Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 613719, 11 pages
http://dx.doi.org/10.1155/2014/613719

Research Article

Dynamic Placement of Virtual Machines with Both Deterministic and Stochastic Demands for Green Cloud Computing

Wenying Yue and Qiushuang Chen

College of Computer and Control Engineering, Nankai University, Tianjin 300071, China

Correspondence should be addressed to Qiushuang Chen; chenqs@nankai.edu.cn

Received 30 April 2014; Accepted 17 June 2014; Published 7 July 2014

Academic Editor: Yoshinori Hayafuji

Cloud computing has come to be a significant commercial infrastructure offering utility-oriented IT services to users worldwide. However, data centers hosting cloud applications consume huge amounts of energy, leading to high operational cost and greenhouse gas emission. Therefore, green cloud computing solutions are needed not only to achieve high level service performance but also to minimize energy consumption. This paper studies the dynamic placement of virtual machines (VMs) with deterministic and stochastic demands. In order to ensure a quick response to VM requests and improve the energy efficiency, a two-phase optimization strategy has been proposed, in which VMs are deployed in runtime and consolidated into servers periodically. Based on an improved multidimensional space partition model, a modified energy efficient algorithm with balanced resource utilization (MEAGLE) and a live migration algorithm based on the basic set (LMABBS) are, respectively, developed for each phase. Experimental results have shown that under different VMs' stochastic demand variations, MEAGLE guarantees the availability of stochastic resources with a defined probability and reduces the number of required servers by 2.49% to 20.40% compared with the benchmark algorithms. Also, the difference between the LMABBS solution and Gurobi solution is fairly small, but LMABBS significantly excels in computational efficiency.

1. Introduction

Cloud computing is a new paradigm for the dynamic provisioning of computing services supported by state-of-the-art data centers. With the rapid development of cloud computing, the data centers are growing at an unprecedented rate. However, an average data center consumes as much energy as 25,000 households [1] and it was estimated that the energy consumption from data centers would have accounted for about two percent of the worldwide energy consumption by 2020 [2]. High power consumption not only produces huge operational cost for data centers, but also significantly contributes to the growing environmental issues of global warming. Therefore, green cloud computing is driven to achieve high level service performance and to minimize energy consumption.

Virtual machines are attractive to modern cloud data centers because they can significantly promote the efficiency and flexibility of resource management helping to cut back the power consumptions [3–5]. However, such an incentive highly relies on a well-designed VM placement scheme [6, 7]. When multiple VMs run on the same physical host or rack, they share all the physical resources. According to how resource-intensive applications are, users may customize different types of VMs such as CPU-intensive, memory-intensive, and bandwidth-intensive VMs. If all VMs in a cloud data center have the same intensive resource, VMP is then reduced to the classical one-dimensional VM placement problem. But a cloud data center often has VMs with different resource intensities [8], and, thus, an efficient multidimensional algorithm is required for the VMs placement.

In cloud data centers the VM placement problem (VMP) has been tackled in both research work [7, 9–17] and commercial products [18, 19]. In most of the research work, VMP is formulated as a multidimensional bin packing problem, aiming to minimize the number of required servers and

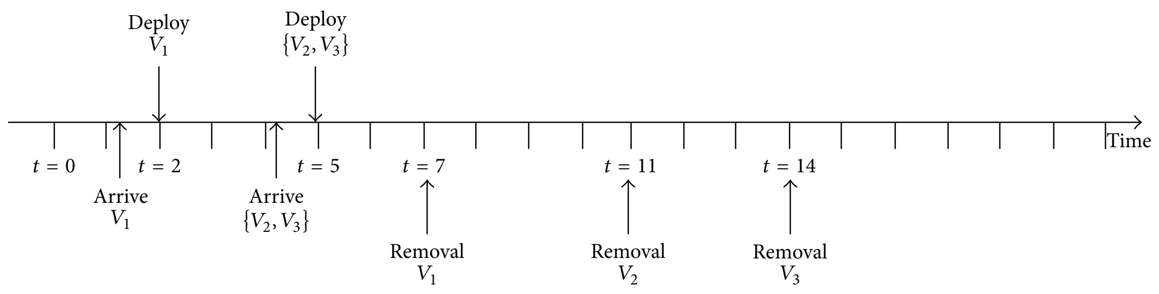


Figure 1: The deployment and removal of VMs.

with constraints from various management fields [4], such as the resource demands of VMs, security/isolation policies (e.g., ensuring that certain VMs will not run on the same server), and compatibility/location policies (e.g., restrict the placement of certain VMs to certain servers). Variants of the classic packing algorithms such as First Fit (FF) and First Fit Decreasing (FFD) [20] are often used to solve VMP. In particular, Li et al. [17] proposed an energy efficient algorithm with balanced resource utilization. However, in these studies, the VMs' demands are assumed to be deterministic for all resources and these algorithms are called deterministic resource algorithms.

Recent studies [11, 21–23] indicate that VMs' demands for some resources like network bandwidth are stochastic and highly volatile. Regarding the placement of VMs with stochastic demands, the deterministic resource algorithm can be adopted by using the mean or maximum of the demand as its estimated value. But this naive approach may lead to either poor user experience or resource waste. Wang et al. [24] and Breitgand and Epstein [25] assumed the VMs' bandwidth demand to be normally distributed. They described VMP as a stochastic bin packing problem and used improved bin packing heuristics to solve it. Nevertheless, only the network resources with stochastic demands were considered in their study while system resources with deterministic demands were not included.

In reality, the VMs' demands for all resources have different characteristics. For some resources, such as CPU and memory, VMs' demands are constant or have a very small fluctuation and, thus, can be considered as deterministic. For other resources, such as network bandwidth, VMs' demands are time varying and have high volatility, and, thus, can be considered as stochastic. Yet, little work has focused on the placement of VMs with both deterministic and stochastic demands. Jin et al. [26] considered both kinds of demands and proposed a polynomial time algorithm to solve the multidimensional stochastic VM placement problem. In their work, the VM deployment requests were given beforehand and the VMs were placed in a static manner. Practically, VM requests dynamically arrive at the cloud data center.

Taking into account VMs' deterministic and stochastic demands, this paper studies the VMP problem in a practical dynamic cloud environment. The paper is organized as follows. Section 2 formulates the instant-deployment problem for VMs and the periodic-consolidation problem for servers. Section 3 presents an improved multidimensional space partition model. Section 4 describes the execution procedures of the proposed two-phase optimization strategy, and two developed algorithms, respectively, for VM deployment and server consolidation based on the model in the previous section. In Section 5, the algorithms are validated by simulation experiments. Section 6 concludes the paper with a summary of results and future work directions.

2. Problem Formulation

Assume that servers are homogeneous in a cloud data center and there are sufficient server resources to meet all the VM requests. For VMs' demands, the server resources fall into two types: deterministic resources and stochastic resources. Denote D the set of deterministic resources and Q the set of stochastic resources. Servers should supply fixed-value deterministic resources required by VMs and provide an availability guarantee of stochastic resources with probability at least $1 - \alpha$, where α is called target overflow probability specified in the service level agreements (SLAs).

Let the time-slots be of equal length Δt. The deployment and removal of VMs only occur at each time instant [27]. As shown in Figure 1, VMs, on arrival, will be placed at the next time instant. In order to ensure high responsiveness, Δt needs to be properly chosen.

For a cloud provider, quick response to the incoming VM requests is crucial for a high service quality. Thus, an instant-deployment strategy is adopted for VMs. However, VMs are being continuously removed over time, as shown in Figure 1. Over a long time period, the infrastructure may be brought to a poor state (e.g., there are only a few VMs in some servers). Therefore, it is necessary to implement server consolidation periodically.

2.1. The Instant-Deployment for VMs.

2.1. The Instant-Deployment for VMs. A newly arriving VM is denoted by v_k. The strategy is to select a suitable physical machine (PM) to deploy v_k in runtime. VM migration is not involved. Considering VMs' deterministic and stochastic demands, a placement scheme is regarded as feasible only if it satisfies the following conditions: (1) for each deterministic resource of a server, the server's capacity should not be exceeded by the total demand of all the VMs on the server; (2) for each stochastic resource of a server, the probability that the server's capacity is exceeded is no larger than α.

Denote VM v_i's demand for a deterministic resource r as d_i^r and server h's corresponding capacity as C_h^r.

Condition (1) can be represented as

$$d_k^r + \sum_{i \in U_h} d_i^r \le C_h^r, \quad \forall r \in D. \tag{1}$$

Condition (2) can be represented as

$$\Pr\left\{d_k^r + \sum_{i \in U_h} d_i^r \le C_h^r\right\} \ge 1 - \alpha, \quad \forall r \in Q \tag{2}$$

U_h is the set of VMs running on server h.

The VM instant-deployment problem is to find a VM placement scheme for a newly arriving VM v_k, which satisfies the above two conditions and minimizes the number of required servers.

2.2. The Periodic-Consolidation for Servers. Server consolidation refers to gathering VMs into as few servers as possible and then turning off the idle servers or putting them in a low energy mode. To formulate the strategy, a stochastic chance-constrained programming model is presented.

Firstly, the notations used are listed in the following.

(i) Parameters:

t: the time instant to trigger server consolidation.

$V(t)$: the set of VMs which have been deployed and will not be removed at time instant t.

$P(t)$: the set of PMs which are running at time instant t, $P(t) = \{1, 2, \dots, h, \dots, n_p(t)\}$.

$p_i(t)$: the PM on which VM v_i is running at time instant t, $i \in V(t)$, $p_i(t) \in P(t)$.

N: the total number of PMs in a cloud data center.

(ii) Decision variables:

the three variables below define the status of PMs and VMs after the server consolidation.

$y_{ih}(t) \in \{0, 1\}$: is equal to 1 if VM i is mapped to PM h, 0 otherwise.

$o_h(t) \in \{0, 1\}$: is equal to 1 if PM h is used, 0 otherwise.

$m_i(t) \in \{0, 1\}$: is equal to 1 if VM i is migrated to another PM (namely, $\sum_{h \in P(t)} h \cdot y_{ih}(t) \ne p_i(t)$), 0 otherwise.

The most effective way to reduce power consumption is to decrease the total number of required servers. Given the overhead of live migration, the objective is to minimize the number of migrations, in addition to the number of required servers.

Thus, the server consolidation problem for the VM set $V(t)$ can be formulated as

$$(\text{Model\_0}) \min f_1 = \sum_{h \in P(t)} o_h(t) \tag{3}$$

$$\min f_2 = \sum_{i \in V(t)} m_i(t) \tag{4}$$

subject to

$$o_h(t) \le \sum_{i \in V(t)} y_{ih}(t), \quad \forall h \in P(t) \tag{5}$$

$$o_h(t) \ge y_{ih}(t), \quad \forall i \in V(t), \ \forall h \in P(t) \tag{6}$$

$$\begin{aligned} -m_i(t) \cdot N \le p_i(t) - \sum_{h \in P(t)} h \cdot y_{ih}(t) \\ \le m_i(t) \cdot N, \quad \forall i \in V(t) \end{aligned} \tag{7}$$

$$\sum_{h \in P(t)} y_{ih}(t) = 1, \quad \forall i \in V(t) \tag{8}$$

$$\sum_{i \in V(t)} y_{ih}(t) \cdot d_i^r \le C_h^r, \quad \forall h \in P(t), \ r \in D \tag{9}$$

$$\Pr\left\{\sum_{i \in V(t)} y_{ih}(t) \cdot d_i^r \le C_h^r\right\} \ge 1 - \alpha, \quad \forall h \in P(t), \ r \in Q \tag{10}$$

$$y_{ih}(t) \in \{0, 1\}, \quad \forall i \in V(t), \ \forall h \in P(t) \tag{11}$$

$$o_h(t) \in \{0, 1\}, \ \forall h \in P(t) \tag{12}$$

$$m_i(t) \in \{0, 1\}, \ \forall i \in V(t). \tag{13}$$

In (3), f_1 defines the number of required servers after the server consolidation. In (4), f_2 defines the number of migrations needed during the server consolidation. Constraints (5)–(7) define the implicit relations between the decision variables. Constraint (8) states each VM ($i \in V(t)$) is placed on a single server. Constraint (9) states that for each deterministic resource of a server, the total demand of all the VMs on the server must not exceed the server's capacity. Constraint (10) states that for each stochastic resource of a server, the demands of all the VMs on the server should be satisfied with the given probability $1-\alpha$. Constraints (11)–(13) define the domain of the variables.

3. An Improved Multidimensional Space Partition Model

3.1. The Model for Deterministic Resources. Most existing algorithms (such as Next Fit and First Fit) simply perform the placement computation by comparing the VM's demand with the server's available capacity, without considering the balanced utilization of multidimensional resources. However, as long as an arbitrary dimensional resource of a server is exhausted, even when all other dimensional resources are sufficient, no more VMs can be placed on the server. In order to avoid this and reduce resource waste, Li et al. [17] proposed a multidimensional space partition model and based on the model presented an **e**nergy efficient **alg**orithm with ba**l**anced r**e**source utilization (EAGLE for short). Figure 2 shows the two-dimensional space partition model. In the figure, the vertical and horizontal axes, respectively, stand for the utilization ratios of resource 1 and resource 2 in server h; the point $\mathbf{E}(1.0, 1.0)$ indicates the two types of resources are exhausted, while the point $\mathbf{O}(0, 0)$ means server h is idle.

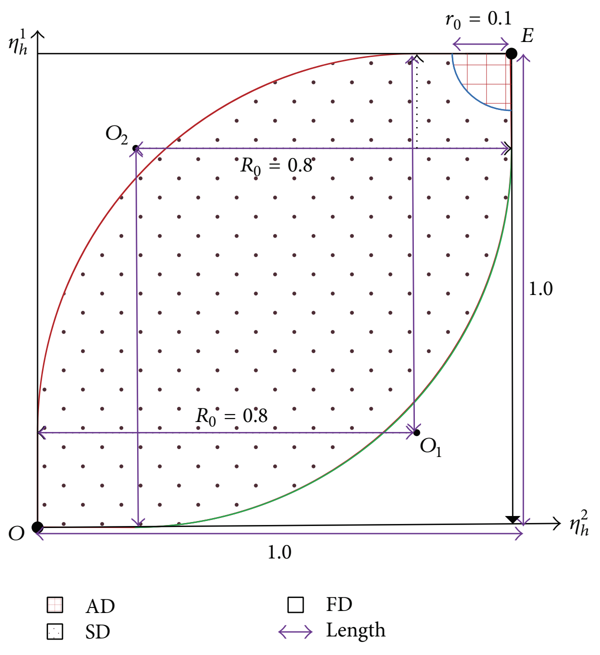


Figure 2: The multidimensional space partition model [17].

In the partition model, there are three domains: the AD domain which implies all resources are nearly exhausted, the FD domain which indicates there is an obvious disequilibrium of multidimensional resource utilization, and the SD domain which represents the balanced case. As shown in Figure 2, the unit square is partitioned into three parts by three quarter circles. The AD domain is one quarter circle with center $\mathbf{E}$ and radius r_0 ($r_0 \in [0, 1.0]$). The FD domain is divided into two subparts by two symmetrical quarter circles, with the same radius R_0 ($R_0 \in [0, 1.0]$). The point $\mathbf{O}_1$ is the center of the red quarter circle and the point $\mathbf{O}_2$ is the center of the green quarter circle. The SD domain is bounded by the three quarter circles and square edges.

Generally, in the M-dimensional space partition model, one point is corresponded to each usage state, expressed as $\mathbf{US_h} = (\eta_h^1, \eta_h^2, \ldots, \eta_h^r, \ldots, \eta_h^M)$. η_h^r represents the utilization ratio of resource r in server h. Given $\mathbf{US_h}$, the parameters r_0 and R_0, the domain determination function $\mathbf{f}$ can be defined as

$$\mathbf{f}(\mathbf{US_h}) = \begin{cases} \text{AD}, & \text{if distance}(\mathbf{US_h}, \mathbf{E}) \le r_0, \\ \text{SD}, & \text{else if } (\forall r, \text{ distance}(\mathbf{US_h}, \mathbf{O_r}) \le R_0) \parallel \\ & \quad (\forall r, \eta_h^r \le 1 - R_0) \parallel (\forall r, \eta_h^r \ge R_0), \\ \text{FD}, & \text{otherwise} \end{cases} \tag{14}$$

distance$(\mathbf{US_h}, \mathbf{O_r})$ means the Euclidean distance between the two points $\mathbf{US_h}$ and $\mathbf{O_r}$. The point $\mathbf{O_r}$ is represented as $(x_1, x_2, \ldots x_k, \ldots, x_M)$, where

$$x_k = \begin{cases} 1 - R_0, & \text{if } k = r; \\ R_0, & \text{otherwise.} \end{cases} \tag{15}$$

Denote v_k as a VM to be deployed. According to EAGLE in [17], for each running PM h, firstly compute the posterior usage state, expressed as $\mathbf{PS_{h,v_k}}$, which refers to the new usage state when VM v_k is placed on it, then determine the domain $\mathbf{f}(\mathbf{PS_{h,v_k}})$ for each PM h, which has sufficient resources to host VM v_k, and finally select a PM Host(v_k) for VM v_k by following the steps below.

(1) Check whether P_{AD} is empty, where P_{AD} is a PM set in which each PM h satisfies $\mathbf{f}(\mathbf{PS_{h,v_k}}) = \text{AD}$;

(2) if P_{AD} is nonempty, then let Host$(v_k) = h_{\text{AD}}$, where $h_{\text{AD}} \in P_{\text{AD}}$ is the PM with the highest overall resource utilization, otherwise check whether P_{SD} is empty, where P_{SD} is a PM set in which each PM h satisfies $\mathbf{f}(\mathbf{PS_{h,v_k}}) = \text{SD}$ and go to the next step;

(3) if P_{SD} is nonempty, then let Host$(v_k) = h_{\text{SD}}$, where $h_{\text{SD}} \in P_{\text{SD}}$ is the PM with the most balanced utilization of multidimensional resource, otherwise turn on a PM h_{New} and let Host$(v_k) = h_{\text{New}}$.

By comparing the experimental results of EAGLE and First Fit, Li et al. [17] validated that EAGLE work efficiently on the balanced utilization of multidimensional resources, and, thus, reduce the number of required servers. Besides, they analyzed the impact of model parameters (R_0 and r_0) on the algorithm performance and got that the least number of servers is required when R_0 is equal to 0.8 and r_0 is equal to 0.1. In this paper, it is adopted that $R_0 = 0.8$ and $r_0 = 0.1$.

3.2. An Improved Model for Both Deterministic and Stochastic Resources. As with [17], this work focuses on the VMP with multidimensional resources and aims to reduce the number of required servers. However, different from [17], VMs' stochastic demands and periodic server consolidation are taken into account in this paper. Considering these characteristics, a new partition is presented for the M-dimensional resource usage state space.

For a PM h with $\mathbf{PS_{h,v_k}} \le 0.2$ and $\mathbf{f}(\mathbf{PS_{h,v_k}}) = \text{SD}$, it can be regarded as a candidate for the subsequent server consolidation, thus the criterion for selecting Host(v_k) from these PMs should focus on migration cost reduction. In view of this, the original SD domain is partitioned and then a new domain LD is obtained where the utilization ratio is less than 0.2 for all resources. Besides, for a usage state $\mathbf{US_h} \ge 0.8$, it takes on a high and balanced utilization of M-dimensional resources, which is the feature of states in the AD domain. So these states are incorporated into the AD domain and then a new domain HD is obtained where the utilization ratio is higher than 0.8 for all resources. Figure 3 shows the improved two-dimensional space partition model. The improved model includes four domains as follows.

(i) Unbalanced utilization domain (NBD): this domain indicates that there is an obvious disequilibrium of multidimensional resource utilization, which should be avoided.

(ii) Balanced utilization domain (BD): this domain represents the balanced case.

(iii) High utilization domain (HD): this domain implies that all resources are nearly exhausted, which is an ideal case for all PMs.

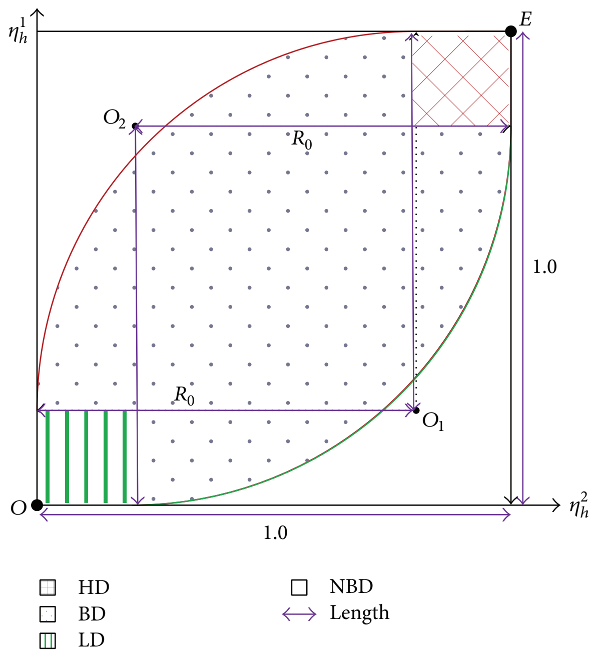


Figure 3: The improved multidimensional space partition model.

(iv) Low utilization domain (LD): this domain indicates that the utilization ratio is very low for all resources. The PMs whose corresponding usage state lies in this domain may be candidates for the subsequent server consolidation.

For the M-dimensional space partition model, according to $\mathbf{US_h}$, the parameters r_0 and R_0, the domain determination function $\mathbf{f}'$ can be defined as

$$\mathbf{f}'(\mathbf{US_h}) = \begin{cases} \text{HD}, & \text{if } \forall r, \eta_h^r \geq R_0 \\ \text{LD}, & \text{else if } \forall r, \eta_h^r \leq 1 - R_0 \\ \text{BD}, & \text{else if } \forall r, \text{distance}(\mathbf{US_h}, \mathbf{O_r}) \leq R_0 \\ \text{NBD}, & \text{otherwise.} \end{cases} \quad (16)$$

In related studies, the stochastic resources refer primarily to the network bandwidth. Moreover, it is shown in [22, 24] that the VMs' bandwidth demands approximately follow a normal distribution. So this paper assumes VM v_i's demand d_i^r $(r \in Q)$ follows a normal distribution $N(\mu_i^r, (\sigma_i^r)^2)$. Based on this assumption, for the stochastic resource r, the upper confidence limits of the total demand of all the VMs on the server h can be calculated as

$$\sum_{i \in U_h} \mu_i^r + \Phi^{-1}(1-\alpha)\sqrt{\sum_{i \in U_h} (\sigma_i^r)^2}, \quad \forall r \in Q, \quad (17)$$

where $\Phi^{-1}(1-\alpha)$ is the quantile of $N(0,1)$ at probability α.

Thus, η_h^r can be defined as

$$\eta_h^r = \frac{\sum_{i \in U_h} d_i^r}{C_h^r}, \quad \forall r \in D,$$
$$\eta_h^r = \frac{\sum_{i \in U_h} \mu_i^r + \Phi^{-1}(1-\alpha)\sqrt{\sum_{i \in U_h} (\sigma_i^r)^2}}{C_h^r}, \quad \forall r \in Q. \quad (18)$$

4. Strategy and Algorithms

4.1. A Two-Phase Optimization Strategy. Figure 4(a) shows the interactions between the two phases in optimization, which are, respectively, for VM deployment and server consolidation. At each time instant t, firstly checks whether there are newly arriving VMs to be deployed; if there are, then enter into the VM deployment phase and place these VMs in the order they arrive. The procedure to deploy a VM is shown in Figure 4(b), where P_1 represents the set of the PMs with some VMs running.

When the server consolidation is triggered, an optimal scheme can be obtained by solving the model in Section 2.2. According to formula (17), the deterministic form equivalent with the chance constraint (10) can be derived as follows:

$$\sum_{i \in V(t)} y_{ih}(t) \cdot \mu_i^r + \Phi^{-1}(1-\alpha)\sqrt{\sum_{i \in V(t)} y_{ih}(t) \cdot (\sigma_i^r)^2} \leq C_h^r,$$
$$\forall h \in P(t), r \in Q. \quad (19)$$

Based on the equivalent form, the Model_0 can be transformed into a deterministic model called Model_1, which includes (3)–(9), (11)–(13), and (19). Model_1 is a multiobject integer programming model. From a service provider's perspective, minimizing the number of required servers can reduce more cost than minimizing the number of migrations, so the two objectives can be arranged in order of importance, given that, the Lexicographic method [28] is adopted to solve Model_1. However, the method is only applied to solve a small-scale problem of server consolidation. As the scale of the problem gets larger, it may happen that the general solver such as Gurobi and Cplex cannot get a feasible solution in a reasonable time. Thus, a heuristic algorithm of live migration is designed in Section 4.3.

4.2. A Modified EAGLE Algorithm for VM Deployment. Based on the improved multidimensional space partition model, a modified EAGLE algorithm (MEAGLE) is proposed. With the MEAGLE algorithm, the steps below can be followed to select Host(v_k) for VM v_k.

(1) For each PM h in P_1, compute the posterior usage state $\mathbf{PS_{h,v_k}}$. Let $P_{\text{fea}}(v_k) = \{h \mid h \in P_1 \text{ and } \mathbf{PS_{h,v_k}} \leq 1\}$.

(2) For each PM h in $P_{\text{fea}}(v_k)$, determine the domain $\mathbf{f}'(\mathbf{PS_{h,v_k}})$ according to the formula (16) and $\mathbf{PS_{h,v_k}}$.

(3) Divide $P_{\text{fea}}(v_k)$ into four sets: P_{HD}, P_{LD}, P_{BD}, P_{NBD}. P_{HD} is a PM set in which each PM h satisfies

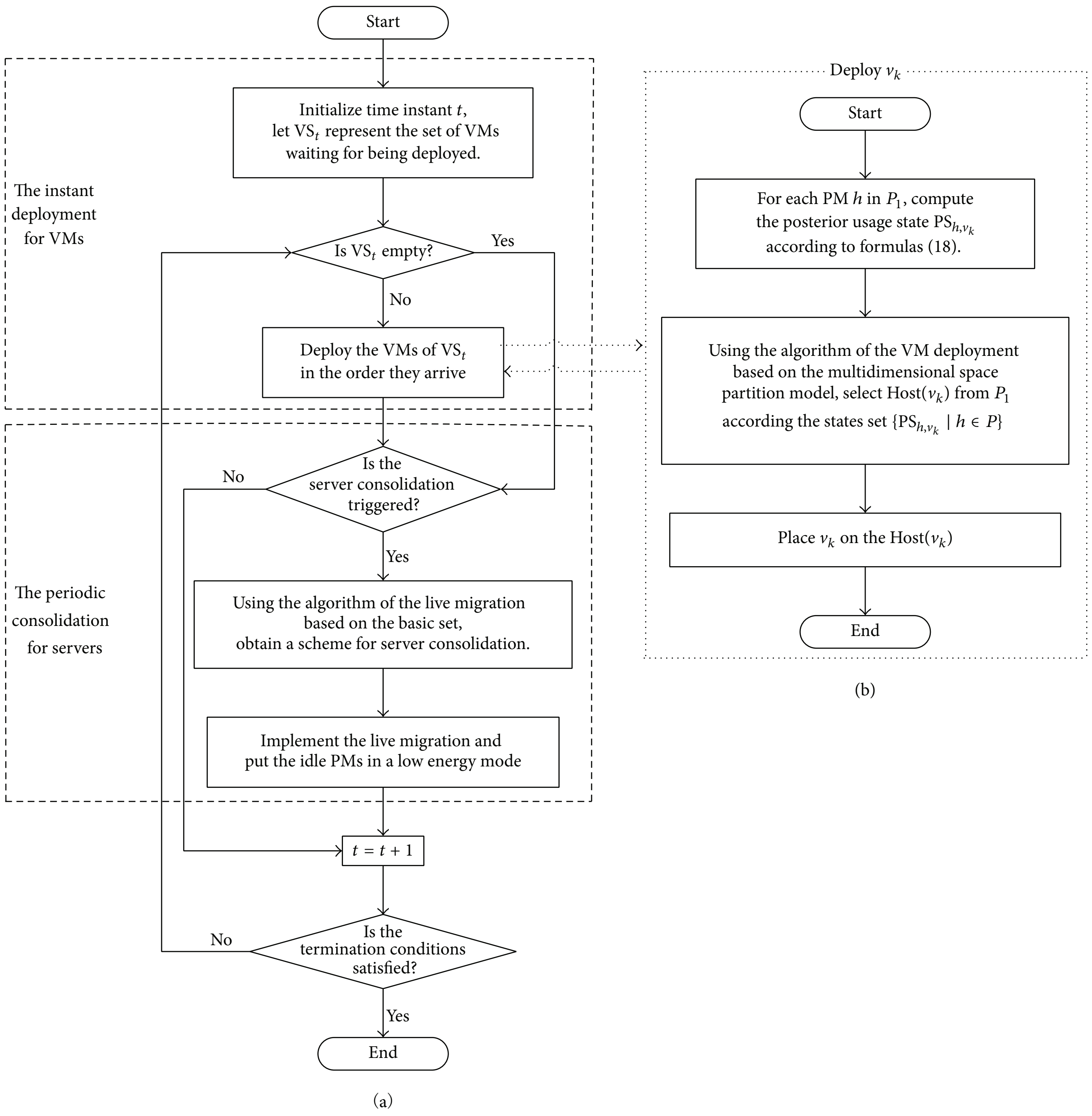

Start
Initialize time instant t, let VS_t represent the set of VMs waiting for being deployed.
The instant deployment for VMs
Is VS_t empty?
Yes
No
Deploy the VMs of VS_t in the order they arrive
Is the server consolidation triggered?
No
Yes
The periodic consolidation for servers
Using the algorithm of the live migration based on the basic set, obtain a scheme for server consolidation.
Implement the live migration and put the idle PMs in a low energy mode
$t = t + 1$
Is the termination conditions satisfied?
No
Yes
End
(a)
Deploy v_k
Start
For each PM h in P_1, compute the posterior usage state PS_{h,v_k} according to formulas (18).
Using the algorithm of the VM deployment based on the multidimensional space partition model, select Host(v_k) from P_1 according the states set $\{PS_{h,v_k} \mid h \in P\}$
Place v_k on the Host(v_k)
End
(b)

FIGURE 4: The flowchart of the two-phase optimization strategy.

$\mathbf{f}'(\mathbf{PS_{h,v_k}}) = \mathrm{HD}$, and the definitions of $P_{\mathrm{LD}}, P_{\mathrm{BD}}, P_{\mathrm{NBD}}$ are similar with P_{HD}.

(4) Check whether P_{HD} is empty. If P_{HD} is nonempty, then let $\mathrm{Host}(v_k) = h_{\mathrm{HD}}$, where $h_{\mathrm{HD}} \in P_{\mathrm{HD}}$ is the PM with the highest overall resource utilization, otherwise go to step (5).

(5) Check whether P_{BD} is empty. If P_{BD} is nonempty, then let $\mathrm{Host}(v_k) = h_{\mathrm{BD}}$, where $h_{\mathrm{BD}} \in P_{\mathrm{BD}}$ is the PM with the most balanced utilization of multidimensional resource, otherwise go to step (6).

(6) Check whether P_{LD} is empty. If P_{LD} is nonempty, then let $\mathrm{Host}(v_k) = h_{\mathrm{LD}}$, where $h_{\mathrm{LD}} \in P_{\mathrm{LD}}$ is the PM with the largest number of VMs, otherwise turn on a PM h_{New} and let $\mathrm{Host}(v_k) = h_{\mathrm{New}}$.

For a PM h with the usage state $(\eta_h^1, \eta_h^2, \ldots, \eta_h^r, \ldots, \eta_h^M)$, the overall resource utilization $\mathfrak{R}_h$ and the equilibrium degree of M-dimensional resource utilization $\mathscr{D}_h$ can be calculated as follows:

$$\mathfrak{R}_h = \frac{1}{M}\sum_{r\in D\cup Q}\eta_h^r$$
$$\mathscr{D}_h = \sqrt{\sum_{r\in D\cup Q}\left(\eta_h^r - \frac{1}{M}\sum_{r\in D\cup Q}\eta_h^r\right)^2}. \tag{20}$$

4.3. A Live Migration Algorithm for Server Consolidation. Based on the improved multidimensional space partition model and MEAGLE algorithm, a live migration algorithm based on the basic set (LMABBS) is proposed. Let the basic set $P_{\mathrm{BS}} = \{h \mid \mathbf{f}'(\mathbf{US_h}) = \mathrm{HD}$ or $\mathfrak{I}_h = \max_{j\in P_1}\mathfrak{I}_j\}$ and the nonbasic set $P_{\mathrm{NBS}} = P_1 \setminus P_{\mathrm{BS}}$, where $\mathfrak{I}_j$ indicates the number of VMs on the PM j. The key idea of LMABBS is to turn off as many PMs in P_{NBS} as possible by live migration. The algorithm is terminated when P_{NBS} is empty or there is no server in P_{NBS} as a candidate for server consolidation. Here a candidate means a PM on which all VMs can be migrated to other PMs.

The function JudgeMigratable(h_i) in Algorithm 1 describes the procedures to determine whether a PM h_i is a candidate and returns a plan for VMs' migrations if so. The main points of the function are as follows.

(1) Firstly according to MEAGLE, check each VM v_i on server h_i in the ascending order of VMs' ID, to see whether it can be migrated into a directly-deployed server, on which the original VMs need not be migrated out before VM v_i's migration.

(2) If there is only one VM v_l without directly-deployed servers, then go on looking for an indirectly-deployed server, into which v_l can be migrated after some original VMs on it are migrated out, otherwise judge that the PM h_i is not a candidate.

(3) In order to keep migration cost as low as possible, three types of indirectly-deployed servers are considered: (a) a PM on which an original VM is needed to migrate out before VM v_i's migration, (b) a PM on which two original VMs are needed to migrate out before VM v_i's migration, and (c) a PM on which an original VM is needed to be exchanged with a VM on the other non-h_i PM before VM v_i's migration.

(4) If there are two VMs on server h_i, and there exists one VM without directly-deployed and indirectly-deployed servers, then return to step (1) and check each VM again in the descending order of VMs' ID.

5. Simulation Experiment

5.1. Simulation Configuration. Consider a cloud data center with 200 servers with the same capacity. Each VM is assumed to have three types of resources demands: CPU, memory, and bandwidth. Set $D = \{\mathrm{CPU}, \mathrm{memory}\}$, $Q = \{\mathrm{bandwidth}\}$, and $\alpha = 5\%$. The ratio of the resource demand to server capacity is employed to identify the demand intensity of each resource. Among all resources, the resource with the largest DCR value is defined as the intensive resource and others are defined as the nonintensive resources. To emulate a typical data center setting, four groups of VMs are generated: CPU-intensive VMs, memory-intensive VMs, bandwidth-intensive VMs, and general VMs, where each group contains a quarter of all VMs. For each VM in the first three groups, the intensive resource randomly selects its DCR value from a higher range $[30\%, 40\%]$ and other nonintensive resources randomly select their DCR values from a lower range $[5\%, 10\%]$, which is based on the data generation method in [26]. For each VM in the fourth group, all resources select their DCR value from a general range $[3\%, 10\%]$ to validate the MEAGLE algorithm when $R_0 = 0.8$. For the stochastic bandwidth demand, the selected DCR value represents the ratio between the mean of the bandwidth and the server capacity. Coefficients of variation for all VMs' bandwidth demands are set 0.2 by default.

Assume that the number of VM requests reaching the cloud data center in a period follows a Poissonian distribution with a mean λ. Here a period includes 60 time instants. In order to evaluate the performance of our MEAGLE and LMABBS, respectively, the experiment consists of two parts: (1) comparing the solutions of MEAGLE and EAGLE and (2) comparing the solutions of LMABBS and Gurobi. The Gurobi solutions are obtained by using Gurobi 5.5.0 to solve Model_1, which is formulated by YALMIP [29] in MATLAB environment. All experiments are performed on a computer with Intel Core i7 3.40 GHz CPU and 4 GB RAM.

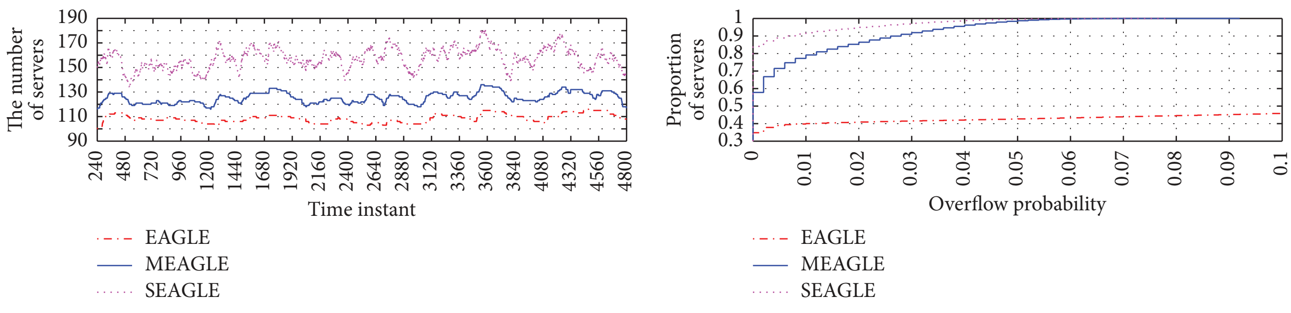
5.2. The Result and Analysis

5.2.1. Comparison between MEAGLE and EAGLE. This subsection is conducted to evaluate the performance of MEAGLE on the instant-deployment for VMs. Assume that there is no VM on each server at $t = 0$. Eighty periods are considered and 220 VMs on average are arriving in a period (i.e., $\lambda = 220$). According to the settings, a VM set is generated and called VmSet which includes all the VMs arriving during the eighty periods. The VMs are deployed instantly as soon as they arrive. For each VM $v_k \in$ VmSet, EAGLE, SEAGLE, and MEAGLE are, respectively, adopted to search Host(v_k). A VM's bandwidth demand is constant and taken as the mean in EAGLE while it is regarded as a random variable in SEAGLE and MEAGLE. When using SEAGLE, firstly for each PM h in P_1, compute the posterior usage state $\mathbf{PS_{h,v_k}}$ according to formula (18) then perform the same steps as EAGLE described in Section 3.

Figure 5(a) shows the number of required servers at each time instant. The reduction of the servers at some time instants is a result of turning off the idle servers as some VMs are removed. To further validate the actual overflow probability, the following procedures are carried out. Firstly, for each VM's bandwidth demand at per time instant, we sample the trace to obtain 1000 samples of the momentary value of bandwidth consumption. Then for each packing obtained by the algorithms at per time instant and for each server in the packing, we sum up the total momentary bandwidth consumption per server and check whether

```
Procedure MigPlan = JudgeMigratable ($h_i$)
(1) Initialize $S_t(h_i)$ as the set of VMs on server $h_i$ at time instant $t$
(2) $n \leftarrow S_t(h_i)$. size
(3) for $k = 1$ to $n$ do
(4)     According to MEAGLE, search for the Host($v_k$) for $v_k$, where $v_k \in S_t(h_i)$.
(5)     if Host($v_k$) $\in P_1 \setminus \{h_i\}$
(6)         MigPlan = [MigPlan; $v_k h_i$ Host($v_k$)]
(7)         Update the usage state of Host($v_k$) after $v_k$ is placed on it.
(8)     else
(9)         $l \leftarrow k$;
(10)        $n$count $\leftarrow$ $n$count + 1
(11)        if $n$count > 1
(12)        /* In $S_t(h_i)$, there exists more than one VM without directly-deployed servers.
(13)            MigPlan = [];
(14)            return MigPlan
(15)        end if
(16)    end if
(17) end for
(18) if $n$count == 1
(19) /* In $S_t(h_i)$, there is only one VM $v_l$ without directly-deployed servers.
(20)     if $\exists h_1, h_2 \in P_1 \setminus \{h_i\}$, $v_l$ can be placed on $h_1$
         after an original VM on $h_1$ is migrated into $h_2$, then
(21)         Update MigPlan
(22)     else if $\exists h_1, h_2, h_3 \in P_1 \setminus \{h_i\}$, $v_l$ can be placed on $h_1$
         after two original VMs on $h_1$ are respectively migrated into $h_2$ and $h_3$, then
(23)         Update MigPlan
(24)     else if $\exists h_1, h_2 \in P_1 \setminus \{h_i\}$, $v_l$ can be placed on $h_1$
         after an original VM on $h_1$ is exchanged with a VM on $h_2$, then
(25)         Update MigPlan
(26)     else
(27)         MigPlan = []
(28)         return MigPlan
(29)     end if
(30) end if
(31) return MigPlan
```

Algorithm 1: The procedures to judge whether a PM h_i is a candidate for server consolidation.



(a) The number of required servers at each time instant

(b) Overflow probability and target overflow probability 0.05

Figure 5: The comparisons of simulation results.

the server capacity constraint is broken. Figure 5(b) shows the cumulative distribution function (CDF) of server compliance with the overflow probability. From Figure 5, it can be seen, by EAGLE the number of required servers at each time instant is the least but only about 42% of servers respect the target overflow probability $\alpha = 5\%$. For MEAGLE and SEAGLE, almost all servers obey the target overflow probability, yet MEAGLE uses a smaller number of servers.

In order to better evaluate the performance of MEAGLE, a series of eight experiments is performed to deploy the VMs in VmSet under different coefficients of variation with respect to VMs' bandwidth demands. Here the coefficient of variation

Table 1: The simulation results with different variation coefficients.

Variation coefficients	EAGLE		SEAGLE		MEAGLE	
	ω_1	τ_1	ω_2	τ_2	ω_3	τ_3
0.01	108.7139	6.03%	107.6639	0.11%	**104.9861**	0.15%
0.05	108.7139	28.28%	114.1208	0.55%	**107.8042**	0.60%
0.1	108.7139	44.58%	127.7708	0.46%	113.0083	1.48%
0.15	108.7139	53.28%	141.9153	0.37%	118.6417	1.42%
0.2	108.7139	57.30%	151.4569	0.37%	122.4278	1.27%
0.25	108.7139	59.13%	157.0264	0.39%	124.9875	1.34%
0.3	108.7139	60.56%	163.4546	0.83%	131.5111	2.48%
0.35	108.7139	62.70%	163.7028	1.64%	136.9597	4.33%

Table 2: The possible results obtained by Gurobi.

The type of results	Δ	The state of solution	The percentage of instances with C_i
C_1	Decrease	Optimal solution	44.52%
C_2	Decrease	The best feasible solution	31.45%
C_3	Invariant	Optimal solution	4.42%
C_4	Invariant	The best feasible solution	5.65%
C_5	—	No solution	13.95%

characterizes the variation in demand. The simulation results are shown in Table 1, where ω_i $(i = 1, 2, 3)$ represents the average number of required servers in a time unit and τ_i $(i = 1, 2, 3)$ refers to the proportion of servers violating the target overflow probability. From Table 1, it can be seen, (1) with the same volatility (or coefficient of variation), τ_1 is significantly larger than τ_2 and τ_3 while the difference between τ_2 and τ_3 is very small; (2) as the coefficient of variation increases, τ_1 is rising sharply while τ_2 and τ_3 are keeping at very low level; (3) ω_3 is 2.49%–20.40% less than ω_2 with the same coefficient of variation and ω_3 is the least when the variation coefficient is 0.01 and 0.05.

From the results shown in Table 1, it can be concluded that SEAGLE and MEAGLE work well to obey the target overflow probability, while EAGLE brings a high proportion of servers violating the target overflow probability as it cannot detect the change of VM's burst level. Furthermore, though MEAGLE is slightly inferior to SEAGLE in lowering the overflow probability of servers as shown in Figure 5(b), MEAGLE uses a smaller number of servers than SEAGLE does. This is primarily due to the fact that by SEAGLE, a VM v_k must not be placed on a PM h with $\mathbf{f}(\mathbf{PS}_{\mathbf{h},\mathbf{v}_\mathbf{k}}) = \text{FD}$ and a new PM will be turned on if there is no proper server to host v_k. VMs in our study have different resource intensities and a PM is more likely to incur an imbalanced utilization of multidimensional resources after it hosts a nongeneral VM (such as a CPU-intensive VM). Therefore, according to the procedures of SEAGLE and MEAGLE, SEAGLE will use more servers than MEAGLE obviously.

5.2.2. Comparison between LMABBS and Gurobi. Within the two-phase optimization strategy, the server consolidation is implemented for every 4 periods. In order to test the performance of LMABBS, $\lambda = 11, 12, \dots, 26$ is set and twenty running states are chosen for each value of λ. A running state for server consolidation is regarded as a problem instance and then 320 instances are generated. For each instance, the corresponding problem is solved, respectively, by the LMABBS and Gurobi. As a note, when solving large scale problems by Gurobi, one bad case that may arise is that solutions cannot be obtained due to out of memory. So the values of λ in this subsection are relatively smaller than that in the previous subsection. Here set the solving time limit as 30 minutes for Gurobi. Then all instances are divided into five groups, according to all possible results ($C_1 \sim C_5$) obtained by Gurobi shown in Table 2. Δ represents the variation in the number of required servers by server consolidation. The state with "the best feasible solution" represents that Gurobi cannot get an optimal solution but can get a feasible solution in 30 minutes, while the state with "no solution" means that Gurobi cannot produce a feasible solution in 30 minutes. Let S_i denote a set of the problem instances, where $i = 1, 2, \dots, 5$. Each instance in S_i has the same result as C_i if it is solved by Gurobi.

By statistical analysis on experimental results, the following points can be found out: (1) for the 80.95% instances in S_1, the LMABBS solution coincides with Gurobi solution on the number of required server and migrations; (2) by LMABBS, though there exist 19.05% instances in S_1 without optimal solutions, there are still 29.17% of these instances with the minimum number of required servers (Let $\widehat{S}_1$ represent the set of the 19.05% instances in S_1); (3) in S_2, there are 74% instances for which the solutions of the two methods are identical to the number of required server and there are 60% instances for which the solutions of the two methods are identical to the number of required server and migrations; (4) for the instances in S_3 and S_4, LMABBS as well as Gurobi cannot obtain a valid consolidation solution which can reduce the number of required servers; and (5) in S_5, there are 76.19% instances with valid consolidation solutions

Table 3: Comparison of LMABBS solution with Gurobi solution.

The sets of instances	ρ_h			ρ_m			ρ_t		
	Ave	Max	Min	Ave	Max	Min	Ave	Max	Min
$\widehat{S}_1$	1.1003	1.2500	1.000	0.6988	1.25	0	0.0197	0.1159	1.5621×10^{-5}
S_2	1.0286	1.2000	1.000	0.9022	1.50	0	0.0001	0.0008	5.0555×10^{-6}
$S_3 \cup S_4$	1.000	1.000	1.0000	—	—	—	4.7288×10^{-5}	0.0002	5.3054×10^{-6}

by LMABBS and the number of required servers declines by 15.98% on average.

For different sets of instances, the comparison results of LMABBS solution with Gurobi solution are shown in Table 3, where ρ_h, ρ_m, and ρ_t, respectively, represent the ratios between the number of required servers, between the migration number, and between the computation time on the LMABBS solution and Gurobi solution. As one can see, ρ_m are both equal to 0 for some instances in $\widehat{S}_1$ and S_2. This is because that the servers to turn off are chosen only from the nonbasic set when implementing server consolidation; in other words, if there exists a server in the basic set that is a candidate for server consolidation, the server will not be turned off by live migration because of high migration cost. Besides, according to the comparison results for S_3 and S_4, it can be seen that LMABBS can rapidly identify a situation that the deployment of VMs on the collection of hosts is compact enough and, thus, spend little time looking for an invalid consolidation solution. Taken together, the comparison results suggest that the difference is fairly small between the number of required servers obtained by LMABBS and Gurobi and LMABBS has a significant advantage in lowering migration cost and improving the computational efficiency.

6. Conclusions

This paper studies the dynamic placement of VMs with deterministic and stochastic demands for green cloud computing. A two-phase optimization strategy is presented including instant VM deployment and periodic server consolidation. The innovations of the strategy can be listed as follows: (1) taking into account stochastic resources, the VMs' resource demands are described as random variables and the equivalent form of chance constraints is introduced to guarantee the availability for these resources; (2) when deploying VMs, the migration cost generated in the subsequent server consolidation is considered; and (3) given that this work focuses on the VMs with different resource intensities and the VMs with complementary demands can be placed on the same server, a temporary imbalanced utilization of multidimensional resources is allowed in our method.

For VM deployment and server consolidation, a modified EAGLE algorithm (MEAGLE) and a live migration algorithm based on the basic set (LMABBS) are proposed, respectively. The simulation results demonstrate that MEAGLE not only virtually ensures as good performance of user application as SEAGLE, but also reduces the number of required servers by 2.49% to 20.40% compared with SEAGLE. In addition, there is fairly small difference between the number of required servers obtained by LMABBS and Gurobi, but LMABBS has a computational efficiency unmatched by Gurobi. For the instances without a feasible solution in a reasonable time by Gurobi, LMABBS can obtain valid consolidation solutions and lower the number of required servers by an average of 15.98%.

For the public clouds that do not provide explicit SLAs guarantees on VM bandwidth today, the application of MEAGLE algorithm will achieve a specified quality-of-service and then help to enhance their credibility and win more customers. Besides, the LMABBS algorithm will help to improve the efficiency of implementing server consolidation and reduce the waste of energy consumption effectively. Thus the proposed two-phase optimization strategy based on MEAGLE and LMABBS will provide a good solution to achieve high level service performance and to minimize energy consumption for green cloud computing. As a future work, we will intend to draw a specific migration plan for the implementation of server consolidation, with considering the intermediate migrations needed when a deadlock occurs in the actual migration process.

Conflict of Interests

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

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant no. 71172071) and the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant no. 20120031110036).

References

[1] J. Kaplan, W. Forrest, and N. Kindler, "Revolutionizing data center energy efficiency," Tech. Rep., McKinsey & Company, 2008.

[2] W. Forrest, "How to cut data centre carbon emissions?" Website, December, 2008.

[3] VMware Infrastructure, https://www.vmware.com/cn/virtualization/.

[4] C. Hyser, B. Mckee, R. Gardner, and B. J. Watson, "Autonomic virtual machine placement in the data center," Tech. Rep. HPL-2007-189, Hewlett Packard Laboratories, 2007.

[5] J. Xu and J. Fortes, "A multi-objective approach to virtual machine management in datacenters," in *Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC '11)*, pp. 225–234, Karlsruhe, Germany, June 2011.

[6] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, 2009.

[7] H. N. Van, F. D. Tran, and J. M. Menaud, "Autonomic virtual resource management for service hosting platforms," in *Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD '09)*, pp. 1–8, Vancouver, Canada, May 2009.

[8] Amazon EC2 Instances, http://aws.amazon.com/cn/ec2/instance-types/.

[9] Z. Huang and D. H. K. Tsang, "SLA guaranteed virtual machine consolidation for computing clouds," in *Proceedings of the IEEE International Conference on Communications (ICC '12)*, pp. 1314–1319, Ottawa, Canada, June 2012.

[10] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in *Proceedings of the 7th International Conference on Autonomic Computing*, pp. 11–20, Washington, DC, USA, June 2010.

[11] M. Chen, H. Zhang, Y. Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective VM sizing in virtualized data centers," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 594–601, Dublin, Ireland, May 2011.

[12] E. Bin, O. Biran, O. Boni et al., "Guaranteeing high availability goals for virtual machine placement," in *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS '11)*, pp. 700–709, Minneapolis, Minn, USA, July 2011.

[13] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, 2011.

[14] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 266–278, 2010.

[15] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS '06)*, pp. 373–381, Vancouver, Canada, April 2006.

[16] S. Mehta and A. Neogi, "ReCon: a tool to recommend dynamic server consolidation in multi-cluster data centers," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS '08)*, pp. 363–370, Salvador, Brazil, April 2008.

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

[18] VMware Infrastructure, "Resource management with VMware DRS," Whitepaper, 2006.

[19] IBM, "System planning tool," http://www-947.ibm.com/systems/support/tools/systemplanningtool/.

[20] R. G. Michael and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, WH Freeman, San Francisco, Calif, USA, 1979.

[21] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*, pp. 267–280, Melbourne, Australia, November 2010.

[22] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of datacenter traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference (IMC '09)*, pp. 202–208, Chicago, Ill, USA, November 2009.

[23] J. Kleinberg, Y. Rabani, and É. Tardos, "Allocating bandwidth for bursty connections," *SIAM Journal on Computing*, vol. 30, no. 1, pp. 191–217, 2000.

[24] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proceedings of the 30th IEEE International Conference on Computer Communications*, pp. 71–75, Shanghai, China, April 2011.

[25] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," in *Proceedings of the 31st Conference on Computer Communications (INFOCOM '12)*, pp. 2861–2865, IEEE, Orlando, Fla, USA, March 2012.

[26] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient VM placement with multiple deterministic and stochastic resources in data centers," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM '12)*, pp. 2505–2510, Anaheim, Calif, USA, December 2012.

[27] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti, "VM placement strategies for cloud scenarios," in *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, pp. 852–859, Honolulu, Hawaii, USA, June 2012.

[28] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[29] J. Löfberg, "YALMIP: a toolbox for modeling and optimization in MATLAB," in *Proceedings of the IEEE International Symposium on Computer Aided Control System Design*, pp. 284–289, Taipei, Taiwan, September 2004.