Energy-efficient and Quality-aware VM consolidation method

Zhihua Li[a,c*], Xinrong Yu[a], Lei Yu[b], Shujie Guo [a,c], Victor Chang[d]

[a] Department of Computer Science and Technology, School of Internet of Things Engineering, Jiangnan University, Jiangsu Wuxi 214122, China
[b] School of Computer Science, College of Computing, Georgia Institute of Technology, Atlanta, GA, 30332, USA
[c] Jiangsu Provincial Engineerinig Laboratory of Pattern Recognition and Computational Intelligence, Jiangnan University, , Jiangsu Wuxi 214122, China
[d] School of Computing, Engineering and Digital Technologies, Teesside University, Middlesbrough, UK

Abstract: To improve resource utilization and energy efficiency, cloud datacenters use VM consolidation to consolidate VMs to less number of physical machines through VM migration. However, improper VM placement may cause frequent VM migrations and constant on-off switch on physical machines (PMs), which results in decreasing service quality and increasing energy consumption. To address this problem, in this paper, we propose an effective and efficient VM consolidation approach called EQ-VMC with the goal to optimize energy efficiency and service quality. In our approach, a discrete differential evolution algorithm is developed to search the global optimum solution of VM placement. By integrating it with a set of algorithms we propose for effective host overloading detection, VM selection and under-loaded host detection, EQ-VMC effectively reduces energy consumption and improves quality of services (QoS). Extensive simulation demonstrates its effectiveness and shows its advantage compared with previous VM consolidation methods.

Keywords: Virtual Machine placement scheme; Optimization model; Discrete DE algorithm; Virtual Machine consolidation

1. Introduction

Virtual machine (VM) consolidation is a critical mechanism to improve the energy efficiency and resource utilization of cloud computing, by migrating VMs to less number of running physical machines (PMs). However, an improper VM placement during VM consolidation may further incur frequent live VM migration and constant on-off switch of PMs, which lead to serious service quality degradation and resource overhead. Thus, the algorithm effectiveness of VM consolidation is key for efficiently fulling its purpose. At the same time, it is a very challenging issue since it involves multiple different types of resource factors, such as CPU, memory, network bandwidth and disk I/O, while VM workloads have the dynamic and uncertain resource demands.

The VM consolidation problem has been receiving significant attention in recent years. Many methods [1-16] have been proposed with addressing several aspects involved in VM consolidation, including host overloading detection, VM selection, VM placement, and under-loaded PMs detection. The methods [1-3] determine hotspots by comparing the current resource utilization measurements with the given

thresholds. The threshold-based methods however cannot adapt to the dynamic resource utilization and demand uncertainty. A number of works on VM consolidation [4-9] analyzed the characteristic of VMs' resource demands, PMs' workload in data centers and propose statistical methods to predict the VMs' resource demand and PMs' workload to perform VM migration. Many intelligence-related VM consolidation methods [3, 4, 15, 17-25] also have been developed, but they easily trap in local optimal regime and are difficult to obtain an ideal balance between energy consumption, resource utilization and QoS. Some VM consolidation methods [10-15] use heuristic algorithms to search the optimal solution of VM placement for reducing energy consumption but they may suffer easily premature convergence, which leads to sub-optimal solutions.

In this paper, we present an energy-efficient and quality-aware VM consolidation (EQ-VMC) method. EQ-VMC method is a heuristic VM consolidation method that targets to minimize the energy consumption of running PMs while ensuring lowest overloading risk of host resources through a dynamic optimization model for VM placement. In our approach, an improved discrete differential evolution (discrete DE) algorithm is developed to search the global optimization solution to find optimal VM placement for the migrated VMs. This algorithm regards all the mappings between VMs and PMs as a population and uses heuristic evolutionary to obtain the optimal VM placement. At the same time, the developed discrete DE algorithm fastens the searching process for the global optimization solution by employing the strategy of multi-evolution routes. Additionally, we propose a set of methods for different phases in VM consolidation including host overloading detection, VM selection and under-loaded host detection. Our major contributions can be summarized as follows.

(1)First, we note that VM placement is an authentic combination optimization issue with multiple resource constraint;

(2)Then, the probable mappings between VMs and PMs are abstracted as a piece of limited search space, and which corresponds to a population of heuristic evolutionary algorithm. Each individual of population is identical to a real mapping between VMs and PMs during a cycle of VMs consolidation;

(3)Next, we define a combination optimization model for handling VM placement to achieve the optimal mapping between VMs and PMs in the search space. The solution of the optimization model is performed by an improved heuristic evolutionary algorithm to guarantee the globally optimal results, namely, the optimum VM placement scheme;

(4)At last, the proposed EQ-VMC method integrates sub-algorithms on host overloading detection, VM selection and under-loaded host detection for VMs consolidation. Comparison and validation are performed using the CloudSim toolkit. The experimental results show that the presented EQ-VMC method is promising in degrading energy consumption and host overloading risk, as well as in improving QoS. Thereby its effectiveness and efficiency have been validated.

The rest of this paper is organized as follows. The related works are introduced in section 2. In section 3, we describe the optimization model and scheme for VM

placement. In section 4, we derive the framework of the EQ-VMC method and propose the overview and the detailed design of it. Extensive experiment results are given in section 5, followed by the concluding remarks and future works in section 6.

2. Related work

A VM consolidation scheme should determine the migrated VMs from where and where to, as well as the PMs that can be turned off, identically, the issue of identifying the source and destination hosts for live VM migration. Many works [3,15,17,18,2,4,20,21,22,23,24,40] handle this issue from different perspectives, such as VM placement [3,15,17,18], host overloading detection [2,3,4,20,21,22,18], and VM migration selection [3,23,24]. In this paper, we focus on heuristic evolutionary based VMs consolidation methods, thus we mainly discuss the topic-related VM placement and VMs consolidation methods.

2.1 VM placement

Efficient VM placement is critical for VM consolidation, due to the fact that an inappropriate VM placement scheme easily reduces resource utilization and increases energy consumption, possibly leading to risk of new host overloading. The core of VM placement addresses the issue of "where to" for live VM migration, and it has been approximately characterized as a bin packing problem [17, 26-28]. Mishra and Sahoo [27] handled this issue as a multi-dimensional bin packing problem, and optimized the mapping between VMs and PMs as specific optimization objectives by using an improved genetic algorithm [26], first-fit decreasing (FFD) algorithm [26], and other intelligence optimization algorithms. Although both Kaaouache et al. [26] and Mishra and Sahoo [27] addressed improving the quality of service (QoS) and resource utilization by optimizing the mapping, both of them did not regard the dynamic scales of the up-allocating VMs and PMs, which easily results in frequent VM migrations and constant switching between on and off on PMs. Best fit decreasing (BFD) [38] is an effectively heuristic algorithm employed to resolve this packing issue. A Power-Aware BFD (PABFD) algorithm [17], which is based on BFD, is presented. The PABFD algorithm first performs the unallocated VMs in descending order based on their CPU resource request and then allocates each VM to the destination host according to this order. Each VM deploys on a PM with minimal increase of energy consumption. However, both BFD and PABFD algorithms do not consider the workload changes of the destination host after VM placement, which may incur a new risk of host overloading and thus cannot ensure the QoS. J. A. Aroca et al. [28] performed competitive ratio analysis on approximate solutions for the VM placement issue with restrictions on the number of VMs and PMs. Melhem et al. [29] proposed a Markov prediction model for VM placement in live VM migration to determine the set of candidate destination hosts that would be able to receive the migrated VMs in a way that avoids their VM migration in the near future. However, they did not consider how to determine the under-loaded, over-loaded, or normal loaded status, which is also important for VM consolidation.

2.2 Heuristic VM consolidation

VM consolidation primarily includes host overload detection, VM migration selection, VM placement, and running hosts shrinking [2, 13]. Due to the complexity of VM consolidation, the issue of VM consolidation [13] was divided into several sub-problems, and the task of VMs consolidation was then conducted by handling and integrating the sub-problems. When using competitive ratio analysis, this type of method was very effective in practice in terms of the viewpoints [39], even though it was unable to guarantee optimal results theoretically. So far, a large number of studies [1-16] have addressed the VM consolidation involved in the different phases, and heuristic algorithms have been implemented for VM consolidation, owing to their outstanding performance in resolving the complex multi-objective optimization model. Several researches [10, 11, 13-15] have explored this issue, and their performance was relatively benchmark. Telenyk et al. [14] aimed to improve QoS in terms of reducing energy consumption and proposed to minimize the imbalance between each resource of the PMs with a simulated annealing algorithm to find the optimal VM placement. However, the presented SA-VMC method did not consider stochastic demands, which could result in new overloading risk for further load imbalance. Generally speaking, fewer running PMs implies lower energy consumption in data centers. Based on this idea, Hallawi et al. [11] developed a multi-objective optimization model, minimizing both the number of running PMs and total resource wastage. The genetic algorithm COFFGA was proposed to perform VM consolidation, using chromosome encoding to represent the order of VM placement and obtaining the best order of VM placement via evolution. However, COFFGA employs an FFD algorithm for VM placement, which easily results in insufficient reserved resources in PMs, and suffers from reduction of QoS to improper optimization objective. Farahnakian et al. [10] proposed to minimize the total number of running PMs with the goal of reducing the frequency of VM migration (VMMs) to improve QoS. They proposed an algorithm called Ant Colony System (ACS) to resolve the optimization model. In ACS, a set of tuples, T, is created, where each tuple consists of three elements: the source PM, the VM to be migrated, and the destination host, of which each tuple represents a sketch of VM migration. ACS finds the best tuple set from the total probable VM migration via the ant colony algorithm. The proposed ACS-VMC decreased energy consumption by consolidating VMs and reducing the total number of running PMs. But the objective of minimizing VMMs easily causes new host overloading risk, further increasing SLA violations. Gao et al. [15] proposed VMPACS, which utilizes an ACS algorithm to address VM consolidation with the goal of reducing energy consumption directly and minimizing resource wastage. VMPACS searches for the optimum VM placement balancing the available resource on each PM along varying dimensions. The method efficiently, simultaneously minimizes total resource wastage and energy consumption. Although, VMPACS is superior to the aforementioned algorithms in reducing energy consumption, it did not regard how to reserve resources of PMs to guarantee QoS. Li et al. [13] proposed to search optimal mapping between VMs and PMs for live VM migration to minimize energy consumption and VMMs with simulating artificial bee colony foraging behaviour. In order to guarantee QoS, the research took overloading

probability as a constraint condition for each running PMs. However, the convergence speed of the algorithm was relatively slow.

The proposed studies treat the researches on VM placement model and the model-related VM consolidation method. The proposed model focuses on the optimal balance between the energy consumption and QoS. Accordingly, we present an EQ-VMC method. The major differences from the previous works are as follows.

(1)First, the presented VM placement model focuses on the balance between energy consumption and hosts' workload stability during ongoing VMs consolidation;

(2)Next, searching the finally optimum results in the solution space (e.g., population) is performed by the improved heuristic evolutionary algorithm, and it not only guarantees the global optimization result but fasten the evolutionary process;

(3)At last, by integrating several sub-algorithms with discrete DE based VM placement algorithm, EQ-VMC globally addresses the issue "where from and where to" of live VM migration, thereby fundamentally guaranteeing the reliability of the results. The extensive experiment results demonstrate that the proposed EQ-VMC manner efficiently reduces energy consumption and host overloading risks while effectively improving QoS.

3. System model and problem formulation

3.1 Data center model

Suppose a data center consists of a set of PMs denoted by $H=\{h_1,h_2,\mathrm{K},h_j,\mathrm{K},h_n\}$, and the VMs are represented as $V=\{v_1,v_2,\mathrm{K},v_i,\mathrm{K},v_m\}$. The deployment relation between VMs and PMs is expressed as a mapping matrix $D_{m\times n}=\left(d_1^T,d_2^T,\mathrm{K},d_i^T,\mathrm{K},d_m^T\right)^T$, in which each vector d_i represents a mapping relation between the virtual machine v_i to all PMs in data centers. The vector d_i is called the deployment vector for the virtual machine v_i. If the virtual machine v_i is deployed on the physical machine h_j, then the j-th element in the vector $d_{i,j}=1$, otherwise $d_{i,j}=0$. Because each VM can only be allocated on a single PM, the deployment vector d_i satisfies $\sum_{j=1}^{n} d_{i,j}=1$, which is a constraint condition for the VM placement.

For each resource, the configured resource capacity of virtual machine v_i is represented by r_i^{res},where $res\in\{cpu,mem,band\}$. Each type of resource capacity on the host h_j is denoted as c_j^{res}. The utilization of each resource on a host is computed as (1).

$$u_j^{res}=\sum_{v_i\in V_j} a_i^{res} \Big/ c_j^{res} \tag{1}$$

where V_j indicates the set of VMs deployed on host h_j, and a_i^{res} represents the real usage of resource of the virtual machine v_i in the host .

As for a physical machine h_j, the total usage for certain type of resources from all deployed VMs cannot be more than the resource capacity in VMs, namely, $a_i^{res}<r_i^{res}$.

So, for the destination host, the total requested resource from the deployed VMs must satisfy the following constraint condition.

$$\sum_{v_i \in V_j} r_i^{res} < c_j^{res} \tag{2}$$

3.2 Energy consumption estimation

Due to the PMs' hardware heterogeneity and running status, the energy consumption of the identical VM deployed on different PMs will also be different. We follow the energy consumption model proposed in the work [13] that characterizes the relation between CPU utilization and energy consumption. It says that given a number p of intervals $\{[0,1/p),[1/p,2/p),\text{K},[(p-1)/p,1]\}$ on the average CPU resource utilization, the host energy consumption linearly increases with the CPU resource utilization ratio in each interval. Thus, the energy consumption of PMs can be estimated as shown in formula (3).

$$PM_j^{power}\left(u_j^{cpu}\right)=\begin{cases}\lambda_1 \cdot u_j^{cpu}+\eta_1, & 0 \le u_j^{cpu} < 1/p \\ \lambda_2 \cdot u_j^{cpu}+\eta_2, & 1/p \le u_j^{cpu} < 2/p \\ \text{M} & \text{M} \\ \lambda_p \cdot u_j^{cpu}+\eta_p, & (p-1)/p \le u_j^{cpu} \le 1\end{cases} \tag{3}$$

where $\lambda_i (i=1,2,\text{K},p)$ is the slope of the linear function of each power interval, $\eta_i (i=1,2,\text{K},p)$ is energy consumption at different load levels in watts with respect to the hardware heterogeneity of PMs (e.g., Table 2).

Based on formula (3), the energy consumption for a host within a certain period of time can be estimated as

$$EC\left(h_j\right)=\gamma_j \cdot \int_{t_0}^{t_1} PM_j^{power}\left(u_j^{cpu}(t)\right)dt \tag{4}$$

where $\gamma_j \in \{0,1\}$, if $\gamma_j = 1$ then the host h_j is running; otherwise, if $\gamma_j = 0$, the host h_j is in sleep mode.

3.3 Host overloading probability estimation

Host overloading probability reflects the overloading risk of PMs under the current deployed VMs. When the resource utilization gets high, the uncertainty of workload will increase the host overloading risk. In this paper，we follow the approach [37]. It characterizes the stochastic variation of VM resource requests by normal distribution, and then estimates the distribution of each resource usage on hosts. The host overloading probability is determined as.

$$P_{over}\left(h_j\right)=1-\prod_{res \in \{cpu,mem,band\}} \Pr_{res}\left(\sum_{v_i \in V_j} r_i^{res} < c_j^{res}\right) \tag{5}$$

where $\Pr_{res}$ is a probability distribution function of various resource usage on a host.

3.4 Problem formulation for VM placement

In terms of the aforementioned energy consumption estimation for data centers and analysis of host overloading risk, the optimization problem for VM placement in datacentres can be defined as shown in formula (6). In fact, formula (6) attempts to obtain the optimum mapping relation between VMs and PMs by taking the

mathematical expectation of energy consumption of PMs with lowest overloading risk to realize the optimization objective.

$$\begin{aligned} \arg\min \quad & f(D) = \sum_{\substack{f:vm_i \to h_j \\ f \in D}} EC(h_j) \cdot e^{\alpha \cdot P_{over}(h_j)} \\ s.t. \quad & \sum_{j=1}^{n} d_{i,j} = 1, i = 1,2,\mathrm{K}\ ,m \\ & c_{j,res} \cdot u_{j,res} + \sum_{i=1}^{m} r_{i,res} \cdot d_{i,j} < c_{j,res}, j = 1,2,\mathrm{K}\ ,n, res \in \{cpu, mem, band\} \end{aligned} \tag{6}$$

where the host overloading probability is $P_{over}(h_j) \in [0,1]$, the function $e^{\alpha \cdot P_{over}(h_j)}$ adjusts the efficacy of the host overloading probability. Once the overloading risk increases during VM placement, the importance of host overloading probability arises exponentially. This policy has the effect of enhancing the sensitivity of the model to host overload probability. Further, the parameter α is used to control the impact of overloading probability on the optimization objective. The greater the value of α is, the more sensitive the destination hosts is to the host overloading probability. Unfortunately, when the host overloading probability tends to be "0", it is easy to cause the phenomenon that the identically migrated VM is able to migrate to multiple under-loaded destination hosts. Under such situation, the VM placement can be determined according to energy consumption. Apparently, it can be seen that formula (6) can fully achieve the optimization objectives of minimizing energy consumption of running PMs with minimal overloading risk.

4. Our VM consolidation approach

In this section, we present our VM consolidation approach. It consists of multi-resource host overloading detection (MHOD) algorithm, QoS-aware VM selection (QVMS) algorithm, discrete DE based VM placement (Discrete DEVMP) algorithm, and under-loaded hosts detection (ULHD) algorithm. These four algorithms cooperate and integrate with each other for live VM migration with guaranteeing QoS, improving resource utilization and reducing energy consumption. We describe them one by one in the follows.

Additionally, for the clarity, all used notations and their meanings are listed below.

v_i	the i-th VM
V	the set of VMs in datacentres
V_{mig}	the set of migrated VMs
h_j	the j-th PM
H	the set of PMs in data center
H_{active}	the set of active PMs
H_{over}	the set of overloading PMs
H_{mig}	the set of new destination PMs
H_s	the set of PMs need under-loaded detection
H_{sp}	the set of PMs are suitable to place VMs from under-loaded PM
m	the number of VMs
n	the number of PMs

$D_{m\times n}$	a mapping matrix denotes the deployment relation between VMs and PMs
d_i	the deployment vector for v_i
$d_{i,j}$	the deployment component for v_i to h_j
res	the resource types of PMs and the set is $\{cpu, mem, band\}$
r_i^{res}	the resource demand of v_i
a_j^{res}	the real amount of resource allocated to the VMs on h_j
c_j^{res}	the resource capacity of h_j
u_j^{res}	the resource utilization of h_j
p	the number of intervals on CPU utilization
λ_i	the slope of the linear function of each power interval
η_i	the intercept of the linear function of each power interval
γ_j	the flag whether h_j is active or not
$\Pr_{res}$	the normal distribution function of various resource usage on a host
α	the weight of overloading probability in optimization model
ω	the safe parameter to adjust threshold
X^t	the population of t-th generation
S	the size of population
D_k^t	the k-th individual in population
D_Δ	the difference matrix
D_{r1}, D_{r2}, D_{r3}	three random selected individuals from population
D_k^m	the k-th mutant individual in population
$d_{k,i}^m, d_{k,i,j}^m$	the deployment vector and deployment component in D_k^m
D_k^c	the k-th crossover individual in population
$d_{k,i}^c, d_{k,i,j}^c$	the deployment vector and deployment component in D_k^c
F	a scalar factor and its value range is $[0,2]$
$randf$	a random variable and its value range is $[0,2]$
CR	a cross constant and its value range is $[0,1]$
$randc$	a random variables and its value range is $[0,1]$
$randi$	a random data in the sequence $\{1,2,\mathrm{K},m\}$
$HMatrix^t(i,j)$	an element in heuristic information matrix of the t-th generation
β	the relative weight of two factors for initial heuristic values
ρ	the parameter adjusts the proportion between previous generation heuristic information and current heuristic information
Δc_{j+i}^{cpu}	the remain CPU resource of h_j after hosting v_i on it
u_{j+i}^{res}	the resource utilization of h_j after hosting v_i on it
v_{mig}	the migrated VMs
a_{mig}^{mem}	the allocated memory for v_{mig}
$\Pr_{res}^{-v_{mig}}$	the normal distribution of a resource usage in which hosts exclude v_{mig}
EXP	the rate of exploratory evolution and its value range is $[0,1]$
$rand_{exp}$	a random value and its value range is $[0,1]$
$\xi_j^{violate}$	the SLAV duration resulting from overloaded CPU resources for h_j
ξ_j	the running time of h_j

R_i^{mig}	the size of the unsatisfied demand for CPU resources as a result of v_i migration
R_i	the size of the demand for CPU resources from v_i

4.1 VM Placement

To resolve the VM placement problem formulated in section 3.4, we present an improved Discrete DE algorithm in the following.

4.1.1 Discrete DE algorithm

Differential evolution (DE) algorithm [30] consists of three operations including mutation, crossover and selection, in which mainly handles continuous variables. The population of traditional DE algorithm consists of several vectors, and the value in each dimension is continuous. However, in this paper, we have to conduct the discrete variable, thus need to discretize the typical DE algorithm. Given a population with S pieces of individual denoted as $X^t = \left\{D_0^t, D_1^t, \mathrm{L}\ , D_k^t, \mathrm{L}\ , D_S^t\right\}$, in which D_k^t is a matrix and t is the generation of population. The detailed processes are as follows.

A. Mutation

Arbitrarily select three individuals $D_{r_1}, D_{r_2}, D_{r_3}$ from the population X^m, in which D_k^m is performed as

$$D_k^m = D_{r1} + F \cdot (D_{r2} - D_{r3}), k = 1, 2, \mathrm{K}\ , S \tag{7}$$

where $F \in [0, 2]$, which is a scalar factor that adjusts the impact to the difference vector. The i-th row in difference matrix D_Δ is determined according to formula (8).

$$d_{\Delta,i} = d_{r2,i} - d_{r3,i} = \begin{cases} d_{r2,i}\ , d_{r2,i} \neq d_{r3,i} \\ \mathrm{r} \\ 0\ ,\quad others \end{cases} \quad i = 1, 2, \mathrm{K}\ , m \tag{8}$$

where $d_{\Delta,i}, d_{r2,i}, d_{r3,i}$ represent the i-th row corresponding to the matrix D_Δ, D_{r2}, D_{r3} respectively. Formula (8) obtains the result (e.g., the row in the difference matrix) by determining whether the deployment vectors at the locations corresponding to the two matrices are equal. Further, calculate the mutation individual D_k^m by formula (9).

$$d_{k,i}^m = d_{r1,i} + F \cdot d_{\Delta,i} = \begin{cases} d_{\Delta,i}\ , 0 < randf < F\ \&\ d_{\Delta,i} \neq \overset{1}{0} \\ d_{r1,i}\ , \qquad\qquad others \end{cases} \quad i = 1, 2, \mathrm{K}\ , m \tag{9}$$

where $d_{k,i}^m$ is the i-th row in matrix D_k^m, $randf$ is a random variable and $randf \in [0,\ 2]$.

B. Crossover

The cross-calculation is examined by formula (10).

$$d_{k,i}^c = \begin{cases} d_i^m, & randc \leq CR \text{ or } i = randi \\ d_i, & randc > CR\ \&\ i \neq randi \end{cases} \quad i = 1, 2, \mathrm{K}\ , m \tag{10}$$

where $d_{k,i}^c$ is the i-th row in the cross-mapping matrix D_k^c, $CR \in [0,1]$ and is a cross constant, $randc$ is a random variables and $randc \in [0,1]$, $randi$ is a random data in the sequence $\{1, 2, \mathrm{K}\ , m\}$.

C. Selection

Selection operation is performed by comparing the value of f one by one between the original individuals in the initial population and corresponding cross individuals, select individuals with small value of f to join next-generation populations. The detailed selection process is shown as formula (11).

$$D_k^{t+1} = \begin{cases} D_k^c, & f\left(D_k^c\right) < f\left(D_k^t\right) \\ D_k^t, & others \end{cases}, k = 1, 2, \mathrm{K}, S \tag{11}$$

4.1.2 Discrete DE based VM placement algorithm

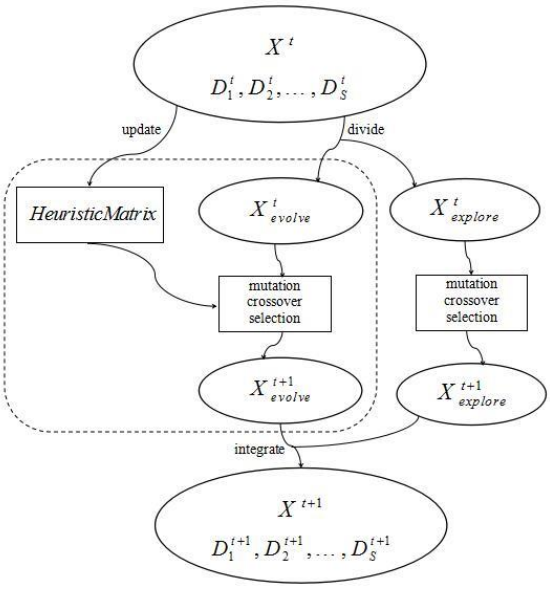

Fig.1 The concept diagram of discrete DE algorithm

The mappings between PMs and VMs in data centers form a limited search space, and it is abstracted as a population corresponding to the X^t in section 4.1.1, where each individual D_k^t in population represents one of the probable mapping matrixes between VMs and PMs. The i-th row in D_k^t is the deployment vector of the virtual machine v_i, namely, $d_{k,i}^t$ essentially refers to the deployment map on all running PMs for virtual machine v_i and is the deployment vector. The corresponding $d_{k,i,j}^t$ represents the deployment component of the deployment vector $d_{k,i}^t$. Here, if the individual obtained by the mutation, crossover, and selection operations by the Discrete DE algorithm is the final optimal mapping relation between VMs and PMs, which is called VM placement scheme in this paper.

In this section, we propose an improved Discrete DE algorithm to solve the VM placement problem given in the model Eq. (6). In the improved Discrete DE algorithm, we additionally employ a policy of heuristic information matrix to fasten the evolutionary process. The concept diagram of discrete DE algorithm shows as Fig.1.

In Fig.1, given generating the i-th generation population X^t, the individuals are randomly separated into two evolution routes including X_{evolve}^t and $X_{explore}^t$. The route of X_{evolve}^t depends on the heuristic information matrix, the $X_{explore}^t$ directly explore the results in the search space. The heuristic information matrix $HMatrix$ is updated with the current generated population at the same time, which speeds up the

evolution and avoiding premature local optimization. Finally, a new population X^{t+1} is achieved.

In the follows, we discuss the different parts including the update of heuristic information matrix, the evolution routs of X^t_{evolve} and $X^t_{explore}$ respectively.

A. The update of heuristic information matrix

In Fig.1, each element in the heuristic information matrix has a one-to-one correspondence with each deployment component. For example, supposing $HMatrix^t(i,j)$ records the heuristic information of the deployed virtual machine vm_i on host h_j in the current population X^t. At this time, if the virtual machine v_i can be treated to deploy on host h_j, the initial value of $HMatrix^0(i,j)$ can be computed as formula (12), else the initial value is 0.

$$HMatrix^0(i,j) = \beta \cdot (\Delta c^{cpu}_{j+i} / \Delta PM^{power}_{j+i}) + (1-\beta) \cdot 1 / std(u^{cpu}_{j+i}, u^{mem}_{j+i}, u^{band}_{j+i}) \quad (12)$$

where $\Delta c^{cpu}_{j+i} = c^{cpu}_j - (a^{cpu}_j + r^{cpu}_i)$ is the remain CPU resource of h_j after hosting v_i on it , $\Delta PM^{power}_{j+i} = PM^{power}_j(1) - PM^{power}_j((\sum_{v_i \in V_j} a^{cpu}_i + r^{cpu}_i) / c^{cpu}_j)$ is the remain power of h_j, the larger $\Delta c^{cpu}_{j+i} / \Delta PM^{power}_{j+i}$ represents the higher cost performance of h_j for other VMs. $std(u^{cpu}_{j+i}, u^{mem}_{j+i}, u^{band}_{j+i})$ is the standard deviation of resources utilization of h_j after hosting v_i, the larger $1/std(u^{cpu}_{j+i}, u^{mem}_{j+i}, u^{band}_{j+i})$ represents more stable resource utilization of h_j after finishing VM placement. β is the weight of these two objectives.

For each D^t_k in population X^t has a corresponding value $f(D^t_k)$ according to the formula (6), the range of f is determined by the deployment vector $d^t_{k,i}\left(e.g., i = 1,2,\mathrm{K}\ ,m\right)$ in the current mapping matrix D^t_k. Usually, the smaller the value of f is, the more favourable the VM placement scheme is, conversely, the worse the VM placement scheme is. Further, quantify the priority level of each deployment components by $1/f(D^t_k)$. In addition, $\sum_{k=1}^{S}\left((1/f(D^t_k)) \cdot d^t_{k,i,j}\right) \Big/ \sum_{k=1}^{S} d^t_{k,i,j}$ records the average priority level of the deployment component in population. At the same time, the heuristic information matrix is updated according to formula (13).

$$HMatrix^{t+1}(i,j) = \rho \cdot HMatrix^t(i,j) + (1-\rho)\left(\sum_{k=1}^{S} \frac{1}{f(D^{t+1}_k)} \cdot D^{t+1}_{k,i,j}\right) \Big/ \sum_{k=1}^{S} D^{t+1}_{k,i,j}, D^{t+1}_k \in X^{t+1} \quad (13)$$

where $\rho \in [0,1]$, and it weights the previous generation of heuristic information and current heuristic information. In terms of the above element value of heuristic matrixes and their update process, it can be found that the situation that a deployment component satisfies $d^t_{k,i,j} = 1$ more frequently appears in D^t_k with a small f value, the heuristic information value $HMatrix^t(i,j)$ of the corresponding deployment component will get bigger. Namely, the element value of the heuristic information matrix reflects the rank of priority and inferiority of the corresponding deployment component, that is, the larger element value of the heuristic information matrix means that the corresponding deployment component of it contributes more weight to the optimization objectives.

B. The evolution of X^t_{evolve}

Under the guidance of the heuristic information matrix, the mutation operation is treated at first. The mutation operation conducts to generate more perfect deployment component of the deployment vector. Given three mapping relation D_{r1}, D_{r2}, D_{r3} in population X^t are randomly selected. The deployment vector $d_{\Delta,i}$ in difference matrix D_Δ is computed as formula (14).

$$d_{\Delta,i} = d_{r2,i} - d_{r3,i} = \begin{cases} d_{r2,i}, & \sum_{j=1}^{n} d_{r2,i,j} \cdot HMatrix^t(i,j) \geq \sum_{j=1}^{n} d_{r3,i,j} \cdot HMatrix^t(i,j) \\ \vec{0}, & others. \end{cases} \quad i = 1,2,\mathrm{K},m \tag{14}$$

where $d_{r2,i}, d_{r3,i}$ denotes the deployment vector of D_{r2}, D_{r3} respectively and $d_{r2,i,j}$, $d_{r3,i,j}$ are the corresponding deployment components. In fact, the formula (14) performs the minus operation by comparison between the heuristic information of the two different deployment vectors. However, when the heuristic information is relatively small, the vector $d_{\Delta,i}$ equals "$\vec{0}$". The obtained differential matrix by formula (14) achieves a relative perfect deployment vector. Additionally, since the obtained differential matrix contains more heuristic information, thus such methods can guide the subsequent evolutionary route to find the optimum VM placement scheme fast. At last, the deployment vector $d^m_{k,i}$ of the mutant individual D^m_k is obtained by formula (9).

Next, the crossover operation treats the mapping relation between the original mapping relation and mutated mapping relation between PMs and VMs. The updated crossover individual may have a mapping relation that contains a slice of superior deployment components identified in heuristic matrix. The crossover individuals D^c_k of the original individuals and mutant individuals are generated by the formula (10). It can be seen that the crossover operation of equation (10) randomly selects the deployment components of mutant individuals to cross individuals. However, when selecting the deployment components of the cross individual one by one, it is necessary to constrain each deployment vector $d^c_{k,i}$ in the crossover individual D^c_k according to the hosts' resource constraints under the presented optimization model, so as to ensure that the crossover individual becomes a VM placement scheme that meets the hosts' resource constraints. When it does not satisfy the hosts' resource constraints, the deployment components of the crossover individual is defined as formula (15).

$$d^c_{k,i,j} = \begin{cases} 1, & \max_j\{HMatrix^t(i,j)\} \,\&\, \sum_{v_i \in V_j} \left(d^m_{k,i,j} \cdot r^{res}_i \right) < c^{res}_j \\ 0, & otherwise \end{cases} \tag{15}$$

Formula (15) targets to find the most appropriate deployment component, namely, getting the i-th deployment vector $d^c_{k,i}$ of the cross-individual D^c_k;

C. The evolution of $X^t_{explore}$

If the individual follows the exploratory evolution route, the corresponding mutant individuals are obtained directly according to formula (8) and (9). Such operation is conducive to fully explore other feasible mutant individuals in the search space.

Similarly, the crossover mapping is the exploration of the original mapping relation to some deployment components, which can effectively avoid the evolution algorithm falling inside local optimization. When the deployment vector $d_{k,i}^{c}$ dissatisfies the the hosts' resource constraints, recalculate each deployment component of $d_{k,i}^{c}$ by formula (16).

$$d_{k,i,j}^{c} = \begin{cases} 1, \sum_{v_i \in V_j} \left(d_{k,i,j}^{m} \cdot r_i^{res} \right) < c_j^{res}, wherein\ random\ j \\ 0, \qquad otherwise \end{cases} \tag{16}$$

where there are randomly selected a deployment component in line with the hosts' resource constraints during VM placement, which fully embodies the generality of the exploratory evolution route.

D. VM placement Algorithm

The optimum mapping relation between VMs and PMs is obtained through mutation, crossover, selection, and heuristic information matrix synchronously updates by Discrete DE algorithm, i.e., the VM placement scheme. Based on this, the Discrete DE based VM Placement (Discrete DEVMP) algorithm is presented. The Discrete DEVMP algorithm is described below.

Algorithm 1：Discrete DE based VM Placement

```
Input：V , H
Output：Map_mig < host, vm >
Generate X^0 = {D_0^0, D_1^0, K , D_S^0} ,  X^t = X^0
Use equation (12) init HMatrix^t
while cur_times < total_times
    Update HMatrix^t using equation (13)
    for all D_k^t in X^t
        if rand_exp < EXP
            Use equation (14),(9) to calculate mutated deployment map D_k^m
            Use equation (10),(15) to calculate cross deployment map D_k^c
        else
            Use equation (8),(9) to calculate mutated deployment map D_k^m
            Use equation (10),(16) to calculate cross deployment map D_k^c
        end if
        D_k^{t+1} = f(D_k^t) < f(D_k^c) ? D_k^t : D_k^c
        Add D_k^{t+1} to X^{t+1}
    end for
end while
D_optimal = arg min_{D_k^t} f(D_k^t)
Convert D_optimal to Map_mig < host, vm >
```

4.2 Host overloading detection

First, we use the normal distribution model to fit the history records of hosts' resource usage, and get the probability distribution function $\Pr_{res}$ of each resource accordingly. Next, the overloading probability of each resource is obtained depend on

the $\Pr_{res}$. Finally, the overloading threshold of each resource is determined according to the overloading probability. The overloading threshold is conducted in terms of formula (17).

$$T_j^{res} = 1 - \omega \cdot \Pr_{res}\left(\sum_{v_i \in V_j} r_i^{res} > c_j^{res}\right) \tag{17}$$

where T_j^{res} is the upper boundary of the current utilization of each resource on host h_j; ω is the adjustment coefficient for weight the overloading threshold. By adjusting its size, the threshold setting can be inclined to make full use of resources or guaranteeing QoS. During host overloading detection, any resource utilization exceeding a corresponding threshold value (e.g., $u_j^{res} > T_j^{res}$) is recognized an occurrence of host overloading. In data centers, all PMs with overloading risk need to be discovered.

Multi-resource host overloading detection (MHOD) algorithm is as follows.

Algorithm 2：MHOD

```
Input：H_active
Output：H_over
for all h_j in H_active
    if u_j^res > T_j^res
        H_over ← H_over ∪ h_j
    end if
end for
```

4.3 VM selection

In data centers, usually, it is necessary to migrate some VMs from the overloaded PMs decrease their overloading risk. Thus this paper proposes two criteria for performing VM selection: (1) the workload on source hosts gets more stable after VM migration; (2) the time for live VM migration should be as short as possible, so as to minimize the negative influence on QoS by VM migration. For the first criterion, it can be measured in terms of the changes of overloading probability on source and destination hosts after VM migration. For the second criterion, using pre-copy mechanism [31], i.e., the smaller the memory usage, the less migration time needs, thus effectively reducing the negative impact by VM migration on QoS. The resource overloading probability of source hosts is estimated in terms of formula (18) after VM migration.

$$P_{over}^{-v_{mig}}\left(h_j\right) = 1 - \prod_{res \in \{cpu, mem, band\}} \Pr_{res}^{-v_{mig}}\left(\sum_{v_i \in V_j} r_i^{res} < c_j^{res}\right), v_{mig} \notin V_j \tag{18}$$

where $\Pr_{res}^{-v_{mig}}$ is the resource utilization of source host h_j after VM migration. v_{mig} represents the migrated VM.

Further, the two above described criteria for selecting the VMs to be migrated are quantified as shown in formula (19), and which is called VM selection criteria.

$$RaP = P_{over}^{-v_{mig}} \cdot \frac{a_{mig}^{mem}}{c_j^{mem}} \tag{19}$$

where a_{mig}^{mem} is the amount of occupied memory by the migrated VM.

For the source host, after VM migration, it is also necessary to continue the

overloading risk assessment to confirm whether the current source host has relieved overloading risk. If the current VM migration does not eliminate the host overloading risk, it needs to repeatedly perform the aforementioned steps until host overloading risk is removed.

QoS-aware VM Selection (QVMS) algorithm is as follows.

Algorithm 3：QVMS

```
Input：H_over
Output：V_migrate
for all h_j in H_over
    while u_j^res > T_j^res
        v_m ← arg min_{v_i ∈ V_j} RaP
        V_mig ← V_mig ∪ v_m ,  V_j ← V_j − v_m
    end while
end for
```

4.4 Under-loaded hosts detection

During VM consolidation, turning off some under-loaded running PMs is useful to reduce energy consumption and improve resource utilization. Therefore, in this section, we address the criteria for under-loaded PMs detection. During the process of shutting down the under-loaded PMs, the following problems need to be addressed: (1) detect the under-loaded PMs; (2) migrate all of the VMs from the under-loaded PM; (3) select the destination host for live VM migration. Obviously, during a cycle of VM consolidation, (1) the under-loaded PMs must be selected from the current running PM set. Because the overloading hosts are in a relatively stable status after VM migration, the previous overloading hosts are impossible to still be at under-loaded status; (2) those hosts that have been selected as destination hosts in the previous cycle of VM consolidation should also be excluded so that the under-loaded PMs candidate set is optimized as $H_s = H_{active} - H_{over} - H_{mig}$. Further, for the under-loaded PM candidate set, the comprehensive utilization of each resource is calculated according to the formula (20), and sorting the comprehensive resource utilization in ascending order, which in turn turns off the low-utilization PMs and migrates all of the VMs on it. The candidate set of destination hosts for the migrated VMs is denoted as $H_{sp} = H_{active} - H_{over}$.

$$u_j = \sqrt{\left(u_j^{cpu}\right)^2 + \left(u_j^{mem}\right)^2 + \left(u_j^{band}\right)^2} \tag{20}$$

Under-loaded hosts detection (ULHD) algorithm is as follows.

Algorithm 4：ULHD

```
Input：H_s , H_sp
Output：Map_s < host, vm >
Sort H_s by equation (20)
for all h_j in H_s
    Get D_s from Discerte DEVMP(V_j , H_sp − h_j)
    if D_s ≠ null
        Convert D_s to map < host, vm >
```

8: $H_{sp} \leftarrow H_{sp} - h_j$
9: end if
10: $Map_s < host, vm > \leftarrow Map_s < host, vm > \text{U} map < host, vm >$
11: end for

4.5 VM consolidation method

The essence of VM consolidation is to optimize the mapping relation between currently running PMs and VMs and improve resource utilization, reduce energy consumption while guaranteeing QoS. This paper takes the optimum mapping relation between VMs and PMs as the optimization objective through minimizing energy consumption of hosts with lowest host overloading risk; and resolves the optimization model with the proposed discrete DE algorithm. A Discrete DE algorithm based VM placement scheme and method are presented. By the full mutation and crossover of the improved discrete DE algorithm, the optimum mapping relation between VMs and PMs is fundamentally guaranteed, and the global optimization result is achieved. In addition, the above presented sub-algorithms, including host overloading detection, VM selection, and under-loaded PMs detection algorithms, are integrated, and a hybrid energy-efficient and quality-aware based heuristic VM consolidation (EQ-VMC) method is proposed.

The detailed EQ-VMC method is as follows.

Algorithm 5：EQ-VMC

1： Input： H， H_{active}
2： Output： $Map < host, vm >$
3： $H_{over} \leftarrow MHOD(H_{active})$
4： $V_{mig} \leftarrow QVMS(H_{over})$
5： $H_{app} = H_{active} - H_{over}$
6： $Map < host, vm > \leftarrow Discrete\ DEVMP(V_j, H_{app})$
7： $H_s = H_{active} - H_{over} - H_{mig}$， $H_{sp} = H_{active} - H_{over}$
8： $Map_s < host, vm > \leftarrow ULHD(H_s, H_{sp})$
9： $Map < host, vm > \leftarrow Map < host, vm > \text{U} Map_s < host, vm >$

5. Experiment

5.1 Simulation Setting

We employed the CloudSim toolkit [32] as the simulation platform. In experiments, since the lower bound on the number of VMs in the employed workload traces is approximately 800, 800 heterogeneous PMs are created in cloud data centers, these hosts include four types, and the specific configuration and the power measurement are shown in Table 1, Table 2 (SPEC) [33]respectively. Besides, six types of VMs are created based on the instance parameters provided by EC2 (Amazon EC2 Product Details)[34], Table 3 shows their configuration parameters. These VMs are randomly deployed on different PMs.

In order to properly simulate the real resource request and response in data centers, two different data sets Bitbrains trace [35] and GoogleClusterTrace [36] are employed to examine the experiments. Bitbrains includes CPU usage, memory, and workload of bandwidth, the selected 10 days trace is shown in Table 4. GoogleClusterTrace just

includes CPU and memory usage. We equally select 10 days data trace, and 1200 items of data trace are selected randomly from each day respectively.The selected trace is shown in Table 5. Additionally, The parameters in EQ-VMC are set as follows, $F=1$, $CR=0.5$, $\alpha=6$, $\beta=0.5$, $\rho=0.5$, $\omega=0.8$, $total\_times=10$, $S=10\times|H|$ and $EXP=0.7$.

Table 1 The PM instances

Type	CPU	RAM(GB)
IBM server X3550 M3-1	Intel Xeon X5670 12 cores 2933Hz	12
IBM server X3550 M3-2	Intel Xeon X5675 12 cores 3067Hz	16
HP Enterprise ProLiant DL 360 Gen9	Intel Xeon E5-2699 v3 36 cores 2300Hz	64
HP Enterprise ProLiant DL 360 Gen10	Intel Xeon Platinum 8180 28 cores 2500Hz	48

Table 2 The energy consumption at different load levels in watts with respect to different PMs

Utilization(%)	0	10	20	30	40	50	60	70	80	90	100
M3-1	66	107	120	131	143	156	173	191	211	229	247
M3-2	58.4	98	109	118	128	140	153	170	189	205	222
Gen9(kWh)	45.0	83.7	101	118	133	145	162	188	218	248	276
Gen10(kWh)	38.7	68.9	82.2	94.6	107	121	138	156	178	210	233

Table 3 The VM instances

Type	CPU frequency (MIPS)	RAM (GB)
m3.medium	2300×1	3.75
m3.large	2300×2	7.5
m3.xlarge	2300×4	15
c4.large	2300×2	3.75
c4.xlarge	2300×4	7.5
c4.2xlarge	2300×8	15

Table 4 The properties of Bitbrains trace

Date	Number of VMs	CPU		Memory		Bandwidth	
		mean(%)	St.dev.(%)	mean(%)	St.dev.(%)	mean(%)	St.dev.(%)
2013-08-02	1237	7.20	5.97	8.83	4.35	0.76	1.84
2013-08-04	1233	8.05	4.83	9.75	4.12	0.80	1.77
2013-08-08	1209	10.27	6.64	9.69	4.52	0.70	1.67
2013-08-10	1205	8.17	5.43	9.47	4.36	0.85	2.00
2013-08-12	1202	9.57	6.36	9.25	3.86	0.79	1.79
2013-08-14	1194	4.88	4.54	8.41	3.33	0.59	1.41
2013-08-18	1189	8.39	3.73	8.95	3.16	0.89	1.88
2013-08-20	1186	8.99	3.45	9.10	3.05	0.79	1.46
2013-08-23	1176	9.44	4.64	9.74	3.90	1.01	2.19
2013-08-25	1175	5.48	4.06	8.40	3.63	1.12	2.39

Table 5 Ten items of GoogleClusterTrace

No.	Number of VMs	CPU	Memory

		mean(%)	St.dev.(%)	mean(%)	St.dev.(%)
1	1200	1.63	0.49	2.16	0.10
2	1200	2.23	0.71	2.30	0.16
3	1200	2.71	0.82	2.31	0.16
4	1200	2.97	0.89	2.25	0.16
5	1200	3.01	0.91	2.27	0.17
6	1200	3.02	0.91	2.24	0.16
7	1200	2.99	0.91	2.23	0.17
8	1200	2.98	0.90	2.21	0.17
9	1200	2.96	0.91	2.20	0.16
10	1200	2.95	0.90	2.20	0.17

5.2 Evaluation Metrics

We use five performance indices to evaluate performance in our experiments [13]: SLA violation time per active host (SLATAH), performance degradation due to migration (PDM), SLA violations (SLAV), energy consumption (EC), and energy and SLA violations (ESV).

a) SLATAH, which measures the service quality of a running PM, is defined as

$$SLATAH = \frac{1}{n}\sum_{j=1}^{n}\frac{l_j^{violation}}{l_j} \tag{21}$$

where $l_j^{violation}$ is the SLAV duration resulting from overloaded CPU resources for a host h_j, l_j the running time of host h_j, and n the number of PMs.

b) PDM, given by (22), reflects the extent of VM migration-related performance decline.

$$PDM = \frac{1}{m}\sum_{i=1}^{m}\frac{R_i^{mig}}{R_i} \tag{22}$$

where R_i^{mig} denotes the size of unsatisfied demand for CPU resources as a result of the migration of a given virtual machine v_i, R_i the size of total demand for CPU resources from v_i, and m the number of VMs.

c) SLAV evaluates the QoS of a data center on a single day:

$$SLAV = SLATAH \times PDM \tag{23}$$

SLATAH, PDM, and SLAV are inversely proportional to QoS.

d) The comprehensive evaluation index ESV, which is defined in formula (24), reflects the energy consumption, VMMs, and service quality.

$$ESV = EC \times SLAV \tag{24}$$

where EC indicates the energy consumption of a data center in a single day, which is determined according to formula (4) in section 3.2. A low ESV value indicates that more energy is saved and guarantees the service quality of data centers.

Since VMs always suspend services during live migration, prolonged VM migrations can further affect QoS. Reducing the number of insignificant VM migrations is beneficial for improving QoS. Therefore, if limited VM migrations can yield ideal effects using a given VM consolidation method, this indicates that the VM consolidation method is highly efficient.

5.3 Experiment Results

In this section, several experiments are arranged to validate the effectiveness and

efficiency of the proposed optimization model and VM consolidation method from different aspects.

5.3.1 Validating the optimization model

To validate the effectiveness and efficiency of the proposed optimization model and VM placement method, this section performs comparison between the presented Discrete DEVMP, PABFD and FFD. We combine and evaluate each of them with 4 identical host overloading detection methods and 3 VM selection methods, resulting in total 36 different combinations. The host overloading detection algorithms [2,3] include benchmark IQR, LR, MAD, ST, the VM selection algorithms [2,3] including MC, MMT, RS. These combined schemes are separated into 12 groups; each group with 3 different colour columns corresponds to Discrete DEVMP, PABFD and FFD respectively. The experimental comparisons using 36 different combination schemes are performed on the same measurement including EC, SLATAH, PDM, SLAV, ESV and VMMs on Bitbrains trace. The results are shown in Figs.2-7.

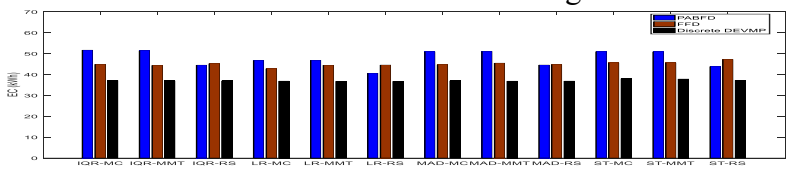

Fig. 2 Comparison of EC

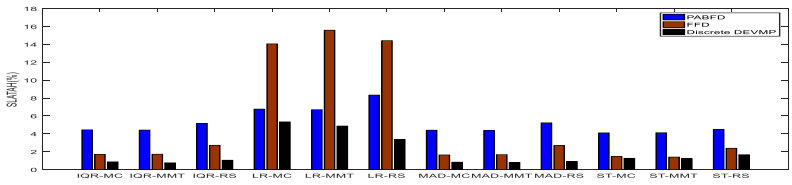

Fig. 3 Comparison of SLATAH

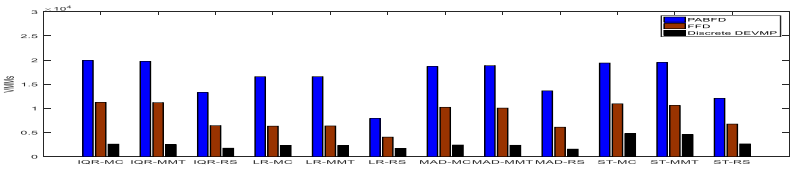

Fig. 4 Comparison of times of VM migrations

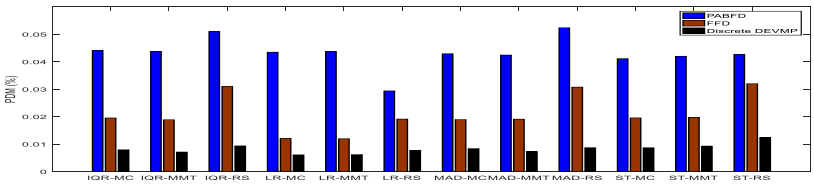

Fig. 5 Comparison of PDM

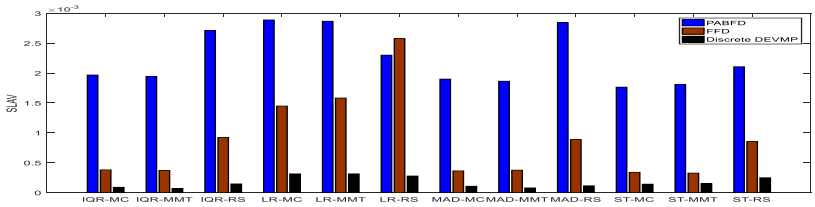

Fig. 6 Comparison of SLAV

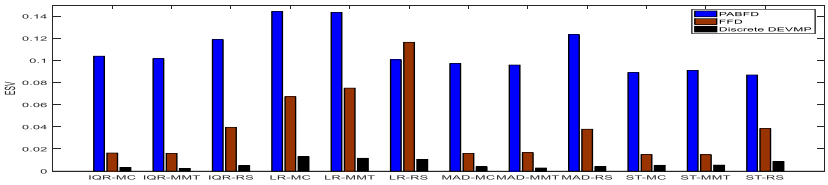

Fig. 7 Comparison of ESV

Figs.2-7 are the comparison results on various performance indices. Fig.2 shows the comparison results based on the EC index between the related schemes. As shown in Fig.2, the EC index of schemes with Discrete DEVMP is the lowest. Fig.3 compares the SLA violation of all 36 combinations. The Discrete DEVMP algorithm performs best in all 36 combinations. Fig. 4 shows VMMs during VM consolidation for each method. The times of VM migration with Discrete DEVMP is the smallest. Fig. 5 shows the PDM-driven performance of combination schemes with Discrete DEVMP is the best, and the schemes using PABFD has the worst performance. Fig. 6 shows the comparison of SLAV on the different schemes. Since the SLAV depends on both SLATAH and PDM, the Discrete DEVMP algorithm outperforms other schemes in terms of SLATAH, PDM SLAV index. Fig. 7 shows the ESV index and the Discrete DEVMP algorithm performs best.

The figures above show that the proposed method is prominent in various indices, which validates the effectiveness of the proposed optimization model and method. In contrast, the VM placement policy employed by PABFD does not sufficiently predict the host overloading risk, which resulting in frequent VM migration and a large number of SLA violations. In addition, due to the local greedy selection of PABFD, it is unable to reduce the number of running PMs immediately, resulting in energy waste. Additionally, as shown in the figures, several combination schemes using LR show a little vibration in EC, SLAV, ESV indices, especially in SLATAH. The most unexpected changes come from the combination of LR and FFD. The primary reasons are that the LR algorithm predicts the resource utilization of hosts through linear regression. When the FFD algorithm performs VM placement according to the prediction results, although it greatly increases the resource utilization, it also leads to the occurrence of excessive VM consolidation and causes the QoS deterioration.

5.3.2 Effectiveness and efficiency evaluation

A. Comparisons in different evaluation metrics

In this section, we compare the proposed EQ-VMC method with four VM consolidation methods, i.e., ACS-VMC [10], SA-VMC [14], EC-VMC [13], COFFGA [11]. The comparison is examined using different data trace on the same indices respectively.

Table 6 Simulation results using Bitbrains trace with four metrics

	EC(kWh)	SLATAH	VMMs	ESV
EQ-VMC	37.31	1.35	1842.2	0.00889
EC-VMC	37.33	1.64	6068.3	0.01197
ACS-VMC	39.39	2.62	3502.9	0.04915
SA-VMC	40.75	1.70	10603.3	0.02241
COFFGA	52.70	1.60	10827.9	0.02363

Table 6 shows the experimental results on the Bitbrains trace with the four indices. For the EC metric, EQ-VMC method is the best, EC-VMC method is the second best,

and COFFGA method is the worst. EQ-VMC and EC-VMC perform very similarly in EC, because they all take the total EC of data centers as one of the optimization objectives for VM consolidation. The COFFGA method aims at minimizing the number of running hosts and resource waste, which does not directly optimize energy consumption and thus the solution process fall into local optimal easily. As for the SLATAH index, EQ-VMC performs the best, ACS-VMC is 94 % more than EQ-VMC, though its EC is only 5.6% more than EQ-VMC. This is because that ACS-VMC targets at decreasing the number of running PMs which can significantly improve resource utilization. Regarding VMMs index, in Table 6, EQ-VMC has the smallest number; ACS-VMC has the second smallest. In terms of the estimation of host overloading risk, EQ-VMC is capable of keeping workload stable for running PMs, which further reduces VM migrations. ACS-VMC establishes an optimization model which contains the optimization objective for minimizing VM migrations, which make it relieve host overloading risk, even if it has less VM migrations. SA-VMC and COFFGA do not consider the QoS directly when it optimizes VMs placement, so they have a common performance on VM migrations. Since ESV is a combination metric covered energy and QoS for VM consolidation. EQ-VMC still has the best performance in ESV, EC-VMC is the second best. The EQ-VMC and EC-VMC methods are superior to other compared methods in both EC and QoS indices, because reducing energy consumption and relieving host overloading risk are the optimization objectives of the established optimization model in them. This verifies that both EQ-VMC and EC-VMC methods realize their optimization objective. In summary, EQ-VMC shows better performance and stability in EC, SLATAH, VMMs and ESV than other methods.

Table 7 Simulation results using GoogleClusterTrace with four metrics

	EC(kWh)	SLATAH	VMMs	ESV
EQ-VMC	22.31	1.34	5298.4	0.00446
EC-VMC	22.03	0.63	2865.7	0.00120
ACS-VMC	24.25	1.70	2395.1	0.00334
SA-VMC	27.17	1.84	14030.2	0.00997
COFFGA	35.76	1.91	14952	0.02155

Table 7 shows the experimental results on the GoogleClusterTrace. As for the EC and SLATAH indices, EQ-VMC is inferior to EC-VMC. In terms of the VMMs and ESV indices, EC-VMC and ACS-VMC are superior to EQ-VMC. We note that on the GoogleClusterTrace, all of the indices by EQ-VMC are inferior to that of EC-VMC, but on Bitbrains trace EQ-VMC is superior to that of EC-VMC. The different performance of EQ-VMC between Bitbrains and GoogleClusterTrace, results from the difference amount of resource requests in two traces and the regular distribution in GoogleClusterTrace, which is shown in Table 4 and Table 5. GoogleClusterTrace has resource demands that approximately follow Gaussian distribution [37] and the demanded resource is less and more stable, which make EQ-VMC hard to know host overloading probability with normal distribution and result in the inferior performance than that of EC-VMC. Next, EC-VMC takes host overloading probability as a constrain condition to PMs rather than taking host overloading probability as an

optimization objective like that in EQ-VMC, which benefits the running PMs to improve resource utilization while avoiding host overloading risk. At last, because of the distinctive workload characteristics like the Gaussian distribution in GoogleClusterTrace, the resource usage of running PMs does not have a strong randomness, and even ACS-VMC has better results in VMMs and ESV indices than EQ-VMC.

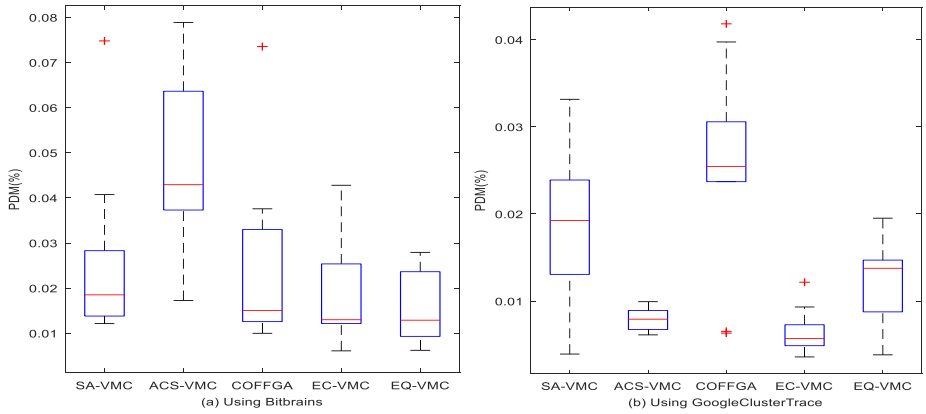

Fig. 8. Comparison of PDM

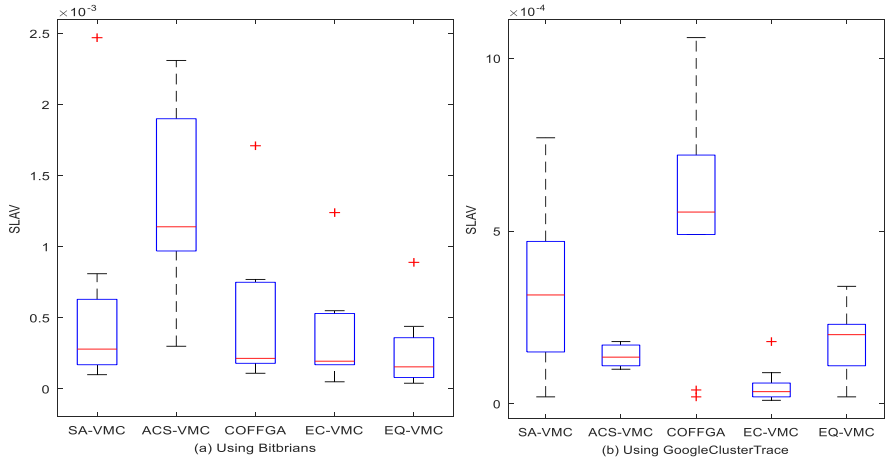

Fig. 9. Comparison of SLAV

Fig. 8 compares different schemes in terms of PDM that depends on VMMs and memory capacity of the migrated VMs. Fig 8 (a) shows the PDM metric on Bitbrains trace. The PDM of EQ-VMC is the lowest and most stable among compared methods, and EC-VMC performance is the second lowest. The VMMs of ACS-VMC shown in Table 6 is lower than that of EC-VMC, SA-VMC and COFFGA, whereas its PDM is the highest among all compared methods. The reason is that the ACS-VMC has to migrate VMs with higher resource request to reduce VM migration. SA-VMC and COFFGA use MMT for VMs selection, so they obtain lower PDM, despite of more VM migrations during their VM consolidation. Fig 8 (b) shows the PDM results on GoogleClusterTrace. As we can see, EC-VMC is the best and EQ-VMC is at the medium level. The primary reason for it is that, when the host overloading probability of PMs is small, the proposed optimization model in EQ-VMC is unable to choose proper destination hosts for VM placement. Unfortunately, it further spurs host overloading risk, frequent VM migrations and higher PDM value. Finally, compared

to SA-VMC and COFFGA, EQ-VMC still performs better.

Fig. 9(a) and 9(b) show the comparison results on the SLAV index using Bitbrains trace and GoogleClusterTrace respectively. We can see that, using Bitbrains trace, EQ-VMC is the best, EC-VMC is the second best, and they are better than the other compared methods. This result indicates the optimization model proposed in this paper facilitate relieving host overloading risk of running PMs effectively. On GoogleClusterTrace, the SLAV value of EC-VMC is the best, and the SLAV value of EQ-VMC is at medium level among all compared methods. The main reason for that is the same as the aforementioned different workload characteristics between GoogleClusterTrace and Bitbrains trace, namely, GoogleClusterTrace subjects to the normal distribution without enough randomness, and thus the heuristic evolutionary optimization-like algorithms are hard to work well with GoogleCluster-like traces.

In summary, by comparing SA-VMC, ACS-VMC, COFFGA, EC-VMC and EQ-VMC on the same indices, we can see that EQ-VMC is capable of decreasing energy consumption and guaranteeing QoS, even if it is inferior on a few indices to the other compared methods on GoogleClusterTrace.

B. Effectiveness Evaluation

To evaluate the practical application of the proposed approach, this part further compares EQ-VMC and other compared methods on the number of running PMs, VMMs, CPU resource utilization, memory resource utilization and bandwidth utilization during VM consolidation. In our simulation a cycle of VM consolidation is conducted every 5 minutes, and the total number of cycles is 288 one day. Figs.10-16 show the results.

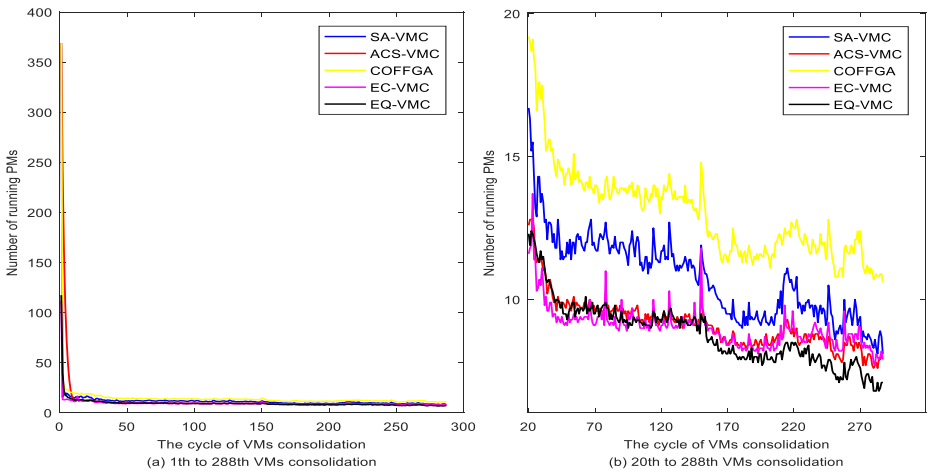

Fig. 10. The number of running PMs varying with the cycle of VM consolidation on Bitbrains

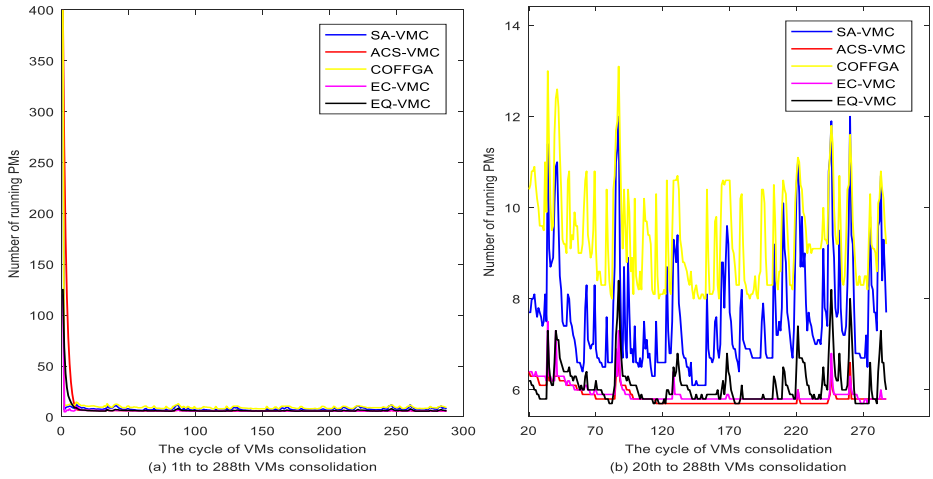

Fig. 11. The number of running PMs varying with the cycle of ongoing VM consolidation using GoogleClusterTrace

Fig. 10 shows the variation for the number of running PMs with respect to each cycle of ongoing VM consolidation using Bitbrains trace. Fig. 10(a) shows the variation from 1-th to 288-th cycle of VM consolidation. With all five VM consolidation methods, the number of running PMs in data centers significantly decreases at the initial stage, thus reducing energy consumption. The variation from the 20-th to 288-th cycle is further displayed in Fig. 10(b) for detailed comparison. As shown in Fig. 10(b), in the initial stages, ACS-VMC, EC-VMC and EQ-VMC have the similar changing trend for the number of running PMs, while EQ-VMC has less running PMs than the other remainder methods, especially in the later stages; this is also supported by the experimental results in Table 6. The EC index in Table 6 demonstrates that fewer running PMs are equivalent to lower energy consumption in data centers. In contrast, COFFGA and SA-VMC have larger number of running PMs than the other compared methods, because that they employ FFD for VM placement that can easily fall into local optimal region and lead to high host overloading risk.

Fig. 11 shows the variation in the number of running PMs within each cycle of the ongoing VM consolidation on GoogleClusterTrace. As shown in Fig. 11(a), the number of running PMs in data centers also is significantly reduced in the initial stage. Fig. 11(b) further shows the variation from the 20-th to the 288-th cycle of VM consolidation. We can see that the number of running PMs of COFFGA and SA-VMC is apparently more than the other methods. This is due to the same reasons as mentioned before, namely, the regular distribution in GoogleClusterTrace. In contrast, the number of running PMs of EQ-VMC, EC-VMC and ACS-VMC are similar, and less than that of the other compared methods. Wherein, with combination analysis of Table 7, the EC index of ACS-VMC is still larger than that of EQ-VMC and EC-VMC. This is because that the employed optimization model in ACS-VMC is unable to find out the proper destination hosts for live VM migration. Unlike ACS-VMC, EC-VMC utilizes ABC-like (e.g., Artificial Bee Colony) search way to attempt to find the optimum VM placement scheme with lowest EC and VMMs. In contrast, EQ-VMC algorithm can effectively reduce the number of running PMs,both on Bitbrains trace and GoogleClusterTrace, due to its embedded discrete DEVMP algorithm which is

able to search the optimum scheme for VM placement with lowest energy consumption and smallest host overloading risk.

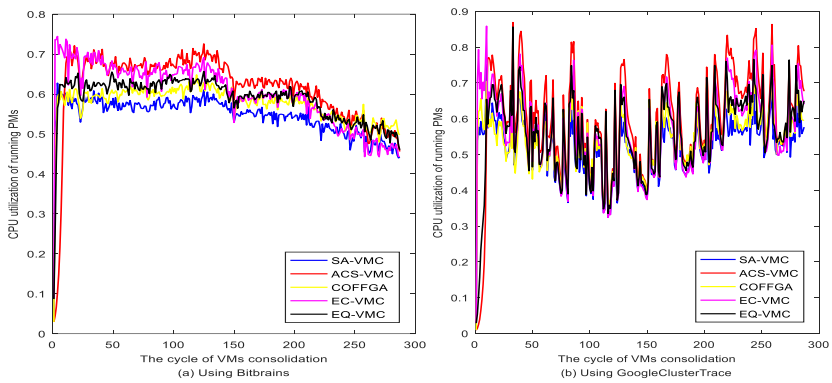

Fig.12. CPU utilization of running PMs

Usually, on one hand, the reduction of the number of running PMs during VM consolidation means that a single PM has to undertake more computing tasks and the CPU resource utilization of it inevitably increases. Fig. 12 shows the variation of CPU utilization on the running PMs during ongoing VM consolidation. On the other hand, the increasing CPU resource utilization indicates a higher energy consumption for that PM . Fig.12 (a) shows the results using Bitbrains trace and Fig.12 (b) shows the results with GoogleClusterTrace. Besides ACS-VMC method, the VM consolidation methods show very similar variation trend in CPU utilization. Combined with Fig.11, it is easy to find that EQ-VMC and EC-VMC always select the destination hosts with large-scale capacity of resources on physical resources, which results in an ideal performance for resource utilization and decreasing energy consumption. In contrast, as shown in Fig.12 (a), the CPU utilization of ACS-VMC is almost the highest during all cycles of VM consolidation. This result means that ACS-VMC has to experience more host overloading risk than that of the other compared methods. With combination analysis of the number of running PMs in Fig.11, it is obvious that the capacity of physical resource of running PMs selected by ACS-VMC method is relatively smaller than that of EC-VMC and EQ-VMC, resulting in increasing the number of running PMs. Additionally, ACS-VMC also shows the highest CPU utilization on GoogleCluster Trace in Fig.12 (b) due to the same reason aforementioned of the regular distribution in it.

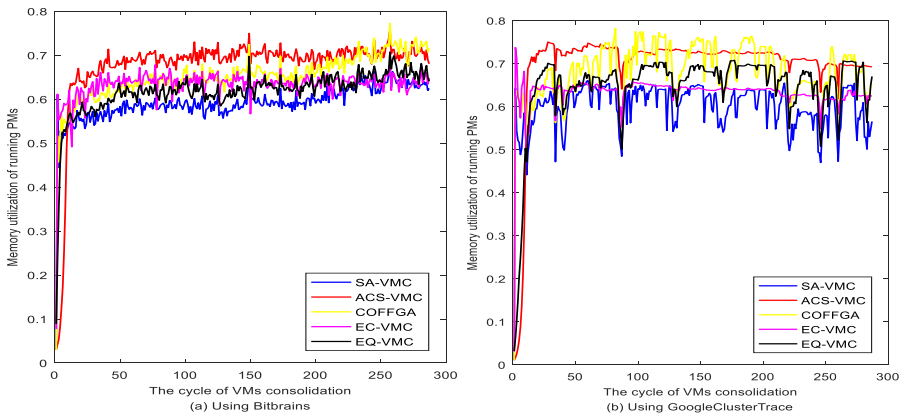

Fig.13. Memory utilization of running PMs

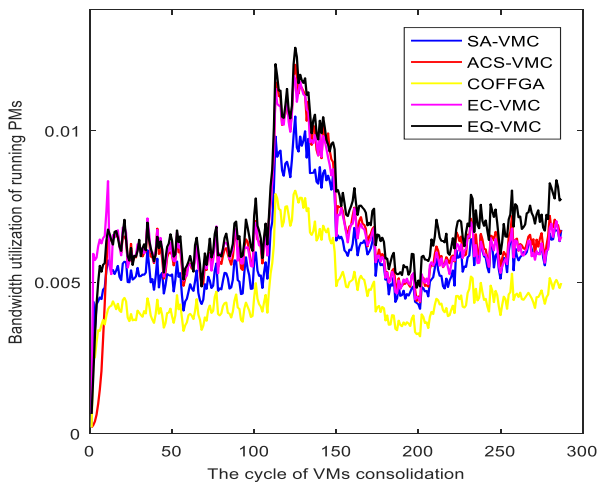

Fig.14. Bandwidth utilization of running PMs

Fig.13 demonstrates the memory utilization of the running PMs during the cycles of VM consolidation. Fig.13 (a) is the results with Bitbrains trace and Fig.13 (b) is the results with GoogleClusterTrace. Since there is no bandwidth trace in GoogleClusterTrace, Fig.14 only shows the experimental results on bandwidth utilization using Bitbrains trace. Combining Fig.12 (a), Fig.13 (a), and Fig.14, we can see that the CPU and memory utilization of EQ-VMC method locate the medium level. This performance results from workload stable yielded by EQ-VMC method, of which is accompanied with efficient network transmission and communications. It is also demonstrated by the high bandwidth utilization in Fig.14. In contrast, ACS-VMC shows the highest resource utilization in CPU and memory, which incurs frequent VM migration, higher host overloading risk and QoS deterioration. Besides the high host overloading risk, frequent VM migration also consumes extra network bandwidth, which results in a higher bandwidth utilization of ACS-VMC method in Fig.14.

Additionally, as for EQ-VMC method using GoogleClusterTrace, we get the same indices distribution due to the identical aforementioned reasons that yield the output in Fig. 12(b) and Fig. 13(b).

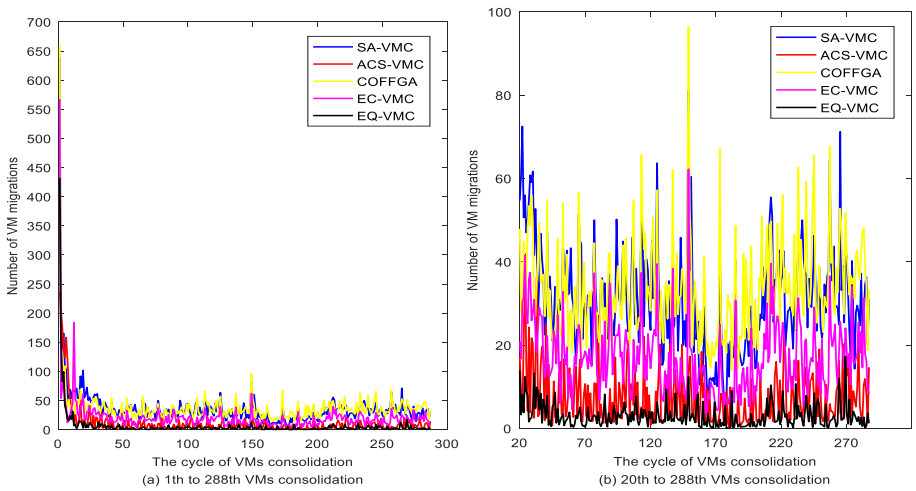

Fig. 15. The VMMs varying with the cycle of VM consolidation on Bitbrains

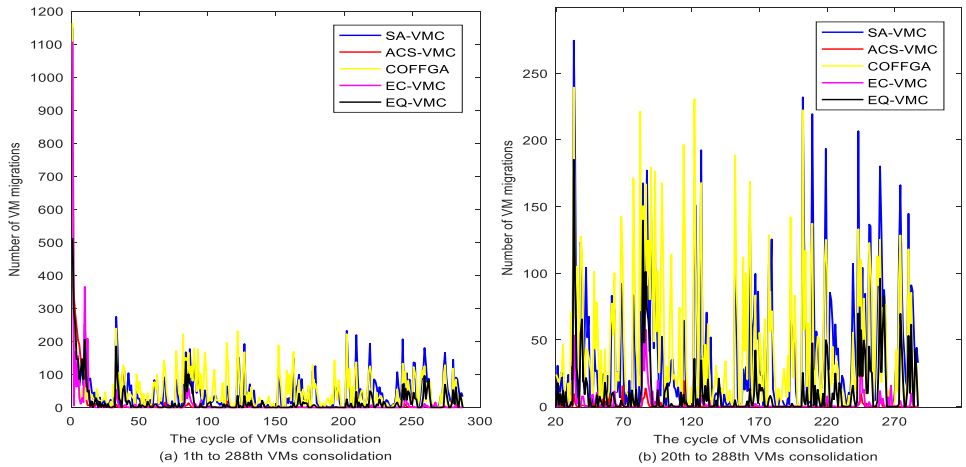

Fig. 16. The VMMS varying with the cycle of VM consolidation on GoogleClusterTrace

The VMMs reflects the stability of QoS. Generally, the fewer the total VMMs result in better QoS. Fig.15 shows the variation of VM migrations during ongoing VM consolidation using Bitbrains trace. Fig.15 (a) describes the changes trend from the 1-th to 288-th cycle. For clearly comparing, Fig.15 (b) is partial of Fig.15 (a), of which shows the variation trend from the 20-th to 288-th cycles of VM consolidation. As shown in Fig.15 (a), since many PMs have been turned off in initial stage, an army of VM migrations occurs. This phenomenon illustrates that these compared methods perform well in the early stages for VM consolidation. After the 20-th cycle of VM consolidation, the VMMs triggered by EQ-VMC is lower than that of the other compared methods. This result indicates that the workload of hosts gets more stable after VM consolidation by EQ-VMC, so fewer virtual machines need to be migrated. In contrast, the SA-VMC and COFFGA conduct a large number of VM migration and vibration, this is supported by their performance in QoS showed in Table 6, Fig. 8(a), Fig. 9(a), of which is inferior to that of EQ-VMC.

Fig.16 shows that the variation of VM migrations during ongoing VM consolidations using GoogleClusterTrace. Wherein, Fig.16 (a) shows the changes from the initialization to the 288-th VM consolidation. Fig.16 (b) is partial of Fig.16 (a). ACS-VMC get the lowest VMMs and the curve in Fig.16 is smoother than the other compared methods, that mainly results from lower resource request and the regular distribution of workload in GoogleClusterTrace, which prevents ACS-VMC from suffering from host overloading risk, thus degrading the VMMs. The changes of VM migrations by EC-VMC throughout all 288 cycles of VM consolidations is just inferior to that of ACS-VMC method, but it get prominent advantage in EC and other QoS metrics than ACS-VMC showed in Table 7, Fig.8 (b), Fig.9 (b). The curve of VM migrations with EQ-VMC throughout 288 cycles of VM consolidation locates medium in all methods. However, EQ-VMC gets obviously improvement in EC than the other compared methods.

Based on the above analytical comparison, the results show that the proposed EQ-VMC method can effectively reduce energy consumption and guarantee QoS. The presented VMs placement scheme efficiently maintains load balancing and relieves host overloading risk. Therefore the optimization objectives are realized through

EQ-VMC.

6. Conclusions and future works

Data center provides access to shared resources on the Internet as a scalable, dynamic and measurable service. The VM consolidation performs live VM migration to appropriate destination hosts in order to be able to improve one or more objectives and thus it is regarded as NP-hard issues. Generally, heuristic algorithms have shown advantages of resolving the complex combination optimization problem. Although VM consolidation using heuristic or meta-heuristic algorithms do not provide the best allocating solutions between VMs and PMs, they offer VM consolidation scheme close to optimal ones.

This paper addresses VM consolidation with respect of heuristic evolutionary algorithms. First, our scheme aims to minimize the mathematical expectation of the energy consumption of running PMs while maintaining their lowest probable risk of host overloading, and establish a dynamic optimization model for VM placement. Next, by abstracting the deployment relationship between each virtual machine and all physical machines into a single deployment vector, the deployment vectors of all VMs are thus equivalent to an item of mapping between VMs and PMs during a cycle of VM consolidation, namely, an individual in the heuristic evolution algorithm. Naturally, all probable mappings between VMs and PMs during ongoing VM consolidation correspond to a population, that is, the search space in evolutionary computations. Afterwards, an improved discrete differential evolution (discrete-DE) algorithm is developed to resolve the aforementioned optimization model by finding the result in the search space which realizes the optimum VM placement for the migrated VMs. Further, a VM placement algorithm is consequently proposed based on the presented optimization model. Finally, depend on the study above, a hybrid heuristic evolutionary-based EQ-VMC method is developed for VM consolidation. Extensive trace-driven experiments are examined to validate the proposed method, and the experimental results demonstrate that it significantly reduces energy consumption, avoids unnecessary host overloading risk, and improves QoS.

However, there are a few limitations that need to be further addressed in our future works. First, this study only focuses on VM placement with optimization objectives of minimizing energy consumption and relieving host overloading risk, the other optimization issues such as mminimizing VMMs, maximizing resource utilization during VM consolidation need further study to improve the presented algorithm. Next, EQ-VMC shows prominent experiment results on Bitbrains trace; but QoS is only at medium level with GoogleClusterTrace, thus the proposed optimization model should be given priority to adapt to the GoogleCluster-like trace, and also the GoogleClusterTrace should be deeply analyzed and data mined for the extremely particularity hidden in it due to such as the large-scale Google data center.

Acknowledgment

This work is supported by the Smart Manufacturing New Model Application Project Ministry of Industry and Information Technology (Grant No.ZH-XZ-18004), Future

Research Projects Funds for the Science and Technology Department of Jiangsu Province (Grant No.BY2013015-23), the Fundamental Research Funds for the Ministry of Education (Grant No. JUSRP211A 41) and the 111 Project (Grant No.B2018)

Reference

[1] X. Zhu, D. Young, B.J. Watson, Z. Wang, J. Rolia, S. Singhal, 1000 Islands: Integrated Capacity and Workload Management for the Next Generation Data Center, in: International Conference on Autonomic Computing, IEEE Computer Society,(2008) pp. 172-181.

[2] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Generation Computer Systems. 28(5) (2012)755-768.

[3] A. Beloglazov, R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers, Concurrency & Computation Practice & Experience, 24(13)(2012),pp.1397－1420.

[4] F. Farahnakian, P. Liljeberg, J. Plosila, LiRCUP: Linear Regression Based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers, Software Engineering and Advanced Applications, IEEE,(2013), pp. 357-364.

[5] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, D. Pendarakis, Efficient resource provisioning in compute clouds via VM multiplexing, in: International Conference on Autonomic Computing, Icac 2010, Reston, Va, Usa. DBLP,(2010),pp. 11-20 .

[6] H. Lin, X. Qi, S. Yang, S. Midkiff, Workload-Driven VM Consolidation in Cloud Data Centers, Parallel and Distributed Processing Symposium, IEEE, (2015), pp.207-216.

[7] J. Cao, Y. Wu, M. Li, Energy Efficient Allocation of Virtual Machines in Cloud Computing Environments Based on Demand Forecast, Journal of Chinese Computer Systems. 7296(4)(2013), pp.137-151.

[8] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints, in: IEEE Transactions on Parallel & Distributed Systems, 24(7)(2013),pp. 1366-1379.

[9] Z. Li, C. Yan, X. Yu, N. Yu, Bayesian network-based virtual machines consolidation method, Future Generation Computer Systems. 69(2017), pp.75-87.

[10] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, Using ant colony system to consolidate vms for green cloud computing, in: IEEE Transactions on Services Computing, 8 (2)(2015), pp.187-198.

[11] H. Hallawi, J. Mehnen, H. He, Multi-capacity combinatorial ordering ga in application to cloud resources allocation and efficient virtual machines consolidation, Future Generation Computer Systems. 69(2016), pp.1-10.

[12] J. Jiang, Y. Feng, J. Zhao, K. Li, Dataabc: a fast abc based energy-efficient live vm consolidation policy with data-intensive energy evaluation model, Future Generation Computer Systems. 74(2016),pp. 132-141.

[13] Z. Li, C. Yan, L. Yu, X. Yu, Energy-aware and multi-resource overload probability constraint-based virtual machine dynamic consolidation method, Future Generation Computer Systems. 80(2018), pp.139-156.

[14] S. Telenyk, E. Zharikov, O. Rolik, Consolidation of virtual machines using simulated annealing algorithm, in: International Scientific and Technical Conference on Computer Sciences and Information Technologies, IEEE, (2017),pp. 117-121.

[15] Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, Journal of Computer & System Sciences.79 (8) (2013), pp.1230-1242.

[16] R.W. Ahmad, A. Gani, S.H.A. Hamid, et al. A survey on virtual machine migration and server consolidation frameworks for cloud data centers, Journal of Network & Computer Applications. 52(C) (2015),pp.11-25.

[17] M.R. Chowdhury, M.R. Mahmud, R.M. Rahman, Implementation and performance analysis of various vm placement strategies in cloudsim, Journal of Cloud Computing. 4(1) (2015),pp.1-21.

[18] S.S. Masoumzadeh, H. Hlavacs, A Cooperative Multi Agent Learning Approach to Manage Physical Host Nodes for Dynamic Consolidation of Virtual Machines, Network Cloud Computing and Applications, IEEE,(2015), pp.43-50.

[19] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future generation computer systems. 28(5)(2012),pp. 755-768.

[20] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, Energy Aware Consolidation Algorithm Based on K-Nearest Neighbor Regression for Cloud Data Centers, in: International Conference on Utility and Cloud Computing, IEEE,(2014), pp. 256-259.

[21] S.S. Masoumzadeh, H. Hlavacs, An Intelligent and Adaptive Threshold-Based Schema for Energy and Performance Efficient Dynamic VM Consolidation. Energy Efficiency in Large Scale Distributed Systems. Springer Berlin Heidelberg,(2013), pp. 85-97.

[22] S.B. Shaw, A.K. Singh, Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center, Computers & Electrical Engineering, 47(2015),pp.241-254.

[23] H. Shen, L. Chen, Distributed autonomous virtual resource management in datacenters using finite-markov decision process, in: Proceedings of the ACM Symposium on Cloud Computing. ACM, (2014),pp.1-13.

[24] S. Sohrabi, I. Moser, The effects of hotspot detection and virtual machine migration policies on energy consumption and service levels in the cloud, Procedia Computer Science, 51 (2015),pp. 2794-2798.

[25]Li, M., Bi, J., & Li, Z. . Improving consolidation of virtual machine based on virtual switching overhead estimation. Journal of Network and Computer Applications, 59(C)2015, 158-167.

[26] M.A. Kaaouache, S. Bouamama, Solving bin packing problem with a hybrid genetic algorithm for vm placement in cloud, Procedia Computer Science. 60 (2015), pp.1061-1069.

[27] M. Mishra, A. Sahoo, On Theory of VM Placement: Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach, in: IEEE, International Conference on Cloud Computing, IEEE Computer Society, 17 (2011), pp.275-282.

[28] J.A. Aroca, A.F. Anta, M.A. Mosteiro, C. Thraves, L.Wang, Power-efficient assignment of virtual machines to physical machines, Future Generation Computer Systems. 54(C) (2016),pp. 82-94.

[29] S.B. Melhem, A. Agarwal, N. Goel, M. Zaman, Markov prediction model for host load detection and vm placement in live migration. IEEE Access. 6 (2018), pp.7190-7205.

[30] R. Storn, K. Price, Differential evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces, Berkeley: ICSI, 1995.

[31] K.Z. Ibrahim, Optimized pre-copy live migration for memory intensive applications, High PERFORMANCE Computing, Networking, Storage and Analysis, IEEE, 26(b) (2011), pp.1-11.

[32] R.N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software Practice & Experience. 41(1) (2011),pp. 23-50.

[33] SPEC [Online]. Available: https:// www.spec.org/power_ssj2008/results.

[34] Amazon EC2 Product Details [Online]. Available: http://www.amazonaws.cn/en/ec2/details.

[35] S. Shen, V.V. Beek, A. Iosup, Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters, in: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, (2015), pp.465-474

[36] [dataset] ClusterData2011_2 traces [Online]. Available: https://github.com/google/cluster-data/blob/master/ ClusterData2011_2.md.

[37] L. Yu, L. Chen, Z. Cai Z, et al., Stochastic Load Balancing for Virtual Resource Management in Datacenters, IEEE Transactions on Cloud Computing, 99(2016), pp.1-1.

[38] F. Farahnakian, T. Pahikkala, P. Liljeberg, et al., Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing, 2015 IEEE 8th International Conference on Cloud Computing, IEEE,(2015),pp. 381-388.

[39] Z.A. Mann, Rigorous results on the effectiveness of some heuristics for the consolidation of virtual machines in a data center. Future Generation Computer Systems 51(2015),pp.1-6.

[40] M. Vitali, B. Pernici, U.M. O'Reilly, Learning a goal-oriented model for energy efficient adaptive applications in data centers. Information Sciences, 319(C) (2015),pp.152-170.