Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 906098, 12 pages
http://dx.doi.org/10.1155/2014/906098

Research Article

Workload-Aware and CPU Frequency Scaling for Optimal Energy Consumption in VM Allocation

Zhen Liu,[1,2] Yongchao Xiang,[1,2] and Xiaoya Qu[1,2]

[1] *School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*
[2] *Engineering Research Center of High-Speed Railway Network Management, Ministry of Education, Beijing 100044, China*

Correspondence should be addressed to Zhen Liu; liuzhenlzsy@gmail.com

Received 13 June 2014; Accepted 27 July 2014; Published 28 August 2014

Academic Editor: Guoqiang Zhang

In the problem of VMs consolidation for cloud energy saving, different workloads will ask for different resources. Thus, considering workload characteristic, the VM placement solution will be more reasonable. In the real world, different workload works in a varied CPU utilization during its work time according to its task characteristics. That means energy consumption related to both the CPU utilization and CPU frequency. Therefore, only using the model of CPU frequency to evaluate energy consumption is insufficient. This paper theoretically verified that there will be a CPU frequency best suit for a certain CPU utilization in order to obtain the minimum energy consumption. According to this deduction, we put forward a heuristic CPU frequency scaling algorithm VP-FS (virtual machine placement with frequency scaling). In order to carry the experiments, we realized three typical greedy algorithms for VMs placement and simulate three groups of VM tasks. Our efforts show that different workloads will affect VMs allocation results. Each group of workload has its most suitable algorithm when considering the minimum used physical machines. And because of the CPU frequency scaling, VP-FS has the best results on the total energy consumption compared with the other three algorithms under any of the three groups of workloads.

1. Introduction

On-demand and effective resources management is crucial for the availability of a scalable cloud datacenter [1, 2]. According to statistics, the resource utilization rate of datacenter is very low, which is only about 30% on average. However, a server is in idle status most of the time during the day. And even an idle server still consumes almost 60% of its full-load power. With its technical advantages and the hardware support, virtualization technology [3] comes out once again. Based on the support of the processor and server, virtual machine (VM) becomes the basic unit for resource management and sharing [4]. It has high utilization efficiency and good isolation characteristics with each other. Nowadays, people also use VMs to solve energy saving problem in datacenter which is so-called packing problem [5]. Packing problem is a global optimal problem. The objective is to pack the pieces into a number of bins, subject to the constraint, that the sum of the sizes of the pieces in each bin is less than or equal to the size of the bin. Similarly, VM packing means to place the VMs as much as possible to the physical machines (PMs), so that the unused PMs can be shut down to save energy. Therefore, VM packing problem is also called VM placement problem.

Different from dynamic VM placement, initial VM placement should have long term effects in datacenter. Because VMs migration will cost time and resources, frequent migration is impractical. Such placement plays an important role in efficient use of resources and energy saving in datacenters [6]. Initial VM placement is subject to the VM requirements for resources, the SLA requirements set by users, and the available resources on PMs. There are many works studying the optimal solution of initial VM placement problem, such as heuristic methods [7–9] or genetic algorithms [10, 11]. However, heuristic algorithm is basically the single point search, and thus it is easy to fall into the local optimal solution. Genetic algorithm does not use the feedback information of the system, which makes

the search blind. Reference [12] proposed an improved ant colony optimization algorithm to find the optimal solution. Reference [11] proposed a genetic algorithm based on NSGA-II to find the optimal solution. There are also many works focusing on the energy or power in a virtualized environment [13–18]. In [12], the author proposed a heuristic method to find the optimal energy consumption in a virtualized enterprise environment. Reference [7] studied the relation between CPU utilization and applications. In this way, the system can decide the placement solution for VMs in order to decrease the energy consumption. Reference [15] proposed algorithms based on first-fit descending to consolidate VMs in datacenter. Reference [16] not only considers the energy but also considers the factor of SLA for optimal service provision according to users resource requirement. References [17, 18] also focus on the objective of energy. They also regard CPU as the main resource for energy consumption and monitor it. The objective of VM migration is to find the optimal energy consumption.

These algorithms are all based on the fixed frequency of CPU. References [19, 20] put forward the model of CPU frequency and energy consumption. Reference [21] proposed a solution under the changeable CPU frequency. But it only adapts to unchanged workload. In [22], the authors addressed the idea that an energy conservation algorithm must consider the workload characteristics of virtual machines. The typical or random workload will actually affect the final energy consumption. Reference [23] describes the relationship between the power consumption and the operating frequency. This energy consumption model has been utilized by many works [24, 25]. However, its assumption is when the CPU operates on full load and the CPU utilization is 100%. Paper [22] has addressed the idea that the different workload will affect the CPU utilization because of its task kinds and algorithms. For example, some workload is CPU intensive while others are data intensive or I/O intensive. Therefore, it is not comprehensive to only consider the relation of power and frequency.

This paper considers the relationship among power consumption and CPU frequency and CPU utilization. Here we consider the condition of workload characteristics. That means the CPU utilization will dynamically change according to their task characteristics. Usually, traditional PM in datacenters runs at a fixed CPU frequency despite what CPU utilization is. Based on the energy consumption model, we deduce that there will be a suitable frequency for certain workload CPU utilization in order to obtain minimum energy consumption. Therefore, we put forward a CPU frequency scaling idea and design a heuristic ant colony algorithm VP-FS (virtual machine placement with frequency scaling) to find the global solution for VMs placement. For the experiments, we simulate three groups of workload with different characteristics of CPU requirements as the evaluation data. CPU utilization for these three group presents linear, even, and bipolar distribution, respectively. For comparison, we also design and realize three algorithms with different greedy policies for initial VM placement. Different weights are considered for CPU, memory, and bandwidth resource requirements separately. The efforts show that different workload has its suitable placement algorithm. Because CPU frequency can dynamically be scaled to fit for certain CPU utilization, VP-FS can lead to minimum energy consumption among these algorithms. The main contributions of this paper are in the following aspects.

(1) Different from the work in [23], we consider the CPU utilization which is also an important factor to the energy consumption. And we verify that there should be a frequency that is best suited for CPU utilization with respect to the minimum energy consumption.

(2) Different from the work in [20], we consider the optimal energy consumption not the energy efficiency. We can have a more direct view of the relation among the PM numbers, CPU frequency, and the workload tasks capacity.

(3) Based on the above analysis, we put forward a CPU frequency scaling algorithm which can dynamically scale the frequency of the PM according to the CPU utilization in order to obtain the optimal energy consumption.

(4) In order to evaluate the effort of ours, we simulate three groups of workloads. Each group has a certain CPU utilization distribution. Then we do experiments using the proposed algorithms. From the results we can see that the solution of VM placements does have the relation with the different workloads. And the energy consumption can be lower by using frequency scaling.

The rest of this paper is organized as follows. The VM packing problem statement is presented and the three greedy algorithms are proposed in Section 2. In Section 3, we modeled the VM placement problem with respect to multi-objectives and frequency scaling. We also deduce the related factors with respect to minimum energy consumption. Based on the above analysis and the idea of CPU frequency scaling, we propose a heuristic ant colony algorithm VP-FS in Section 4. VP-FS is to find the optimal VM placement solution by searching the global solution space. In Section 5, we design the workload with different characteristics of CPU requirements. We then do the experiments and evaluations of the proposed algorithms. Conclusion is finally given in Section 6.

2. Problem Formulation and the VM Packing Algorithms

2.1. Problem Formulation. For future reference, we summarize the notation that is used throughout this paper in Notation section. In Notation section, each u_r denotes a resource utilization ratio (RUR), such as CPU, memory, or the bandwidth of a VM. With the notations presented in Notation section, we use $\sum_{r=1}^{d} w_{ri}u_{ri}$ to represent the comprehensive RUR of V_i. Thus, VM packing problem can be formulated as follows: given a certain $ru_j > 0$, $u_r > 0$, $w_r > 0$, $1 \le i \le n$, $1 \le j \le m$, $1 \le r \le d$, we should find a placement solution

Input: n VM, m PM
Output: $S = \{\langle V_i, P_j \rangle \mid M_{ij} = 1, 1 \le i \le n, 1 \le j \le m\}$
(1) **for** $i = 1$ to n do
(2) **for** $r = 1$ to d do
(3) $u[r] = \text{VM}[i][r]$;
(4) $\text{uVM}[i] = \sum_{r=1}^{d} w_{ri} u_{ri}$;
(5) sort(uVM, oVM, descending);
(6) **for** $i = 1$ to m do
(7) **for** $u = 1$ to d do
(8) $u[i] = \text{PM}[i][u]$;
(9) $\text{uPM}[i] = \sum_{r=1}^{d} w_{ri} u_{ri}$;
(10) sort(uPM, oPM, ascending);
(11) **for** $i = 1$ to n do
(12) **for** $j = 1$ to m do
(13) if $\text{PM}[j] > \text{VM}[\text{oVM}[i]]$
(14) $\text{PM}[j] - = \text{VM}[\text{oVM}[i]]$;
(15) $\text{M}[\text{oVM}[i], j] = 1$;
(16) adjust($\text{PM}[j]$), $\text{oPM}[j]$);
(17) break;
(18) **end**

Algorithm 1: VPBFD (VM, PM).

$S = \{\langle V_i, P_j \rangle \mid M_{ij} = 1, \forall i, j\}$, that is, $\sum_{r=1}^{d} w_{ri} u_{ri} < ru_j$ and $\min \sum_{i=1}^{n} \sum_{j=1}^{m} e_{ij}$ $(e_{ij} = 0 \mid \langle V_i, P_j \rangle \notin S)$.

2.2. RUR Based VM Packing Algorithms with Different Greedy Policies. We will present three algorithms with different greedy policies in this section.

(1) VPBFD Algorithm. The idea of VPBFD (virtual machine placement with best fit descending policy) algorithm is as follows. Compute the comprehensive RUR of each VM. And sort the VMs in descending order according to their RUR. Sort the PMs in ascending order according to their RUR. Search for each VM, finding the first PM that can satisfy the RUR of the VM. If a VM found its host PM, then recalculate the surplus RUR of PM and find its new place in the ascending PM queue. The pseudocode of VPBFD is presented in Algorithm 1. uVM[] is an array used for saving the comprehensive RUR of each VM. And the VM label i $(1 \le i \le n)$ is recorded in array oVM[].

VPBFD algorithm will satisfy VM with highest resource requirements and select the PM that just can satisfy the VM requirements, so it can leave much more room for other VMs.

(2) VPWFD Algorithm. The idea of VPWFD (virtual machine placement with worst fit descending policy) algorithm is as follows. Compute the comprehensive RUR of each VM. And sort the VMs in descending order according to their RUR. Sort the PMs in descending order according to their RUR. Search for each VM, finding the first PM that can satisfy the RUR of the VM. If a VM has found its host PM, then recalculate the remnant RUR of PM and find its new place in the descending PM queue. The only difference of VPWFD algorithm with VPBFD algorithm is to sort PM in descending order instead of ascending order. So the pseudocode of VPWFD will not be presented in this paper.

Different from VPBFD algorithm, VPWFD algorithm will first select the PM with highest resource surplus. Therefore, such PM can have room for other VMs.

(3) VPRandom Algorithm. The idea of VPRandom (virtual machine placement with random policy) algorithm is as follows. Compute the comprehensive RUR of each VM. It will not sort VM or PM. The algorithm will just begin to search for each VM, finding the first PM that can satisfy the RUR of the VM. The pseudocode of VPRandom algorithm is presented in Algorithm 2. As Algorithm 1, here the uVM[] is an array used for saving the comprehensive RUR of each VM. And the VM label i $(1 \le i \le n)$ is recorded in array oVM[].

Different from Algorithms 1, 2, and 3, VPRandom algorithm will not sort VM or PM. In totally random order, the algorithm will select the first satisfiable PM for each VM. In some cases, this method may obtain the ideal solution.

3. Multiobjective VM Placement with Frequency Scaling

3.1. VM Placement Problem with Variable Frequencies. In this paper, we will put n VMs to m PMs; the basic objective is to minimize energy consumption of all the PMs. Therefore, the solution space is m^n. If we consider h frequencies of each PM and PM can select a proper CPU frequency according to its workload, then the solution space will be $h^m m^n$. Generally speaking, different type of workload has different resource requirements. And only if all the resources can satisfy VM

Input: n VM, m PM
Output: $S = \{\langle V_i, P_j \rangle \mid M_{ij} = 1, 1 \le i \le n, 1 \le j \le m\}$
(1) **for** $i = 1$ to n do
(2) **for** $r = 1$ to d do
(3) $u[r] = \mathrm{VM}[i][r]$;
(4) $\mathrm{uVM}[i] = \sum_{r=1}^{d} w_{ri} u_{ri}$;
(5) **for** $i = 1$ to n do
(6) **for** $j = 1$ to m do
(7) **if** $\mathrm{PM}[j] > \mathrm{VM}[\mathrm{oVM}[i]]$
(8) $\mathrm{PM}[j] -= \mathrm{VM}[\mathrm{oVM}[i]]$;
(9) $\mathrm{M}[\mathrm{oVM}[i], j] = 1$;
(10) break;
(11) **end**

Algorithm 2: VPRandom (VM, PM).

requirements, the resources can be allocated to the VM. However, the unbalance of resources utilization will easily lead to a waste of resources. If some VMs can meet a resource balance in different resource requirement, these VMs should be placed in one PM to obtain better resource utilization. For example, one VM has a CPU-intensive workload and another VM has a data-intensive workload; then, these two VMs can be packing together in one PM to obtain better resource utilization. This may be reasonable for an optimal placement solution. Thus, besides the energy consumption objective, this paper also proposes another objective to measure the solution, that is, resource balancing degree.

3.2. The Objectives of the Problem

(1) Energy Consumption. Consider

$$e = c + k * f^3 * u_{\mathrm{cpu}}, \tag{1}$$

where e denotes the instantaneous power of a PM. It depends on the energy consumption of PM in its idle time and the instantaneous frequency and CPU utilization, u_{cpu} denotes the CPU utilization, and k is a coefficient, which indicates that dynamic energy consumption of CPU is proportional to the cubic of frequency and utilization [21]. In [23], the energy consumption is depicted as $e = c + k * f^3$. However, its assumption is that the workload works on 100% CPU utilization. Actually, a workload task can be depicted as the multiple of CPU frequency, CPU utilization, and a coefficient [20]. Therefore, in Formula (1), we put the u_{cpu} as a multiplier to the f.

(2) Resources Balancing Degree. Consider

$$b = \left(1 - \frac{u_{\mathrm{cpu}}}{u}\right)^2 + \left(1 - \frac{u_{\mathrm{mem}}}{u}\right)^2 + \left(1 - \frac{u_{\mathrm{bw}}}{u}\right)^2, \tag{2}$$

$$u = \frac{u_{\mathrm{cpu}} + u_{\mathrm{mem}} + u_{\mathrm{bw}}}{3}. \tag{3}$$

In this paper, we mainly consider three kinds of resources. They are CPU, memory, and bandwidth. In formula (2), u_{mem} and u_{bw} denote the utilization of memory and bandwidth on PMs, respectively. u is the average utilization of the three resources. We normalize u_{cpu}, u_{mem}, and u_{bw} in formula (2) so that b can depict the balancing of the three kinds of resources in a PM. If the value of b is small, then it means the three kinds of resources utilization in a PM are balancing.

(3) Objective Function. Based on formulas (1) and (2), the objective function of VMs initial placement can be depicted as follows.

Minimize

$$a_1 * \sum_{j=0}^{m-1} e_j + a_2 * \sum_{j=0}^{m-1} b_j \tag{4}$$

subject to

$$u_{\mathrm{cpu}} \le u_{\mathrm{sla}}, \quad u_{\mathrm{mem}} \le u_{\mathrm{sla}}, \quad u_{\mathrm{bw}} \le u_{\mathrm{sla}}, \tag{5}$$

where e_j indicates the energy consumption of P_j, b_j indicates the resources balancing degree of P_j, and a_1 and a_2 are weights which stand for the degree of importance on energy consumption and resources balancing. The objective function is to find the minimal value of energy consumption and best resource balancing degree. Objective SLA is implemented by constraining resources utilization in e_j and b_j; u_{sla} indicates the upper threshold of resources utilization on PMs.

3.3. Frequency Scaling for Lowest Energy Consumption.

Formula (1) shows the relation of instantaneous power of a PM with its instantaneous frequency f and CPU utilization u_{cpu}. Given a set of workloads, each of whom has its own resource requirements. And each workload may ask for different CPU resource according to its load capacity. Theoretically speaking, if the PM can work on its best CPU frequency according to the CPU utilization, then the energy efficiency will be the optimal. Therefore, we need to answer the following two questions. (1) Is there a CPU frequency that is best suitable

for a certain workload that can make the minimum energy consumption? (2) What can such frequency be depicted as?

For convenience, we use u to denote u_{cpu} in this section. If there are m PMs with each work on a frequency f_i, then, as [23], the total required computing power of all the PMs is $F = \sum_{i=1}^{m} f_i$. According to formula (1), the total energy consumption of all the m PMs is given by

$$E = mc + k\left[\sum_{i=1}^{m-1} f_i^3 u_i + \left(F - \sum_{i=1}^{m-1} f_i\right)^3 u_m\right]. \tag{6}$$

As formula (6) shows, the total energy consumption has two variables, which are CPU frequency and utilization. If all the CPU frequencies and utilization in each PM are the optimal ones, then the total energy consumption E can be the minimum one. Therefore, we will deduct the relations of E with respect to CPU frequency or CPU utilization separately in the following.

(1) Total Energy Consumption with respect to CPU Frequency. As the first question asked about if we want to obtain the lowest energy consumption, then there is optimal values for CPU frequency of each PM. Based on formula (6), we make a first partial derivative of the total energy consumption with respect to CPU frequency. Consider

$$\frac{\partial E}{\partial f_i} = 3kf_i^2 u_i + k\frac{\partial\left(F - \sum_{i=1}^{m-1} f_i\right)^3}{\partial f_i}. \tag{7}$$

In formula (7),

$$\begin{aligned}
&\frac{\partial\left(F - \sum_{i=1}^{m-1} f_i\right)^3}{\partial f_i} \\
&= \frac{\partial\left(F^3 - 3F^2\sum_{i=1}^{m-1} f_i + 3F\left(\sum_{i=1}^{m-1} f_i\right)^2 - \left(\sum_{i=1}^{m-1} f_i\right)^3\right)}{\partial f_i} \\
&= -3F^2 + 3F\frac{\partial\left(\sum_{j=1,j\neq i}^{m-1} f_j + f_i\right)^2}{\partial f_i} - \frac{\partial\left(\sum_{j=1,j\neq i}^{m-1} f_j + f_i\right)^3}{\partial f_i} \\
&= -3F^2 + 3F\left(2\sum_{j=1,j\neq i}^{m-1} f_j + 2f_i\right) \\
&\quad - \partial\left(\left(\sum_{j=1,j\neq i}^{m-1} f_j\right)^3 + 3\left(\sum_{j=1,j\neq i}^{m-1} f_j\right)^2 f_i \right. \\
&\qquad \left. + 3\left(\sum_{j=1,j\neq i}^{m-1} f_j\right) f_i^2 + f_i^3\right) \times (\partial f_i)^{-1} \\
&= -3F^2 + 6F\sum_{i=1}^{m-1} f_i - \left(3\left(\sum_{j=1,j\neq i}^{m-1} f_j\right)^2 \right. \\
&\qquad \left. + 6\left(\sum_{j=1,j\neq i}^{m-1} f_j\right) f_i + 3f_i^2\right) \\
&= -3F^2 + 6F\sum_{i=1}^{m-1} f_i - 3\left(\sum_{j=1,j\neq i}^{m-1} f_j + f_i\right)^2 \\
&= -3\left(F - \sum_{i=1}^{m-1} f_i\right)^2.
\end{aligned} \tag{8}$$

According to (8), we can rewrite (7) as

$$\frac{\partial E}{\partial f_{\text{i}}} = 3kf_i^2 u_i - 3k\left(F - \sum_{i=1}^{m-1} f_i\right)^2 u_m. \tag{9}$$

In order to get the minimum E, let formula (9) equate 0, and then we have

$$3kf_i^2 u_i = 3k\left(F - \sum_{i=1}^{m-1} f_i\right)^2 u_m, \tag{10}$$

$$f_i^2 u_i = f_m^2 u_m. \tag{11}$$

(2) Total Energy Consumption with respect to Amount of PM. We substitute each $f_i u_i$ in formula (6) with (11). Formula (6) can be rewritten as

$$\begin{aligned}
E &= mc + k\sum_{i=1}^{m} f_i^3 u_i = mc + kf_m^2 u_m \sum_{i=1}^{m} f_i \\
&= mc + kf_m^2 u_m F.
\end{aligned} \tag{12}$$

If PM i works at frequency f_i and the CPU utilization is u_i, we define T as

$$T = \sum_{i=1}^{m} f_i^2 u_i. \tag{13}$$

Therefore, according to (11), we have

$$f_i^2 u_i = f_m^2 u_m = \frac{T}{m}. \tag{14}$$

Then we can rewrite formula (12) as

$$E = mc + k\frac{T}{m}F. \tag{15}$$

We make the first derivative of the total energy consumption E according to (15) with respect to the number of PM. This can be expressed as

$$\frac{\mathrm{d}E}{\mathrm{d}m} = c - kF\frac{T}{m^2}. \tag{16}$$

Setting $\mathrm{d}E/\mathrm{d}m = 0$, then we have $c = kF(T/m^2)$.

Thus,

$$m = \sqrt{\frac{kFT}{c}}. \tag{17}$$

Equation (17) means that besides the constants c and k and the number of PMs m, the optimal energy consumption also relates to each frequency f_i in each PM.

4. Dynamic Programming Algorithm

To find the optimal solution for energy consumption, there should be a most suitable frequency to each PM according to the workload running on it. Based on this idea, this paper proposes an ant colony optimization algorithm VP-FS (virtual machine placement with frequency scaling). It can select the suitable frequency by scaling it according to the CPU utilization of the running workload. The basic idea is discussed as follows.

Initialize each V_i and each P_j as a two tuples $\langle V_i, P_j \rangle$, which means a placement between V_i and P_j. We call the two tuples as a path. Set for each path an initial pheromone $t_{ij} = 1$. The ant will choose a path randomly for V_i according to the value of t_{ij}. Of course, the path will be easily selected if it has a big t_{ij}. Once the ant has finished path choosing for each V_i, an initial placement solution is formed.

If there are n ants, then there will be n placement solution. We take all the solutions that meet objective SLA as the solution space for the second objective $\min \sum_{j=0}^{m-1} e_j$. In the solution space that meets the second objective, we select the first p% subset solution as the solution space for the third objective $\min \sum_{j=0}^{m-1} b_j$. Then we will find the best solution. For the paths in the final solution, their pheromone will increase multiplied by q ($q > 1$). Iterating the above processes till the value of the objective function as formula (4) does not change or changes in a small enough range. The pseudocode of VP-FS algorithm is presented in Algorithm 3.

In Algorithm 3, F denotes the value of the objective function of formula (4). Variant a represents the ant. It will loop from 1 to the A. Each ant will produce a placement solution. Variants i and j are used to find a proper P_j for a V_i. While choosing the proper P_j for V_i, it will depend on the probability $t_{ij} / \sum_{i=1,j=1}^{n,m} t_{ij}$. If any remnant resource is insufficient, that is, $ru_j > u_{\text{sla}}$, then undo the allocation, else record the allocation $\langle V_i, P_j \rangle$. *choose_energy(p, S)* means to select the p% solution from set S. And *choose_balance(S)* means to search the best solution with minimal resources balancing degree b from set S. The ending condition of the iteration is that the difference of two results is less than a small enough value ϵ.

5. Experiments

Our experiments run on a cluster containing 10 PMs with OpenStack as its cloud infrastructure platform. All PMs are connected by a gigabit Ethernet. CPU in each PM has four cores. CPU highest frequency of each core is 2.7 GHz. The memory and bandwidth of each PM are 2 GB and 100 Mbps. We also set 30 VMs and let each VM deploy on one of the cores of the PM. We use sysbench to simulate each workload.

5.1. CPU-Intensive Workloads. Different workloads have different resource requirements variation, which may ask for different initial placement algorithms for best performance. In this paper, we simulate three groups of workloads. In each group of workload, we mainly focus on the CPU resource. The distribution of CPU utilization in each group is shown in Figure 1.

In Figure 1(a), the CPU utilization requirements of the three VM groups present different distribution. The 1st group is a linear distribution, the 2nd group is an even distribution, and the 3rd group is a bipolar distribution. Figure 1(b) is the memory and bandwidth resources, which are almost the same in each group.

The comprehensive RUR of V_i is $\sum_{r=1}^{d} w_{ri} u_{ri}$. In our experiments, $d = 3$. That means we consider three types of resource, CPU, memory, and bandwidth. Because CPU is the most effective factor in energy and we propose a CPU frequency scaling approach in this paper, so we set w_1, w_2, and w_3 as 7, 3, and 2, respectively. In the realization of the proposed VP-FS algorithm, we set $p = 0.05, q = 1.2$.

5.2. Placement Solution of Each Algorithm. For VMs in each group, we use the four algorithms (VPBFD, VPRandom, VPWFD, and VP-FS) in this paper to allocate them. 30 VMs are allocated to 10 PMs. Figure 2 is the allocation results under each VM group.

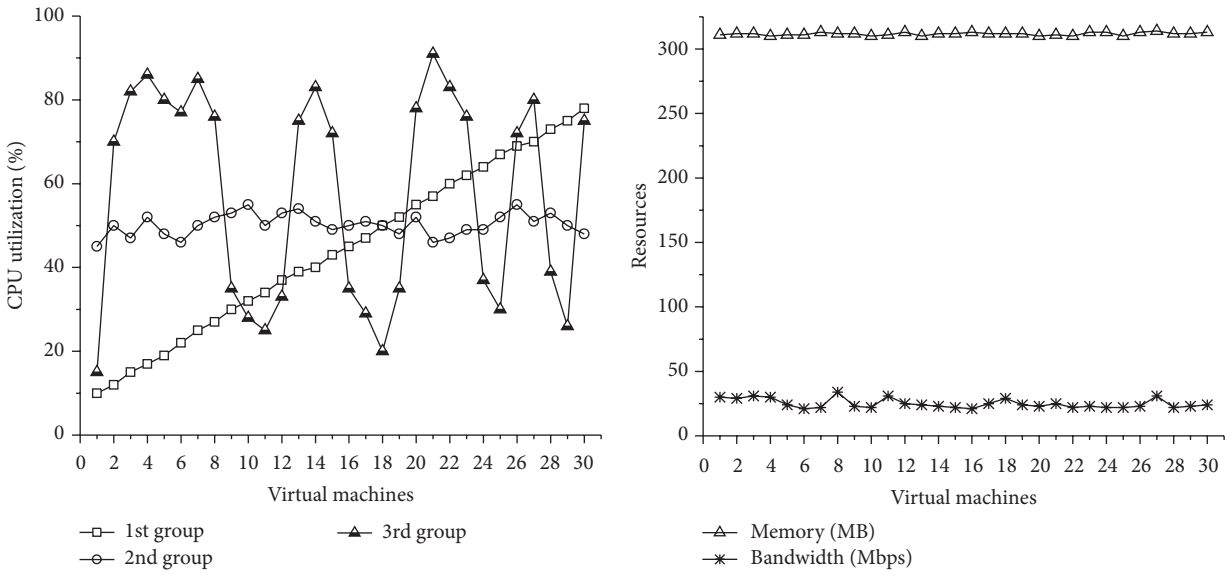
Because of different resource distribution in each group, the allocation results are different, as Figures 2(a), 2(b), and 2(c) show. x-Coordinate is the 10 PMs and y-coordinate is the 30 VMs. For example, in Figure 2(a), according to VPBFD, the allocation result is that VMs numbers 30, 29, 28, and 27 will be placed to PM number 1. According to VP-FS, VMs numbers 1, 19, and 3 will be placed to PM number 1. As the allocation results using different four algorithms show, *firstly*, the amount of used PMs is different. VPBFD and VPRandom use the least PMs, that is, 8 PMs instead of 10 PM for the 30 VMs, while VPWFD and VP-FS allocate 30 VMs in all the 10 PMs. *Secondly*, the distributed pattern is different. Because VPRandom algorithm does not sort the PM or the VM in advance, so the 30 VMs in Figures 2(a), 2(b), and 2(c) are all sequentially allocated from PM 1 to 10 if only the PM can afford the required resources. This is the same for the algorithm of VP-FS. When using VPWFD algorithm, the 30 VMs under three groups are all disorderly distributed in each PM, because VPWFD algorithm always tries to find the PM with the largest resources.

5.3. Resource Balancing Degree of Each Algorithm. After VM placement, we calculate the value of b in each PM according to formula (2). The results are shown in Figure 3.

In Figure 3, we compare the value of b in each PM under different algorithms. If the value of b is low, then the PM has a good resource balance. Otherwise, the PM has a bad resource balance. Using the first group with the linear distributed

Input: n VM, m PM, A ants, error margin ϵ, parameter $p, q(q > 1)$
Output: $S = \{\langle V_i, P_j \rangle \mid M_{ij} = 1, 1 \le i \le n, 1 \le j \le m\}$
(1) **for** each $\langle i, j \rangle$ $t_{ij} = 1$
(2) $F = 0$
(3) **do**{
(4) $S = []$
(5) $F_1 = F$
(6) **for** each a from 1 to A
(7) $S_a = []$
(8) **for** each i from 1 to n
(9) choose host j from hosts
(10) Allocate(V_i, P_j)
(11) If $ru_j > u_{\text{sla}}$
(12) then Dealloc(V_i, P_j), back to step 9
(13) $S_a = S_a + [\langle V_i, P_j \rangle]$
(14) $S = S + S_a$
(15) S = choose_energy(p, S)
(16) S = choose_balance(S)
(17) **for** t_{ij} in S
(18) $t_{ij} = t_{ij} * q$
(19) F = energy_sla_sum(S)
(20) } **while**($|F_1 - F| \ge \epsilon$)

Algorithm 3: VP-FS (VM, PM).



(a) CPU requirement of each group

(b) Memory and bandwidth requirements

Figure 1: Resources utilization of each group of workloads.

CPU utilization. We get the results in Figure 3(a). The results obtained by VPBFD and VPRandom are fluctuating wildly from top to bottom. That means only some of the PMs can have a very good resource balance after being allocated with VMs. Because VPBFD firstly considers allocating the VMs to the used PM if only the PM has enough resources, therefore the resources of some PMs can be used effectively. VPWFD firstly considers allocating the VMs to the PM with the largest resources, and therefore all the 10 PMs have been used. Although some of the PMs have the lowest b value under VPBFD, both the VPWFD and VP-FS have an average resources balance among each of the 10 PMs. Comparatively speaking, under linear CPU utilization workloads, b value of VPWFD is a little better than VP-FS. Figure 3(b) is the results under CPU RU even distribution, and we can see taht all the b value of each PM are all in an average manner by using any of the algorithms. Figure 3(c) is the results under CPU RU bipolar distribution. The b values of each PM by using VP-FS

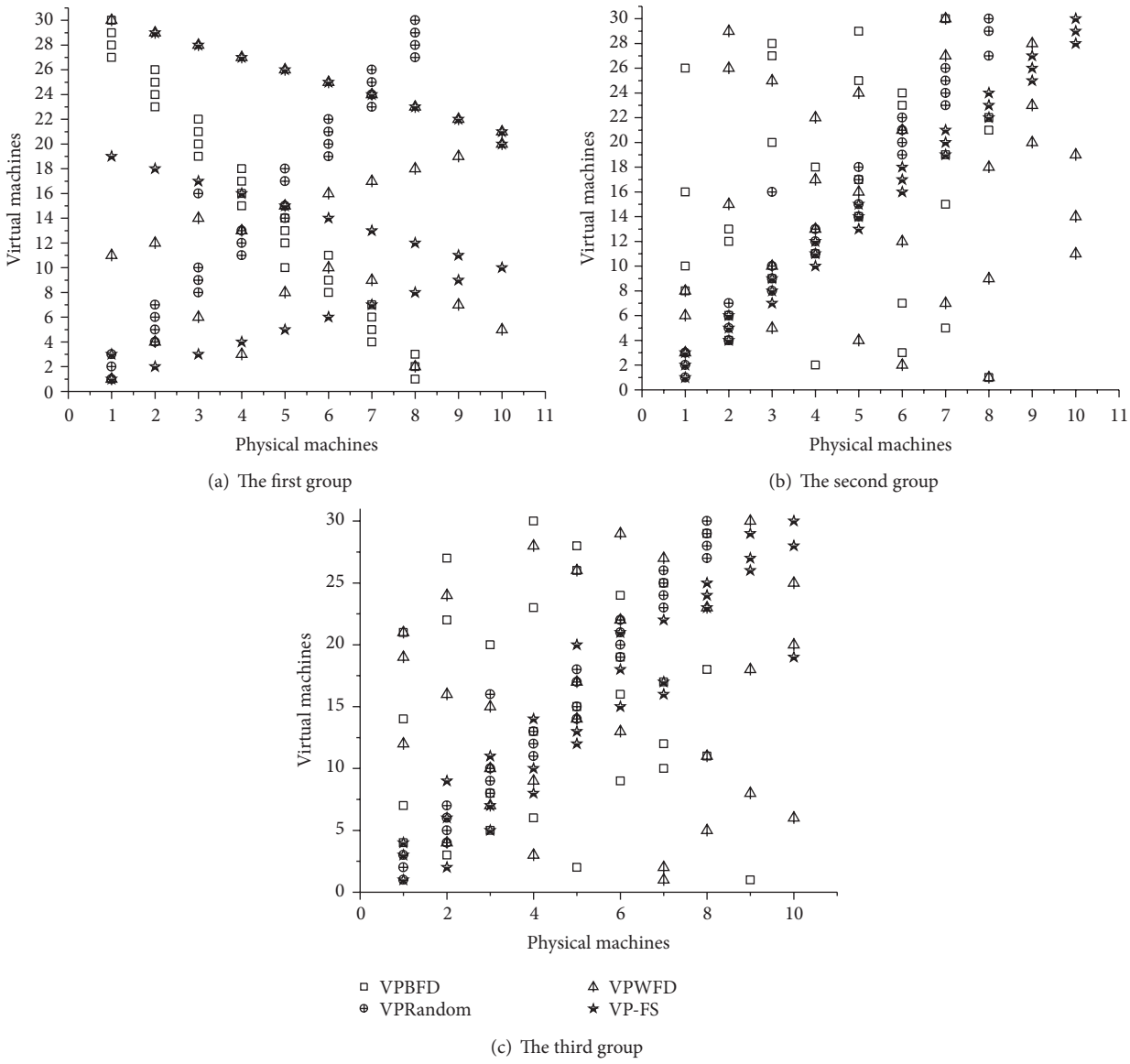


Figure 2: Allocation results under three groups.

are the smoothest compared to the other three algorithms. That means, by using VP-FS, under the workloads in bipolar distribution, all the PMs can obtain the best resource balance. We can also use the standard deviation of b value of the total 10 PMs to describe the resource balance of each algorithm. As Table 1 shows, the standard deviation of b value of VP-FS is small compared with the other algorithms. That means the resource balance is well in each of the 10 PMs under VP-FS.

5.4. Energy Consumption of Each Algorithm. The VMs run for 2 hours after having been placed on the PMs. We use power meters to measure the instantaneous power and the whole power consumption of each PM. When the PMs are in their idle status, we can obtain the basic energy consumption of each PM, as Table 2 shows.

Using the placement results under the 1st group data, the consumption energy of each PM is shown in Figure 4(a), so as to the 2nd group data in Figure 4(b) and the 3rd group data in Figure 4(c). Because VPBFD and VPRandom algorithms only use 8 PMs, their energy consumption in each PM is higher than that of VPWFD and VP-FS in the first 8 PMs. VP-FS leads to almost the least energy consumption in each PM under each group, except for few PMs.

Workload distribution in each group really affects the energy consumption result in each PM. *In the first group*, CPU RU is in a linear distribution pattern. In the four algorithms, VPBFD method will sort the VMs by their resource utilization in descend and VM in the first place will be placed firstly. So the energy consumption from PM 1 to PM 8 is also in a descending manner in Figure 4(a). VPRandom method will not sort VMs in advance and the placement is in a random manner, so the result in Figure 4(a)

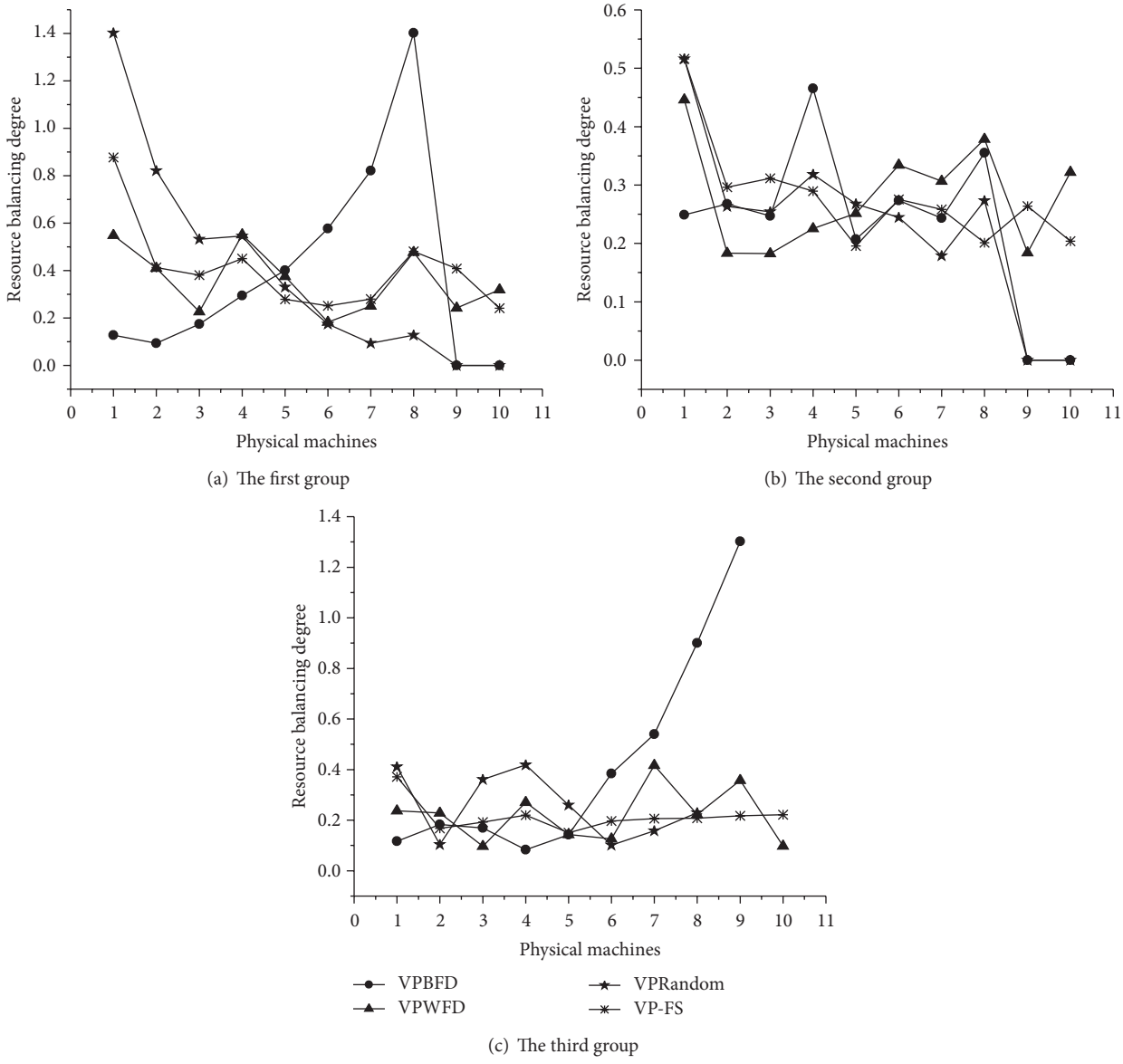


Figure 3: Resource balancing degree under three groups.

Table 1: Standard deviation of b value.

	VPBFD	VPWFD	VPRandom	VP-FS
Group 1	0.4197	0.1285	0.4200	0.1768
Group 2	0.1347	0.0863	0.1426	0.0877
Group 3	0.3984	0.1026	0.1494	0.0565

is totally the same manner in ascending order as the CPU RU distribution. *In the second group*, CPU RU is in an even distribution pattern; according to the above analysis, the energy consumption results of VPBFD and VPWFD are also in an even manner in Figure 4(b). *In the third group*, CPU RU is in a bipolar distribution pattern; although VPBFD will sort VMs in advance, the big RU values are much more than the first group, so VMs in the front positions will be placed to the sequential PMs. Thus, the energy consumption results of VPBFD and VPRandom are also in disordered manner. In any of these three groups, without sorting the VMs in advance while considering proper frequency scaling, VP-FS method leads to similar energy consumption in each PM.

As we all know, basic energy consumption contributes to the large part of the total energy consumption of an active PM. Including the basic energy consumption of the PM number 9 and PM number 10, VP-FS still has the lowest total energy consumption in the four VM placement algorithms.

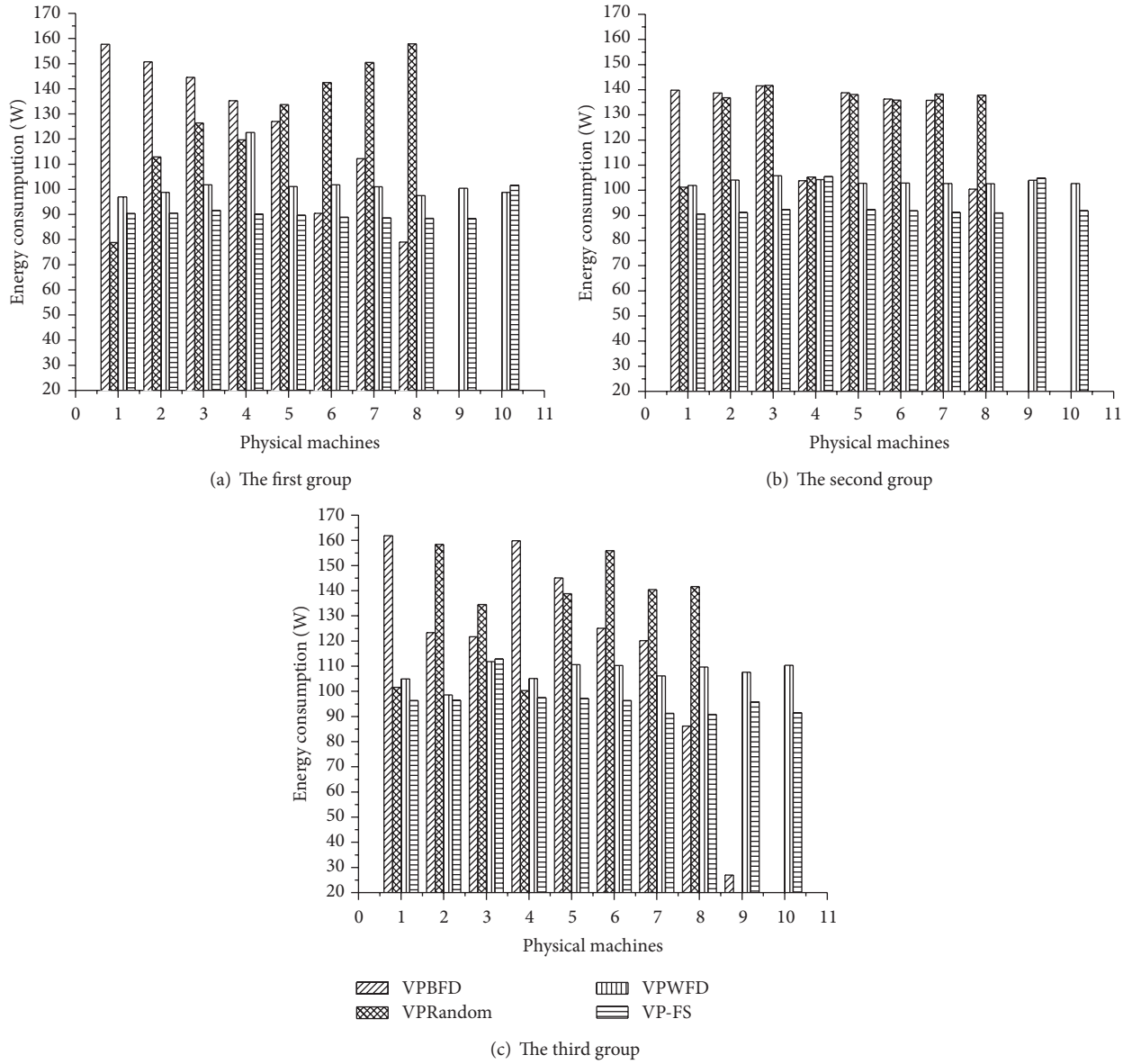


Figure 4: Energy consumption results under three groups.

Table 2: Basic energy consumption of each PM.

Physical machine	PM 1	PM 2	PM 3	PM 4	PM 5	PM 6	PM 7	PM 8	PM 9	PM 10
Basic energy consumption (W)	94.57	95.35	99.42	96.84	95.88	94.78	94.50	94.81	95.12	94.37

Table 3: Energy saving between VP-FS and other algorithms.

	VP-FS/VPBFD	VP-FS/VPRandom	VP-FS/VPWFD	Average
1st group	8.90%	11.17%	11.03%	10.37%
2nd group	8.944%	8.946%	8.81%	8.89%
3rd group	1.84%	9.87%	10.16%	7.29%
Average	6.56%	9.99%	10%	8.85%

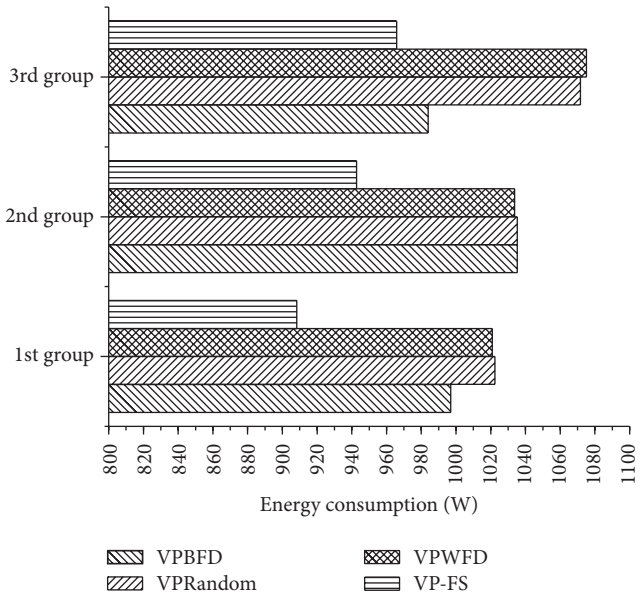


Figure 5: Total energy consumption under three groups.

Figure 5 is the *total energy consumption* under each group of data. In each of the three groups, VPRandom and VPWFD methods get almost the same energy consumption. Table 3 is detailed data which show the energy saving results between VP-FS and other algorithms. From Figure 5 and Table 3, in the first group, when CPU workload is in a linear distribution pattern, VP-FS has the best energy saving results of 10.37% to the other traditional methods. VPBFD has the second best energy consumption result. In the third group, the advantage of VP-FS is not very obvious compared to VPBFD. The energy saving of VP-FS compared to VPBFD is only 1.84%. That means when workloads is in a bipolar distribution, VPBFD is almost as good as VP-FS method. On the average, VP-FS has saved 6.56%, 9.99%, and 10% total energy compared with VPBFD, VPRandom, and VPWFD, respectively. Obviously, using frequency scaling in initial VM placement, PM can find the proper CPU frequency for the certain VM that allocated to it.

6. Conclusions

In this paper, we consider workload characteristics with dynamic CPU utilization and energy consumption with CPU frequency scaling to the ordinary VM placement problem. If a PM has h different CPU frequencies, then the solution space of the VM placement problem will be expanded from m^n to be $h^m m^n$. Using energy consumption model, we verify that there will be a CPU frequency that best fit for a CPU utilization in PM with respect to the minimum energy consumption. We then modeled the objectives of energy, resource balance and SLA for optimal VM placement solution, and propose an ACO-based CPU frequency scaling algorithm VP-FS. In order to compare the effect, we put forward three typical greedy algorithms, which are VPBFD, VPRandom, and VPWFD. Each of them has different greedy policy. We design three groups of VMs with different resource distribution, so that we can have an evaluation of the impact which the workloads bring to the algorithms in VMs consolidation and energy saving. Our efforts show that, for workloads with different CPU resource utilization, running under different algorithms will produce different VMs allocation results. If we consider the numbers of used PMs, then different group of workload will have its most suitable algorithms for minimum used PMs. VP-FS is not the best algorithms considering the number of used PMs. However, because of its frequency scaling policy, it has the lowest energy consumption compared with the other three algorithms under three different groups of VM workloads. In the future, we will further consider workload awareness to dynamic VM allocation.

Notation

n: Number of VMs
m: Number of PMs
h: Frequencies of a PM
f: Instantaneous frequency of a PM
c: Energy consumption of PM in its idle time
V_i: VM i $(1 \le i \le n)$
P_j: PM j $(1 \le j \le m)$
d: Dimension of the resources
u_r: Utilization ratio of resource r, $u_r \in [u_1 \cdots u_d]$
w_r: Weight of resource r
ru_j: Remnant of the comprehensive resource utilization of P_j
M_{ij}: Placement solution; if V_i is placed in P_j, then $M_{ij} = 1$, else $M_{ij} = 0$
e_{ij}: Energy consumption of V_i in P_j.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This paper is sponsored by the National Natural Science Funds of China under Grants nos. 61202429, 61379145.

References

[1] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] P. Kurp, "Green computing," *Communications of the ACM*, vol. 51, no. 10, p. 1113, 2008.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[4] U. R. M. Saif, U. K. Samee, and K. S. Sudarshan, "Modeling and analysis of state-of-the-art VM-based cloud management platforms," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 50–63, 2013.

[5] E. G. Coffman, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: a survey," in *Approximation Algorithms for NP-Hard Problems*, pp. 46–93, PWS Publishing, Boston, Mass, USA, 1997.

[6] W. Zhang, Y. Song, L. Ruan, M. Zhu, and L. Xiao, "Resource management in internet-oriented data centers," *Journal of Software*, vol. 23, no. 2, pp. 179–199, 2012.

[7] A. Verma, P. Ahuja, and A. Neoqi, "pMapper: power and migration cost aware application placement in virtualized systems," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pp. 243–264, 2008.

[8] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "EnaCloud: an energy-saving application live placement approach for cloud computing environments," in *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, pp. 17–24, September 2009.

[9] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '10)*, pp. 826–831, May 2010.

[10] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proceedings of the IEEE/ACM International Conference on & International Conference on Cyber, Physical and Social Computing (CPSCom '10), Green Computing and Communications (GreenCom '10)*, pp. 179–188, Hangzhou, China, December 2010.

[11] Q. Li, Q. Hao, L. Xiao, and Z. 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.

[12] Z. Wenjun and C. Jian, "Cloud computing resource scheduling strategy based on prediction and ACO algorithm," *Computer Simulation*, vol. 29, no. 9, pp. 239–242, 2012.

[13] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM '09)*, pp. 327–334, IEEE, Piscataway, NJ, USA, June 2009.

[14] A. Verma, G. Dasgupta, T. Nayak et al., "Server workload analysis for power minimization using consolidation," in *Proceedings of the Conference on USENIX Annual Technical Conference*, p. 28, USENIX Association, Berkeley, Calif, USA, 2009.

[15] S. Takeda and T. Takemura, "A rank-based VM consolidation method for power saving in datacenters," *IPSJ Transactions on Advanced Computing Systems*, vol. 3, no. 2, pp. 138–146, 2010.

[16] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science (MGC '10)*, Bangalore, India, December 2010.

[17] Y. O. Yazr, C. Matthews, and R. Farahbod, "Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis," in *Proceedings of the IEEE 3rd International Conference on Cloud Computing*, pp. 91–98, 2010.

[18] S. Kumar, V. Talwar, V. Kumar, P. Ranganathan, and K. Schwan, "Vmanage: loosely coupled platform and virtualization management in data centers," in *Proceedings of the 6th International Conference on Autonomic Computing (ICAC '09)*, pp. 127–136, Barcelona, Spain, June 2009.

[19] E. N. Elnozahy, M. Kistler, and R. Rajamony, "Energy-Ecient server clusters," in *Proceedings of the 2nd International Workshop on Power-Aware Computer Systems*, pp. 179–197, Springer, Cambridge, Mass, USA, 2003.

[20] J. Song, T. Li, Z. Yan, J. Na, and Z. Zhu, "Energy-efficiency model and measuring approach for cloud computing," *Journal of Software*, vol. 23, no. 2, pp. 200–214, 2012.

[21] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of virtual machines for real-time Cloud services," *Concurrency Computation Practice & Experience*, vol. 23, no. 13, pp. 1491–1505, 2011.

[22] J. Yang, P. Liu, and J. Wu, "Workload characteristics-aware virtual machine consolidation algorithms," in *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom '12)*, pp. 42–49, December 2012.

[23] H. S. Abdelsalam, K. Maly, R. Mukkamala, M. Zubair, and D. Kaminsky, "Analysis of energy efficiency in clouds," in *Proceedings of the Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns (COMPUTATIONWORLD '09)*, pp. 416–421, Athens, Greece, November 2009.

[24] Y. Y. Chen, A. Das, W. B. Qin, and A. Sivasubramaniam, "Managing server energy and operational costs in hosting centers," in *Proceedings of the International Conference on Measurements and Modeling of Computer Systems (SIGMETRICS '05)*, D. L. Eager, C. L. Williamson, and S. C. Borst, Eds., pp. 303–314, Banff, Canada, 2005.

[25] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1374–1381, 2011.