Virtual Machine Placement Based on the VM Performance Models in Cloud

Hui Zhao, Qinghua Zheng, *Member*, *IEEE*, Weizhan Zhang *Member*, *IEEE*, Yuxuan Chen, Yunhui Huang
SPKLSTN Lab, Department of Computer Science and Technology, Xi'an Jiaotong University
No.28, Xianning West Road, Xi'an, Shaanxi, 710049, P.R. China
zhangwzh@mail.xjtu.edu.cn

***Abstract*—Cloud service providers can offer users virtual machines (VMs) on-demand as a service over the Internet. VMs running on top of a physical machine (PM) share physical resources (CPU, memory, and/or bandwidth), and there may be a great resource contention among them, which results in VMs performance degradation. To prevent this, cloud providers need to study how to place VMs on PMs efficiently. However, the existing virtual machine placement (VMP) methods mainly tried to optimize the cloud resources instead of the VM performance. In this paper, we propose a VMP method based on the VM performance models in cloud. Firstly, with a real OpenStack cloud platform, we study the virtualization resource scheduling principle, analyze the interaction among VMs with shared hardware, consider the relationship between VMs and the host PM, and then we introduce the VM performance models to present the VM performance degradation problem. Secondly, to choose an appropriate PM for placing VM, we take into consideration the application-aware resource consumption characteristic, the VM resource requirement and the VM performance models, so as to minimize the PM performance degradation and guarantee the VM performance. Finally, we take the streaming media services for examples, and conduct some experiments to evaluate our method. The results show it works better than others and guarantees the VM performance significantly.**

Keywords*—cloud; virtual machine placement; performance degradation; VM performance models.

I. Introduction

The public cloud service providers can utilize virtualization technology to offer various different server instances (storage, processors, bandwidth, and virtual machine) as a service like electricity, water, and gas to customers over the Internet. Each of these instances provides a certain amount of dedicated resources and charges per instance-hour. Using public cloud, customers can eliminate their purchasing and maintaining cost, and scale up and down resources dynamically based on their needs, which can help them to focus on their business.

To increase the resource utilization in cloud, the cloud service providers use virtualization technology to execute several VMs concurrently on top of a PM. Each VM employs a partition of resource capacity of PM. To host a VM, a PM must have enough available resources, including CPU, memory, storage, and bandwidth. Though VMs in a PM run individually with their proprietary resources, they share the same physical resources, which may result in VMs performance degradation. If two VMs with intense CPU requirements are placed on the same PM, the resource utilization in PM may be imbalance, e. g., there is little CPU left but a lot of memory and bandwidth remaining. This CPU scarcity may block the placement of a new VM on this PM and degrade the VM performance.

To prevent this, cloud service providers should study how to place multiple VMs with different requirements on PMs efficiently, known as virtual machine placement (VMP) method. VMP is the process of mapping VMs on to appropriate PMs by resolving constraints of customers and cloud service provider. Generally, VMP can be divided into two different types, that is, static (initial) placement and runtime (dynamic) placement. In static placement, VMs are created for applications and placed at appropriate PMs, while in dynamic placement, VMs can be migrated online among PMs when VMs are already is running. In this paper, we focus on the initial placement of VMs. There are also some existing works about initial VMP methods, but they mainly focus on optimizing the cloud resources, such as cost [1]–[3], energy [4]–[10], traffic [11]–[13], network performance [14], [15] or multi-objectives [16]–[18]. However, they did not take the VM performance into account, while the VM performance is a very important factor to impact user's experience.

In this paper, we propose a VMP method based on the VM performance models. It aims to offer the VMs with best performance to users. Firstly, we studied how to formulate the VM performance models. We establish a real cloud platform with OpenStack. With the methods of theoretical analysis and experimental evaluation, we study the nature of virtualization resources scheduling principle on OpenStack cloud platform, investigate the interaction among VMs with shared hardware, and consider the relationship between VM performance and the host PM resource utilization, we then introduce our models to formulate the VM performance under various resource consumptions of PM. It can present the performance degradation problem caused by resource sharing and contention among VMs in the same host PM. Specifically, the performance is actually the relative performance as the ratio between the VM performance with and without resource contention. Based on the VM performance models we can calculate how the performance will be changed when a new VM placed on a PM, which is a fundamental base for the following VMP method. Of course, there are many resources which affect VMs greatly, but we mainly focus on three resources, that is, CPU, memory, and networks in this paper.

Secondly, we apperceive the service features of VMs with different applications, that is, we analyze and obtain the resource consumption characteristic of different applications. For example, the VMs running video live streaming media applications should require lots of bandwidth, while VMs running video transcoding need powerful CPU resources. We can know that different applications have different demands. To choose an appropriate PM as the placed machine, we should take into consideration the application-aware resource consumption characteristic, the VM resource requirement and the VM performance models.

Finally, we conduct experiments in our OpenStack cloud platform to evaluate our method and other VMP methods. In our OpenStack cloud platform, we provide some streaming media services, including live streaming, video-on-demand, and video transcoding, so we take these applications as examples to evaluate the VM performance. The results show that our VMP method can work better than other methods and can provide VMs with the better performance.

Our main contribution lays particular emphasis on the VMP method. To summarize, the contributions of this paper are as follows:

1) We propose VM performance models to present the performance degradation caused by resource sharing and contention among VMs on the same host PM. With the models, we can predict the future VM performance, which is a fundamental base for the following placement.

2) Based on VM performance models, we propose a VMP method, which takes the application-aware resource consumption characteristic, the VM resource requirement and the VM performance models into account to place VMs on appropriate PMs.

3) We establish a real cloud platform with OpenStack, and conduct experiments to evaluate our VMP method. The results show it works better than other methods and can significantly ensure the VM performance.

The remainder of this paper is organized as follows. Section II summarizes related works. Section III describes our VM performance models. Section IV gives the VMP method based on VM performance models in detail. We then evaluate our VMP method in Section V. Section VI concludes the paper and gives the future works.

II. Related Works

There have been some researches about VMP, but they mainly focus on how to optimize the cloud resources in different aspects, such as cost, energy, traffic, and network performance.

Saving cost [1]–[3] in cloud is one of the hottest researches in VM placement. Simarro et al. [1] proposed a cost-efficient scheduling model for optimizing VMs placement across available cloud offers. They used some variables such as average prices or cloud prices trends for an optimal deployment. Chen et al. [2] proposed an optimal VM placement model to capture the intrinsic trade-off between the electricity cost and wide-area-network cost. They then introduced a cost-aware two-phase metaheuristic algorithm, Cut-and-Search, that approximated the best trade-off between the two costs. Le et al. [3] studied the impact of load placement policies on cooling and maximum data center temperatures in cloud service providers, and they proposed dynamic load distribution policies that considered all electricity-related costs as well as transient cooling effects through VMP in high performance computing clouds.

Besides cost, many approaches aimed to reduce energy consumptions in cloud [4]–[10]. Dong et al. [4] proposed a VM placement scheme meeting multiple resource constraints, such as the physical server size (CPU, memory, storage, bandwidth, etc.) and network link capacity to improve resource utilization and reduce both the number of active physical servers and network elements so as to finally reduce energy consumption. Wang et al. [5] proposed a heuristic greedy algorithm, called as CMBFD, to implement VM deployment and live migration to maximize total resource utilization and minimize energy consumption, which is based on energy-aware and quadratic exponential smoothing method to predict the workloads. Some other works save energy by reducing the quantity of PMs in cloud [6]–[8]. Huang et al. [6] explored the balance between server energy consumption and network energy consumption to present an energy-aware joint virtual machine placement, which can efficiently reduce the number of used physical machines. Li et al. [7] proposed a VMP algorithm, EAGLE, which can balance the utilization of multi-dimensional resources, reduce the number of running PMs, and thus lower the energy consumption. Zhang et al. [8] formulated VMP problem as a multi-objective nonlinear programming, and proposed an offline and an online algorithms to minimize the PMs resource used in cloud.

Traffic [11]–[13] and network performance [14], [15] are also two main problems in consolidation of VMs on physical servers in cloud. Meng et al. [11] proposed a traffic-aware VMP algorithm, which reduced traffic among VMs by optimizing the placement of VMs on host machines, e.g. VMs with large mutual bandwidth usage were assigned to the same PM or to PMs in close proximity. Vu and Hwang [12] presented an algorithm that improves communication performance by reducing overall traffic cost of VMs. Li et al. [13] focused on the optimized placement of VMs to minimize the network traffics cost (N-cost) and PMs cost (PM-cost). They also presented an effective binary-search-based algorithm to make a trade-off between N-cost and PM-cost. Dong et al. [14] proposed a multi-resource constraints VMP scheme to improve network performance in cloud. They minimized network maximum link utilization to balance network traffic distribution and reduce network congestion. Wen et al. [15] presented VirtualKnotter, including an online VM placement algorithm and an efficient VM migration scheduling algorithm, to reduce network congestion.

There are also some works studied how to optimize multi-objectives in cloud [16]–[18]. Zheng et al. [16], [17] tried to minimize power consumption, resource waste, server unevenness, inter-VM traffic, storage traffic and migration time at the same time. Liu et al. [18] proposed a multi-objective VMP model to minimize the number of active PMs, minimize communication traffic, and balance multi-dimensional resource in data center simultaneously.

In brief, all these studies focused on VMP based on optimizing the cloud resources utilization. However, they were not based on performance optimization, and they did not take the VM performance into account, while the VM performance is the really one of the main issues what users will pay close attention to.

TABLE I. HARDWARE OF FIVE SERVERS

Quantity	Hardware	
1	CPU: Xeon E5620 2.0G 4*1 Memory: 8G Hard Disk: 250GB Network Cards: 100Mbps*4	Controller Node
4	CPU: Xeon E5620 2.4G 4*2 Memory: 24G Hard Disk: 2TB Network Cards: 100Mbps*4	Compute Node

III. VM PERFORMANCE MODELS

In the Section I, we showed that the VM performance can be significantly affected with other VMs running on the same PM because of the resource contention among VMs. It is important to accurately model the VM performance to characterize the resource contention in PM. There have been some studies about performance modeling in virtualized environment [19]–[23]. For example, Watson et al. [19] examined the probabilistic relationships between virtualized CPU allocation, CPU contention, and application response time in virtualized environments. Ramakrishnan et al. [20] examined the performance of various interconnect technologies and characterized the virtualization overhead and its impact on performance. In this paper, we share the same idea on exploiting examining evaluation, and study the nature of virtualization resource scheduling on cloud platform to model the VM performance. The performance modeling has two main procedures as follows.

A. Constructing the performance models

In this paper, we construct the VM performance models based on a real cloud platform built by OpenStack, which is an open source software that can be used to deploy and manage cloud infrastructure in both large-scale and small-scale environment. OpenStack was initiated by Rackspace and NASA under the Apache 2.0 License in July 2010, and it is completely written in Python. Our OpenStack cloud testbed is deployed on five high performance servers with Icehouse software version and KVM hypervisor. The information of the five servers is shown in Table I. One computer is used as "Controller Node", and other four computers are treated as "Compute Node" to provide computing service and VMs. The controller node consists of mainly six different components: Nova (compute service), Neutron (network service), Horizon (dashboard service), Glance (image management service), Keystone (identity service), and Cinder (block storage). The controller node has four network interfaces. Two interfaces are used for communicating with two public ISPs. The third one is used for internal communication of different components of cloud, and the last one is used for internal communication among the VMs. The compute node has Nova-Compute service, which manages the lifecycles of VMs. There are two network interfaces too, which are the same as controller node except the public ISPs communication.

As we mainly focus on CPU, memory, and networks, we study the virtualization resource (CPU, memory, network) scheduling principle in OpenStack cloud platform, and construct our performance models for CPU, memory, and network. Table II summarizes some important symbols in the generalized system model. Given a PM, let M_j denotes its workload

TABLE II. SYMBOLS USED TO DESCRIBE THE MODELS

Symbols	Meaning
V_i	The virtual machine i, $1 \le i \le N$
vP^i	The VM performance vector of V_i
M_j	The physical machine j, $1 \le j \le M$
M_c^j	The total CPU resource of M_j
M_m^j	The total memory resource of M_j
M_n^j	The total network resource of M_j
p_c^j	The current VM CPU relative performance in M_j
p_m^j	The current VM memory relative performance in M_j
p_n^j	The current VM network relative performance in M_j
mP^j	The VM performance vector in M_j
$p_c^i(v)$	The CPU relative performance of V_i in M_j
$p_m^i(n)$	The memory relative performance of V_i in M_j
$p_n^i(b)$	The network relative performance of V_i in M_j

capacity vector, $M_j = (M_c^j, M_m^j, M_n^j, p_c^j, p_m^j, p_n^j)$. Here, M_c^j, M_m^j, and M_n^j are total physical CPUs, memory, and network bandwidth of M_j. p_c^j, p_m^j, and p_n^j are current VM relative performance (RP) in M_j. If VM V_i in running on PM M_j, then $p^i = p^j$. The VM relative performance can be calculated by $p^i = P_{vm}^i / P_d^i$. Here, P_{vm}^i is the current VM performance of V_i in M_j, and P_d^i is the desired performance if there is only one VM in M_j. In other words, P_d^i is the VM performance without hardware contention. The performance can be measured as the runtime (e.g., video transcoding time), throughput (e.g., video streaming), latency (e.g., retrieval time in database), etc.

When the VM V_i running on M_j, its CPU relative performance can be denoted as:

$$p_c^i \propto \begin{cases} 1 & if(\sum_k V_c^k \le M_c^j - M_{c,r}^j) \\ \frac{M_c^j - M_{c,r}^j}{\alpha \sum_i V_c^i} & else \end{cases}$$

where V_c^k is the number of vCPU kernel of VM V_k in PM M_j, $M_{c,r}^j$ is the reserved CPUs for running itself, $\alpha \ge 1$ is the CPU performance degradation parameter.

The memory relative performance can be denoted as:

$$p_m^i \propto \frac{M_m^j - M_{m,r}^j}{\beta \sum_k V_m^k}$$

where, $M_{m,r}^j$ is the reserved memory for running itself, $\beta \ge 1$ is the memory performance degradation parameter.

The network relative performance is:

$$p_n^i \propto \frac{M_n^j}{\gamma \sum_k V_n^k}$$

where, $\sum_k V_n^k < M_n^j$, $\gamma \ge 1$ is the network performance degradation parameter.

B. Generating the performance parameters

To test and verify our performance models, we choose and run some typical benchmarks to test different physical resources (CPU, memory, network) in our real OpenStack cloud platform, so as to collect test results and then train the VM performance models.

CPU: Super PI [24] is a computer program that calculates π to a specified number of digits after the decimal point. It is a popular benchmark to test the CPU performance. However, Super PI is single threaded, so its relevance as a performance

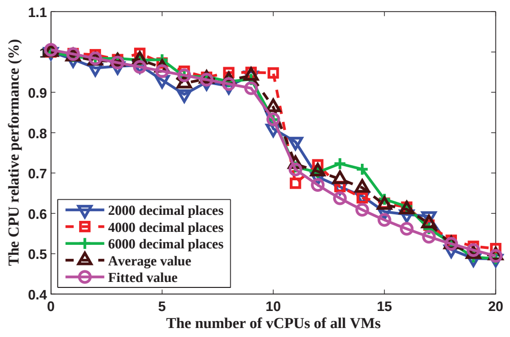

Fig. 1. The test results of CPU performance.

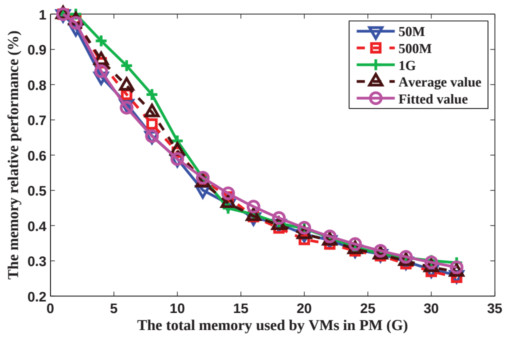

Fig. 2. The test results of memory performance.

measure of the multi-core machines is diminishing quickly. To test stability on multi-core CPU, Hyper PI [25] is developed to support multiple threads of Super PI to be run at the same time. The test steps are as follows.

Step1: Choose a compute node (e.g., Compute Node 3) in cloud, and place one VM (e.g., 1 vCPU, 2G memory). Run Hyper PI benchmark on the VM, and get the runtime as the desired performance (P_d^i);

Step2: Continue to place 2, 3, ..., n VMs, then run Hyper PI benchmark on the VMs several times, and calculate the average runtime P_{vm}^i.

Step3: Calculate the VM CPU relative performance by $p^i = P_{vm}^i / P_d^i$.

In this experiment, we get the relative performance of calculating π to 2000, 4000, 6000 digits after the decimal point respectively, and then we calculate their average value. The results are shown in Fig. 1. From the results, with the increasing of number of vCPUs, the VM CPU relative performance decreases slowly at first. And then, the VM CPU relative performance decreased rapidly after a certain value of the threshold. In the figure, the dividing point is at 8, which is the number of CPU cores of PM. When the number of vCPUs of all VMs is over 8, there is a great resource contention among these VMs, which results in a great performance degradation. For example, when the number of vCPUs of all VMs is 20, the VM CPU relative performance is just around 0.5.

We also fit our CPU performance model using the results to get the parameters, and then we get the theoretical value (line "Fitted value" shown in Fig. 1 too) using the fitted model. We can see the fitting result is proper, and the fitting error is about 0.27%, which shows that the CPU model can describe the CPU performance degradation trend very well. The fitted model is:

$$p_c^i(v) = \begin{cases} -0.0155v + 1.03 & if (v = \sum_k V_c^k \le M_c^j - M_{cr}^j) \\ 5.96/v + 0.1435 & else \end{cases}$$

Memory: We use memtester [26], [27] to test memory performance. When there are multiple VMs running on the PM, they need to read and write the same memory at the same time frequently. As the memory is shared and competed, this leads to performance degradation in reading and writing speed. We run memtester benchmark and get the runtime to test the memory performance. The test steps are similar to those of testing CPU, and they are as follows.

Step1: Choose a compute node (e.g., Compute Node 3) in cloud, and place one VM (e.g., 1 vCPU, 2G memory). Run memtester benchmark on the VM, and get the runtime as the desired performance (P_d^i);

Step2: Continue to place 2, 3, ..., n VMs, then run memtester benchmark on the VMs several times, and calculate the average runtime P_{vm}^i.

Step3: Calculate the VM memory relative performance by $p^i = P_{vm}^i / P_d^i$.

We test memtester to read and write different memory sizes (e.g., 50M, 500M, 1G) under different memory utilizations of PM, and the results are shown in Fig. 2. We can see that the memory performance decreases with the increasing of memory used in PM, which shows that the memory consumption (the sum of all VMs memory) greatly affects the VM memory performance. Meantime, we fit the memory performance model using the results with a fitting error 0.277%. The fitting model is:

$$p_m^i(n) = \frac{M_m^j - M_{m,r}^j}{9.718 + n} \quad (n = \sum_k V_m^k)$$

Network: We use Netperf [28], [29] to test the maximum bandwidth of virtual network interface of a single VM. Netperf is a software application that provides network bandwidth testing between two hosts on a network. It contains two components, netServer for receiving data and netperf for sending data. The test steps are as follows.

Step1: Choose a compute node (e.g., Compute Node 3) in cloud, and place one VM (e.g., 1 vCPU, 2G memory). Deploy netServer. Run netperf deployed on a computer to send data to netServer, and get the maximum throughput as the desired performance (P_d^i);

Step2: Continue to place 2, 4, ..., 2*n VMs, and deploy netServer on them. Then send data to these VMs at the same time, and get the maximum throughput P_{vm}^i.

Step3: Calculate the VM network relative performance by $p^i = P_{vm}^i / P_d^i$.

The results are shown in Fig. 3. We can see that the network performance decreases with the increasing of VMs. We fit the

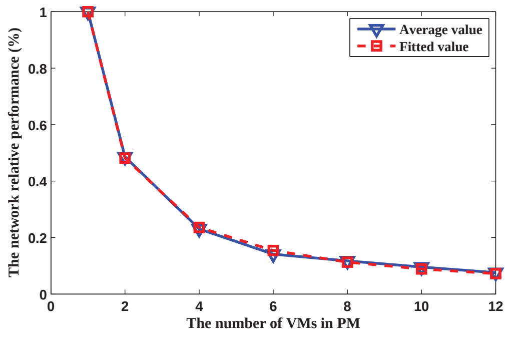

Fig. 3. The test results of network performance.

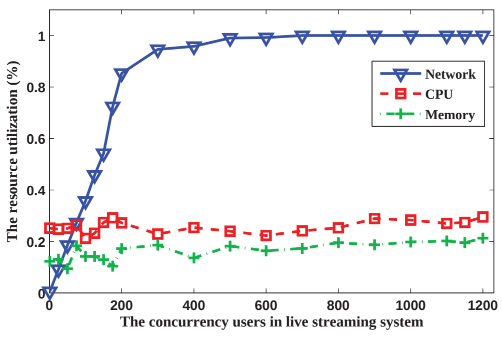

Fig. 4. The resource consumption of live streaming.

network performance model using the results with a fitting error 0.029%, and then we get the following formula.

$$p_n^i(b) = 0.9816/b - 0.0094 \quad (b = \sum_k V_n^k)$$

C. *The sum*

The above describes our performance models for CPU, memory, and network of VMs. They can describe the performance degradation caused by resource contention among VMs placed on the same PM. The proposed modeling method can be generalizable to a generic application, because the only knowledges needed for the models are some performance parameters for cloud platform, all of which can be retrieved from previously "learning". Note that, the performance parameters are some dependence on the hardware. As long as the hardware of cloud platforms is kept the same, those parameters can be applied directly. For other cloud platforms, those parameters should be trained again.

IV. VM Placement Method

In this section, we present our VMP method based on VM performance models. We consider the resource consumption characteristics of applications, the resource requirement of VMs, and the server capacity of PMs in the process. The application-aware resource consumption characteristic is introduced firstly. The VMP algorithm will be presented in detail in the following subsection.

A. *Application-aware resource consumption characteristic*

The applications running on VMs are different, so the VM resource requirements are different. For example, for video live streaming applications, they need to support large-scale users watching videos simultaneously, so they are bandwidth-intensive and require lots of bandwidth to transmit data to users. Another example is video process, such as video transcoding. It is a computation intensive and time-consuming task, which requires lots of computation (CPU) resources. So we need to study the resource consumption characteristics of different applications.

Define $W_i = (w_c^i, w_m^i, w_n^i)$ as the resource consumption characteristic vector of application running in VM V_i. Here, w_c^i, w_m^i, and w_n^i are the CPU, memory, and network utilizations, and they can be obtained by monitoring the VM performance. For example, Fig. 4 shows the resource (CPU, memory, bandwidth) utilizations of live streaming service, which proves that it is a bandwidth-intensive application. With the increase of users, the bandwidth has been drained, while the memory and CPU usages are around 20% and 30%, respectively. So we can use $W_i = (0.3, 0.2, 1)$ for live streaming applications. Of course, we can also get the resource consumption characteristics of other applications using the similar way.

B. *Problem formulation*

Generally, we assume that there are N VMs and M PMs ($N \geq M$). No VM requests more resource than a single PM can offer. For each VM $V_i (i = 1, 2, ..., N)$, its resource requirement vector is denoted as $Y_i = (v_c^i, v_m^i, v_n^i)$. Here, v_c^i, v_m^i, and v_n^i are vCPU, memory, and bandwidth requirements. Meantime, the resource consumption vector of application running on V_i is $W_i = (w_c^i, w_m^i, w_n^i)$. According to the resource requirement vector Y_i and the resource consumption vector W_i of V_i, we can calculate its real resources requirement vector $R_i = Y_i \cdot W_i = (v_c^i \cdot w_c^i, v_m^i \cdot w_m^i, v_n^i \cdot w_n^i)$. For each PM $M_j (j = 1, 2, ..., M)$, its workload capacity vector $M_j = (M_c^j, M_m^j, M_n^j, p_c^j, p_m^j, p_n^j)$, $mP_{curr}^j = (p_c^j, p_m^j, p_n^j)$ is its current VM relative performance vector before placement, and its current resource consumed by all VMs $C_{before}^j = (v, n, b)$ can calculated by $v = \sum_k V_c^k$, $n = \sum_k V_m^k$, and $b = \sum_k V_n^k$. Let matrix $X_{NM} = \{x_{ij} | x_{ij} = 0, 1\}$ indicates the VMP solution, where $x_{ij} = 1$ means VM V_i is placed on M_j, and $x_{ij} = 0$ implies V_i is not placed on M_j.

After VMs placement, all VMs resource consumed of M_j will be $C_{after}^j = (v', n', b')$, here $v' = v + \sum_{x_{ij}=1} v_c^i \cdot w_c^i$, $n' = n + \sum_{x_{ij}=1} v_m^i \cdot w_m^i$, and $b' = b + \sum_{x_{ij}=1} v_n^i \cdot w_n^i$. Then we calculate the new relative performance vector mP_{after}^j with C_{after}^j using our VM performance models in last section. Let vP_{after}^i denotes the VM performance of V_i after placement. If $x_{ij} = 1$, then $vP_{after}^i = mP_{after}^j$. On the one hand, to offer the best VMs, we should maximize the total performance of all VMs, e.g., $max(\sum_{i=1}^N ||vP_{after}^i||)$. On the other hand, to maximize the cloud resource utilization, we should minimize the performance degradation of all PMs, e.g.,

$min(\sum_{j=1}^{M} ||mP^j_{curr} - mP^j_{after}||)$.

Therefore, we can define our optimization problem as follows (denoted as Π):

$$\begin{aligned} \Pi : & max(\sum_{i=1}^{N} ||vP^i_{after}||) \\ and\ & min(\sum_{j=1}^{M} ||mP^j_{curr} - mP^j_{after}||) \\ s.t.\ & x_{ij} = \{0,1\}, 0 \le vP^i, mP^j \le 1 \\ & \sum_{j=1}^{M} x_{ij} = 1, 0 \le \sum_{i=1}^{N} x_{ij} \le N \end{aligned}$$

C. Problem solving

The proposed VM placement attempts to obtain two objectives. Many methods convert the original problem with multiple objectives into a single-objective optimization problem, which is called a scalarized problem. In this paper, we use a linear scalarization to formulate it as a single-objective optimization problem. Let $R = max(\sum_{i=1}^{N} ||vP^i_{after}||)$ and $Q = min(\sum_{j=1}^{M} ||mP^j_{curr} - mP^j_{after}||)$, then we convert problem Π as $max(\omega R - (1-\omega)Q)$, here ω is the scalarization parameter. If $\omega = 1$, then the VM performance is the main objective. $\omega = 0$ means the cloud resource utilization is the main objective. When $0 < \omega < 1$, there is a trade-off between the two objectives.

To obtain the optimal solution, we have to test every possible solutions. If there are N VMs and M PMs, then getting the optimal result demands a time-complexity of $O(M^N)$. We can see that it is impractical in real-life situations. In this paper, we use a greedy algorithm to get an approximate optimal solution with $O(M \cdot N)$. The problem can be divided into N sub-problems. For each VM placement, we choose a PM M_j, which can get the maximum value, that is, $max(\omega||vP^i_{after}|| - (1-\omega)||mP^j_{curr} - mP^j_{after}||)$, to place the VM. At last, we can get the global approximate optimal solution for all VMs. Algorithm 1 presents the algorithm in the pseudo code form.

Algorithm 1 Our VMP algorithm

```
Input: VMs $V = \{V_i\}$, PMs $M = \{M_j\}$, $\omega$
Output: The VMP solution $X_{NM}$
 1: Set $X_{NM}$ with zeros;
 2: for each VM $V_i$ in $V$ do
 3:     Get its resource requirement $Y_i = (v^i_c, v^i_m, v^i_n)$;
 4:     Get its resource consumption characteristic $W_i = (w^i_c, w^i_m, w^i_n)$;
 5:     Calculate the real resources requirement vector $R_i = Y_i \cdot W_i = (v^i_c \cdot w^i_c, v^i_m \cdot w^i_m, v^i_n \cdot w^i_n)$;
 6:     $max = 0$; $pm = -1$;
 7:     for each PM $M_j$ in $M$ do
 8:         Get relative performance vector $mP^j_{curr}$;
 9:         Get resource consumed by all VMs $C^j_{curr} = (v, n, b)$;
10:         Calculate the new resource consumed $C^j_{after} = C^j_{curr} + R_i = (v', n', b')$; // if place $V_i$ on $M_j$
11:         Calculate the new relative performance vector $mP^j_{after}$ with $C^j_{after}$ using our performance models;
12:         Calculate $\Delta = (\omega||vP^i_{after}|| - (1-\omega)||mP^j_{curr} - mP^j_{after}||)$;
13:         if $\Delta > max$ then
14:             $max = \Delta$;
15:             $pm = j$;
16:         end if
17:     end for
18:     if $pm \neq -1$ then
19:         $X_{i,pm} = 1$; // Place $V_i$ on $M_{pm}$;
20:     end if
21: end for
22: return $X_{NM}$
```

V. Experiments

A. Experimental setting

We use our OpenStack cloud platform as the testbed, and we set the parameter ω to 1, 0.5, and 0 respectively. To evaluate our VMP method based on performance models, the existing scheduling methods in scheduling module of Nova (Nova-VMPs), such as ChanceScheduler, SimpleScheduler, ZoneScheduler, and AbsractScheduler, are considered in our experiments, respectively. The main task of scheduling module of Nova is to choose a PM for placing a VM instance. ChanceScheduler (Random) is a random scheduling algorithm, and SimpleScheduler (Simple) realizes the minimum load scheduling algorithm. ZoneScheduler (Zone) is used to select a random PM in an available zone. Different from the ChanceScheduler, ZoneScheduler first selects an available zone, then gets the PMs in the available zone, then randomly selects a PM. AbsractScheduler (Weighted) gives different weights for different PMs, and chooses a PM with suitable weight for placing VMs. Besides the VMP methods in Nova of OpenStack, we also compare our VMP method with CMBFD [5]. There may be a little unfair if competing our method with CMBFD, because the latter tried to maximize total resource utilization and minimize energy consumption, instead of improving VM performance.

We take two applications, live streaming and video transcoding, as our test applications. We randomly generate two VMs request sequences (TestCase1 and TestCase2), both of which have 32 random VMs requirements with different configurations and applications respectively. Once a VM request is coming, we place it using our VMP and other VMPs respectively. If the new VM is for live streaming, we add it into live streaming server cluster, and count the maximum concurrency for live streaming of the cluster. If the new VM is for video transcoding, we run the same transcoding task on all transcoding VMs simultaneously, and then calculate the average transcoding time of all transcoding VMs. We repeat the above processes until all 32 VMs are place on PMs. We use the maximum concurrency and the average transcoding time as the VM performance metrics for live streaming and transcoding services.

B. Experimental results

The detailed results are shown in Fig. 5. From the Fig. 5(a) and 5(b), we can see that our VMP methods ($\omega = 1, 0.5, 0$)

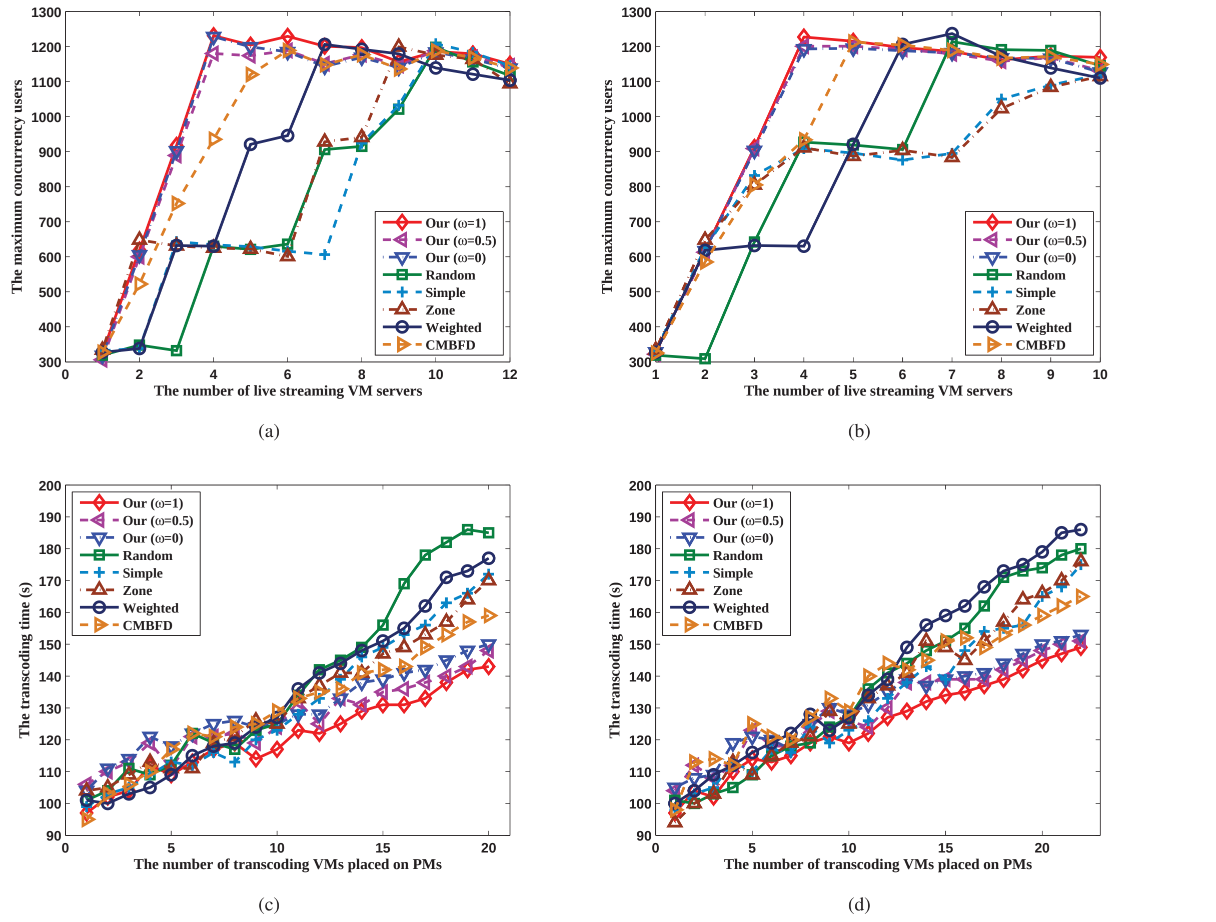

Fig. 5. The results comparison with various methods. (a) and (b) are the maximum concurrency results for live streaming in TestCase1 and TestCase2; (c) and (d) are the transcoding results for video transcoding in TestCase1 and TestCase2.

perform better than others in live streaming tests. For example, our VMP methods only needs 4 VMs that can achieve the maximum concurrency (1200) for live streaming, while CMBFD needs 5 VMs and Nova-VMPs need more than 6 VMs to reach the same result. Although CMBFD is better than Nova-VMPs, it is a little worse than our VMPs. CMBFD aims to optimize energy consumption in cloud, so it can't guarantee VM performance. Our VMP methods take the VM performance models into account, which can reflect the VM performance loss caused by resource sharing and contention on a PM, so as to provide VMs with best performance to users. Of course, all methods stop increasing when the maximum concurrency for live streaming is over 1200, which is limited by the resource constraint of our cloud platform, so the maximum number of users supported by the streaming media servers will not increase with the increase of VMs.

Meanwhile, from the video transcoding test results shown in Fig. 5(c) and 5(d), we can see that the performance difference among all VMP methods is not clear with the small scale of VMs scheduling. However, with the increase of the number of transcoding servers, especially when there are more than 10 transcoding servers, our VMP methods work much better than others. At first, when there are several VMs on the cloud platform, there is no resource competition problem among VMs, so the transcoding times of all VMP methods are almost the same. When the number of VMs increases, the more VMs there are, the more resource competition there will be, so the VM performance degrades clearly. However, our VMP methods can place VMs with the best performance, so they outperform the others in average transcoding time.

VI. Conclusion

This paper addresses the issue of VM performance degradation when placing VMs on PMs. After analyzing the shortcomings of existing VMP method, our application-aware VMP method based on the VM performance models is proposed. It introduces VM performance models, which are the fundamental bases for the following placement. Then it takes the application-aware resource consumption characteristic into consideration to place VMs on appropriate PMs, which can guarantee the VM performances and ensure customers' QoE. Lastly, we take the streaming media services for examples and conduct some experiments to evaluate our VMP method in our real OpenStack cloud platform. The results show that it works better than other methods.

As for future work, we will study performance models of other resources, such as disk I/O, which also affects the VM performance greatly. Besides, our method is a static placement. Once the VMs are placed in a PM, it can't place the VMs again in another PM. Next, we intend to study how to dynamically place VMs.

Acknowledgment

This research was partially supported by the National Science Foundation of China under Grant Nos. 61221063, 61472317, 61428206, 61532015, 91118005, 91218301, the MOE Innovation Research Team No.IRT13035, the National Key Technologies R&D Program of China under Grant No.2013BAK09B01, the Co-ordinator Innovation Project for the Key Lab of Shaanxi Province under Grant No.2013SZS05-Z01, Changjiang Scholars Program of MOE, the Fundamental Research Funds for the Central Universities, and the China Scholarship Council.

References

[1] J.L.L. Simarro, R. Moreno-Vozmediano, R.S. Montero, and I.M. Llorente, "Dynamic placement of virtual machines for cost optimization in multi-cloud environments," in *Proceedings of 2011 International Conference on High Performance Computing and Simulation (HPCS 2011)*, pp.1-7, 2011.

[2] K.Y. Chen, Y. Xu, K. Xi, and H.J. Chao, "Intelligent virtual machine placement for cost efficiency in geo-distributed cloud systems," *IEEE International Conference on Communications*, pp.3498-3503, 2013.

[3] K. Le, J. Zhang, J. Meng, R. Bianchini, Y. Jaluria, and T. D. Nguyen, "Reducing electricity cost through virtual machine placement in high performance computing clouds," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2011)*, 2011.

[4] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-Saving virtual machine placement in cloud data centers," in *Proceedings of 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2013)*, pp.618-624, 2013.

[5] J. Wang, C. Huang, K. He, X. Wang, X. Chen, and K. Qin, "An energy-aware resource allocation heuristics for VM scheduling in cloud," in *Proceedings of 2013 IEEE International Conference on High Performance Computing and Communications (HPCC 2013) and 2013 IEEE International Conference on Embedded and Ubiquitous Computing (EUC 2013)*, pp.587-594, 2013.

[6] D. Huang, D. Yang, H. Zhang, and L. Wu, "Energy-aware virtual machine placement in data centers," in *Proceedings of 2012 IEEE Global Telecommunications Conference (GLOBECOM 2012)*, pp.3243-3249, 2012.

[7] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Mathematical and Computer Modelling*, vol. 58, no.5-6, pp.1222-1235, 2013.

[8] J. Zhang, Z. He, H. Huang, X. Wang, C. Gu, and L. Zhang, "SLA aware cost efficient virtual machines placement in cloud computing," in *Proceedings of 33rd IEEE International Performance Computing and Communications Conference (IPCCC 2014)*, 2014.

[9] H. Goudarzi and M. Pedram, "Energy-efficient virtual machine replication and placement in a cloud computing system," in *Proceedings of 2012 IEEE 5th International Conference on Cloud Computing (CLOUD 2012)*, pp.750-757, 2012.

[10] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic consolidation of virtual machines in self-organizing cloud data centers," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp.215-228, 2013.

[11] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of 2010 IEEE INFOCOM*, 2010.

[12] H.T. Vu and S. Hwang, "A traffic and power-aware algorithm for virtual machine placement in cloud data center," *International Journal of Grid and Distributed Computing*, vol. 7, no. 1, pp.21-31, 2014.

[13] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: Towards traffic aware virtual machine placement in data centers," in *Proceedings of 2014 IEEE INFOCOM*, pp.1842-1850, 2014.

[14] J.K. Dong, H.B. Wang, Y.Y. Li, and S.D. Cheng, "Virtual machine placement optimizing to improve network performance in cloud data centers," *Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 3, pp.62-70, 2014.

[15] X. Wen, K. Chen, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "Virtual knotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter," in *Proceedings of 2012 International Conference on Distributed Computing Systems*, pp.12-21, 2012.

[16] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, et al., "Virtual machine consolidated placement based on multi-objective biogeography-based optimization," *Future Generation Computer Systems*, Available online 30 March 2015, 2015.

[17] Q. Zheng, R. Li, X. Li, and J. Wu, "A multi-objective biogeography-based optimization for virtual machine placement," in *Proceedings of 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2015)*, pp.687-696, 2015.

[18] C. Liu, C. Shen, S. Li, and S. Wang, "A new evolutionary multi-objective algorithm to virtual machine placement in virtualized data center," in *Proceedings of 2014 5th IEEE International Conference on Software Engineering and Service Science (ICSESS 2014)*, pp.272-275, 2014.

[19] B. J. Watson, M. Marwah, D. Gmach, Y. Chen, M. Arlitt, and Z. Wang, "Probabilistic performance modeling of virtualized resource allocation," in *Proceeding of the 7th International Conference on Autonomic Computing (ICAC '10) and Co-located Workshops*, pp.99-108, 2010.

[20] L. Ramakrishnan, R.S. Canon, K. Muriki, I. Sakrejda, and N.J. Wright, "Evaluating interconnect and virtualization performance for high performance computing," in *Proceedings of the 2nd International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computing Systems (PMBS'11), Co-located with SC'11*, pp.1-2, 2011.

[21] R.C. Chiang and H.H. Huang, "TRACON: Interference-aware schedulingfor data-intensive applicationsin virtualized environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, pp.1349-1358, 2014.

[22] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp.931-945, 2011.

[23] S. Kundu, R. Rangaswami, K. Dutta, and Z. Ming, "Application performance modeling in a virtualized environment," in *Proceedings of 2010 IEEE 16th International Symposium on High Performance Computer Architecture (HPCA 2010)*, pp.1-10, 2010.

[24] "Superpi." [Online]. Available: http://www.superpi.net

[25] "Hyperpi." [Online]. Available: http://virgilioborges.com.br/hyperpi/

[26] Q. Kang, C. Jin, Z. Zhang, and A. Zhou, "MemTest: A novel benchmark for in-memory database," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8807, pp.34-46, 2014.

[27] "Memtester." [Online]. Available: http://pyropus.ca/software/memtester/

[28] S.S. Kolahi, S. Narayan, D.D.T. Nguyen, and Y. Sunarto, "Performance monitoring of various network traffic generators," in *Proceedings of 2011 UKSim 13th International Conference on Modelling and Simulation (UKSim 2011)*, pp.501-506, 2011.

[29] "Netperf." [Online]. Available: http://www.netperf.org/netperf/