Research on Dynamic Virtual Machine Scheduling Strategy Based on Improved Genetic Algorithm

Jingmei Li, Shuang Yang*, Jiaxiang Wang, Linfeng Yang
College of Computer Science and Technology, Harbin Engineering University
*a Corresponding author: author@e-mail.org

Abstract. In the actual application, when the virtual machine is created and the virtual machine is dynamically migrated, only the memory utilization of the host is considered, and other resource indicators, such as CPU utilization, are not considered, so the resource utilization may be unbalanced. This leads to unreasonable allocation of cloud computing resources in virtual machine scheduling. In order to solve the above problems, this paper proposes a dynamic virtual machine scheduling strategy based on improved genetic algorithm, which fully considers the CPU utilization and memory utilization of the host, and obtains the minimum dynamic migration overhead virtual machine scheduling scheme under the condition that the above resource utilization is relatively balanced. Finally, the algorithm was verified by CloudSim simulation experiment. Experimental data prove that the proposed algorithm has balanced resource utilization and effectively solves the load imbalance problem in virtual machine scheduling.

1. Introduction
In recent years, cloud computing has become a popular computing model. It provides users with more targeted, cheaper and faster ways to achieve resource management and use through the integration of storage resources, computing resources and software services. Separation. OpenStack belongs to the most popular open source IaaS (Infrastructure as a Service) cloud platform in the cloud computing field. With the widespread use of cloud computing, how to make the host resources of the entire cloud data center fully and balancedly utilized has become one of the most important issues in virtual machine scheduling in cloud computing environments[1].

2. Dynamic virtual machine scheduling

2.1 Virtualization technology
Currently mainstream server virtualization technologies are Vmware, Xen and KVM. Among them, KVM is based on the Linux kernel, and OpenStack support for KVM is also the best[2]. Although KVM supports CPU over-commitment, if the CPU load is close to 100% when it is over-used, it may cause system instability[3].

The default CPU over-proportion ratio in OpenStack is 1:16. For example, a server host (referred to as a host in this article) has n logical CPUs, so the number of vCPUs (virtual CPUs) of all virtual machines allocated by this host cannot be More than n*16, where each virtual machine has a vCPU less than n. The default memory over-proportion ratio in OpenStack is 1:1.5, which means that one host has m (unit: GB) memory, then the total memory of all virtual machines allocated by this host
must not exceed 1.5*m (unit: GB). Memory, where each virtual machine’s memory is less than m (unit: GB). Based on this, the scheduling of virtual machines in this paper follows the following principles: The CPU’s over-proportion ratio is calculated as 1:16. The number of vCPUs per virtual machine is less than the minimum number of logical CPUs in all hosts; The memory over-proportion ratio is calculated as 1:1.5, and the memory of each virtual machine is less than the minimum memory of all hosts.

2.2 Virtual machine scheduling mode

In the virtual machine scheduling of cloud computing, two types of scheduling modes are included, namely static scheduling (offline scheduling)\(^\text{(4)}\) and dynamic scheduling (online scheduling)\(^\text{(5)}\). In static scheduling, the main emphasis is on the pre-scheduling scheme. The relevant scheduling scheme has been developed before the actual scheduling. Therefore, effective allocation is required before scheduling, and adjustment cannot be performed after scheduling which requires strong judgment. For the dynamic cloud computing virtual machine scheduling algorithm, it can be adjusted in time during execution or in the running process, which has good execution efficiency and application value.

2.3 OpenStack virtual machine scheduling policy

OpenStack defaults to a simple algorithm, as shown in Figure 1. The simple algorithm is divided into two steps. The first step is to filter, find the host that meets the running conditions of the virtual machine to be scheduled from all the hosts, and then find the most suitable host from the filtered hosts. The Scheduler can filter multiple filters in turn, and the filtered nodes select the most suitable nodes by calculating the weights\(^\text{(6)}\).

![Figure 1. OpenStack virtual machine scheduling strategy](image)

This overly simple scheduling strategy can easily lead to unbalanced use of hardware resources, which will increase the energy consumption of the cluster; a simple scheduling algorithm cannot solve the problem of how to efficiently schedule during long-term use. Therefore, the virtual machine scheduling used by OpenStack does not realize the reasonable allocation of cloud computing resources. For the convenience of research, this paper selects CPU and memory as indicators to measure host load.

2.4 Dynamic virtual machine scheduling model

Dynamic virtual machine scheduling (virtual machine includes a1 virtual machines to be scheduled and a2 scheduled virtual machines) refers to a1 (in the case of a1=1 in this paper), the virtual machines to be scheduled are allocated to b hosts according to certain policies. Run, a2 scheduled virtual machines according to a certain policy to determine whether each virtual machine is dynamically migrated. Dynamic virtual machine scheduling problems can be represented by triples \(S=(H,V,Q)\).

\[H=\{H_1,H_2,...,H_i,...,H_b\}, 1\leq i \leq b\}\) represents a collection of b hosts, \(H_i\) represents the i-th host, \(H_i=\{\text{Num}_\text{CPU}_H_i,\text{Num}_\text{RAM}_H_i\}\), \(\text{Num}_\text{CPU}_H_i\) indicates the number of physical CPUs on the host \(H_i\), \(\text{Num}_\text{RAM}_H_i\) indicates the size of the actual memory on the host \(H_i\).

\[V=\{V_1,V_2,...,V_{a_1},...,V_{a_1+a_2}\}\) represents a collection of all virtual machines. When \(1\leq j \leq a_1\), \(V_j\) represents a virtual machine to be scheduled. When \(a_1 < j \leq a_1+a_2\), \(V_j\) represents a scheduled virtual machine. For any virtual machine,
\[ V_j = (\text{Num}_vCPU_{V_j}, \text{Num}_vRAM_{V_j}) \quad 1 \leq j \leq a_1 + a_2 \quad V_j \] represents the jth virtual machine. \( \text{Num}_vCPU_{V_j} \) indicates the number of CPUs required by the virtual machine \( V_j \), and \( \text{Num}_vRAM_{V_j} \) represents the required memory size of the virtual machine \( V_j \).

\[ Q = \{ Q_{ij} \mid 1 \leq i \leq b, a_i \leq j \leq a_i \} \] indicates that the collection of virtual machine and host mapping relationships has been scheduled. Where \( Q_{ij} \) indicates that the virtual machine \( V_j \) is running on the host \( H_i \).

According to the principle of virtual machine scheduling in the virtualization technology of 2.1, the CPU's over-proportion ratio is calculated as 1:16, as shown in formula (1). \( \text{Num}_vCPU_H_i \) indicates the number of virtual CPUs that the host \( H_i \) can provide. The memory over-proportion ratio is calculated as 1:1.5, as shown in formula (2). \( \text{Num}_vRAM_H_i \) indicates the amount of virtual memory that the host \( H_i \) can provide.

\[ \text{Num}_vCPU_H_i = 16 \times \text{Num}_CPU_H_i \quad (1) \]
\[ \text{Num}_vRAM_H_i = 1.5 \times \text{Num}_RAM_H_i \quad (2) \]

### 3. Dynamic Virtual Machine Scheduling Strategy Based on Genetic Algorithm in OpenStack

Genetic algorithm is a computational model that simulates the natural evolution of Darwin's biological evolution theory and the biological evolution process of genetic mechanism[7]. The main idea of the genetic algorithm is to calculate the fitness of each chromosome from the initial population, select excellent genes for genetic manipulation, replace the previous generation with better genes, and gradually improve the solution of the problem through such a process[8].

#### 3.1 Chromosome coding

In this paper, chromosomes are encoded in a tree structure, and each mapping scheme corresponds to a chromosome tree C-Tree. C-Tree has a three-layer structure: the first layer is the root node; the second layer node represents the host; the third layer node represents the virtual machine running on the host represented by its parent node. Figure 2 shows a C-Tree example diagram where the hosts are \( H_1, H_2, H_b, \ldots, H_b \). Virtual machines \( V_1 \) and \( V_2 \) run on host \( H_1 \), virtual machines \( V_3, V_4, \) and \( V_5 \) run on host \( H_2 \), and so on.

![C-Tree example diagram](image)

**Figure 2. C-Tree example diagram**

Before and after the virtual machine is scheduled, the C-Tree changes are shown in Figure 3. The to-be-scheduled virtual machine \( V_{at+1} \) is scheduled to run on the host \( H_b \). Before the dynamic virtual machine is scheduled, the virtual machines \( V_1 \) and \( V_2 \) run on the host \( H_2 \). After the dynamic virtual machine is scheduled, the virtual machines \( V_1, V_3, \) and \( V_4 \) run on the host \( H_1 \). It can be seen that the virtual machine \( V_4 \) is dynamically migrated to the host \( H_f \) after scheduling.
3.2 Genetic algorithm specific steps

The specific execution flow of the genetic algorithm is as follows: First, the initial population is formed according to a certain initialization method; secondly, in each evolution, the chromosomes in the population are selected, and then the chromosome is further manipulated according to the fixed cross mutation probability. At the end of the evolution, the better chromosomes are selected; finally, the better chromosomes are formed into new populations, and genetic operations are resumed.

1) Population initialization

Population initialization is a key step in genetic algorithms. A better initial population can shorten the time to get the optimal solution. Set the genetic algebra counter $t = 0$ to set the maximum evolution algebra $T$. When initializing the population, this paper randomly selects $M$ chromosomes.

2) Calculating fitness values

The fitness function is a method for judging whether a chromosome is excellent in a genetic algorithm. The CPU and memory utilization balance of each host is an important measure. The specific steps are:

1) Calculate the CPU and memory utilization of each host. As shown in formula (3) and (4), $U_{CPU}$ and $U_{RAM}$ respectively represent the CPU and memory utilization of the virtual machine scheduled to the host $H_i$.

$$U_{CPU}^i = \frac{\sum \text{Num}_v \text{CPU}_V - V_i \in Q}{\text{Num}_v \text{CPU}_H - H_i}$$  \hspace{1cm} (3)

$$U_{RAM}^i = \frac{\sum \text{Num}_v \text{RAM}_V - V_i \in Q}{\text{Num}_v \text{RAM}_H - H_i}$$  \hspace{1cm} (4)

2) Calculates the average CPU and memory utilization of hosts in the cluster. According to formula (4) and (5), the average utilization ratios $U_{CPU}^{avg}$ and $U_{RAM}^{avg}$ of CPU and memory of each host can be obtained. In the formula, $b$ is the number of hosts in the cluster.
3) Calculate the utilization balance of CPU and memory resources. Solve using formula (7) and formula (8), where the role of the constant $\eta$ is to prevent the calculation result from being too small and the error is too large, in this paper $\eta=100$.

$$
D_{CPU}^{CPU} = \sqrt{\frac{1}{b} \sum_{i=1}^{b} (\eta * U_{CPU}^i - \eta * U_{avg}^{CPU})^2}
$$

(7)

$$
D_{RAM}^{RAM} = \sqrt{\frac{1}{b} \sum_{i=1}^{b} (\eta * U_{RAM}^i - \eta * U_{avg}^{RAM})^2}
$$

(8)

4) Calculate the migration cost. The migration cost is calculated according to formula (9), where $u$ represents the number of migrations, $K$ is a constant, indicating the combined cost of one migration, and finally the migration cost $Q$ of the chromosome is obtained. This article is convenient for the calculation order $K=1$.

$$
Q = u * K
$$

(9)

5) Calculate the fitness value. The fitness value can be calculated by the formula (10), where $R$ is an adjustment constant, and the effect is to ensure that the fitness value is a positive number. $\alpha , \beta$ and $\gamma$ are adjustment factors. Their role is to balance the importance of CPU and memory and migration costs. In this paper, the adjustment factor is 1.

$$
D = R - \alpha * D_{CPU}^{CPU} - \beta * D_{RAM}^{RAM} - \gamma * Q
$$

(10)

(3) Selection operation

Using the adaptive value ratio selection method, the fitness value of the chromosome is directly proportional to the probability of being selected, and the probability that each chromosome is selected can be calculated by formula (11).

$$
P_i = \frac{F_i}{\sum_{i=1}^{M} F_i} (i = 1, 2, 3, ..., M)
$$

(11)

In equation (11), $P_i$ represents the probability that the i-th chromosome is selected, $F_i$ is the fitness of the i-th chromosome, and $M$ represents the population size.

(4) Crossover operation

According to the fitness ratio selection method in the selection operation, two chromosomes $C_1$ and $C_2$ are selected, the even-numbered virtual machines in $C_1$ are deleted, the odd-numbered virtual machines in $C_2$ are deleted, and finally $C_1$ and $C_2$ are combined to obtain the sub-chromosome $C_3$.

(5) Mutation operation

Mutation refers to the process of randomly changing some genes in the parent chromosome to generate new chromosomes according to the probability of mutation $P_t$. Usually, the value of $P_t$ is very small. Select two virtual machines for exchange on the two target physical machines.
\[ P_t = e^{-\lambda t M (a_1 + a_2)} \]

\((12)\)

\(P_t\) is an adaptive mutation probability. In formula (12), \(\lambda\) is a constant, \(t\) is the genetic algebra, \(M\) is the population size, and \(a_1 + a_2\) is the number of virtual machines.

(6) Termination condition judgment

If \(t = T\), the genetic algebra \(t\) is equal to the given maximum evolution algebra \(T\) when the exit operation. Then, the chromosome with the largest fitness obtained in the evolution process is used as the optimal solution output, and the calculation is terminated.

4. Simulation

Simulation experiments were performed using the CloudSim simulation software in a Java environment.

Suppose there are 50 virtual machines and 5 physical machines, and the genetic algorithm of this paper is as follows: \(M=50\), variation 1.5 \(= 1.5\), adjustment constant \(R=100\), adjustment factor. The configuration of the five physical machines is shown in Table 1.

|              | H₁  | H₂  | H₃  | H₄  | H₅  |
|--------------|-----|-----|-----|-----|-----|
| CPU (core)   | 4   | 8   | 4   | 8   | 16  |
| RAM (G)      | 32  | 32  | 16  | 16  | 64  |

In the experiment, the number of virtual machine tests was 10, 20, 30, 40, and 50, respectively. The experimental results are shown in Figure 4.

![Figure 4](image_url)

Figure 4. Comparison of Optimal Fitness Values of Three Algorithms with Different Number of Virtual Machines

It can be seen from Fig. 4 that compared with the simple algorithm and the random selection algorithm of the OpenStack default virtual machine scheduling strategy proposed in this paper, the optimal fitness value is stable and the advantages are obvious, and it is more balanced on CPU and memory usage. The experiment proves the effectiveness of the genetic algorithm applied in the virtual machine scheduling strategy.

5. Conclusion

This paper introduces an improved genetic algorithm for the shortcomings of the default scheduling algorithm in OpenStack. In the fitness function of the genetic algorithm, the utilization of CPU and memory is considered comprehensively, and the virtual machine scheduling strategy based on improved genetic algorithm is realized. Through simulation experiments on CloudSim cloud simulation platform, the experimental results show that the virtual machine scheduling strategy based on improved genetic algorithm has a more balanced resource utilization, which improves the overall resource utilization to some extent and effectively solves the load imbalance problem in virtual machine scheduling.
Acknowledgements

This work was supported by National Key Research and Development Plan of China (No 2016YFB0801004).

References

[1] Santra S, Mali K. A new approach to survey on load balancing in VM in cloud computing: Using CloudSim[C]. International Conference on Computer, Communication and Control. IEEE, 1(2016).

[2] Sonkar S K, Kharat M U. A review on resource allocation and VM scheduling techniques and a model for efficient resource management in cloud computing environment[C]. International Conference on ICT in Business Industry & Government. IEEE, 2017:1-7.

[3] Wenying Yue, Yoshinori Hayafuji. Dynamic Placement of Virtual Machines with Both Deterministic and Stochastic Demands for Green Cloud Computing [J]. Mathematical Problems in Engineering, 2014

[4] Renuga Kanagavelu, Bu-Sung Lee, Nguyen The Dat Le, Luke Ng Mingjie, Khin Mi Mi Aung. Virtual machine placement with two-path traffic routing for reduced congestion in data center networks[J]. Computer Communications. 2014

[5] Qunying Huang, Chaowei Yang, Kai Liu, Jizhe Xia, Chen Xu, Jing Li, Zhenglong Li. Evaluating open-source cloud computing solutions for geosciences[J]. Computers and Geosciences. 2013

[6] Sharrukh Zaman. Combinatorial auction-based allocation of VM instances in clouds [J]. Journal of Parallel and Distributed Computing. 2012

[7] Vahora S, Patel R. CloudSim-A Survey on VM Management Techniques[J]. 2015:128-133.

[8] Weiwei Lin, Chen Liang. A Threshold-based Dynamic Resource Allocation Scheme for Cloud Computing[J]. Procedia Engineering. 2011.