An Initial Resource Scheduling Algorithm for OpenStack

Feng-Jun Shang, Lu-Zhong Li and Xuan-Ling Chen

ABSTRACT

In view of resource scheduling problems in OpenStack, we put forward the initial resource scheduling algorithm based on matching hungry degree, which named as MHD-VMP. The algorithm computes the matching degree between the demand resource of the virtual machine and the available resources of the physical machine and the hungry degree of all the virtual machines in the physical machines, then we can get the matching hungry degree of matching pairs, which can select the most appropriate physical machine to place virtual machine. Finally, the simulation results demonstrate the proposed algorithm significantly outperforms the initial virtual machine placement algorithm of OpenStack.

INTRODUCTION

In recent years, cloud computing has become a popular computing model [1]. The biggest contribution of cloud computing is that it changes the allocation of physical hardware resources. The cloud data center constructs the virtual resource pool by CPU, memory, storage and network through virtualization technology. The virtual resource management technology is used to realize automatic resource deployment, dynamic expansion and on-demand distribution [2]. But the data show that: the resource utilization of cloud data center is generally not high, only about 10% on average, and the server is idle most of the time. In spite of this, it will bring 60% of the power consumption of server which is full load [3]. The root cause of the problem is the unreasonable resource scheduling mechanism, which will ultimately lead to lower resource utilization of the platform [4]. It can be seen that the reasonable placement of VM is an important way to solve this problem.

Feng-Jun Shang¹, Lu-Zhong Li² and Xuan-Ling Chen³
¹College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, China
²College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, China
³College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, China
RELATED WORK

In this section, the existing research on the initial placement of virtual machine in cloud computing is divided into traditional heuristic methods such as first-time matching algorithm, best matching algorithm and biological intelligent heuristic algorithm, such as genetic algorithm and ant colony algorithm. In [5], the authors introduce an ant colony algorithm based on performance matching to perform batch deployment of virtual machines, but he doesn’t take the virtual machine dynamic resource requirements into account. In [6], the authors propose a heuristic approach based on an improved ant colony algorithm (ACA) to solve the virtual machine placement problem, named as GACA-VMP, but it is only currently deployed on homogeneous cloud environment. In [7], the authors propose a virtual machine placement method based on improved simulated annealing algorithm (ISA) to solve the problem that traditional heuristic algorithm costs long time to reach an optimal allocation. However, this article does not consider factors such as response time. In [8], the authors propose an exact Integer Nonlinear Program (INLP) and a heuristic to solve the Reliable VM Placement (RVMP) problem. However its running time is significantly larger than the most of heuristics. All of the above algorithms can perform well under certain conditions, but there is unstable performance necessarily for OpenStack. Therefore, this paper proposes an improved initial placement algorithm of virtual machines based on matching hungry degree of OpenStack, named as MHD-VMP.

VM PLACEMENT ALGORITHM BASED ON MATCHING HUNGRY DEGREE

In OpenStack, the nova-scheduler has two virtual machine placement mechanisms: random placement algorithm and filter-weight placement algorithm. The first is not used by default and it’s not practical. The second placement algorithm is divided into three steps: First, the PM is selected according to the physical resource that can meet resource request of virtual machine. Then the weights according to the remaining available memory index of the selected PM are calculated. Finally, the PM of largest weight is been chosen in sorted list of hosts to place virtual machines. It can be seem that there are many shortcomings.

VM Placement Algorithm Model

The placement problem of virtual machine is a Bin packing problem, that is a classical NP-hard problem. "Boxes" (PM) and "objects" (VM) may be heterogeneous. A physical machine can deploy more than one virtual machine, but virtual machine can only be deployed on one physical machine. The virtual resource scheduling is a process of mapping the virtual resources to the physical resources rationally. Virtual resources can be abstracted as: \( \{ \text{vm}_i \}_i \) and the physical machine resources can be abstracted into \( \{ \text{pm}_j \}_j \), and \( c \) is CPU, \( m \) is memory, \( d \) is disk, \( b \) is the bandwidth. Corresponding to the virtual resources \( v_m \{ c_i, m_i, d_i, b_i \} \) and physical resources \( p_m \{ c_j, m_j, d_j, b_j \} \), for a given set of \( PM \{ \text{pm}_1, \text{pm}_2, \ldots, \text{pm}_n \} \) and \( VM \{ \text{vm}_1, \text{vm}_2, \ldots, \text{vm}_n \} \), the PM assigned to the
VM is represented by \( R = \{ (vm_i, pm_j) | M_{ij} = 1, \forall i, j \} \), where \( \sum c_{ij} < c_{ji} \), and \( i \) is the label of the virtual machine placed on the physical machine \( pm_j \), indicating that the total CPU resources of all the virtual machines on a physical machine can't exceed the CPU resources of the physical machine. The other three resource properties correspond to such formulas.

**Matching Hungry Degree-VMP Algorithm**

*Sort-matching method based on matching degree*

The main idea of the sort-matching method is to match the performance requirements of the virtual machine with the idle performance of the physical machine, and find the best matching physical machine to place the virtual machine.

**Definition 1**: The matching degree \( P_{id} \). We use the Euclidean distance to determine the performance of the matching degree between virtual and physical machine. We consider the main four resources: CPU, memory, storage, and bandwidth. And note that the four resources parameters have different contribution when the type of matching pairs of virtual and physical machines is different. For example, the matching process of the CPU intensive virtual and physical machine, we focus the CPU obviously, that is the matching degree of CPU will have a greater impact. Therefore, we design different weights \( w_i \) according to the type of matching pairs of virtual and physical machine. The formula shown in Eq.(1):

\[
P_{id} = \sqrt{w_1 (vm_{cpu} - pm_{cpu})^2 + w_2 (vm_{ram} - pm_{ram})^2 + w_3 (vm_{storage} - pm_{storage})^2 + w_4 (vm_{bw} - pm_{bw})^2}
\]

(1)

The \( P_{id} \) is smaller, that physical machine can adapt to the performance of the virtual machine request, also shows the best match in performance of the physical and virtual machines, and the system resources can be fully used, the load is more balanced, the number of resource fragmentation is the least. The weight is determined by the type of matching pairs of virtual machines and physical machines, \( w_i \) all need to satisfy the constraints: \( \sum_{i=1}^{n} w_i = 1, 0 < w_i < 1, i = 1, 2, \ldots, n \) inevitably. Here we define the \( w_i \) of the CPU intensive matching pair is \( w_i = 0.55, w_2 = 0.15, w_3 = 0.15, w_4 = 0.15 \). The remaining three types of matching pairs are similar. Finally, we sort the PM according to the \( P_{id} \).

**MHD-VMP Algorithm**

As we mentioned, cloud service providers adopt the pre-allocated and on-demand allocation of resources according to the requirements of users. And the virtual machine share physical machine resources, it can't be kept 100% requirement on time. Most of the time, it is enough when the pre-allocated resources can meet the performance requirements, and most of resources will be idle. So we adopt the pseudo excess usage of resource, it is seem that the preassigned resources of VM is more than the sum of the real resources of PM, but it is not real. In fact, some VM will share idle resources temporarily to others. When the second source virtual machine in the face of high-performance request, it
needs to retreat the resources. If the resources utilization of PM is very high, the probability of failure to retreat resource will increase, which will inevitably lead to the VM migration probability, and this has a significant impact on the performance of the cloud platform.

**Definition 2:** The pseudo excess usage of resource. We assume the resource of physical machine is 1.25 times the actual amount physical resources. This will ensure that the resource utilization is relatively high, but it will not really over-use.

**Definition 3:** The degree of VM resources hunger (Hungry). The resources hunger of virtual machine are the demand expectations of VM to retreat the secondment of resources essentially. Demand expectations of retracing resources, that are the retreat of the desire and degree for the secondment of virtual machine resources, namely degree of hunger and thirst for borrowing out of resources. The degree of hunger is shown in Eq.(2):

\[ H = \sum V \times E \]

Where \( H \) is the strength of the retracement secondment resources of VM, \( V \) represents the volume of resources that the VM retrace and \( E \) is the probability of VM to retrace resources.

This part of the discussion is the virtual machine resources retracement expectations of secondment to go out, because we don't know the specific virtual machine actual borrowing out how many resources, we will put \( V_h = V - V_{\text{use}} \) (\( V \) is the pre-allocated resources of VM, \( V_{\text{use}} \) is the actual use of resources of VM, so \( V_h \) is the hunger for resources quantity) as the resources of VM to share with others. So we assume that the physical machine \( pm_j \) have \( n \) virtual machines, that is \( pm_j(\text{vm}_{j1}, \text{vm}_{j2}, \cdots, \text{vm}_{jm}) \), as each virtual machine \( \text{vm}_j \) amount of hunger for four resources can be regarded as \( \text{vm}_{ji}(V_{jiCPU}, V_{jiRam}, V_{jiStorage}, V_{jiBw}) \), that is the hungry of virtual machines \( i \) on the physical machine \( j \), and \( V_{jiCPU}, V_{jiRam}, V_{jiStorage}, V_{jiBw} \) is the hunger of the four resources. The hunger of physical machine is the sum of the amount of VM's hunger, here we define each physical machine's hunger for resources in Eq.(3):

\[ \text{Hungry}_{pm_j} = \sum_{i=1}^{n} \left( V_{jiCPU} \times e_{jiCPU} + V_{jiRam} \times e_{jiRam} + V_{jiStorage} \times e_{jiStorage} + V_{jiBw} \times e_{jiBw} \right) \]

Where \( u_{jiCPU} \) denotes the resource utilization of CPU and \( e_{jiCPU} \) is the probability of CPU resources to retreat of the \( \text{vm}_i \) on \( pm_j \). But we are unable to know the probability of the four resources to retreat accurately, so we think it is identical. So we define the hungry probability \( e_{ji} \) as 1 conveniently. Finally, hunger can be expressed as Eq.(5):

\[ \text{Hungry}_{pm_j} = \sum_{i=1}^{n} \left[ \text{vm}_{jiCPU} \times (1-u_{jiCPU}) + \text{vm}_{jiRam} \times (1-u_{jiRam}) + \text{vm}_{jiStorage} \times (1-u_{jiStorage}) + \text{vm}_{jiBw} \times (1-u_{jiBw}) \right] \]

Where \( u_{jiCPU} \) is the CPU utilization of the \( \text{vm}_i \) on \( pm_j \). We know \( u_{ji} \) is bigger, \( \text{Hungry}_{pm_j} \) is smaller. In general, the resource utilization of virtual machine is higher and the resource utilization of physical machine is lower, we think the host is more suitable. Then we take \( P_{edj} \) into account. Finally the possibility of a physical machine to be selected is the utility function of matching degree and hunger. And the possibility of the matching pairs in Eq.(6):
Probability_{ij} = \alpha \cdot P_{e_{ij}} + \beta \cdot u_{PM_{i}} + \gamma \cdot Hungry_{j}, \alpha + \beta + \gamma = 1 \tag{6}

Here we define $\alpha = 0.5, \beta = 0.25, \gamma = 0.25$. According to Algorithm MDH-VMP, we select the pair of the smallest Probabilityij as the matching pair (vmi, pmj) of PM and VM.

SIMULATION EXPERIMENT AND RESULT ANALYSIS

As the experimental environment is limited, so the proposed algorithm has been evaluated with CloudSim in this paper. CloudSim is a cloud computing emulation software from the Grid Labs at the University of Melbourne in Australia. The user-developed scheduling algorithm can be implemented in it. And we designed an experimental scenario: the establishment of data center with 100 PMs, then we generate a certain number of VM randomly. The four resources parameters of PM and VM as shown in the table 1.

<table>
<thead>
<tr>
<th></th>
<th>CPU(MIPS)</th>
<th>RAM(G)</th>
<th>STORAGE(G)</th>
<th>BW(G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM</td>
<td>[1860,2660]</td>
<td>{4,4}</td>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td>VM</td>
<td>{2500,2000,1000,500}</td>
<td>{0.87, 1.74, 1.74, 0.613}</td>
<td>250</td>
<td>0.1</td>
</tr>
</tbody>
</table>

From the simulation results depicted in Fig.1 and Fig.2, it is completely obvious that the MHD-VMP algorithm significantly in the numbers of the running physical machines and the energy consumption outperforms the initial VM placement algorithm of OpenStack.

CONCLUSION

This paper studies the virtual machine placement algorithm of OpenStack, and puts forward the VM placement algorithm (MHD-VMP) of OpenStack based on matching hungry degree, and the algorithm can improve the resource utilization and reduce the number and the power consumption of physical machines, so it is practical. The next step of this paper will focus on the placement method, which can balance the resource utilization, service quality and energy consumption. More importantly, it can reduce the probability of subsequent migration of VM. It can optimize the placement of VM in large-scale data center.
ACKNOWLEDGMENTS

The author would like to thank the Chongqing Basic and Frontier Research Project under Grant NO. cstc2016jcyjA0590. The work is partly funded by the National Nature Science Foundation of China (No. 61672004).

REFERENCES