An Adaptive Virtual Machine Location Selection Mechanism in Distributed Cloud

Shukun Liu[1], Weijia Jia[2]
[1] School of Information Science and Engineering, Central South University, 410083 Changsha, China
[e-mail: liu_shukun@csu.edu.cn]
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University, 200240 Shanghai, China
[e-mail: weijiaj@gmail.com]
*Corresponding author: Shukun Liu

Received July 7, 2015; revised September 22, 2015; accepted October 18, 2015;
published December 31, 2015

Abstract

The location selection of virtual machines in distributed cloud is difficult because of the physical resource distribution, allocation of multi-dimensional resources, and resource unit cost. In this study, we propose a multi-object virtual machine location selection algorithm (MOVMLSA) based on group information, doubly linked list structure and genetic algorithm. On the basis of the collaboration of multi-dimensional resources, a fitness function is designed using fuzzy logic control parameters, which can be used to optimize search space solutions. In the location selection process, an orderly information code based on group and resource information can be generated by adopting the memory mechanism of biological immune systems. This approach, along with the dominant elite strategy, enables the updating of the population. The tournament selection method is used to optimize the operator mechanisms of the single-point crossover and X-point mutation during the population selection. Such a method can be used to obtain an optimal solution for the rapid location selection of virtual machines. Experimental results show that the proposed algorithm is effective in reducing the number of used physical machines and in improving the resource utilization of physical machines. The algorithm improves the utilization degree of multi-dimensional resource synergy and reduces the comprehensive unit cost of resources.

Keywords: Distributed cloud, location selection, multi-dimension, immune memory

This work is supported in part by the National Natural Science Foundation of China under Grant Numbers 61272151 and 61472451, the International Science & Technology Cooperation Program of China under Grant Number 2013DFB10070, the China Hunan Provincial Science & Technology Program under Grant Number 2012GK4106, and the "Mobile Health" Ministry of Education - China Mobile Joint Laboratory (MOE-DST No. [2012]311). We express our thanks to the editor and all the reviewers for our manuscript.

http://dx.doi.org/10.3837/tiis.2015.12.003 **ISSN : 1976-7277**

1. Introduction

With the growth of network application services, information technology architecture and a variety of resources should be effectively integrated to manage physical resources effectively, improve the utilization rate of resources, and reduce resource unit costs [1]. Virtual machine technology is a key to virtualization, and it is widely applied to distributed cloud [2]. With the rapid popularity of cloud computing, the unlimited use of limited resources can be achieved by users in the future. Under this condition, users can obtain the physical resources they actually require, similar to how people purchase fuel or natural gas for their daily routines. However, users need to select the most appropriate interface to obtain such resources; otherwise, a considerable amount of resources will be wasted. The selection of the proper purchase window has thus become a key and fundamental problem that necessitates urgent solutions. Nowadays, the resources needed by cloud users are mainly embodied in the form of a virtual machine. To make a virtual machine perform efficiently, the host resource is mapped to the application layer, and the resource scheduling process is encapsulated in the search process of the virtual machine [3, 4]. Thus, the key problem in the resource allocation process is the rapid and proper selection of a virtual machine for the corresponding physical nodes under the premise of satisfying all service-level targets for different applications [5].

Physical resource utilization and user satisfaction can be greatly improved with an adaptive selection mechanism that allows users to automatically select a virtual machine according to comprehensive factors, which depend on a physical machine, during the allocation of virtual machine resources. Therefore, as the key issue in virtual machine deployment, the location selection of virtual machines needs to be solved urgently.

In this study, we first formulate our optimization problem as a bin packing based on multi-dimensional resource utility to determine the optimal location selection of virtual machines to physical machines, with consideration of the requirements for dependability. This method is different from the traditional way. Second, this study proposes a new population updating method based on immune memory and on a new multi-object genetic algorithm for the location selection of virtual machines based on the structure of a doubly linked list. Finally, during the coding process of virtual and physical machines, we used a mechanism of information grouping mode, through which the similar scale virtual machines can be allocated to the proper locations in shorter time compared with the traditional way.

The rest of this paper is organized as follows. Section 2 surveys related works. Section 3 states the problem that we will address and presents related definitions. Section 4 describes the detail of gene encoding and evaluation function design. Section 5 describes the MOVMLSA algorithm. Section 6 presents the experimental evaluation and analysis. Finally, Section 7 concludes this paper and suggests a future research direction.

2. Related Work

The mapping problem of virtual machines to physical nodes can be regarded as a multi-dimensional vector packing problem [6–8] and as a NP-hard problem. At present, this problem is mainly solved with a heuristic algorithm [8, 9]. However, the current research on the location selection of virtual machines in a cloud platform is mainly aimed at the optimization of only one particular dimension. For example, existing research only aims to guarantee service-level objectives, minimize the number of physical nodes, and reduce virtual machine migration and energy consumption [10]. The optimization goals in some cases are inconsistent and contradicting. For example, a virtual machine is placed on a minimal number of physical nodes to reduce the number of used nodes; in this way, idle nodes can be saved, and energy consumption and management cost can be reduced [11]. Nevertheless, extensive virtual machine migration occurs. If the goal is to reduce such migration, the number of used physical nodes is likely to increase.

In [12, 13], a genetic algorithm was used to tackle the problem of the static placement of virtual machines without consideration of the overhead costs involved in the virtual machine migration. In [14], node virtualization integration was described as a packing and random optimization problem in the cloud data center, but the considerations focused only on processor resources and not on other dimensions, such as memory and input/output (I/O). In [15, 16], the authors proposed a management framework for virtual machine placement in cloud computing, but they failed to consider resource costs and system energy consumption. In [17, 18], the authors proposed a scheduling strategy for virtual machines based on a genetic algorithm, the historical data of the cloud computing system, and the current state of the system [19]. This method achieves an ideal load balance and incurs minimal overhead for virtual machine migration; however, it ignores resource utilization and energy consumption in the data center. In [13, 15], the problem of the location selection of virtual machines was divided into a multi-objective optimization problem and a bin packing problem but without the consideration of the cost of virtual machine migration; the study focused only on the static placement of virtual machines and disregarded dynamic deployment based on virtual machine migration. In [18, 20], the problem was considered as a combinatorial optimization problem based on bin packing; similarly, the authors only considered the static placement of virtual machines and disregarded dynamic placement based on virtual machine migration.

Most optimization methods for virtual machine placement are implemented in several phases to solve a single-objective optimization problem. Multiple targets are rarely optimized simultaneously. Consequently, only a local rather than a global optimization solution is obtained. In sum, extensive research results have been obtained in the cloud placement of virtual machines, but several serious problems, including those enumerated below, have yet to be solved.

(1) Most studies determine data centers by selecting the required physical nodes and not the most appropriate nodes from the cloud data center. A complete strategy for the location selection of virtual machines should be divided into two levels, namely, data center and physical machine.

(2) Many of the existing studies on the virtual machine placement strategy are mainly based on a single dimension of target optimization under certain rules. Consequently, the generated optimal placement method may be based on only one certain condition. An efficient strategy for the location selection of virtual machines should consider the dependencies between dimensional constraints and comprehensive balance.

(3) Many new challenges need to be addressed to dynamically allocate and manage the shared physical and virtual resources of data centers. For example, an adaptive framework for the location selection of effective virtual machines is lacking. In location selection, the cost of resources, system performance, energy consumption, and other factors must be considered. An efficient algorithm for the location selection of virtual machines must also be designed such that it is adaptable to different user goals and business needs.

3. Problem Statement and Related Definitions

The total amount of resources utilized by users in a cloud computing platform is equivalent to the particular resources of a virtual machine. Each user application runs in its own, independent virtual machine. The efficient use of cloud computing resources and the reduction in the cost of such resources are key academic research issues.

3.1 Problem Statement

The multi-dimensional collaborative problem of the location selection of virtual machines can be considered a multi-objective combinatorial optimization problem. The available resources of each physical machine, such as the central processing unit (CPU), memory, disk, and I/O devices, can be used as multi-dimensional vectors. Every dimension is a physical resource for physical machines. Each resource required by a virtual machine corresponds to a multi-dimensional vector. The goal of the location selection of virtual machines is to place different virtual machines to multiple physical nodes according to the different needs of multiple users. The location selection process must be based on an effective adaptive framework. The problem of the location selection of virtual machines, which is based on multiple targets, can be described as follows.

In order to describe the problem and definitions conveniently and clearly, we designed two notation tables in which the basic meanings of the main symbols are defined accordingly.

Tabel 1. Notation table (1)

SYMBOL	DESCRIPTION
H	number of data centres
V	virtual machine set
v_j	the *jth* virtual machine
M	physical machine set
m_i	the *ith* physical machine
t	task set of a virtual machine
R	resource set of a data centre
r_i	resource set of *ith* physical machine
r_i^q	the *qth* dimension resource of the *ith* physical node
L	resource service period
S	useful resource service vector
s_i	the useful service ratio of r_i
t_{jk}^q	the *qth* dimension resource of *kth* task on v_j
W	task generation ratio set of virtual machine
w_j	task generation ratio of v_j
$p_{n\times m}$	probability distribution matrix of data source
$s_{n\times m}$	cloud resource scheduling matrix
μ	load balance variance of all physical nodes

Table 2. Notation table (2)

SYMBOL	DESCRIPTION
$U_{i,q}$	qth dimension resource utility of m_i
$U_{j,q}^{i}$	utility of qth resource of v_j which is assigned to m_i
v_{ji}^{q}	the qth dimension resource of v_j which is assigned to m_i
m_i^q	the qth dimension resource of m_i
E	remaining resource biggest matrix of physical node
F	virtual machine resource matrix
f_j	the required resource for v_j
f_j^q	the required resource number of qth dimension for v_j
C_i	resource billed vector of m_i
c_i^q	the cost of obtain qth dimension from m_i
G	mapping matrix of v_j to m_i
N	migration number of virtual machine
Y	overhead of placing v_j on m_i
Q	the max number of resource dimension

A cloud platform comprises a number of data centers. In this study, we suppose that H data centers exist in a cloud platform and denote all resources in the data centers as $\sum_{h=1}^{H} R_h$. One data center comprises n physical nodes. $R = \{r_{1,} r_{2,} r_{3......} r_{i......,} r_n\}$ denotes the total resources of one data center. $r_i = \{r_i^1, r_i^2, r_i^3, ..., r_i^q\}$ denotes a set of q different dimensions resource of m_i . During an entire service period, which is denoted as L , the useful service time obeys the Poisson distribution [21], and all the resources are independent of one another, i.e., $\forall r_i \in R, \mathop{\Big|}_{i=0}^{n} r_i = \phi$. In the resource set of a data center, which is denoted as R , the useful service vector is denoted as $S = \{s_1, s_2, ..., s_n\}$, and the useful service rate of r_i is denoted as $s_i \, (0 \le i \le n)$.

A virtual machine set is denoted as $V = \{v_1, v_2, v_3, ..., v_m\}$, which is placed on physical nodes. The set comprises m virtual machines. In this study, we suppose that v_j only has one task t_j at any time. The kth task, which is denoted as $t_{j_k} = \{t_{jk}^1, t_{jk}^2, t_{jk}^3, ..., t_{jk}^q\}$ $,0 \le j \le m$, includes q different attributes. The task generation rate of a virtual machine can be described as $W = \{w_1, w_2, w_3, w_4, ...w_j..., w_m\}$, $0 \le j \le m$, where w_j denotes the task production rate of the jth virtual machine. All independent tasks satisfy $\forall t_j \in V, \mathop{\Big|}_{j=0}^{m} t_j = \phi$, $\forall r_i \in R, t_j \in V$. If r_i can satisfy all the requirements of t_j , then t_j can be assigned to r_i , and all operations can be run. In some cases, many r_i can satisfy the requirements of t_j . To achieve an effective balance among all resources, t_j is allocated to r_i with a probability of P_{ij} . The probability distribution matrix of data resources is denoted as $P_{n \times m} = \left(P_{ij}\right)_{n \times m}$, with the condition that $P_{ij} \ge 0$ and $\sum_{i=0}^{n} P_{ij} = 1$. The cloud resource scheduling matrix is denoted as $S_{n \times m} = \left(S_{ij}\right)_{n \times m}$. If v_j is assigned to m_i , then $S_{ij}=1$; otherwise, $S_{ij}=0$. During all the allocation processes, the following conditions must be fulfilled:

$$\begin{cases} \sum_{i=0}^{n} P_{ij} \cdot S_{ij} = 1 \\ \sum_{i=0}^{n} P_{ij} \cdot \mathrm{S}_{ij} \cdot t_j \le \mathrm{s}_i \\ \sum_{i=0}^{n}\sum_{j=0}^{m} P_{ij} \cdot S_{ij} \cdot t_j \le \sum_{i=0}^{n} s_i \end{cases} . \tag{1}$$

In this work, we discuss the proper placement of virtual machines according to the usage information of multi-dimensional vectors of such virtual machines in the location selection process. We evaluate the resulting performance by referring to Microsoft's 2008 report on virtual machine management techniques in the evaluation standards for physical server resources, i.e., according to the CPU, memory, network bandwidth, and disk I/O [15]. The problem is described as follows:

(1) In n physical machine nodes ($M=\{\mathrm{m}_1, m_2, m_3, ..., m_n\}$), n refers to the number of physical nodes in the physical cluster. The main resources include memory, CPU, disk, bandwidth, and I/O.

(2) In m virtual machine nodes ($V=\{v_1, v_2, v_3, ..., v_m\}$), m refers to the number of virtual machines. The main resource requirements of a virtual machine include memory, CPU, disk, bandwidth, and I/O.

(3) A mapping between virtual and physical machines must be established to satisfy all the requirements of virtual machines and to reduce the physical nodes assigned to these virtual machines. During the mapping process, the sum of the assigned virtual machine resources cannot exceed the amount of resources of the physical machine.

3.2 Related Definitions

Definition 1: The largest resource vector of m_i can supply $r_i = \{r_i^1, r_i^2, r_i^3, ..., r_i^q\}^T$, where r_i^q denotes the maximum number of dimension resource supplied by m_i , $1 \le i \le n$. The remaining resource matrix of the physical machine node set can be defined as follows:

$$E = \begin{bmatrix} r_1^1 & r_1^2 & r_1^3 & \ldots & r_1^q \\ r_2^1 & r_2^2 & r_2^3 & \ldots & r_2^q \\ r_3^1 & r_3^2 & r_3^3 & \ldots & r_3^q \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ r_n^1 & r_n^2 & r_n^3 & \ldots & r_n^q \end{bmatrix} .$$

Definition 2: Given an adequate amount of physical resources, a virtual machine runs appropriately. The required amount of resources of v_j can be denoted as $f_j = \{f_j^1, f_j^2, f_j^3, ..., f_j^q\}^T$. f_j^q denotes the required number of qth dimension resource, $1 \le j \le m$. The required resource matrix of a virtual machine is denoted as follows:

$$F=\begin{bmatrix} f_1^1 & f_1^2 & f_1^3 & \cdots & f_1^q \\ f_2^1 & f_2^2 & f_2^3 & \cdots & f_2^q \\ f_3^1 & f_3^2 & f_3^3 & \cdots & f_3^q \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ f_m^1 & f_m^2 & f_m^3 & \cdots & f_m^q \end{bmatrix}.$$

Definition 3: Resource cost

The required amount of resources is billed. The billed vector can be marked with $\mathrm{C}_i=\{\mathrm{c}_i^1,\mathrm{c}_i^2,\mathrm{c}_i^3,\ldots\ldots,\mathrm{c}_i^q\}^T$, where c_i^q denotes the cost of obtaining qth dimension resource from m_i.

Definition 4: Mapping matrix of v_j to m_i

$$G=\begin{bmatrix} g_{11} & g_{12} & g_{1j} & \cdots & g_{1m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{i1} & g_{i2} & g_{ij} & \cdots & g_{im} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{n1} & g_{n2} & g_{nj} & \cdots & g_{nm} \end{bmatrix}$$

Here, $1\le i\le n$, $1\le j\le m$, and $g_{ij}\in\{0,1\}$. g_{ij} denotes whether v_j is placed on m_i. $g_{ij}=1$ indicates that V_j is placed on the m_i; otherwise, $\mathrm{g}_{ij}=0$.

Definition 5: Let $x_i\in\{0,1\}$. $x_i=1$ indicates that a virtual machine migrates from one physical node to another physical node. Otherwise, the virtual machine does not migrate. We suppose that S denotes the number of migrations undertaken by a virtual machine and that $S=\sum_{i=1}^{n}x_i$, $x_i\in\{0,1\}$. According to this definition, the overhead of placing a virtual machine on a physical node can be denoted as Y, where $\mathrm{Y}=\mathrm{F}^{\mathrm{T}}\times G\times C_i$.

Definition 6: $P=\sum_{i=1}^{n}p_i$, $p_i\in\{0,1\}$. $p_i=1$ indicates that at least one virtual machine is assigned to a physical node. Otherwise, no virtual machine is assigned to a physical node.

Definition 7: $\mu=Min(\sum_q D_i/q)$ denotes the balanced load variance of all the physical machines. D_i denotes the ith dimension variance, and q denotes the number of dimensions. $\mathrm{D_i}=\sum_n(\mathrm{p_{ij}}-\overline{\mathrm{p_i}})^{\wedge 2}/n$, where n denotes the number of physical nodes. $\overline{p_i}$ denotes the average value of the performance of the ith dimension for all physical nodes. p_{ij} is the ith performance value of the m_i. All the performance values are normalized. In this study, the optimization goal for virtual machine placement is as follows:

$$\begin{cases} \min(\mu) \\ \min(Y) \\ \min(\mathrm{P}) \\ \max \sum_{i=1}^{N} \sum_{q=1}^{Q} U_{i,\,\mathrm{q}} \end{cases} . \qquad (2)$$

Constrained conditions:

At any time, the total amount of all types of resources required by all virtual machines cannot exceed the total amount of resources of all physical machines. The amount of resources required for each virtual machine should also not exceed that required for each physical machine. The concrete constrained conditions can be formalized as follows:

$$\begin{cases} (a)\forall r_i \in R, \bigcap_{i=0}^{n} r_i = \phi \\ (b)\sum U_{j,q}^{i} \leq T_{i,\mathrm{q}} \\ (c)\forall q, \sum_{i=1}^{n} \sum_{j=1}^{m} g_{ij} \times f_j \leq r_i \\ (d)\sum_{j=1}^{m} f_j \leq R \\ (e)\forall q, \sum_{j=1}^{m} v_{ji}^{q} \times x_{ji} \leq m_j{}^{q} \end{cases} \qquad (3)$$

$T_{i,q}$ denotes the threshold of the resource q for m_i, and $U_{j,q}^{i}$ denotes the useful ratio of the resource q in v_j, which is assigned to m_i. The inequality (e) above indicates that the qth dimension resource (i.e. CPU, memory, disk space, bandwidth, and I/O) of the virtual machine that are assigned to the physical machine m_i cannot exceed those of the physical machine m_i. If a user application can be placed on X virtual machines, then the performance factors of the virtual machines supplied to the application P must satisfy service-level agreements (SLAs).

4. Chromosome Gene Encoding and Evaluation Function Design

In this part, we will introduce the process of chromosome gene encoding and evaluation function design. Using the GA (Genetic Algorithm) technology, we can explain the coding information of virtual machine and physical machine clearly and easily. In addition, evaluation function can be expressed combined with the coding information.

4.1 Encoding

Encoding is generally the first factor that can affect the search efficiency of evolutionary algorithms. Encoding can reflect the mapping relations of solutions to chromosomes [16]. For the location selection of virtual machines, the solution coding of the physical nodes to which virtual machines are assigned can be considered a chromosome. The assigned virtual machine is the gene value. The advantage of this coding is that the number of chromosome genes is determined by the total number of physical servers [22]. Therefore, this situation cannot reduce computing speed because of the emergence of extensive coding. During the subsequent crossover and mutation operations, the hardware constraints among all servers remain the same [22]. In the present study, the problem of the location selection of virtual machines can be regarded as a bin packing problem. Three chromosome encoding mechanisms are available: coding based on cases, coding based on items, and coding based on groups [3]. Coding based on cases and coding based on groups focus on individual items. The goal function of bin packing depends on item groups. In this study, the encoding mechanism is mainly based on item groups. According to the combined redundancies of group encoding [23], a doubly linked list coding method is proposed.

M virtual machines are assigned to N physical nodes. M is generally larger than N. A random virtual sequence that includes M virtual machines is generated. For chromosome encoding, a priority heuristic algorithm is employed to place a random virtual machine sequence on physical nodes. This priority heuristic algorithm selects a physical node from used physical nodes. If the selected node can satisfy the five resource requirements (CPU, memory, network bandwidth, disk, and I/O) of the first virtual machine, then the virtual machine can be placed on the physical node. Otherwise, the subsequent physical nodes are selected until the physical node that satisfies such requirements is found (let the resources of N physical machines satisfy the resource requirements of M virtual machines). If no proper physical node from the used physical nodes can satisfy the resource requirement, then the first new physical node that has not been used is selected for the placement of the virtual machine (when the algorithm is implemented initially, all physical machine nodes are not used). According to the description above, all virtual machines can be placed on physical nodes. The concrete algorithm is described as follows:

Algorithm 1: Priority heuristic algorithm

```
For (i=0; i<=n; i++)
{
    VMFlag=0;
     For (j=0;j<=m;j++)
     {
        If (VMpj ==1)
            {
                If  Rpj(c,m,s,b,i/o)>Rvi(c,m,s,b,i/o)
                    {
                        VMi stored in Pj;
                        VMFlag=1;
                        Break;
                    }
                Else
                 j++;
            }
        }
     If (VMFlag==0)
     VMi stored in the first unused Pj
 }
```

The encoding process of assigning M virtual machines to N physical nodes, which is based on group information and a doubly linked list, is shown in the example. Let us assume the availability of M virtual machines, which need to be allocated at a certain period, as well as a sequence number of $1,2,3,......,m-1,m$. The sequence number is random. Virtual machines are assigned to physical nodes randomly according to the priori algorithm, and the initial code can then be generated. The initial code is optimized according to the doubly linked list packing method.

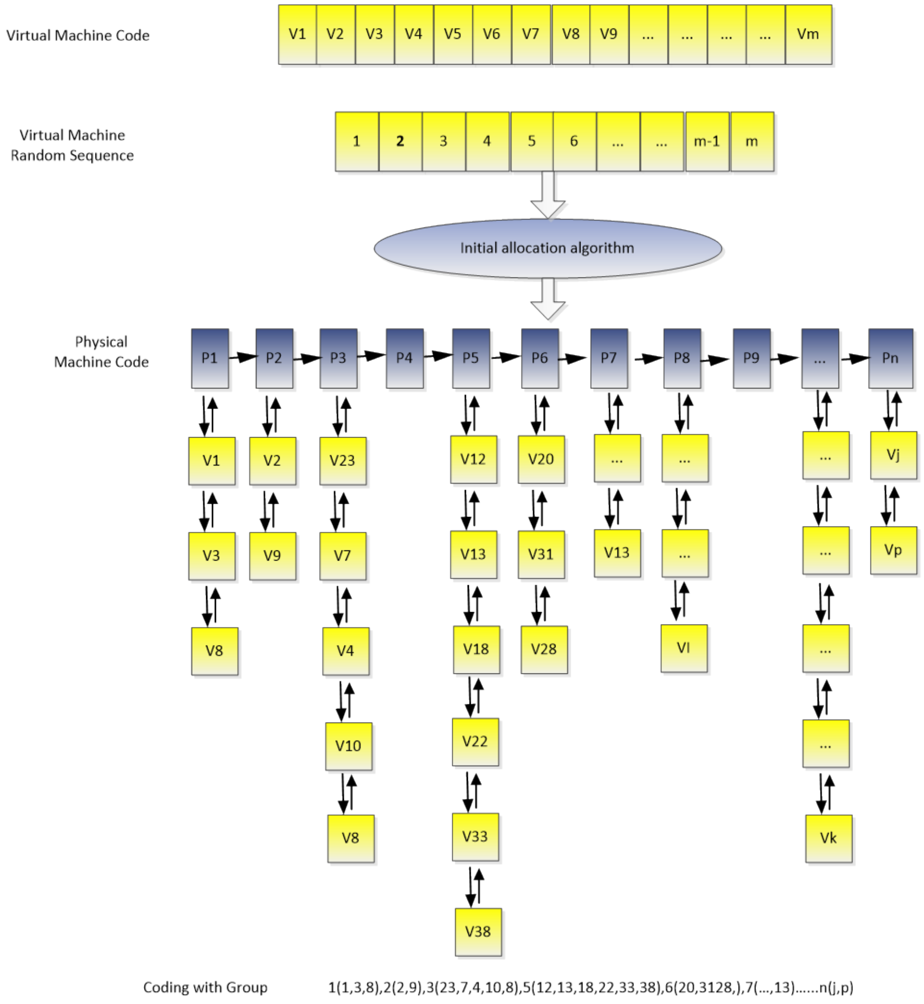

Fig. 1. Encoding process

4.2 Evaluation Function

The optimal goals are a minimal number of used physical machines, low unit price costs of resources for users, a high performance of an application system achieved with a balanced load, and a low useful ratio of physical nodes. An evaluation function can be used to evaluate individuals. The first parameter is a key factor to evaluate the chromosome load performance of used physical nodes. A small load variance denotes the good load performance of used physical nodes. The second parameter is the lowest possible resource cost for users. The third parameter is used to evaluate the energy consumption level in a chromosome using the number of physical nodes. A minimal number of used physical nodes denote insignificant energy consumption. The fourth parameter is used to evaluate the physical machine utilization of a specific resource type. In this study, the fitness function is defined as follows:

$$Fitfunction(\mu, M, P, U_{i,q}) = \{\min(\mu), \min M(Y), \min(P), \max \sum_{i=1}^{N} \sum_{q=1}^{Q} U_{i,q} \}. \qquad (4)$$

4.3 Evolution Operators

For the problem of assigning a virtual machine to physical machine, a new coding method based on group information and doubly linked list nodes is used.

4. 3. 1 Crossover operator

The virtual machine code contains the precursor and subsequent nodes. A chromosome gene is expressed in different groups and in an internal bidirectional chain structure. Considering that the arrival of the virtual machine sequence is random, the sequence length is not fixed. The number of virtual machines that each physical machine can accommodate also differs. The variable length of the chromosomes referred by the crossover operator is not fixed as well.

The genetic algorithm involves physical machine coding and virtual machine group coding. Based on the proposed coding method, a single-point crossover method with the longest doubly linked list is put forward in this study. The crossover process in the genetic algorithm mainly allows the offspring to inherit excellent genes from parent generation. The crossover process is divided into two parts. One part is the process based on cross-group coding to minimize the number of used physical machines. The other part is the process based on cross-resource coding to maximize the utility of physical machine resources. In the first part (as shown in the **Fig. 2** below), the main steps of the single-point crossover optimization method for sub-individual coding optimization based on the largest chain length (for the optimization of individual species) are as follows:

Step a) Two parent nodes are selected randomly. The longest group in parent individual A is selected and replaced with the corresponding group of parent individual B.

Step b) After the crossing operation for parent individual B, the corresponding physical nodes that comprise repeated virtual machines are removed from the collection of used physical nodes and are then added to the unused physical node set; in this case, the virtual machines are deleted [16].

Step c) The unallocated virtual machine series is generated on the basis of the retained virtual machine chain structure of parent individual B.

Step d) The priority heuristic algorithm is used to generate new child entities from the unallocated virtual machine sequence.

Step e) A group is randomly selected from parent individual B and is then used to replace the corresponding group in parent individual A. Individuals B are generated according to Steps a)–d).

For example, 12 virtual machines and 4 physical nodes are simulated. The process of the single-point crossover method [16] is shown in **Fig. 2**.

4. 3. 2 Mutation operator

Two scenarios emerge in chromosome variation. One is based on the group encoding variations of a doubly linked list, i.e., deleting a virtual machine in the parent node list randomly. The other scenario refers to resource coding variants, in which we can delete a doubly linked list node of a virtual machine. After such deletion, the precursor and subsequent nodes change simultaneously. The occurrence of these two types of mutation does not follow a specific discipline; they may occur in the form of a single individual or in a specific moment. When mutation occurs, X physical nodes are randomly selected from the chromosome (where X < M / 2). X physical nodes are removed from the collection of used physical nodes and are then added to the collection of unused physical nodes. The virtual machines deployed on the physical nodes are also deleted. The unallocated virtual machine series is then generated on the basis of the chain structure of individual virtual machines, which preserves individual variation. Finally, the unallocated sequence of virtual machines is redistributed in accordance with the distribution of the priority heuristic algorithm.

The value of parameter X, which can affect the performance of the multi-objective evolutionary algorithm, depends on mutation rate. The value is determined with an experiment [16]. We take note of the following details. 1. When virtual machines have been integrated, we will ensure that the amount of resources of the physical machine is less than or equal to the threshold of each type of resource value during the process of crossover or mutation. 2. In the process of crossover or mutation based on resource coding, the remaining resources of the cross section selected by the physical machine must be greater than those of the selected virtual machine. The process of the mutation operator method is shown in **Fig. 3**.

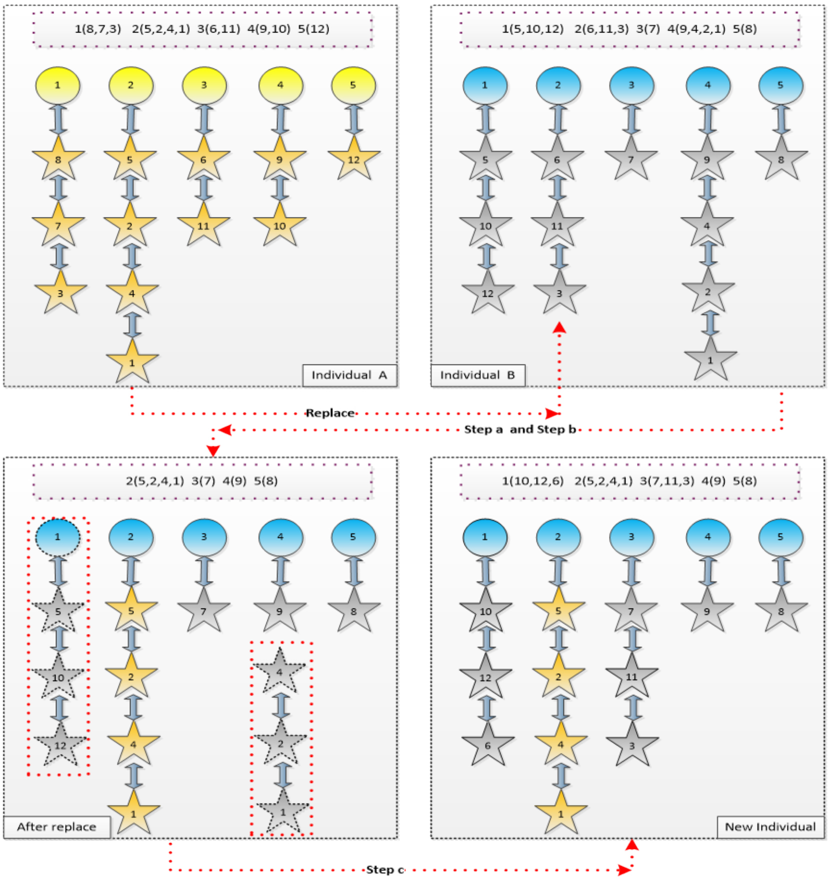

Fig. 2. Single-point crossover process

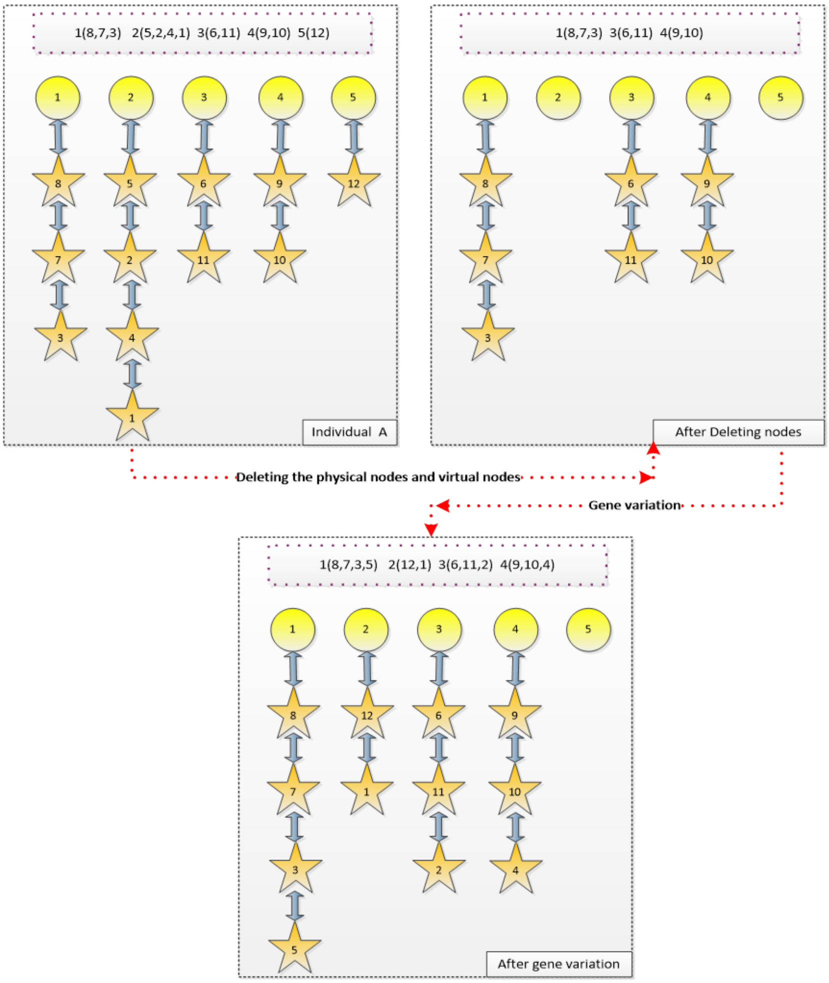

Fig. 3. Mutation process

4. 4. 3 Population update method

The ε dominant relation is a weakened form of the Pareto dominance relation [24]. In this study, the ε form is added for the given $\varepsilon \in R^k, \varepsilon_i > 0, \forall i \in k$. The concrete definition of this form is as follows:

Definition 8: ε dominant relation

Let $f : R^m \to R^k, x_1, x_2 \in \Omega \subseteq R^m$, $\varepsilon \in R^k, \varepsilon_i > 0$, which is called x_1 **ε dominate** x_2, if and only if $\forall i \in \{1, 2, \ldots\ldots, k\}, f_i(x_1) - \varepsilon_i \le f_i(x_2)$ and $\exists i, f_i(x_1) - \varepsilon_i < f_i(x_2)$, which is marked as $x_1 \le \varepsilon_i x_2$.

Definition 9: ε-Pareto optimal approximate solution set

Set X^a is called a ε-Pareto optimal approximate solution set of X if and only if for any $x \in X$, the attribute $x' \in X^a$ is always true and $x' \le_\varepsilon x$ is always true [24].

Definition 10: ε-Pareto solution set

Set X^ε is called a ε-Pareto solution set of X^ε if and only if X^ε is a ε-Pareto optimal approximate solution set of X and $X^\varepsilon \subseteq X^*$ [24].

Definition 11: Immune memory

Immune memory is an important characteristic of the immune system. When the body is exposed to an antigen for the second time, the incubation period of the antibodies relative to the first response time is obviously short. In this case, the antibody levels rise rapidly. When the same antigen invades the body again, a strong primary immune system develops; the phenomenon in which antibodies gain a higher degree of affinity is referred to as immune memory [25].

The current non-dominant individuals are stored on the basis of the memory set. $\varepsilon-dominant$ is an effective form of the relaxation mechanism of the Pareto dominance, which is used to maintain the uniform distribution of a solution. This form is widely applied to maintain diversity domain. $\varepsilon-dominant$ is adopted to update the memory population. For a multi-goal optimization problem that includes m goal functions, if $1\leq f_i \leq K$, $\left(i=1,2,...,m\right)$, then the goal space can be divided into $((K-1)\ /\varepsilon)^m$ sub-spaces according to the rules of $\varepsilon-dominant$. In every sub-space, only one individual exists, and other individuals are deleted from the population space.

The $\varepsilon-dominant$ mechanism is sensitive to a variety of problems. Given such variety, different numbers of antibodies are kept.

From a practical viewpoint, deciders cannot know the geometric distribution form of the Pareto frontier in advance. If a target value of multiple antibodies is less than a certain value, deciders do not view two antibodies as different. In this case, the use of the $\varepsilon-dominant$ mechanism is appropriate. We adopt a control mechanism to provide decision rights that can meet such a situation and to maintain population diversity. In practice, decision makers can dynamically control the values of a vector. Allocating an identification vector for individuals is a simple method that can be used to divide the antibodies of a living space [25]. The identification vector can be defined as $B\left(X_a\right)=\left(B_1\left(X_a\right),B_2\left(X_a\right),...,B_m\left(X_a\right)\right)$ and $Bi\left(Xa\right) = fi\left(X_a\right)/\varepsilon,\left(i=1,2,...,m\ \right)$. The current memory set can be denoted as $MS\left(t\right)=\left(MS_1,MS_2\ ,...,MS_n\right)$. A new generation population is generated after the crossover and mutation operations and is then crossed with sub-populations. The result is sorted with the rule of ε non-dominated sorting. According to the domination relationship among individuals, the memory information of antibodies, the Euclidean distance, and super volume [26], the subsequent generation of species evolves further. The update mechanism of concrete super body populations is described as follows:

a) If the current generation is the first generation, then the non-dominant individual should be selected in the initial population. Otherwise, the non-dominant individual of the current population and the parent population is assigned to the individual population set. The large individual population is sorted after being overlaid according to the dominant relation. Let the grade number be n, and let the non-dominant set be denoted as $L1,L2,.......,Ln$. The new identification vector for the non-dominant population is assigned. If some of the identification vectors of some individuals are the same, then they belong to the same super individual. All identification vectors can be used to extract and identify super individuals.

b) Let the population scale be M, and let the individual number of $L1$ be M. The set of $L1$ can be regarded as the subsequent generation.

c) If the individual number of $L1$ is more than M, then all the memory information of the antibodies and the contribution value of the super individuals are calculated in $L1$, except for some individuals with goal limit values. The $n-2$ contribution values that are relatively large are selected, and the individuals whose values satisfy the goal value are selected as the subsequent generation population and marked as $Z(g+1)$.

d) If the individual number of $L1$ is less than M, then all the individuals of M are added to the subsequent generation marked as $Z(g+1)$. The individuals of $L2, L3, \ldots\ldots, Ln$ are added to $Z(g+1)$ until the scale of the population reaches M. After such addition, if the number of individuals of L_i is more than M, then the Euclidean distance of the vector individuals with the same super individuals is calculated. The individuals with small values of Euclidean distance join the subsequent generation until the population scale reaches M.

5. Immune Memory-based Algorithm for the Location Selection of Virtual Machines

The optimization process with the multi-objective optimization algorithm has obvious disadvantages despite the wide use of the non-dominated sorting genetic algorithm 2 and the double F-shaped resonator (DFR). For the problem of multi-target optimization, improved performances and optimal results can be achieved for different frontal shapes. In this study, the basic evolution operators of cloud resource allocation are thus embedded in SMS-EMOA [16]. We propose a new multi-objective virtual machine location selection algorithm (MOVMLSA) according to the multi-objective selection process based on dominated hyper-volume [27]. The main process of the algorithm is described as follows:

a) According to the group information based on doubly linked list encoding, parent individuals are initialized to form the parent population with a scale of M.
b) The fitness evaluation function is calculated in view of the population individuals. The parent population that has been evaluated is sorted with the ε non-dominated rule. The control level of the individuals in the population is divided.

c) Two parent individuals are selected from populations using the tournament selection method, and a hybrid individual is generated using the single-point crossover method for the two parent individuals.

d) Intelligent hybrid individual variation occurs, and individuals are generated. The child individuals are first evaluated with the fitness evaluation function. The child individuals are then added to the population pool. This step is repeated until the population size reaches M.

e) The best M individuals are selected as the subsequent generation of the parent population on the basis of the update mechanism of the immune memory.

f) For the new generation of the parent population, steps c)–f) are repeated until the evolution condition is met (maximum number of iterations). The algorithm is then terminated.

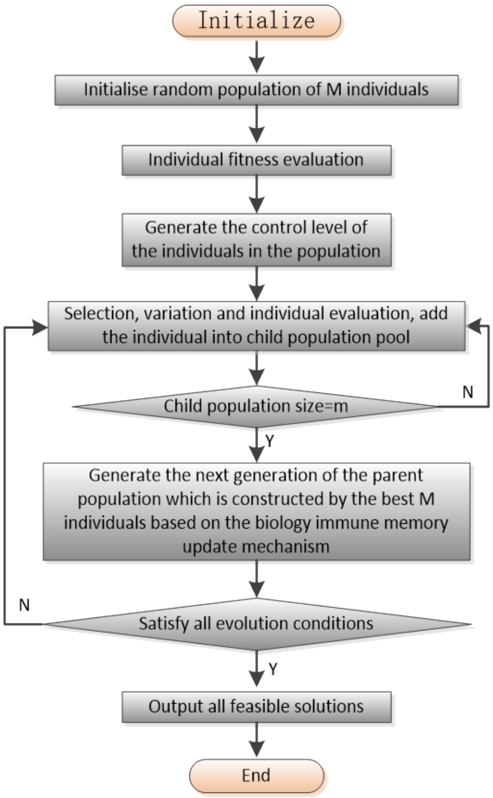
Initialize
Initialise random population of M individuals
Individual fitness evaluation
Generate the control level of the individuals in the population
Selection, variation and individual evaluation, add the individual into child population pool
Child population size=m
N
Y
Generate the next generation of the parent population which is constructed by the best M individuals based on the biology immune memory update mechanism
Satisfy all evolution conditions
N
Y
Output all feasible solutions
End

Fig. 4. Algorithm flow

6. Experimental Results and Analysis

The experimental results show that the proposed MOVMLSA based on immune memory can determine the reasonable placement of virtual machines, improve the utilization of physical resources, and reduce the cost of resources for users. The algorithm features a certain degree of feasibility and high accuracy.

A priority heuristic algorithm (i.e., the proportional hazards model (PHM)), the classic NSG-2 algorithm, the multi-dimensional dominant resource fairness (DRF) scheduling algorithm [28], and the multi-objective evolutionary algorithm based on immune memory are all simulated in this study to verify the performance of the immune system-based multi-dimensional strategy for the location selection of virtual machines. The number of used physical machines, the load balance of the physical machines, and the resource cost are all recorded; these values can be used to compare the performance of the different algorithms. The simulation experimental platform is the CloudSim, which was proposed by the teams from the Grid Laboratory of the University of Melbourne in Australia and the Gridbus Project [6]. Some corresponding classes based on the base classes of CloudSim are modified to implement all the algorithms. Several inherited classes and methods are also designed. The existing useful resource vector of every physical machine can be obtained through the host class. The required resource vector of a virtual machine can be generated from the Datacenter class. The VMProvisioner class is used to achieve the mapping from a physical machine to a virtual machine. The VMAllocation policy is an abstract class that is used to achieve the location selection of virtual machines. The HostForVm allocation method can realize the placement of a special virtual machine to a fixed physical machine.

The inherited VMAllocationPolicy class can provide the assignment strategy of virtual machines. The allocation algorithm of virtual machines based on the multi-objective genetic immune memory can be implemented by editing the user-defined inheritance class named VMAllocation Policy. With this class, virtual machines can select proper physical machines to locate. With the method of extending classes in CloudSim, all classes can be rebuilt.

Forty physical nodes are simulated in the CloudSim simulation platform. Every node is equipped with two processors (Intel(R) Core(TM) i5-3317U 1.7 GHz), a 4 GB memory storage, 8 MB L2 cache, and two disks of 7,200 turns with 500 GB. The virtual numbers are 30, 45, 55, 65, and 80 in turn.

Table 3. Parameter design of the algorithm for the location selection of virtual machines in the CloudSim platform

Number of physical nodes	Number of virtual machines	Algorithm
40	30	PHM, NSG-2, DRF, MOVMLSA
40	45	PHM, NSG-2, DRF, MOVMLSA
40	55	PHM, NSG-2, DRF, MOVMLSA
40	65	PHM, NSG-2, DRF, MOVMLSA
40	80	PHM, NSG-2, DRF, MOVMLSA

The population size of the evolution algorithm and the evolution number are set to 50 and 1,000, respectively. The crossover and mutation ratios are set to 0.6 and 0.01, respectively.

Crosser and mutation rates have very important influence on the experimental results. For general generic algorithm, the crosser and mutation rates are always constant. However, determining the concrete value of both rates is difficult. If the value is extremely small, an optimal solution is always more approximately achieved; however, ensuring that the solution is the global optimal solution is difficult. On the other hand, if the value of crosser and mutation rates are too large, the number of iterations also become large. Moreover, the searching ability becomes extremely low. At the same time, the algorithm is difficult to converge. To determine the experimental parameters, we conducted experiments, which can be used to determine the correlation between mutation rate and convergence speed, 1000 times.

Table 4. Class crossover and mutation rates

Crossover rate	Mutation rate	Maximum evolution number	Population size
0.6	0.01	1,000	50

An evolution experiment is conducted according to the above parameters in **Table 3** and **Table 4**. The experimental results are shown in **Table 5**.

Table 5. Performance comparison of four algorithms

Algorithm	Number of virtual machines	Number of physical machines	Load balance variance of physical set	Resource utilization
PHM	30	29	0.0046	60
NSG-2		13	0.0042	63
DRF		10	0.0026	67
MOVMLSA		11	0.0023	70
PHM	45	30	0.0047	63
NSG-2		19	0.0041	66
DRF		15	0.0024	69
MOVMLSA		14	0.0024	73
PHM	55	32	0.0042	65
NSG-2		24	0.0035	68
DRF		19	0.0022	73
MOVMLSA		16	0.0021	77
PHM	65	35	0.0037	66
NSG-2		27	0.0033	70
DRF		22	0.0023	77
MOVMLSA		20	0.0022	80
PHM	80	40	0.0039	69
NSG-2		30	0.0032	73
DRF		27	0.0021	79
MOVMLSA		26	0.0018	82

The intuitive data graph shown in **Figs. 5–7** is used to further analyze the algorithm performance data shown in **Table 4**. The numbers of the enabled physical nodes in the location selection algorithm based on immune memory and the DFR algorithm are approximately similar, but they are lower than those in the PHM and NSG-2 algorithms. A low number of enabled physical nodes indicates considerable savings in energy and resources. For the load performance of a cluster, the load performance variance in the location selection algorithm based on the multi-objective genetic algorithm is less than those in the other three algorithms. A small load performance variance denotes the good effect of the load-balancing server cluster. The preceding analysis implies that the location selection algorithm based on the multi-objective genetic algorithm can greatly reduce the number of used physical servers and can achieve a good load-balancing server cluster effect.

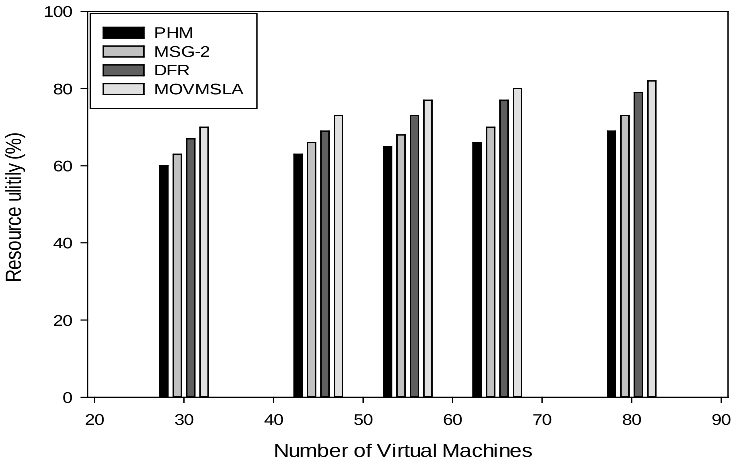

Fig. 5. Comparison of resource utility

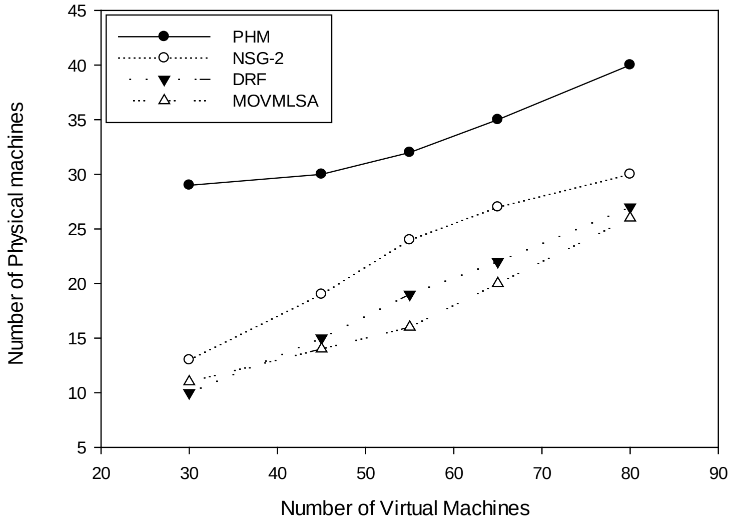

Fig. 6. Comparison of the number of enabled physical machines

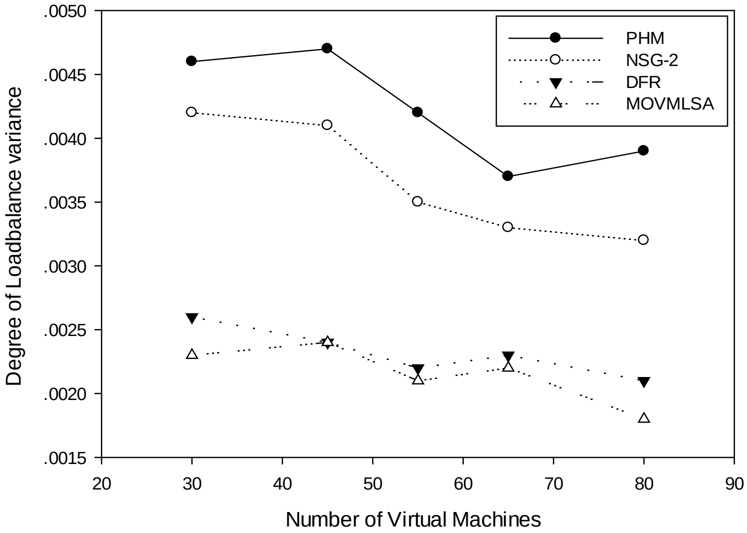

Fig. 7. Comparison of load balance degrees

To obtain the comparison results of PHM, NSG-2, DFR, and MOVMLSA, we performed five independent group experiments (the required numbers of virtual machines are separately set to 30, 45, 55, 65, and 80). Thus, the physical machine number, resource load balance, and resource utility can be analyzed adequately.

(1) Analysis comparison of comprehensive resource utility: The comprehensive resource denotes CPU, memory, network, and I/O resource. From the comparative experiments of the first group data to the fifth group data, the changes were recorded according to the changes in the number of virtual machines. The resource utility situation of PHM, NSG-2, DRF, and MOVMLSA are shown in **Fig. 5**. As shown in the figure, under the same conditions, the

physical resource utility is the highest when MOVMLSA is used. When the number of virtual machines is adjusted from 30 to 80, the comprehensive resource utility using MOVMLSA is 10%–14% higher than that of PHM, NSG-2, and DRF. Thus, MOVMLSA can obviously improve resource utility and save energy.

(2)Analysis comparison of the number of enabled physical machine: The number of virtual machines is adjusted, and the required physical machine number is shown in **Fig. 6**. The figure reflects the difference between NSG-2, DRF, and MOVMLSA. When different algorithms are used to assign the virtual machine to the physical machine, MOVMLSA can assign the virtual machine efficiently, which is reflected by using physical machines at least under the same conditions. Of course, the number of required physical machines increases with the increase in the number of virtual machines. The experimental results show that compared with PHM, which consumes plenty of resources, MOVMLSA can degrade the comprehensive resource utility from 35% to 62%. Compared with NSG-2 and DRF, MOVMLSA can degrade the comprehensive resource utility as well. Thus, MOVMLSA can improve resource utilization.

(3) Analysis comparison of resource load balance: In the experiments, resource threshold value, attributes of physical machines, and virtual machine tasks are consistent with one another, except for the deployment method. The cross and mutation rates are set at 0.6 and 0.01, respectively, and the maximum genetic algebra is set at 1000. The virtual machine deployment process was recorded from 30 virtual machines to 80 virtual machines in the experiment. The main goal of this experiment is to obtain the comparison of the result of resource load balance in a fixed number of virtual machines with different location selection algorithms. The comparison results are presented in **Fig. 7**. The experiments show that with the increase in the number of virtual machines, the algorithm can decrease the unbalanced degree of the resources. From the longitudinal contrast, under the same conditions relative to PHM, NSG-2, and DFR, MOVMSLA can reduce the resource imbalance degree from 0.03% to 0.23%. Compared with three other algorithms, MOVMLSA contributes more effectively to the performance of resource load balance. However, this advantage is not particularly obvious because many uncertain factors in determining genetic algorithm parameters exist. In our future work, the aspects of optimization parameters based on the characteristics of the algorithm itself would be improved.

In theory, the multi-object virtual machine location selection problem, which is proposed in this paper, is a combinatorial optimization problem. In cloud computing environment, however, combinatorial optimization problems are likely to lead to combination explosion. Prioritization process between targets can be avoided because the genetic algorithm can be processed parallel to each target. Thus, the genetic algorithm can effectively reduce the invalid combination. Therefore, the algorithm is suitable for solving a multi-object optimization problem [10]. MOVMLSA is a multi-object location selection algorithm based on immune memory, which is also classified as a kind of genetic algorithm. MOVMLSA is effective for solving multi-object combinatorial optimization problems. However, at present, the genetic algorithm faces the problem of slow convergence speed. To improve the convergence speed of the genetic algorithm, MOVMLSA can choose excellent individuals from each generation population; then, the immune information is extracted from the excellent individuals, and a vaccination for immunization is prescribed to the diagnosed offspring. Immune memory can speed up the breeding of good modes and repair the damaged modes by crossover and mutation model of excellence. Virtual machine population and immune memory information can interact and cooperate with one another, which can greatly improve the convergence speed of MOVMLSA. MOVMLSA can automatically obtain the immunization information of an individual, and this information is dynamically updated with the evolution of the population.

In this way, we can find the subspace that may be included in the optimum solution and improve the searching efficiency of the algorithm. The scale of immune memory library is an important factor that can influence the efficiency of the algorithm. However, in this study, we did not examine the relationship between the population size and the immunization type database scale, which will be investigated in our future work.

7. Conclusion

The MOVMLSA based on the multi-objective evolutionary algorithm and that based on immune memory are discussed in this work. To solve the problem of mapping physical machines to virtual machines, we transform the problem of the location selection of virtual machines into a multi-objective packing problem according to the multi-dimensional resource characteristics of virtual machines and the characteristics of optimization goals. The solution of the multi-objective genetic algorithm is obtained on the basis of the genetic memory information. A chromosome evaluation function and a group chain code are designed according to the doubly linked list and group information. The maximum lengths of the cross operator, single-point crossover operator, and mutation operator of the X-point mutation are designed according to the code information. Compared with other evolutionary algorithms, the multi-objective genetic algorithm based on immune memory shows a better performance for different frontal shapes. An algorithm for the location selection of virtual machines based on immune factors is designed in a cloud computing platform. The algorithm update policy is based on the volume of the update mechanism, which can ensure the population quality and diversity. PHM, NSG-2, DFR, and MOVMLSA are simulated to verify the performance of the new algorithm. The experimental results show that the MOVMLSA based on immune operators performs better than the other algorithms in terms of resource utilization, cluster load balance, and resource cost.

References

[1] D. Breitgand and A. Epstein, “Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds,” *IEEE INFOCOM 2012-IEEE Conference on Computer Communications*, pp. 2861-2865, 2012. Article (CrossRef Link).

[2] M. Alicherry and T. Lakshman, “Optimizing data access latencies in cloud systems by intelligent virtual machine placement,” *IEEE INFOCOM 2013-IEEE Conference on Computer Communications*, pp.647-655, 2013. Article (CrossRef Link).

[3] Xiaojiao Chen, Shihping Chen and Fang Fang, “Virtual machine resource allocation algorithm in cloud computing,” *Computer Application Research*, vol. 31, no. 9, pp.2584-2587, 2014. Article (CrossRef Link).

[4] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, “Resource allocation algorithms for virtualized service hosting platforms,” *Journal of Parallel and Distributed Computing*, vol. 70,no. 9, pp. 962-974, 2010. Article (CrossRef Link).

[5] M. Gahlawat and P. Sharma, “Survey of virtual machine placement in federated Clouds,” in *Proc. of Advance Computing Conference (IACC), 2014 IEEE International*, pp.735-738, 2014. Article (CrossRef Link).

[6] Bo Xu, Chao Zhao, Yanjun Zhu and Zhiping Peng, “Virtual machine resource scheduling multi-objective optimization in cloud computing,” *Journal of System Simulation*, vol. 26,no. 3, pp. 592-595, 2014. Article (CrossRef Link).

[7] X. Kong, C. Lin, Y. Jiang, W. Yan, and X. Chu, "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction," *Journal of Network and Computer Applications*, vol. 34, no. 4 pp. 1068-1077, 2011. Article (CrossRef Link).

[8] F. Hao, M. Kodialam, T. Lakshman, and S. Mukherjee, "Online allocation of virtual machines in a distributed cloud," in *Proc. of IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp. 10-18, 2014. Article (CrossRef Link).

[9] D. Warneke and O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 985-997, 2011. Article (CrossRef Link).

[10] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *Proc. of IEEE INFOCOM 2012-IEEE Conference on Computer Communications*, pp.702-710, 2012. Article (CrossRef Link).

[11] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. of IEEE INFOCOM 2012-IEEE Conference on Computer Communications*, pp.2876-2880, 2012. Article (CrossRef Link).

[12] K. Sato, M. Samejima, and N. Komoda, "Dynamic optimization of virtual machine placement by resource usage prediction," in *Proc. of 2013 11th IEEE International Conference on Industrial Informatics*, pp.86-91, 2013. Article (CrossRef Link).

[13] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proc. of Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pp. 179-188, 2010. Article (CrossRef Link).

[14] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective VM sizing in virtualized data centers," in *Proc. of Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pp.594-601,2011. Article (CrossRef Link).

[15] Q. Li, Q.-F. Hao, L.-M. Xiao, and Z.-J. Li, "Adaptive management and multi-objective optimization for virtual machine placement in cloud computing," *Chinese Journal of Computers*, vol. 34, no.12, pp.2253-2264, 2011. Article (CrossRef Link).

[16] Haojun Ai, Suwen Gong and Yuanming Yuan, "Research of cloud computing virtual machine allocated strategy on Multi-objective evolutionary algorithm," *Computer Science*, vol. 41, no.6, pp. 48-53, 2014. Article (CrossRef Link).

[17] S. Sawant, "A genetic algorithm scheduling approach for virtual machine resources in a cloud computing environment," *Master's Projects. Paper 198*, 2011. Article (CrossRef Link).

[18] J. Gu, J. Hu, T. Zhao, and G. Sun, "A new resource scheduling strategy based on genetic algorithm in cloud computing environment," *Journal of Computers*, vol. 7, no.1 pp.42-52, 2012. Article (CrossRef Link).

[19] Li MF, Bi JP, Li ZC, "Resource Scheduling Waiting Aware Virtual Machine Consolidation," *Journal of Software*, vol.25, no. 7, pp.1388-1402, 2014. Article (CrossRef Link).

[20] G. Hamilton, "Distributed virtual machine migration for cloud data center environments," University of Glasgow, 2014. Article (CrossRef Link).

[21] Da-wei Sun, Guiran Chang, Fengyun Li, Chuan Wang, and Xingwei Wang, "Optimizing multi-dimensional QoS cloud resource scheduling by immune clonal with preference," *Acta Electronica Sinica*, vol. 39, no.8, pp.1824-1831, 2011. Article (CrossRef Link).

[22] Aiping Yuan and Canjun Wan, "Virtual machine deployment strategy based on improved genetic algorithm in cloud computing environment," *Journal of Computer Applications*, vol. 34, no. 2, pp.357-9,364, 2014. Article (CrossRef Link).

[23] Ö. Ülker, E. E. Korkmaz, and E. Özcan, "A grouping genetic algorithm using linear linkage encoding for bin packing," in *Proc. of Parallel Problem Solving from Nature–PPSN X, ed: Springer*, pp. 1140-1149,2008. Article (CrossRef Link).

[24] Dongdong Yang, Llicheng Jiao,Maoguo Gong and Hang Yu, "Clonal selection algorithm for solving preference multi-objective optimization," *Journal of Software*, vol. 21, no.1, pp. 14-33, 2010. Article (CrossRef Link).

[25] Yong Liu, Xinhua Wang, Changming Xing and Shuo Wang, “Resources scheduling strategy based on ant colony optimization algorithms in cloud computing,” 2011. Article (CrossRef Link).

[26] M. Emmerich, N. Beume, and B. Naujoks, “An EMO algorithm using the hypervolume measure as selection criterion,” *Lecture Notes in Computer Science*, vol.3410, pp.62-76, *2005*. Article (CrossRef Link).

[27] N. Beume, B. Naujoks, and M. Emmerich, “SMS-EMOA: Multi-objective selection based on dominated hypervolume,” *European Journal of Operational Research*, vol. 181, no. 3, pp.1653-1669, 2007. Article (CrossRef Link).

[28] Ping Guo and Qi li, “Load balance scheduling algorithm based on the load on the server status classification,” *Journal of Huazhong University of science and technology: Natural Science Edition*, vol. 40, no. 1, pp.62-65, 2012. Article (CrossRef Link).

Shukun Liu received the M.S. degree in computer science and technology from University of South China, PR China, in 2007. Currently, He has been a Ph.D. candidate at Central South University since September 2013, China. His major research interests include cloud computing, virtualization technology, performance analysis, computer networks, database technology, data mining and software engineering. He has published nearly fifteen papers in related journals. And he is a member of CCF and a member of ACM.

Prof. Weijia Jia is a Zhiyuan Chair Professor in the Department of Computer Science and Engineering, Shanghai Jiaotong University. He joined German National Research Center for Information Science (GMD) in Bonn (St. Augustine) from 1993 to 1995 as a research fellow. During 1995-2013, he joined Department of Computer Science, City University of HK as a professor. Weijia Jia received BSc and MSc from Center South University, China in 1982 and 1984 and Master of Applied Sci. and PhD from Polytechnic Faculty of Mons, Belgium in 1992 and 1993 respectively, all in Computer Science. He is the guest Professor of Beijing University of Science and Technology of China, Beijing Jiao Tong University, Jinan University, Guangzhou, Chengdu University, China. He has served as the editors of IEEE TPDS and ComCom and PC chairs and members/keynote speakers for various prestige international conferences. He is the Senior Member of IEEE and the Member of ACM. His research interests include next generation wireless communication, protocols and heterogeneous networks; distributed systems, multicast and anycast QoS routing protocols. In these fields, he has a number of publications in the prestige international journals (IEEE Transactions, e.g., ToN, TPDS, TC, TMC etc.), books/chapters and refereed international conference proceedings (e.g. ACM CCS, WiSec, MobiHoc, SenSys, IEEE ICDCS, INFOCOM etc.). He (with W. Zhou) has published a book “Distributed Network Systems” by Springer where the book contains extensive research materials and implementation examples. He has received the best paper award in a prestige (IEEE) conference.