed for many times independently. The total cost is regarded as a measure to evaluate the advantages and disadvantages of an algorithm, which can more accurately reflect the performance of different scheduling algorithms than other metrics in practice.

Fig. 6 shows the number of application rejections under different workloads and different server reservations.

In the experiments, it is found that when the reservation ratio for server in the CRS is set to be too small, the number of application rejections becomes high. In the case of heavy workloads, if the reservation ratio of servers is low, it does not fully respond to failures and results in more application rejections due to timeout. The number of application rejections does not decrease when a high reservation ratio of servers is set. Conversely, a high reservation ratio of servers not only increases energy consumption, but also reduces the processing rate of DC. Therefore, we consider different fixed reservation ratio of servers and different flexible reservation ratios of servers, respectively. The result shows that failures affect application differently as the workloads change. In the case of light workloads, we can reduce power consumption by reducing the reservation ratio of servers. Therefore, a flexible reservation ratio of servers should be set to cut down both the total cost and the number of application rejections.

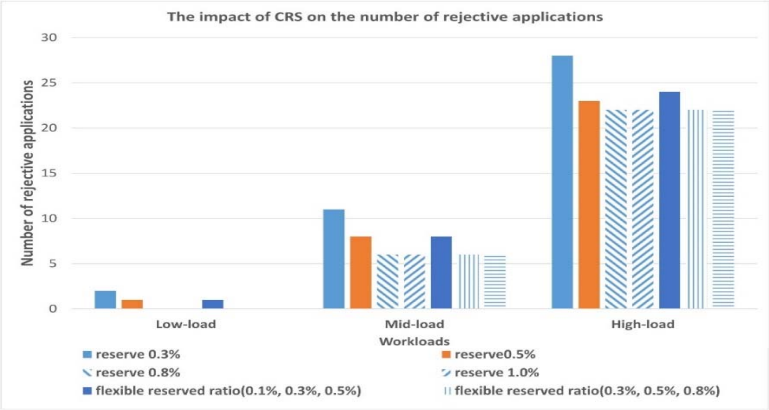

Fig. 6. The impact of CRS on the number of application rejections.

6. CONCLUSION

This work focuses on the energy consumption for scheduling high-performance computing applications under the premise of servers' failures. It aims to minimize application

rejections and deadline violations caused by server failures to improve resource reliability and save energy. Experimental simulations show that the proposed scheduling method reduces total cost by an average of 18% over the upper bound, which is determined by the highest possible performance in DDCs. Moreover, the method can effectively reduce additional application rejections and violations caused by server failures, with an average reduction by 20%. The results show that the strategies relying on only energy consumption to execute applications may not produce the best energy savings in all cases, nor may they achieve the best results for reducing application rejections. Furthermore, the results show that there is an interdependent relationship between the proposed scheduling decision method and the characteristics of the submitted applications. The reservation number of servers not only affects the results of server failures, but also has an impact on the decision for an application to select a cloud.

ACKNOWLEDGMENT

This research is partially supported by Key-Area Research and Development Program of Guangdong Province, China under Grant 2020B010166006 and the Natural Science Foundation of Guangdong Province, China under Grant 2020A151501482.

REFERENCES

1. H. Yuan, J. Bi, and M. Zhou, "Spatiotemporal task scheduling for heterogeneous delay-tolerant applications in distributed green data centers," *IEEE Transactions on Automation Science and Engineering*, Vol. 16, 2019, pp. 1686-1697.
2. S. Di, D. Kondo, and C. Wang, "Optimization of composite cloud service processing with virtual machines," *IEEE Transactions on Computers*, Vol. 64, 2015, pp. 1755-1768.
3. C. Wang, W. Pedrycz, M. Zhou, and Z. Li, "Sparse regularization-based fuzzy C-means clustering incorporating morphological grayscale reconstruction and wavelet frames," *IEEE Transactions on Fuzzy Systems*, Vol. 29, 2020, pp. 1826-1840.
4. J. Bi, H. Yuan, W. Tan, and B. Li, "TRS: Temporal request scheduling with bounded delay assurance in a green cloud data center," *Information Sciences*, Vol. 360, 2016, pp. 57-72.
5. Y. Wu, M. Tornatore, C. U. Martel, and B. Mukherjee, "Content fragmentation – A redundancy scheme to save energy in cloud networks," *IEEE Transactions on Green Communications and Networking*, Vol. 2, 2018, pp. 1186-1196.
6. C. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," *IEEE Communications Surveys & Tutorials*, Vol. 18, 2016, pp. 2244-2281.
7. H. Yuan, J. Bi, M. Zhou, and A. C. Ammari, "Time-aware multi-application task scheduling with guaranteed delay constraints in green data center," *IEEE Transactions on Automation Science and Engineering*, Vol. 15, 2018, pp. 1138-1151.
8. J. Bi, H. Yuan, and M. Zhou, "Geographical scheduling of multi-application tasks for cost minimization in distributed green data centers," in *Proceedings of IEEE International Conference on Systems*, *Man*, *and Cybernetics*, 2018, pp. 3171-3176.

9. P. Zhang and M. Zhou, "Dynamic cloud task scheduling based on a two-stage strategy," *IEEE Transactions on Automation Science and Engineering*, Vol. 15, 2018, pp. 772-783.
10. A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, Vol. 28, 2012, pp. 755-768.
11. H. B. Alla, S. B. Alla, and A. Ezzati, "A novel architecture for task scheduling based on dynamic queues and particle swarm optimization in cloud computing," *International Conference on Cloud Computing Technologies and Applications*, 2016, pp. 108-114.
12. F. Cao and M. M. Zhu, "Energy-aware workflow job scheduling for green clouds," in *Proceedings of IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber*, *Physical and Social Computing*, 2013, pp. 232-239.
13. S. Iturriaga, S. Nesmachnow, A. Tchernykh, and B. Dorronsoro, "Multiobjective workflow scheduling in a federation of heterogeneous green-powered data centers," in *Proceedings of the 16th IEEE/ACM International Symposium on Cluster*, *Cloud and Grid Computing*, 2016, pp. 596-599.
14. C. Nandhakumar and K. Ranjithprabhu, "Heuristic and meta-heuristic workflow scheduling algorithms in multi-cloud environments – A survey," in *Proceedings of International Conference on Advanced Computing and Communication Systems*, 2015, pp. 1-5.
15. G. S. Aujla, M. Singh, N. Kumar, and A. Y. Zomaya, "Stackelberg game for energy-aware resource allocation to sustain data centers using RES," *IEEE Transactions on Cloud Computing*, Vol. 7, 2019, pp. 1109-1123.
16. W. Zhang, Y. Wen, L. L. Lai, F. Liu, and R. Fan, "Cost optimal data center servers: A voltage scaling approach," *IEEE Transactions on Cloud Computing*, Vol. 9, 2021, pp. 118-130.
17. R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: challenges and opportunities," in *Proceedings of International Conference on High Performance Computing and Simulation*, 2009, pp. 1-11.
18. M. Dickinson, S. Debroy, P. Calyam, S. Valluripally, Y. Zhang, R. B. Antequera, T. Joshi, T. White, and D. Xu, "Multi-cloud performance and security driven federated workflow management," *IEEE Transactions on Cloud Computing*, Vol. 9, 2021, pp. 240-257.
19. C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Future Generation Computer Systems*, Vol. 37, 2014, p. 141.
20. M. Xu, V. D. Amir, and R. Buyya, "Energy efficient scheduling of cloud application components with brownout," *IEEE Transactions on Sustainable Computing*, Vol. 1, 2016, pp. 40-53.
21. W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proceedings of International Symposium on High-Performance Computer Architecture*, 2008, pp. 123-134.
22. R. Ge, X. Feng, and K. W. Cameron, "Performance-constrained distributed DVS

scheduling for scientific applications on power-aware clusters," in *Proceedings of ACM/IEEE Supercomputing Conference*, 2005, p. 34.
23. J. Li, Z. Li, K. Ren, and X. Liu, "Towards optimal electric demand management for internet data centers," *IEEE Transactions on Smart Grid*, Vol. 3, 2012, pp. 183-192.
24. M. Herlich, N. Bredenbals, and H. Karl, "Delayed (de-)activation in servers with a sleep mode," *Sustainable Computing: Informatics and Systems*, Vol. 10, 2016, pp. 48-55.
25. S. K. Panda and R. K. Jana, "An efficient task scheduling algorithm for heterogeneous multi-cloud environment," *Arabian Journal for Science and Engineering*, Vol. 43, 2018, pp. 919-933.
26. Y. Laalaoui and J. Al-Omari, "A planning approach for reassigning virtual machines in IaaS clouds," *IEEE Transactions on Cloud Computing*, Vol. 8, 2020, pp. 685-697.
27. J. Simao and L. Veiga, "Partial utility-driven scheduling for flexible SLA and pricing arbitration in clouds," *IEEE Transactions on Cloud Computing*, Vol. 4, 2016, pp. 467-480.
28. H. Wang, Y. Li, Y. Zhang, and D. Jin, "Virtual machine migraion planning in software-defined networks," *IEEE Transactions on Cloud Computing*, Vol. 7, 2019, pp. 1168-1182.
29. F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Future Generation Computer Systems,* Vol. 78, 2018, pp. 257-271.
30. H. Yuan, J. Bi, and M. Zhou, "Revenue-sensitive scheduling of multi-application tasks in software-defined cloud," in *Proceedings of Conference on Automation Science and Engineering*, 2017, pp. 1566-1571.
31. R. Xie and X. Jia, "Data transfer scheduling for maximizing throughput of big-data computing in cloud systems," *IEEE Transactions on Cloud Computing*, Vol. 6, 2018, pp. 87-98.
32. Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, 2016, pp. 1344-1357.
33. D. A. Alboaneen, H. Tianfield, and Y. Zhang, "Glowworm swarm optimisation based task scheduling for cloud computing," in *Proceedings of the 2nd International Conference on Internet of Things*, *Data and Cloud Computing*, 2017, pp. 1-7.
34. A. Alsughayyir and T. Erlebach, "A bi-objective scheduling approach for energy optimisation of executing and transmitting HPC applications in decentralised multi-cloud systems," in *Proceedings of the 16th International Symposium on Parallel and Distributed Computing*, 2017, pp. 44-53.
35. C. Gu, Z. Li, H. Huang, and X. Jia, "Energy efficient scheduling of servers with multi-sleep modes for cloud data center," *IEEE Transactions on Cloud Computing*, Vol. 8, 2020, pp. 833-846.
36. A. B. Alam, M. Zulkernine, and A. Haque, "A reliability-based resource allocation approach for cloud computing," in *Proceedings of IEEE 7th International Symposium on Cloud and Service Computing*, 2017, pp. 249-252.
37. H. Yuan, J. Bi, W. Tan, and B. H. Li, "CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers," *IEEE Transactions on Automation Science and Engineering*, Vol. 13, 2016, pp. 976-985.

38. H. Yuan, J. Bi, and M. Zhou, "Profit-sensitive spatial scheduling of multi-application tasks in distributed green clouds," *IEEE Transactions on Automation Science and Engineering*, Vol. 17, 2020, pp. 1097-1106.
39. S. Fu, "Failure-aware resource management for high-availability computing clusters with distributed virtual machines," *Journal of Parallel and Distributed Computing*, Vol. 70, 2010, pp. 384-393.
40. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, Vol. 41, 2011, pp. 23-50.
41. D. G. Feitelson, D. Tsafrir, and D. Krakov, "Experience with using the parallel workloads archive," *Journal of Parallel and Distributed Computing*, Vol. 74, 2014, pp. 2967-2982.

Qing-Hua Zhu (朱清華) received his Ph.D. degree in Industrial Engineering from Guangdong University of Technology, Guangzhou, China. He is currently an Associate Professor at Guangdong University of Technology, Guangzhou, China. His research interests include scheduling and optimization, cloud computing, edge computing, and discrete event systems.

Jia-Jie Huang (黃嘉杰) received the B.S. degree in Information Management and Information System from Heilongjiang University, Harbin, China. He is currently pursuing the M.S. degree at the School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China. His research interests include scheduling and optimization, cloud computing, and edge computing.

Yan Hou (侯艷) received the Ph.D. degree in Industrial Engineering from Guangdong University of Technology, Guangzhou, China. She is currently an Associate Professor at Guangdong University of Technology, Guangzhou, China. Her current research interests include intelligent scheduling and discrete event systems.