2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing

Balancing Resource Utilization for Continuous Virtual Machine Requests in Clouds

Xin Li*, Zhuzhong Qian†, Ruiqing Chi*, Bolei Zhang*, and Sanglu Lu†
State Key Laboratory for Novel Software Technology
Department of Computer Science and Technology, Nanjing University, China
*{lixin,chiruiqing,zhangbolei}@dislab.nju.edu.cn, †{qzz,sanglu}@nju.edu.cn

***Abstract*—The placement of virtual machines (VMs) on a cluster of physical machines (PMs) is a primary task in clouds. We can benefit a lot from appropriate placement policy, e.g. cost saving. In this paper, we raise a continuous virtual machine placement problem and propose an on-line algorithm to reduce the power consumption of clouds. Generic methods which deal with this problem with greedy manner will result in resource leak, a phenomenon of resource wasting. The direct consequence of this phenomenon is larger number of running PMs, which will cause higher power consumption. We prove it is an NP-complete problem to minimize the power consumption during the placement of VMs while the VM requests reach the clouds continuously. We propose a balanced algorithm with the principle of avoiding resource leak. The balanced algorithm handles the VM requests in real time and guarantees high resource utilization. The number of PMs and power consumption can be cut down as a result of high resource utilization. We evaluate the balanced algorithm via experiments with various simulations. Experimental results show that the balanced algorithm can reduce the number of running PMs effectively, so that, it has a significant efficiency on power saving.**

I. Introduction

It is believed that clouds can significantly reduce the IT infrastructure cost and virtualization provides significant benefits in cloud computing. Along with the growing of cloud computing and virtualization, the placement of virtual machines (VMs) on a cluster of physical machines (PMs) becomes a primary task in clouds with virtualization. Meanwhile, it is a highly complex task caused by various constrains, including performance ([3], [24]), scalability ([3], [22]), availability ([20]), network, traffic or bandwidth ([3], [11], [22], [21]), cost and power ([13]), elasticity ([18]), etc.

Among these domains, one important issue for VM placement is power consumed by the running PMs which host the VMs. Generally, power consumption is a significant fraction of the cost for clouds, companies pay millions of dollars for this fraction ([23], [2]). Existing works ([8], [9]) show the importance of VM placement problem and we can benefit from the appropriate placement policy on power saving. Minimizing the number of utilized PMs is helpful to cut down the power consumption drastically ([8]).

Previous works ([20], [3], [21], [22]) are mainly about static VM placement, the set of VMs is given first in the static situation. However, it is not suitable for all cases to give the set of VMs beforehand. We focus on the continuous VM placement problem, the VM is created when the cloud receive a VM request and the VM requests reach the cloud continuously. A VM request means the cloud tenant requests a VM with resource requirements, while each resource requirement consists of three types of resource: CPU, memory, and disk. Tests indicate that the power of a PM is still a high value, close to the peak power, although the workload is low. So, from the point of view of power saving, it is necessary to minimize the number of running PMs while guaranteeing all VM requests could be satisfied in real time.

The VM placement problem can be viewed as an N-dimensional vector bin packing problem which is an NP-complete problem. A simple strategy to deal with continuous VM requests is selecting a PM to host the VM with a greedy manner, it is similar to First-Fit Decrease (FFD). But, the phenomenon called "*resource leak*" may occur due to the nonuniform resource utilization. It is fully loaded for a PM if any type of resource is exhausted, and the PM can not host VM anymore. For a PM, if some type of resource is exhausted while other types of resource are still surplus, the surplus resource is wasted because the PM is fully loaded already. We call this as "*resource leak*". Therefore, it is not an excellent algorithm to adopt greedy manner, resource leak occurs and results in low resource utilization.

Different from the algorithm with greedy manner, we propose a balanced algorithm that selects or starts a PM with the principle of avoiding resource leak while receiving a VM request. High utilization can be achieved by taking the manner of balancing resource utilization. When a VM request reaches, the PM that satisfies the following constrains will be selected to host the new VM.

- The available resource is sufficient for hosting the new VM.
- It will not result in resource leak after hosting the new VM.
- It has the most reasonable resource usage pattern if it hosts the new VM.

If there is no PM was selected because of resource limitation or avoidance of resource leak, a new PM will be started to host the new VM. Compared with the greedy algorithm, the balanced algorithm guarantees higher resource utilization, thereby reducing the number of PMs and cutting down power consumption.

In this paper, we investigate the continuous VM placement problem from the point of view of power saving. The main

978-0-7695-4684-1/12 $26.00 © 2012 IEEE
DOI 10.1109/IMIS.2012.72

contributions of our work in this paper can be summarized as follows:

1) We raise the continuous virtual machine placement problem in clouds.
2) We propose an on-line algorithm with the principle of avoiding resource leak to increase resource utilization.
3) Experiments show our algorithm has a significant efficiency on power saving.

The rest of the paper is organized as follows. Section II reviews related work. Section III introduces the preliminaries. The problem statement is given in Section IV and our algorithm is proposed in Section V. Section VI presents the experimental results. Finally, the paper concludes in Section VII.

II. Related Work

As already noted in Section I, the placement of virtual machines (VMs) on a cluster of physical machines (PMs) is a primary task in clouds under the emerging virtualization technology. Various factors were considered in the process of VM placement. [3] presents a high level overview of VM placement, and proposes a VM placement system architecture design adopting autonomic VM placement to achieve cost saving from better utilization of computing resources. But it lacks of resource management policy.

Network, traffic or bandwidth is the factor that attracts lots of researchers. [21] optimizes the placement of VMs in a traffic-aware way. Traffic patterns among VMs were aligned with the communication distance between them, VMs with mutual bandwidth usage are assigned to PMs in close proximity. [22] takes network condition into account to minimize the data transfer time consumption and maintain the application performance. To achieve better performance, live migration is adopted in their work. Similarly, [11] formulates the VM consolidation with network bandwidth constrain into a stochastic bin packing problem. The optimal solution is given, the number of PMs required is within $(1+\epsilon)(\sqrt{2}+1)$ of the optimum for any $\epsilon > 0$.

Availability is the concerned issue in [20], the authors define a new high-availability property for a VM. As long as there are up to k PM failures, it is guaranteed that a VM can be relocated to a non-failed PM without relocating other VMs, when a VM is marked as *k-resilient*. Application profiling technique was employed is [18] to improve resource utilization in the process of VM placement. There is a tradeoff between application performance and resource usage. The task of VM placement is widely investigated under various constrains and different objectives. However, most works place the VMs in a static manner, which is totally different from continuous VM placement in this paper.

Elastic resource provision is another widely investigated topic in cloud computing. Researcher have done many works on this topic, e.g., [17], [16], [15], [13]. Commercia platform like Amazon EC2 ([1]) also provide resizable compute capacity in the cloud. To achieve elasticity, VM live migration are always used in cloud systems ([22], [26], [27], [12]). In the setting of this paper, VMs hosted in PMs are isolated, and applications are deployed in the VMs with the property of strongly delay sensitive or the applications can't be disrupted absolutely. Therefor, live migration is forbidden for satisfying QoS (Quality of Service) and SLA (Service Level Agreement).

In a static manner, the placement of VMs on PMs is likely to generate a mapping between VMs and PMs. Despite there are various constrains under the placement problem, the basic placement algorithms are based on the same precondition. In this paper, we deal with the problem of continuous VM placement. Different from static VM placement, the VM requests reach the cloud continuously, and a PM should be selected to host the new created VM real time.

In this paper, we deal with continuous VM placement problem, which is quite different from previous static VM placement. The objective of the placement is to minimize the number of PMs for hosting VMs, and then the power consumption can be cut down. We propose an on-line algorithm to minimize power consumption by improving resource utilization.

III. Preliminaries

In this section, we provide some preliminaries of the static VM placement problem and continuous VM placement problem. At the first, we give the problem preconditions and formulated notations, and then the resource constrains are given. Finally, we formalize the static VM placement problem and prove its hardness by reducing 3-dimensional vector bin packing problem.

A. Precondition

Considering a cloud platform that offers the service of providing computational capability. The cloud accepts the VM requests from tenants and allocate VMs with the required resource to them. Tenants deploy their applications on the VMs, and the applications are delay sensitive and can't be disrupted. So, live migration is forbidden to protect the VMs from interruption. It is important for guaranteeing QoS (quality of service) and satisfying SLA (service level agreement). From the viewpoint of infrastructure provider (or cloud owner), it is worthy to minimize the number of running PMs to reduce power consumption, while the continuous VM requests are handled in real time. For each VM request, some PM is selected to host the new requested VM.

In summary, we make the following assumptions:

- The PMs in the cloud are equivalent on compute capacity.
- The VM requests reach the cloud continuously, there is one request in a time slot.
- Live migration is forbidden to ensure QoS and SLA.
- The VMs hosted on the PMs are isolated for safety.
- Network bandwidth inner-cloud is sufficient for network traffics between VMs.
- Each PM or VM consists of three types of resource: CPU, memory, and disk. Here CPU was considered

as usable CPU time slices. The resources of each VM is isolated.

B. Notations

We'll formula related notations about resource type here. As mentioned above, three types of resource are taken into account, and we use C, M and D to refer to *CPU*, *Memory* and *Disk* respectively. Also, an abstract expression R is introduced to refer to any resource type. In brief, it can be formulated as $R \in \{C, M, D\}$.Based on these notations, we define the resource capacity of each PM as $\{C_{cap}, M_{cap}, D_{cap}\}$, which can be stated as R_{cap} in short.

For each PM, the resources are partitioned into two parts: management cost and available resource for VMs. The management cost fraction is used to manage the PM. The available resource means the resource that can be allocated to VMs. The management cost and available resource are represented as R_{cost} and R_{avl} respectively, where R has the same meaning with the previous statement. For example, M_{cost} means the "memory" part of management cost, and D_{avl} means the available disk space at current time.

The same as PM, each VM consists of the three types of resource. As mentioned above, VM requests reach cloud continuously, while each VM request has different resource requirement. We formulate the resource requirement of VM request as R_{req}. For example, the resource requirement of the i^{th} VM request should be represented as $\{C^i_{req}, M^i_{req}, D^i_{req}\}$, however, we usually use the abstract expression R^i_{req} to replace the complete one.

C. Resource Constraint

Essentially, the VM placement is the process of allocating the resource to the VMs. However, the resource is limited for all PMs. So, the placement algorithm must satisfy the following resource constraints.

$$R^j_{cap} = R^j_{cost} + R^j_{avl} \tag{1}$$

$$R^j_{cap} \geq R^j_{cost} + \sum_{i=1}^{n} R^i_{req} \tag{2}$$

In equation (1), R^j_{cap} means the capacity of the j^{th} PM. Similarly, R^j_{cost} and R^j_{avl} refer to the management cost and available resource of the j^{th} PM.

In equation (2), n is the number of VMs hosted on the j^{th} PM. R^i_{req} means the resource requirement of i^{th} VM, while the n VMs were placed on the j^{th} PM.

The equation (1) means that the resource is partitioned into two parts as mentioned in section III-B. Equation (2) implies that the resource provided to the VMs should be no more than the available resources. The two equations are adequate for all PMs.

D. Static VM Placement

Based on the upper notations and resource constraints, let's define the static VM placement problem here.

Static VM Placement. (S-VMP)
Given the following input:

1) *A set of VMs VM with resource requirement $\overrightarrow{VM_i} = \{C^i_{req}, M^i_{req}, D^i_{req}\}$, $1 \leq i \leq n$, where n is the number of VMs.*
2) *A set of PMs PM with resource capacity $\overrightarrow{PM_j} = \{C^j_{cap}, M^j_{cap}, D^j_{cap}\}$, $1 \leq j \leq m$, where m is the number of PMs.*

Subject to:

- $R^j_{cap} = R^j_{cost} + R^j_{avl}$
- $R^j_{cap} \geq R^j_{cost} + \sum_{k=1}^{H_j} R^k_{req}$

Where H_j is the number of VMs hosted in the j^{th} PM.
Output:
Find a feasible placement $\mathbb{P}(V, P)$ that satisfies the resource constrains. $\mathbb{P}(V, P) = \{(i, j) | 1 \leq i \leq n, 1 \leq j \leq m\}$, where n and m are the number of VMs and PMs respectively. The two-tuples (i, j) means that the i^{th} VM will be placed on the j^{th} PM. The objective of the placement is minimizing the number of PMs.

The static VM placement problem can be viewed as a N-dimensional vector bin packing problem. In this paper, three types of resource are taken into account, it is a 3-dimensional vector bin packing problem here. It was known that N-dimensional vector bin packing problem is NP-hard. Like the hardness of bin packing problem, the static VM placement problem is NP-hard too. The computational complexity will be proved next, but let's formulate the 3-dimensional vector bin packing problem first.

3-dimensional Vector Bin Packing Problem. *Given a set $\mathcal{C}$ of n cuboid with width w_j, height h_j, and depth d_j ($j \in J = \{1, 2, ..., n\}$), and an unlimited number of identical three dimensional containers (can be call "bins") having width W, height H, and depth D. Find a partition (packing) of $\mathcal{C}$ into A_1, A_2,..., A_m such that:*

$$\sum_{j=1}^{n} a_{ij} w_j \leq W$$

$$\sum_{j=1}^{n} a_{ij} h_j \leq H$$

$$\sum_{j=1}^{n} a_{ij} d_j \leq D$$

for all A_i, where $i \in I = \{1, 2, ..., m\}$, and

$$a_{ij} = \begin{cases} 1, & \text{if } j^{th} \text{ cuboid is placed in } i^{th} \text{ bin;} \\ 0, & \text{otherwise.} \end{cases}$$

The objective is to minimize the value of 'm', number of bins or partitions (packings).

To find the optimal solution of 3-dimensional vector bin packing problem is known to be NP-hard, then, we'll prove the static VM placement problem is also NP-hard.

Lemma 1. *Static VM placement problem is NP-hard.*

Proof: We reduce the *3-dimensional vector bin packing* problem to static VM placement problem to prove its hardness. Given a set $\mathcal{C}$ of n cuboid with width w_j, height h_j, and depth d_j ($j \in J = \{1, 2, ..., n\}$), and an unlimited number of identical three dimensional bins having width W, height H, and depth D. We map the width, height, and depth into CPU, memory, and disk respectively. So, a cuboid is considered as a VM. We set the value of R_{cost} be a positive constant, then the available resource of PM is viewed as a bin with C_{cap}, M_{cap}, and D_{cap}. Obviously, to find the optimal solution of 3-dimensional vector bin packing problem is equal to minimizing the number of PMs.

The reduction is polynomial. Due the hardness of 3-dimensional vector bin packing problem, we get the static VM placement problem is NP-hard. ■

IV. Problem Statement

In this section, we define the metric of power consumption and then formulate the continuous VM placement problem on power consumption under the setting of VM live migration is forbidden. Then, we prove that continuous VM placement problem is NP-hard.

A. Power Consumption

The objective of our work is cutting down power consumption. For each PM, power consumption is mainly determined by CPU usage. It can be defined as the formulation approximately :

$$power = a * U + b,$$

where U is CPU utilization ratio. Experiments show that b is a very larger value than a. It means that the power consumption is very similar though the CPU utilization varies, so long as the PM is active. So, in static VM placement problem, power consumption is proportional to the number of running PMs, it can be formulated as

$$Power = \alpha * m,$$

where m is the number of running PMs, α is the power consumption factor of a PM in a time slot.

Different from static VM placement, in the scenario of continuous VM placement, the number of PMs varies along with the VM requests reach the cloud continuously. We define the following metric to measure the power consumption in continuous VM placement.

$$Power = \sum_{i=1}^{n} \alpha * m_i$$

where m_i is the number of running PMs when i VM requests have reached the cloud, n is the number of VM requests.

B. Continuous VM Placement

Based on the preliminaries and power consumption metric, we formulate the power consumption problem in continuous VM placement as follows.

Problem Statement. (Continuous VM Placement) *Given a cloud consists of PMs with the same resource capacity R_{cap}. The cloud accepts VM requests with resource requirement R_{req} continuously. Some PM is selected in real time to host the new VM according to some policy, new VM is created based on the accepted resource requirement. Resource constraints are similar to those in static VM placement for each PM.*

$$R_{cap} = R_{cost} + R_{avl}$$

$$R_{avl} \geq \sum_{i=1}^{n} a_{ij} R^i_{req}, \forall j \in J$$

where n is the current number of VMs, and

$$a_{ij} = \begin{cases} 1, & \text{if } i^{th} \text{ VM is placed on } j^{th} \text{ PM;} \\ 0, & \text{otherwise.} \end{cases}$$

where, $j \in J = \{1, 2, ..., m_i\}$, m_i is the number of running PMs when i VM requests have been handled. Power consumption is represented as:

$$Power = \sum_{i=1}^{n} \alpha * m_i$$

The objective is minimizing the value of $Power$.

From the problem statement, power consumption $Power$ is proportional to the number of running PMs when the n (the number of VMs) is given. We'll prove it is an NP-hard problem that minimizing power consumption in continuous VM placement procedure.

Theorem. *Continuous VM placement problem is NP-hard.*

Proof: Suppose that there are $2n$ VM requests totally. then the power consumption can be described as:

$$\begin{aligned} Power &= \alpha m_1 + \alpha m_2 + ... + \alpha m_{2n} \\ &= \alpha(\sum_{k=1}^{n} m_k + \sum_{k=n}^{2n} m_k) \end{aligned}$$

where α is the power consumption factor of a PM, m_k is the number of PMs when k VM requests have been handled.

We prove the theorem by showing that a special case of the problem is NP-hard. Let the resource requirement of the $j^{th}(n \leq j < 2n)$ VM request is equal to the available resource of a PM, it means that $m_{j+1} = m_j + 1, (n \leq j < 2n)$. So, the $Power$ should be represented as:

$$Power^* = \alpha(\sum_{k=1}^{n} m_k + n * m_n + \sum_{i=1}^{n} i)$$

To minimize the value of $Power^*$, it is necessary to minimize the value of m_n. In the above equation, minimized m_n is the

optimal solution of static VM placement while given n VMs. From lemma 1, we know that find the minimal m_n is NP-hard. It indicates that minimizing $Power^*$ is NP-hard.

$Power^*$ is a special case of $Power$, and we have proved it ia an NP-hard problem to minimize $Power^*$, so that minimizing $Power$ is NP-hard. ■

Due to the hardness of the problem, heuristic algorithms are proposed in the next section.

V. Our Algorithm

In this section, we'll propose our algorithm to solve the problem of minimizing power consumption during continuous VM placement. Due to its hardness as proved in section IV-B, two heuristic algorithms are proposed. When the cloud accept a VM request with resource requirement R_{req}, a new VM will be created on a PM in real time. There can be various policies to select the PM to host the new VM. We propose two algorithms, *greedy algorithm* and *balanced algorithm*, according to two different policies. The greedy algorithm is firstly introduced, which can give the local optimum solution, while it may be not a good solution for long-term. *Resource leak* occurs in the placement process in greedy manner, then a balanced algorithm is proposed to avoid resource leak.

A. Greedy Algorithm

We propose a locally optimal algorithm in greedy manner to place the VM as shown in Algorithm 1. When a VM request reaches the cloud, the firstly scanned one with sufficient available resource to host the VM will be selected. To avoid unnecessary scanning, the fully loaded PMs should not be scanned. So we introduce two states for each PM, fully loaded and standby. For a PM, if the resource utilization is greater than a threshold for every type of resource, we think the PM can not host new VM any more, the PM will be added to the fully loaded list. we'll just scan the PMs in standby list.

While the cloud accepts a VM request with resource requirement R_{req}, the set P_{std} consists of standby PMs will be scanned. If the i^{th} PM's available resource $P^i_{std}.R_{avl}$ is sufficient to host the new VM, the PM will be selected and the new VM will be created on the PM. For the selected PM, resource utilization μ_R changed after hosting the new VM; if the new value of μ_R is greater than the threshold θ_{full}, the PM has been in fully loaded state and will be added to the fully loaded set P_{full} and be deleted from the standby set P_{std}.

If there is no PM can satisfy the VM request after scanning all of the $P_{std}.size$ PMs in P_{std}, a new PM P_{new} will be started and the new VM will be created on the new PM. Similarly, the resource utilization μ_R is updated and the new started PM will be categorized as fully loaded PM or standby PM according to the resource utilization.

So, the power consumption $Power$ in the next time slot is determined by the current number ($P_{all}.size$) of all PMs P_{all} including fully loaded PMs and standby PMs. For each PM, the power consumption in a time slot is α. Power consumption $Power$ accumulates at each time slot while handling a VM request.

Algorithm 1 Greedy Algorithm

Input: Continuous VM requests REQ (one at each time slot) with resource requirement R_{req}.

Output: Power consumption $Power$ during the process of continuous VM placement.

```
1:  Power = 0;
2:  while REQ != NULL do
3:    selected = false;
4:    for i = 1; i ≤ P_std.size; i = i + 1 do
5:      if P^i_std.R_avl ≥ R_req then
6:        selected = true;
7:        create a new VM on P^i_std;
8:        update μ_R of P^i_std;
9:        if μ_R ≥ θ_full then
10:         P_full = P_full ∪ {P^i_std};
11:         P_std = P_std − {P^i_std};
12:       end if
13:       break;
14:     end if
15:   end for
16:   if selected = false then
17:     start a new PM P_new;
18:     P_all = P_all ∪ {P_new};
19:     create a new VM on P_new;
20:     update μ_R of P_new;
21:     if μ_R ≥ θ_full then
22:       P_full = P_full ∪ {P_new};
23:     else
24:       P_std = P_std ∪ {P_new};
25:     end if
26:   end if
27:   Power = Power + α * P_all.size;
28: end while
29: return Power;
```

B. Balanced Algorithm

Due to low resource utilization caused by resource leak in greedy algorithm, we propose a balanced algorithm (shown in Algorithm 2) to increase resource utilization by avoiding resource leak. There are four *potential* states for each PM while handling a VM request, listed as follows:

1) ***Resource Limited*** The available resource of the PM is not sufficient to create a new VM with the resource requirement R_{req}.
2) ***Fully Loaded*** The PM will be fully loaded after satisfying the VM request.
3) ***Resource-Leak*** The resource utilization is abnormal which will result in resource leak if this PM host the new created VM.
4) ***Standby*** The resource usage will be still regular when the new VM is created on this PM.

Here, we give the quantitative definition of *resource leak*. A PM is in resource-leak state if there exists such R that ($\mu_R > \theta_\Lambda$) and ($|\max\{\mu_R\} - \min\{\mu_R\}| \geq \theta_\Delta$), where μ_R

Algorithm 2 Balanced Algorithm

Input: Continuous VM requests REQ (one at each time slot) with resource requirement R_{req}.

Output: Power consumption $Power$ during the process of continuous VM placement.

```
1: Power = 0;
2: while REQ != NULL do
3:    f = 0,   mu*_R = 0;
4:    s = 0,   sigma(mu*_R) = infinity;
5:    for i = 1; i <= P_std.size; i = i + 1 do
6:       if P^i_std.R_avl >= R_req then
7:          if mu'_R >= theta_full then
8:             if mu'_R > mu*_R then
9:                mu*_R = mu'_R;
10:               f = i;
11:            end if
12:         end if
13:         if j = 0 then
14:            if (exists R, mu'_R > theta_Lambda) &&
               (|max{mu'_R} - min{mu'_R}| >= theta_Delta) then
15:               continue;
16:            else if sigma(mu'_R) < sigma(mu*_R) then
17:               sigma(mu*_R) = sigma(mu'_R);
18:               s = i;
19:            end if
20:         end if
21:      end if
22:   end for
23:   if j > 0 then
24:      create a new VM on P^j_std;
25:      P_full = P_full U {P^i_std};
26:      P_std = P_std - {P^i_std};
27:   else if s > 0 then
28:      create a new VM on P^s_std;
29:   else
30:      start a new PM P_new;
31:      P_all = P_all U {P_new};
32:      create a new VM on P_new;
33:      update mu_R of P_new;
34:      if mu_R >= theta_full then
35:         P_full = P_full U {P_new};
36:      else
37:         P_std = P_std U {P_new};
38:      end if
39:   end if
40:   Power = Power + alpha * P_all.size;
41: end while
42: return Power;
```

refers to the resource utilization of the PM, θ_Λ and θ_Δ are two thresholds to quantify the occasion when the resource leak occurs, $\max\{\mu_R\}$ and $\min\{\mu_R\}$ are abstract expression of $\max\{\mu_R|R \in \{C, M, D\}\}$ and $\min\{\mu_R|R \in \{C, M, D\}\}$ respectively. In short, if there is a large gap (θ_Δ) between some two resource utilization, while one of them seems not sufficient to satisfy another VM request. That situation results in resource leak.

Different from greedy algorithm, each PM in PM_{std} is scanned and categorized into four cases, as the listed four *potential* states. It is firstly determined whether the available resource $P^i_{std}.R_{avl}$ is sufficient to create the requested VM, if not, the PM is resource limited and fall into case[*resource limited*]. If the PM can host the new VM, the algorithm makes a judgement that to which case the PM will belong according to μ'_R, which refers to the subsequent resource utilization if the PM allocate resource to the requested VM. As shown in Algorithm 1, if μ'_R is greater than threshold θ_{full}, the PM will be classified as case[*fully loaded*]. The algorithm skips the determination of case[*resource-leak*] and case[*standby*] if there is already some PM resided in case[*fully loaded*]. If there is still no PM in case[*fully loaded*], the algorithm judges whether the placement will cause resource leak. According to the definition of resource leak, the PM will be categorized into case[*resource leak*] or case[*standby*].

If the number of PMs in case[*fully loaded*] is not zero, the one with maximum mean-value $\overline{\mu'_R}$ of subsequent resource utilization will be selected as resource provider. Otherwise, if there is no PM in case[*fully loaded*], the one with minimum variance $\sigma(\mu'_R)$ in case[*standby*] will be selected to provide the required resource. A new PM will be started up if there's no PM in case[*fully loaded*] and case[*standby*].

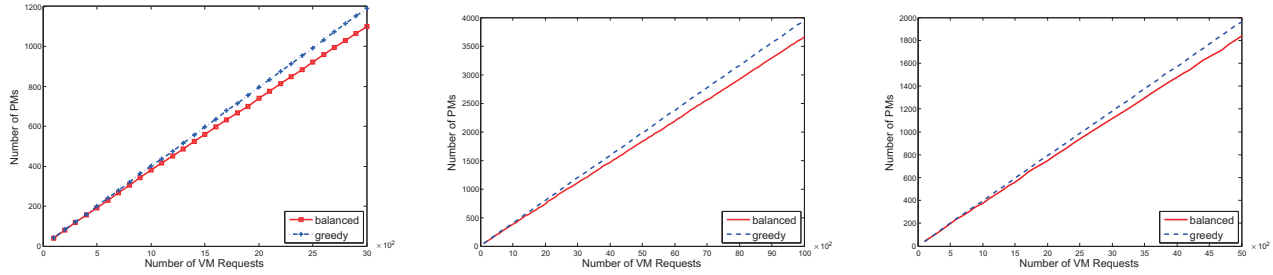
The details are shown in Algorithm 2, two notations f and s are explained as follows:

- f record the position of the PM with maximum $\overline{\mu'_R}$ in P_{std}, while the variable $\overline{\mu^*_R}$ is used to record the current maximum value of $\overline{\mu'_R}$, where $\overline{\mu'_R}$ refers to the mean value of $\{\mu'_R|R \in \{C, M, D\}\}$.
- s record the position of the PM with minimum $\sigma(\mu'_R)$ in P_{std}, while the variable $\sigma(\mu^*_R)$ is used to record the current minimum value of $\sigma(\mu'_R)$, where $\sigma(\mu'_R)$ means the variance of $\{\mu'_R|R \in \{C, M, D\}\}$.

It is noticeable that if $f > 0$, the value of s is useless. Other notations like P_{all}, P_{full}, P_{std}, P_{new} and $Power$ have the same meaning with their appearance in Algorithm 1. The key point of balanced algorithm different from greedy algorithm is the attitude of regarding resource utilization μ_R. The balanced algorithm guarantees higher resource utilization by starting up a new PM at the right moment to avoid resource leak.

VI. Experimental Evaluation

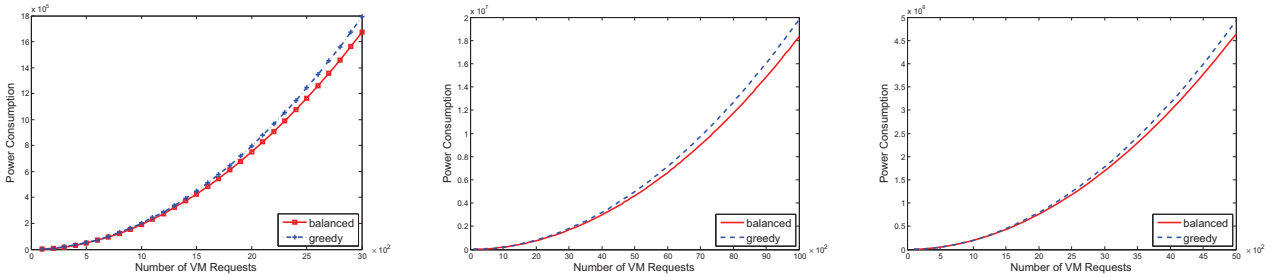
In this section, we use simulation experiments to evaluate the performance of our algorithm. Greedy algorithm acts as a benchmark, we compare the balanced algorithm with it. We assume the VM requests reach the cloud is regular and there's one request at each time slot. Actually, the distribution of request arrival has no influence on resource allocation procedure and potential resource leak. The management cost R_{cost} of PM is not considered in the simulation, it is easy to understand that the cost is assumed to be a constant value for the PMs with same capacity, all of the resource can be



(a) number of PMs when the distribution of resource requirement is uniform distribution, and $REQ.size = 3000, \min. = 25, \max. = 75$.

(b) number of PMs when the distribution of resource requirement is uniform distribution, and $REQ.size = 10000, \min. = 20, \max. = 80$.

(c) number of PMs when the distribution of resource requirement is uniform distribution, and $REQ.size = 5000, \min. = 15, \max. = 85$.



(d) power consumption when the distribution of resource requirement is uniform distribution, and $REQ.size = 3000, \min. = 25, \max. = 75$.

(e) power consumption when the distribution of resource requirement is uniform distribution, and $REQ.size = 10000, \min. = 20, \max. = 80$.

(f) power consumption when the distribution of resource requirement is uniform distribution, and $REQ.size = 5000, \min. = 15, \max. = 85$.

Fig. 1. Simulation results when the distribution of resource requirement is uniform distribution. The top row shows the number of PMs with different request size and range, while the bottom row shows the related power consumption with the same setting.

allocated to the VMs. Three types of resource are taken into account: CPU, memory and disk.

We focus on the two questions: (1)How the number of the PMs varies when the VM requests reach the cloud continuously? (2)How is the power consumption varies under the same setting? So two measurements are used to measure the performance of the algorithms: number of PMs and power consumption.

A. Simulation Setup

In the simulations, the three types of resource are independent to each other. We consider two kinds of distribution of the required resource value: uniform distribution and normal distribution. Three data sets with different parameters are investigated for uniform distribution, and there is one data set for normal distribution. The cases are shown in Table I. For each PM, the value of available resource on R is 150.

For each of the four data sets, there are many instances with the same distribution and parameter setting. The experimental result refers to the average value for each data set.

B. Experimental Results

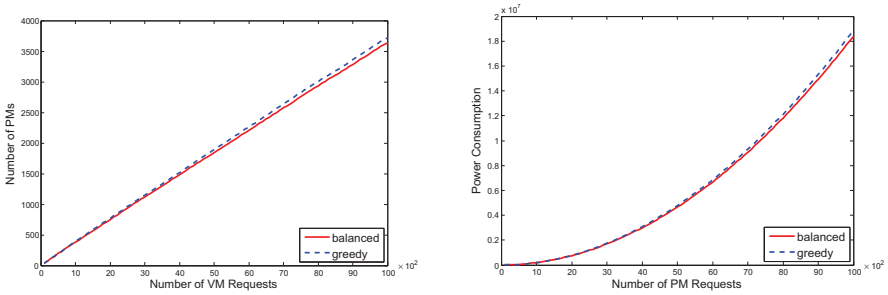
Figure 1 shows the results of number of PMs and power consumption when the resource requirement obeys uniform distribution with three different parameters. The horizontal axis of each sub-figure is the number of VM requests, the top line shows the results about the number of PMs, while the results about power consumption are shown by the bottom line.

TABLE I
DISTRIBUTIONS OF DATA SETS

Data Set	Distribution	Sample Size
DS_1	*uniform distribution* $\min. = 25$ $\max. = 75$	3,000
DS_2	*uniform distribution* $\min. = 20$ $\max. = 80$	10,000
DS_3	*uniform distribution* $\min. = 15$ $\max. = 85$	5,000
DS_4	*normal distribution* $\mu = 50$ $\sigma = 12$	10,000

It is obviously from the figures that the balanced algorithm (red line) performances better than greedy algorithm (blue line) in minimizing the number of PMs and power consumption. The slower increase means that it needs lower power to satisfy the continuous VM requests. We find that nearly 10% power consumption can be saved in some cases.

For the case of normal distribution, Figure 2 illustrates the results. Because of the concentricity of resource requirement, there is less resource leak occurrence than uniform distribution. However, there are still considerable power saving when the number of VM requests is large and it still have an active effect on cost saving when we must pay millions of dollars on power consumption.



(a) number of PMs when the distribution of resource requirement is normal distribution, and $REQ.size = 10000, \mu = 50, \sigma = 12$.

(b) power consumption when the distribution of resource requirement is normal distribution, and $REQ.size = 10000, \mu = 50, \sigma = 12$.

Fig. 2. Simulation results when the distribution of resource requirement is normal distribution. The upper one shows the number of PMs, while the lower one shows the related power consumption with the same setting.

VII. Conclusion

In this paper, we investigate the continuous virtual machine placement problem from the viewpoint of power saving. We distinguish it from static virtual machine placement, and formulate the power consumption for continuous placement process. It is proved to be an NP-hard problem to minimize the power consumption in the cloud. We discovered the phenomenon called *resource leak* which cause wasting of resource and increasing of power consumption. A balanced algorithm is proposed to avoid resource leak and increase resource utilization. The power consumption can be cut down due to the reduced number of physical machine. Experimental results shows that the balanced algorithm performs well and it can reduce the power consumption significantly.

Acknowledgment

This work is partially supported by the National Natural Foundation of China under Grant No. 61073028, 61021062; the National Basic Research Program of China (973) under Grant No. 2009CB320705; the Key Technology Research and Development Program of Jiangsu under Grant No. BE2010179; Jiangsu Natural Science Foundation under Grant No. BK2011510.

References

[1] Amazon EC2, http://aws.amazon.com/ec2/.
[2] J. Hamilton, "Cost of Power in Large-scale Data Centers," http://perspectivvs.mvdirona.com/, Nov. 2009.
[3] C. Hyser, B. McKee, R. Gardner, and B. J. Watson, "Autonomic Virtual Machine Placement in the Data Center," *HP Labs Technical Report*, February 2008.
[4] M. R. Garey, R. L. Graham and D. S. Johnson, "Resource Constrained Scheduling as Generalized Bin Packing," *Journal of Combinatorical Theory*, vol. 21, no. 3, pp. 257-298, 1976.
[5] M. R. Garey and V. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," *W.H. Freeman and Co.* 1976.
[6] W. F. de la Vega and G. S. Lueker, "Bin Packing can be Solved with $1+\epsilon$ in Linear Time," *Combinatorica*, vol. 1, no. 4, pp. 349-355, 1981.
[7] G. J.Woeginger, "There is no asymptotic PTAS for two-dimensional vector packing," *Information Processing Letters*, vol. 64, no. 6, pp. 293-297, 1997.
[8] R. Bianchini and R. Rajamony, "Power and Energy Management for Server Systems," *IEEE Computer*, vol. 37, 2004.
[9] W. Vogels, "Beyond Server Consolidation," *ACM QUEUE*, vol. 6, no. 1, pp. 20-26, 2008.
[10] S. Martello, D. Pisinger, and D. Vigo, "The Three-Dimensional Bin Packing Problem," *INFORMS*, vol. 48, no. 2, pp. 256-267, March-April 2000.
[11] M. Wang, X. Meng, and L. Zhang, "Consolidating Virtual Machines with Dynamic Bandwidth Demand in Data Center," *In Porc. of INFOCOM 2011 (Mini-Conference)*.
[12] H. Liu, H. Jin, X. Liao, C. Yu, and C.-Z. Xu, "Live Virtual Machine Migration via Asynchronous Replication and State Synchronization," *IEEE TPDS*, vol. 3, no. 22, pp. 426-438, 2011.
[13] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A Cost-aware Elasticity Provisioning System for the Cloud," *In Proc. of ICDCS 2011*.
[14] Y.-H. Han, C.-M. Kim, and J.-M. Gil, "A Greedy Algorithm for Target Coverage Scheduling in Directional Sensor Networks," *JoWUA*, vol. 1, no. 2/3, pp. 96-106, 2010.
[15] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Kingfisher: Cost-aware Elasticity in the Cloud," *In Proc. of INFOCOM 2011 (Mini-conference)*.
[16] Z. Gong, X. Gu, and J. Wilkes, "PRESS: PRedictive Elastic ReSource Scaling for cloud systems," *In Proc. of CNSM 2010*.
[17] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: Elastic Resource Scaling for Multi-Tenant Cloud Systems," *In Proc. of SOCC 2011*.
[18] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Y. Zomaya, and B. B. Zhou, "Profiling Applications for Virtual Machine Placement in Clouds," *In Proc. of Cloud Computing 2011*.
[19] J. Rao, X. Bu, K. Wang, and C.-Z. Xu, "Selt-adaptive Provisioning of Virtualized Resource in Cloud Computinng," *In proc. of SIGMETRICS 2011*.
[20] E. Bin, O. Biran, O. Boni, E. Hadad, E. K. Kolodner, Y. Moatti, and D. H. Lorenz, "Guaranteeing High Availability Goals for Virtual Machine Placement," *In Proc. of ICDCS 2011*.
[21] X. Meng, V. Pappas, and L. Zhang, "Improving the Scalability of Date Center Networks with Traffic-aware Virtual Machine Placement," *In Proc. of INFOCOM 2010*.
[22] J. T. Piao, and J. Yan, "A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing," *In Proc. of GCC 2010*.
[23] M. Lin, A. Wierman, L. H. Andrew, and E. Thereska, "Dynamic Right-sizing for Power-proportional Data Center," *In Proc. of INFOCOM 2011*.
[24] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environments via Lookahead Control," *Cluster Computing*, Vol. 12, pp. 1-15, 2009.
[25] Y. Shiraishi, M. Mohri, and Y. Fukuta, "A Server-Aided Computation Protocol Revisited for Confidentiality of Cloud Service," *JoWUA*, vol. 2, no. 2, pp. 83-94, 2011.
[26] V. Shrivastava, P. Zerfos, K. Lee, H. Jamjoom, Y-H Liu, and S. Banerjee, "Application-aware Virtual Machine Migration in Data Centers," *In Proc. of INFOCOM 2011 (Mini-conference)*.
[27] W. Voorsluys, J. Broberg, S. Venugopal and R. Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," *In Proc. of CloudCom 2009*.