Performance-aware Energy-efficient Virtual Machine Placement in Cloud Data Center

Xiaoning Zhang[1], Yangming Zhao[2], Shuai Guo[1], Yichao Li[1]
[1]Key Laboratory of Optical Fiber Sensing and Communication, Education Ministry of China
University of Electronic Science and Technology of China
[2]Department of Computer Science and Engineering
State University of New York at Buffalo

***Abstract*—This paper studies how to consolidate Virtual Machines (VMs) on physical severs in order to minimize the energy consumption in Cloud data centers. In Cloud data centers, a major energy consumption component is the physical servers, and it is a way to save the energy by consolidating VMs onto fewer VMs and shut down the idle physical servers. However, executing more VMs on fewer physical servers should degrade the VM performance. Accordingly, how to consolidate VMs but guarantee the VM performance is an important issue to study. To this end, we first formulate the VM consolidation problem as a mathematic model and then prove that it is equivalent to a generalized constrained minimum k-cut problem, which is the hardest NP-hard problem. Due to the problem complexity, an efficient heuristic named PEVMP (Performance-aware Energy-efficient Virtual Machine Placement) is proposed to solve it. In addition, we extend the PEVMP to be an online version to handle service dynamics. Extensive simulation results show that PEVMP can save up to 36.78% energy, and the performance of online algorithm is close to the offline result.**

I. Introduction

With the increasing popularity of cloud computing, data centers are burgeoning around the world and result in a great amount of energy consumption. Such energy consumption not only leads to significant emissions of carbon dioxide into the environment, but also increases the operation cost of data centers. It has been reported that energy cost achieves 75% of the overall operating cost of a data center by 2014 [1]. Accordingly, energy efficiency is an important issue in Cloud data center.

The energy cost in data centers is due to physical servers, networking devices, other auxiliary equipments such as cooling systems [2]. Among these, physical servers take the lion's share. The energy consumed by the CPU for computation reaches around 45% of the total energy consumption of a data center [3]. However, the CPU utilization for thousands of servers was found to be between 10 and 50 percent of their maximum utilization most of the time [4]. Hence, many researchers have proposed to reduce the energy consumption by consolidating VMs onto a few physical servers and shut down the idle physical servers [5].

This is not the whole of this problem. When we consolidate VMs onto fewer physical servers, the processing time of jobs on each VM would increase since multiple VMs should complete for the physical resources [6]. This may degrade the performance of jobs/applications executing in the data center. Accordingly, how to guarantee the job/VM performance is another ingredient that should be considered when we consolidate VMs to save energy.

In this paper, we are to study such energy-efficient VM placement problem under the constraint of VM performance, i.e. the VM execution time. To this end, we first formulate the energy efficiency problem as a mathematical model, and then indicate that this model is equivalent to constrained minimum k-cut problem [7]. Since this problem is well known to be the hardest NP-hard problem, which we cannot derive a constant-approximation algorithm with polynomial time complexity, we propose an efficient heuristic named PEVMP (Performance-aware Energy-efficient Virtual Machine Placement) to solve it. Extensive simulation results show that our proposed algorithm can save energy consumption of physical servers by up to 36.78% while satisfying the requirement of VM processing time.

The main technical contributions of this paper can be summarized as follows:

- We formulate the performance-aware VM placement problem as a mathematical model and indicate it is equivalent to constrained minimum k-cut problem.
- We propose an efficient heuristic named PEVMP to solve such constrained minimum k-cut problem, which is well-known NP-hard. We extend the offline algorithm to adopt to serve dynamic VM requests.
- Extensive simulation results show that our algorithm can save up to 36.78% energy and ensure the VM performance requirement. The performance of online algorithm is close to offline optimization results.

The rest of this paper is organized as follows. In Section II, we briefly review the literatures related to our work. In Section III, we illustrate our motivation through a simple example. In Section IV, we model the problem to solve as a mathematical model and indicate it is equivalent to a constrained minimum k-cut problem. In Section V, we propose an efficient heuristic called PEVMP (Performance-aware Energy-efficient Virtual Machine Placement) to solve

This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grant (91438117, 61671124, 61201129, 61401070).

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

TABLE I
NOTATION LIST

Notation	Description
n	the number of VMs to be placed on physical servers
k	the number of physical servers
vm_i	the i-th virtual machine, where $0 \leq i \leq n$
s_j	the j-th physical server, where $0 \leq j \leq k$
$u(vm_i)$	CPU utilization of vm_i, where $0 \leq u(vm_i) \leq 1$
$t(vm_i)$	ideal processing time of vm_i on a non-competitive server
$p(s_j)$	energy consumption of server j
$t'(vm_i)$	processing time of vm_i on a competitive server
$Th(vm_i)$	the threshold of VM processing time for vm_i
P_{total}	total energy consumption of all physical servers

the problem. We present simulation results in Section VI and conclude our paper in Section VII.

II. RELATED WORKS

There has been a significant amount of prior work on energy saving by VMs consolidation. There are two kinds of approaches in VMs consolidation. The first one is to make VM migration when VMs are detected to be placed dispersedly. The other one is to avoid the situations where VMs may be dispersedly placed in advance [8]. Although VM migration can consolidate workload and well reduce the number of servers needed to stay power on, the cost for migration is considerable. Sometimes, it may overshoot the benefit from shutting down of servers [9]. In this paper, we restrict our study to consider only VM placement, ignoring live migration. The study of VM placement investigates allocating virtual machines to physical servers in a data center. It is often formulated as a Bin Packing (BP) problem. Since BP is NP-complete, heuristic turns to be a general method to deal with this kind of problems [10]. Verma et al. [11] applied First Fit (FF) descending to generate an initial assignment of VMs, and later used Best Fit (BF) to reassign VMs picked from PMs which violate resource constraints. Goiri et. al. [12] has developed a score-based heuristic which is hill-climbing algorithm search for best match pairs. Beloglazov et al. [13] considered offline allocation of VMs by modified BF descending heuristics. Our work in this paper differs from all the previous work by considering not only minimization of energy consumption, but also the constraint of VM processing time that is SLA/performance assurance of running jobs.

III. BACKGROUND AND MOTIVATION

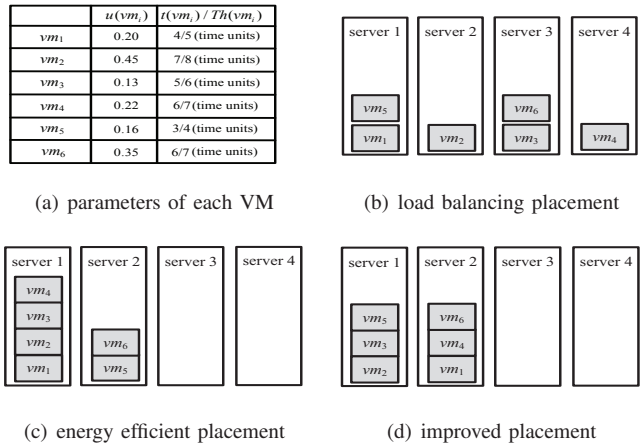
In this section, we first introduce the background, such as the energy model and the VM processing time model used in our work. After that, we motivate our work with a simple example. For clear presentation, we first summarize the notations that will be used in our paper in Tab. I.

A. Background

When we place n virtual machines on server j, energy consumption of server j is:

$$p(s_j) = \sigma_p + \mu_p(\sum_{i=1}^{n} u(vm_i))^{\alpha} \quad (1)$$

where σ_p represents the fixed cost of maintaining a server powered on and available, and μ_p, α are parameters of the device which characterizes its dynamic energy consumption as a function of CPU utilization. In our study, we set σ_p=126.46, μ_p=4.55 and α=0.86 [14].



(a) parameters of each VM (b) load balancing placement

(c) energy efficient placement (d) improved placement

Fig. 1. Example of four VM placement schemes

Lim et al. [15] present a mathematical model for estimating the processing time of VMs in a shared physcial servers by treating it as a conventional job scheduling problem. Based on the model, the new VM processing time on server j is:

$$t'(vm_i) = t(vm_i) * (1 + u(vm_i) * \sum_{i=1}^{n} u(vm_i) - u(vm_i)^2) \quad (2)$$

where $t(vm_i)$ represents ideal processing time of vm_i on a non-competitive server, i.e. there is only one VM (i.e., vm_i) placed on the server. When multiple VMs are consolidated in the same server, the new processing time $t'(vm_i)$ would increase since resource contention.

In our study, to assure the performance of running jobs on the VM, the new $t'(vm_i)$ must be satisfy the corresponding constraint of VM processing time $Th(vm_i)$:

$$t'(vm_i) \leq Th(vm_i) \quad (3)$$

B. Motivation Example

Conventionally, most of VM placement schemes pursue load balancing of physical servers to minimize resource utilization or VM processing time. It induces all physical servers are always powered on even in low-load period. Thus a large amount of energy consumption is wasted. In fact, VM processing time is a loose constraint, and hence we can aggregate some of VMs on fewer physical servers to save energy under the constraint of VM processing time, though some of the tasks are slightly delayed.

More detailedly, we illustrate our motivation with the example shown in Fig.1. We suppose that there are 6 VMs (vm_i, i=1,...,6), which are placed to 4 identical physical servers. The CPU utilization $u(vm_i)$, the ideal processing time $t(vm_i)$ and the constraint of processing time $Th(vm_i)$ are given in Fig.1 (a). Fig.1 (b) shows the traditional load balancing VM placement scheme. In this scheme, VMs are uniformly distributed onto all physical servers. Thus vm_1 and vm_5 are

both placed onto server 1, vm_2 is placed onto server 2, vm_3 and vm_6 are both placed onto server 3, and vm_4 is placed onto server 4. From Eq. (1), the total energy consumption is 513.68 (128.35+128.75+128.88+127.70) energy units. From Eq. (2), the new VM processing time of the 6 VMs is 4.13, 7, 5.23, 6, 3.20, and 6.30 time units, respectively. The new VM processing time of each VM satisfies the corresponding constraint. Fig.1 (c) shows the energy-efficient VM placement scheme. In this scheme, VMs are consolidated onto fewer servers to save energy. Thus vm_1, vm_2, vm_3, vm_4 are placed onto server 1, vm_5 and vm_6 are placed onto server 2. From Eq. (1), the total energy consumption is 260.02 (131.01+129.01) energy units. From Eq. (2), the new VM processing time of the 6 VMs is 4.64, 8.73, 5.57, 7.03, 3.17 and 6.34 time units, respectively. We can see that the new processing time of vm_2 and vm_4 violates the corresponding constraint. Fig.1 (d) shows the improved VM placement scheme, which minimizes energy under the constraint of VM processing performance. In the scheme, vm_2, vm_3, vm_5 are placed onto server 1; vm_1, vm_4, vm_6 are placed onto server 2. From Eq. (1), the total energy consumption is 260.06 (129.97+130.09) energy units; From Eq. (2), the new VM processing time of the 6 VMs is 4.46, 7.91, 5.40, 6.73, 3.28 and 6.88 time units, respectively. The new VM processing time of each VM satisfies the corresponding constraint.

As shown in Fig.1, the load balancing VM placement scheme minimizes the VM processing time, but consumes a large amount of energy. The energy-efficient VM placement scheme minimizes energy consumption, but the new VM processing time of two VMs violates the corresponding constraint since four VMs are placed on the server. The improved VM placement scheme minimizes energy consumption while satisfying the constraint of VM processing time. *From the above example, we know that the solution of energy-efficient VM placement problem cannot provide SLA/performance guarantee. This problem should be solved with consideration of the VM processing time constraint, which is essential for many applications running on the VM.*

IV. Problem Formulation and Analysis

In our study, the goal is to minimize energy consumption of physical servers under the constraint of VM processing time. Accordingly, we can formulate this problem as:

$$\text{minimize:} \quad P_{total} = \sum_{i=1}^{k} p(s_j) \tag{4}$$

$$\text{subjec to:} \quad t'(vm_i) \leq Th(vm_i) \quad 1 \leq i \leq n \tag{5}$$

Consider a undirected graph $\boldsymbol{G} =< \boldsymbol{V}, \boldsymbol{E} >$, where $\boldsymbol{V}$ is the set of vertexes and $\boldsymbol{E}$ is the set of edges. Each vertex $v \in \boldsymbol{V}$ denotes a virtual machine to be placed on physical servers. Thus there are altogether n vertexes in the graph. For each vertex pair (v_i, v_j) in $\boldsymbol{G}$, if the corresponding virtual machines (vm_i, vm_j) can be placed on the same server, i.e. the corresponding processing time constraint can be satisfied, which can be formally formulated as:

$$t'(vm_i) \leq Th(vm_i) \quad and \quad t'(vm_j) \leq Th(vm_j) \tag{6}$$

there is an edge between vertex v_i and v_j. The weight of edge $e(v_i, v_j)$ is set to be the saved energy by placing vm_i and vm_j on the same server, i.e.:

$$\begin{aligned} w_{e(i,j)} =& (2\sigma_p + \mu_p * u(vm_i)^\alpha + \mu_p * u(vm_j)^\alpha) - \\ & (\sigma_p + \mu_p * (u(vm_i) + u(vm_j))^\alpha) \end{aligned} \tag{7}$$

On graph $\boldsymbol{G}$, if two vertexes are connected by an edge with larger weight, more energy can be saved by consolidating the corresponding virtual machines onto the same server.

Consider the case that we place multiple VMs on the same server, all the energy-saving value represented by the links on the graph $\boldsymbol{G}$ can be concreted. Contrary, if two vertexes are connected by an edge on graph $\boldsymbol{G}$, but their corresponding VMs are placed on two different servers, we loss part of the optimization space in terms of reducing energy consumption in a cloud data center. The amount of energy we loss to save is at least the weight of the deleted edge. Accordingly, to determine the VM placement scheme, we can delete some edges in the graph $\boldsymbol{G}$ and divide it into some partitions. All the VMs associated with the vertexes of a partition have to be placed on the same server. To minimize energy consumption, we should minimize the weight sum of all the edges that are deleted for graph partition purpose.

Actually, this problem is a more general constrained minimum k-cut problem. Different from the conventional minimum k-cut problem, where the number of partitions is known as an algorithm input parameter, the number of partitions in our problem unknown and determined by the VM performance requirement.

Theorem 1: The constrained minimum k-cut problem in our study is an NP-hard problem.

Proof. We consider a special case that the CPU utilization and processing time of each VM is the same. Since all servers are assumed identical in our study, the number of VMs which can be mapped to each server is the same. In this case, the constrained minimum k-cut problem is changed to the load-balanced minimum k-cut problem [7], which is a well-known NP-hard problem. Actually, it is the hardest NP-hard problem, i.e. we cannot even find an algorithm with constant approximation ratio. Accordingly, the original constrained minimum k-cut problem in our study is an NP-hard problem. ■

V. The PEVMP Algorithm

Since the problem to solve in our work is NP-hard, we propose a heuristic named PEVMP (Performance-aware Energy-efficient Virtual Machine Placement) algorithm to minimize energy consumption of physical servers under the VM processing time constraint.

A. The Algorithm Description

Alg.1 shows the abstract of PEVMP algorithm. We first generate an auxiliary graph *G* as discussed in Section IV(line 1). Then we partition the graph *G* into multiple unconnected sub-graphs with *EnergySavingCut* function (line 2). If some nodes are located in one sub-graph, it means the corresponding virtual machines are placed onto the same server. After line 2, we can get a feasible VM placement solution. To further

Algorithm 1: PEVMP Algorithm

Require: The n virtual machines $\{vm_1, vm_2,..., vm_n\}$ to be deployed, and vm_i ($1 \leq i \leq n$) has three parameters ($u(vm_i)$, $t(vm_i)$, $Th(vm_i)$); The identical k physical servers $\{s_1, s_2,..., s_k\}$

Ensure: The VM placement result with the minimal energy consumption under the VM processing time constraint

1: Generate the auxiliary graph model $G(V, E)$ from n virtual machines
2: Run $EnergySavingCut()$ function to partition the graph G into multiple unconnected sub-graphs
3: Run $GreedyKnapsack()$ function to merge VMs onto fewer servers for energy efficiency
4: Return the final VM placement result and calculate total energy consumption of all physical servers

Algorithm 2: EnergySavingCut Function

Require: The auxiliary graph model $G(V, E)$, $Th(vm_i)$

Ensure: The set of unconnected sub-graphs G_1, G_2,..., G_{num} from G and the corresponding VM placement result

1: Initialize $tree\_node\_number$=0
2: **while** ($tree\_node\_number$<n) **do**
3: Randomly select a new node v (which means v does not exist in the tree) from G join in the tree T
4: **if** ($tree\_node\_number$>0) **then**
5: Randomly connect node v to a node v' of the tree with probability $1/tree\_node\_number$
6: Set the capacity of the link (v, v') of T as the maximum flow of node pair (v, v') in G
7: $tree\_node\_number$=$tree\_node\_number$+1
8: **end if**
9: **end while**
10: Sort all links of the tree T according to the ascending order of link capacity and store the sorted links to the link set L
11: Initialize $delet\_time$=0
12: Select the first element l with the minimal capacity from L and remove l in L
13: $delet\_num$=$delet\_num$+1
14: Assume link l connects node pair (v_i, v_j) in T, accordingly delete minimum cut links between node pair (v_i, v_j) of G. Thus G would be further divided into $delet\_num$+1 unconnected sub-graphs $\{G_1, G_2,..., G_{delet\_num+1}\}$
15: For the graph set $\{G_1, G_2,..., G_{delet\_num+1}\}$, assume there are num_i nodes in the graph G_i, which means the corresponding virtual machines are placed on one server
16: **for** (all unconnected sub-graphs G_i with num_i) **do**
17: **if** (the VM processing time constraint is not satisfied) **then**
18: Break and go back to line 12
19: **end if**
20: **end for**
21: Return the final VM placement result

improve energy efficiency, we merge VMs onto fewer servers with *GreedyKnapsack* function (line 3). At last, PEVMP returns the final VM placement result and calculates energy consumption of all physical servers (line 4).

The key points in PEVMP are $EnergySavingCut$ and $GreedyKnapsack$ functions. The $EnergySavingCut$ function divides the graph model G into multiple partitions to get the initial VM placement solution. The $GreedyKnapsack$ function merges different partitions further to save energy. The detail of $EnergySavingCut$ function is shown in Alg.2. In this function, to get the energy-efficient VM placement result for the constrained minimum k-cut problem, we first construct *Gomory-Hu* tree T of the auxiliary graph model G (line1-9). It should be noted that the *Gomory-Hu* tree should be a random one. The word "random" has two meanings. The first meaning is that a new node v (i.e., node v is not included in the tree) is randomly selected from G to join in the tree (line 5). The second meaning is that node v is randomly connected with an arbitrary node of T (line 6). We assume node v is connected to node v' of T. The capacity of link (v, v') in T equals the maximum flow of node pair (v, v') on G. Thus all nodes of G join in T. After the *Gomory-Hu* tree T is established, we can divide the tree T by deleting links successively. Since the link capacity of T equals to the maximum flow of the graph G, when we delete a link from T, the corresponding minimum cut links are deleted from G. Then the graph G would be divided into multiple unconnected partitions (line10-20). If some of nodes locate in the same partition, it means that the corresponding VMs are placed onto one server. Whenever we delete minimum cut links from G, we check if the present VM placement result satisfies the VM processing time constraint. If so, $EnergySavingCut$ function returns the VM placement result and stops; otherwise $EnergySavingCut$ function loops back to line 12 to delete more links from T.

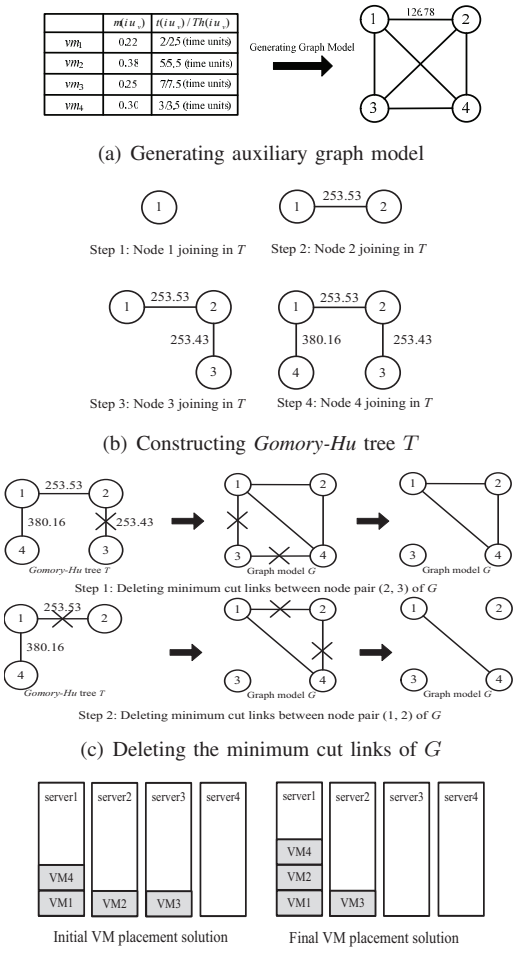
As shown in Alg.3, $GreedyKnapsack$ function further improves energy efficiency by merge virtual machines onto fewer physical servers with greedy knapsack method. We first sort all VM_Set_i according to the descending order of its energy consumption and store the result to the set of $VM\_Sorting$ (line 2). We prefer to merge the VM set consuming the largest energy with the subsequent VM set in $VM\_Sorting$ (line 4-10). If successful, it means that the VM processing time constraint is satisfied. Then we can merge the two VM sets into the same server (line 5-6). The function adopts traversal method until there are no virtual machines can be merged to reduce energy consumption of physical servers.

B. An Illustration for PEVMP Algorithm

Fig.2 shows an illustration of our proposed PEVMP algorithm. Fig.2 (a) illustrates four VMs with parameters and the result of generating auxiliary graph model *G*. In the graph *G*, node i (i=1,2,3,4) denotes vm_i. The number nearby the link is capacity of link, which denotes the saved energy when consolidating the corresponding virtual machines on the same

Algorithm 3: GreedyKnapsack Function

Require: The initial VM placement solution from EnergySavingCut Function
Ensure: The energy-efficient final VM placement solution
1: For the initial VM placement solution, assume the set of virtual machines VM_Set_i (it contains num_i virtual machines) is allocated to physical server s_i
2: Sort all VM_Set_i according to the descending order of its energy consumption and store the result to the set of $VM\_Sorting$
3: Initialize i=1
4: **for** (all $VM\_Sorting[i]$) **do**
5: **for** (all $VM\_Sorting[i+1]$) **do**
6: **if** (the VM processing time constraint is satisfied after merging) **then**
7: Merge $VM\_Sorting[i+1]$ into $VM\_Sorting[i]$
8: **end if**
9: **end for**
10: **end for**
11: Return the final energy-efficient VM placement result



(a) Generating auxiliary graph model

(b) Constructing *Gomory-Hu* tree T

(c) Deleting the minimum cut links of G

(d) Greedy knapsack

Fig. 2. Example of PEVMP algorithm

server. For example, when we place vm_1 and vm_2 onto one server, the consumed energy is 129.39. So the saved energy is 126.74 (i.e., (127.70+128.43)-129.39) energy units and the capacity of link (1, 2) is 126.74. We specially note that in the graph *G* only node pair (2, 3) cannot be connected because the corresponding virtual machines (vm_2, vm_3) cannot be placed onto one server (i.e., the VM processing time of vm_3 is 7.665 time units which is larger than the threshold value $Th(vm_3)$). Fig.2 (b) illustrates constructing the *Gomory-Hu* tree T in *EnergySavingCut* function. In step 1, node 1 is randomly selected. In step 2, node 2 is randomly selected and joins in the tree T, the capacity of link (1, 2) of T is the maximum flow of node pair (1, 2) in *G* (i.e., 253.53). In the same way, node 3 and node 4 join in T sequentially. Fig.2 (c) illustrates deleting the minimum cut links of the graph G in *EnergySavingCut* function. In step 1, link (2, 3) with the minimum capacity in T is deleted from the tree. Accordingly, the minimum cut links (i.e., link (3, 1) and (3, 4)) between node pair (2, 3) in the graph G are deleted. In the same way, link (1, 2) is then deleted from T, and the minimum cut links (i.e., link (2, 1) and (2, 4)) in the graph G are deleted. Thus the graph G is divided into three unconnected sub-graphs: {1, 4}, {2} and {3}. The corresponding VM placement result is as following: vm_1 and vm_4 are both placed onto server 1; vm_2 is placed onto server 2; vm_3 is placed onto server 3. We find the VM placement solution can satisfy the VM processing time constraint. Fig.2 (d) illustrates greedy knapsack. First, we sort the VMs according to the descending order of its energy consumption. The sorting result is as following. The first "item" is vm_1 and vm_4 with the biggest energy consumption, the second "item" is vm_1, and third "item" is vm_3. We merge the items on physical servers with greedy method. The final VM placement result of the example is as following: vm_1, vm_2 and vm_4 are placed onto server 1; vm_3 is placed onto server 2. The final virtual machine placement solution minimizes energy consumption of physical servers while satisfying the VM processing time constraint.

C. Analysis of Online VM Placement Algorithm

Since the dynamics of user requests, in Cloud data center virtual machines are dynamically created and deleted. This makes the VM placement algorithm to response in real-time manner. In this subsection, we study how to extend our proposed offline algorithm adapt to online system. We propose P-PEVMP (Partial PEVMP) algorithm. The P-PEVMP algorithm does not reallocate existing virtual machines on physical servers. It only allocates the batch of new arriving virtual machines using the PEVMP algorithm. We think that the P-PEVMP has a low computation complexity and is suitable for online VM placement in Cloud data center.

VI. Simulation Results

In this section, we evaluate the performance of PEVMP algorithm through extensive simulations. At first, we study the performance of our offline algorithm in Section VI.A, and

then survey its performance when it is extended into the online case in Section VI.B.

A. Performance of Offline Algorithms

In the simulation, we compare our proposed PEVMP algorithm with the Energy-Efficient Virtual Machine Placement (EEVMP) algorithm and the Load Balancing Virtual Machine Placement (LBVMP) algorithm in offline way. The method of two algorithms is BF (Best Fit) heuristic summarized from ref.[8-13]. The EEVMP and the LBVMP both place VMs in a decreasing order of CPU utilization (that is proportional to energy consumption from Eq. (1)), and at each step, the next VM is placed to the best available server. The main difference is how to select the "best" server to place on. The EEVMP algorithm prefers consolidates VMs on fewer servers for energy efficiency, and the LBVMP algorithm disperses VMs on more servers for load balancing. It is a greedy manner to place the VMs and, by this way, we can achieve a local optimization. We assume the number of physical servers is 100. The number of VMs ranges from 50 to 300. The CPU load of VM is random with a uniform distribution *U*[0.1, 0.5]. The processing time unit of VM is random with a uniform distribution *U*[10, 40]. For brevity, we assume the VM processing time constraint for all VMs is 50 time units (i.e., $Th(vm_i)$=50).

Fig.3 shows simulation results of three algorithms, including energy consumption (see in Fig.3 (a)), the maximum VM processing time (see in Fig.3 (b)) and the number of power-on servers (see in Fig.3 (c)). From Fig.3, we can see that the LBVMP consumes the maximum energy consumption and the number of power-on physical servers, but the maximum VM processing time is minimal among the three algorithms. The reason is that the LBVMP disperses VMs on as many servers as possible for load balancing. Thus the largest number of physical servers is powered on and energy consumption is wasted. Compared with the LBVMP, the PEVMP can save up to 36.78% energy in average, and energy-saving effect of the PEVMP is close to that of the EEVMP (only increasing by 7.67% in average). But the maximum VM processing time in the PEVMP is obviously less than that of the EEVMP (decreasing by 34.42% in average). We explain it as the following. The PEVMP first generates a novel auxiliary graph to evaluate energy-saving effect. And the link weight of the graph model is set as the saved energy when consolidating the corresponding virtual machines to one server. The graph is divided into unconnected partitions by deleting minimum cut links so that the links with larger weight are retained. It means that we can save more energy by placing VM corresponding to the nodes in the same partition onto the same physical server. It is exactly what the PEVMP does. So compared to the classic EEVMP algorithm, the PEVMP achieves the same level of energy efficiency. On the other hand, the PEVMP takes the VM processing time constraint into account when it tries to minimize energy consumption. But the EEVMP does not consider this factor. Thus the maximum VM processing time in the EEVMP is much larger than the threshold, ranging from 66.97 to 84.02 time units. Accordingly, the EEVMP cannot guarantee the performance of running jobs on VM, though it achieves energy efficiency. From Fig.3, we can see that our proposed PEVMP derives a fair and efficient solution.

Intuitively, for our proposed PEVMP algorithm the value of $Th(vm_i)$ would affect simulation results. We evaluate the impact of $Th(vm_i)$ on energy consumption, the maximum VM processing time and the number of power-on servers. In the simulation, we assume the number of physical servers is 100. The number of VMs ranges from 50 to 300. The CPU load of VM is random with a uniform distribution *U*[0.1, 0.5]. The processing time unit of VM is random with a uniform distribution *U*[10, 40]. For brevity, we assume the VM processing time constraint for all VMs is from 20 to 80 time units (i.e., $Th(vm_i)$=20, 30,...,80).

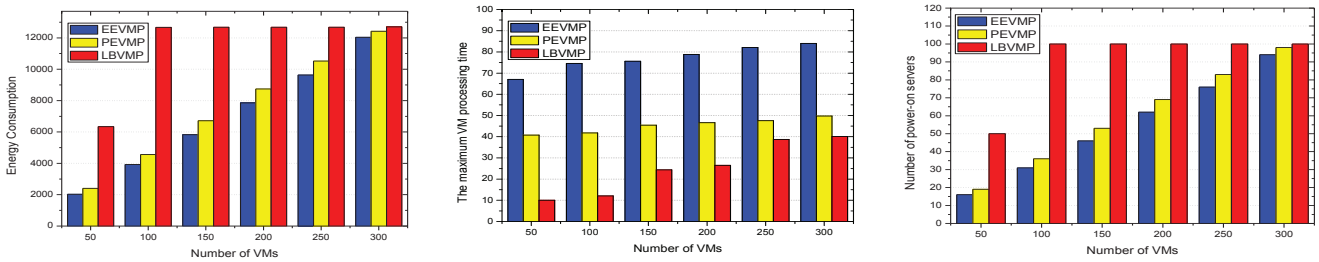
Fig.4 shows simulation results of the PEVMP with variation of $Th(vm_i)$. From Fig.4, we can see that energy consumption and the number of power-on servers gradually decreases with the increase of $Th(vm_i)$, and the maximum VM processing time increases with the increase of $Th(vm_i)$. The reason is that the bigger value of $Th(vm_i)$ means a looser performance requirement. Thus in the PEVMP algorithm, with the increase of $Th(vm_i)$ more VMs can be consolidated onto fewer servers for energy efficiency. When value of $Th(vm_i)$ equals to 20, the performance of the PEVMP is close to that of the LBVMP; when value of $Th(vm_i)$ equals to 80, the performance of the PEVMP is close to that of the EEVMP. So we can set the appropriate value of $Th(vm_i)$ to make the PEVMP get a proportional tradeoff between energy efficiency and performance efficiency.

B. Performance of Online Algorithms

We evaluate the performance of online VM placement algorithms: the P-PEVMP and the C-PEVMP. Different from the P-PEVMP, the C-PEVMP reallocates existing virtual machines in physical servers when a batch of VMs needs to be deployed. We assume multi-branch of VMs dynamically arrives every other 5 time units. Each batch has 50 VMs to be placed onto 100 physical servers. All other parameters are identical with offline scenario. In online system, when processing of each VM is finished, the VM will be deleted from server.

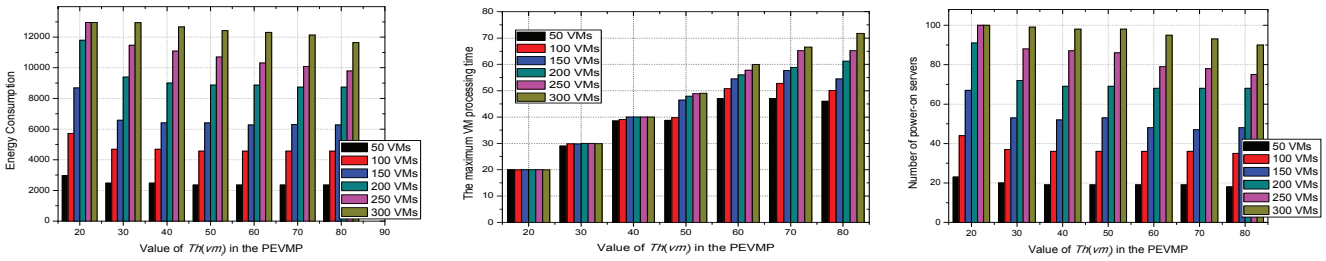
Fig.5 shows energy consumption of two online algorithms when there are totally 20 batches of VMs to be deployed on physical servers. From Fig.5, we can see that the performance gap of two algorithms is not outstanding. Compared to the C-PEVMP, energy consumption of the P-PEVMP only increases 1.43% in average. This is because the C-PEVMP pursues global optimization with reallocating existing VMs on physical servers when new batch of VMs arrives. The P-PEVMP only places new batch of VMs without moving existing VMs on physical servers. The P-PEVMP algorithm can be regarded as a local optimization method.

Table II shows CPU execution time of the C-PEVMP and the P-PEVMP. We do five experiments with total bathes of VMs from 10 to 50. From Table II, we can see that CPU execution time of the C-PEVMP far outweighs that of the P-PEVMP. The reason is that the C-PEVMP allocates all VMs in



(a) Energy consumption (b) Maximum VM processing time (c) Number of power-on servers

Fig. 3. Simulation results of three algorithms



(a) Energy consumption vs $Th(vm_i)$ (b) Maximum VM processing time vs $Th(vm_i)$ (c) Number of power-on servers vs $Th(vm_i)$

Fig. 4. Simulation results of the PEVMP with variation of $Th(vm_i)$

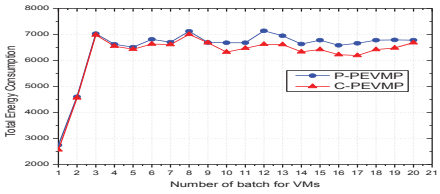


Fig. 5. Simulation results of online algorithms

the servers when new batch of VM arrives, which induces the computation complexity growing greatly. So the C-PEVMP is not suitable for online system.

VII. Conclusion

In this paper, we study on how to optimize energy consumption of physical servers with virtual machine placement. We propose a novel PEVMP algorithm to save energy while guaranteeing the VM execution performance. Simulation results show that the PEVMP can save up to 36.78% energy under the VM processing time constraint. And performance of the corresponding online algorithm is close to that of offline optimization results.

References

[1] Q. Zheng, R. Li, X. Li, et al, "Virtual machine consolidated placement based on multi-objective biogeography-based optimization," Future Generation Computer Systems, vol. 24, no. 1, pp. 95-122, 2016.

TABLE II
CPU execution time of two algorithms (unit: seconds)

batches of VMs	10	20	30	40	50
C-PEVMP	121s	271s	451s	584s	732s
P-PEVMP	5.35s	10.26s	15.64s	20.73s	24.89s

[2] J. Koomey, "Growth in Data Center Electricity use 2005 to 2010," A report by Analytical Press, Completed at the request of The New York Times, 2011.
[3] M.H. Ferdaus, M. Murshed, R.N. Calheiros, R. Buyya, "Virtual machine consolidation in cloud data centers using aco metaheuristic," in: Euro-Par 2014 Parallel Processing, Springer, 2014, pp. 306-317.
[4] M. Melo, P. Maciel, J. Araujo, et al, "Availability study on cloud computing environments: Live migration as a rejuvenation mechanism," in Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2013, pp. 1-6.
[5] M. Wang, X. Meng, L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in: 2011 Proceedings IEEE INFOCOM, IEEE, 2011, pp. 71-75.
[6] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: quantifying effects of shared on-chip resource interference for consolidated vitual machines, " In ACM SOCC, 2011.
[7] N. Guttmann Beck, R. Hassin, "Approximation algorithms for minimum K-cut," Algorithmica, vol. 27, no. 2, pp. 198-207, 2000.
[8] M. Tang and S. Pan, "A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers," Neural Processing Letters, vol.1, no. 1, pp. 1-11, 2014.
[9] L. Jian-ping, X. Li, and C. Min-rong,"Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers," Expert Systems with Applications, vol.4, no. 1, pp. 1-13, 2014.
[10] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in Proceedings of INFOCOM. IEEE, 2010, pp. 1-9.
[11] A. Verma, P. Ahuja, and A. Neogi, "Pmapper: power and migration cost aware application placement in virtualized systems," in Middleware. Springer, 2008, pp. 243-264.
[12] I. Goiri, F. Julia, R. Nou, J. L. Berral, J. Guitart, and J. Torres, "Energyaware scheduling in virtualized datacenters," in Proceedings of IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2010, pp. 58-67.
[13] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," Future Generation Computer Systems, vol. 28, no. 5, pp. 755-768, 2012.
[14] X. Leon, L. Navarro, "Limits of energy saving for the allocation of data center resources to networked applications," in IEEE INFOCOM, 2010.
[15] S. Lim, J. Huh, Y. Kim, G. Shipman, C. Das, "D-factor: a quantitative model of application slow-down in multi-resource shared systems," in Proceedings of the 12th ACM SIGMETRICS, 2012.